Subspace Constrained Gaussian Mixture Models for Speech Recognition

Scott Axelrod, *Member, IEEE,* Vaibhava Goel, *Member, IEEE,* Ramesh A. Gopinath, *Senior Member, IEEE,* Peder A. Olsen, *Member, IEEE,* and Karthik Visweswariah, *Member, IEEE*

***Abstract*— A standard approach to automatic speech recognition uses Hidden Markov Models whose state dependent distributions are Gaussian mixture models. Each Gaussian can be viewed as an exponential model whose features are linear and quadratic monomials in the acoustic vector. We consider here models in which the weight vectors of these exponential models are constrained to lie in an affine subspace shared by all the Gaussians. This class of models includes Gaussian models with linear constraints placed on the precision (inverse covariance) matrices (such as diagonal covariance, MLLT, or EMLLT) as well as the LDA/HLDA models used for feature selection which tie the part of the Gaussians in the directions not used for discrimination. In this paper we present algorithms for training these models using a maximum likelihood criterion. We present experiments on both small vocabulary, resource constrained, grammar based tasks as well as large vocabulary, unconstrained resource tasks to explore the rather large parameter space of models that fit within our framework. In particular, we demonstrate significant improvements can be obtained in both word error rate and computational complexity.**

I. Introduction

STATE of the art automatic speech recognition systems typically use continuous parameter Hidden Markov Models (HMMs) for acoustic modeling. One of the ingredients of the HMM models is a probability distribution $p(x|s)$ for the acoustic vector $x \in \mathbf{R}^d$ at a particular time, conditioned on an HMM state s. Typically, the distribution for each state is taken to be a Gaussian mixture model. When selecting these Gaussian mixture models, the goal is to invest the modeling power and computational time required for decoding wisely. A general approach to achieving this goal is to determine a family of Gaussian models to which each component of the GMM for each state is constrained to belong. In this work we describe a framework for constraining parameters that naturally generalizes many previous successful approaches and provides the system designer flexibility in choosing an optimal compromise between speech recognition performance and computational complexity.

The simplest approach to constraining the Gaussians of a model is to require all of the covariance matrices to be diagonal. Diagonal models have traditionally been used because they are simpler and much less computationally expensive than unconstrained "full covariance" models having an equal number of Gaussians. However, since the diagonal Gaussians have weaker modeling power, it has become commonplace to consider systems with very large numbers of diagonal Gaussians. This then leads back to the same problems that occurred for full covariance models – computational cost and potential overtraining due to large numbers of parameters.

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

Approaches such as Factor Analysis [1] and Probabilistic Principal Component Analysis [2] generalize diagonal models by adding a set of rank-one terms to a diagonal covariance model. They have been applied to speech recognition with moderate success [3]–[5]. However, these direct covariance modeling techniques suffer from computational complexity problems.

One approach that yields significant improvements in accuracy over simple diagonal modeling at minimal computation cost is the Maximum Likelihood Linear Transformation (MLLT) technique [6] where one still uses diagonal Gaussians, but chooses the basis in which the Gaussians are diagonal to be the columns of a linear transform which maximizes likelihood on the training data. One generalization of MLLT models are the "semi-tied full covariance models" of [7] in which the Gaussians are clustered and each cluster has its own basis in which all of its Gaussian are diagonal.

Another class of models generalizing the MLLT models are the Extended Maximum Likelihood Linear Transformation (EMLLT) models [8], [9]. These models constrain the precision (a.k.a *inverse* covariance) matrices to be in a subspace of the space of symmetric matrices spanned by D ($D \geq d$) rank one matrices, so that they may be written as follows:

$$P_g = \Sigma_g^{-1} = \sum_{k=1}^{D} \lambda_g^k \, a_k a_k^T \ . \tag{1}$$

The d-dimensional vectors $\{a_k\}$ are the "tied" parameters shared by all the Gaussians and the $\{\lambda_g^k\}$ are "untied" parameters specific to the individual Gaussians. The EMLLT models provide a class of models that smoothly interpolate between the MLLT models (which is the special case when $D = d$) and full covariance models (where D equals the full dimension $d(d+1)/2$). Experiments in [8], [9] showed that EMLLT models with $D > d$ have lower error rate than equally computationally expensive semi-tied full covariance models with varying numbers of Gaussians and cluster centers.

The MLLT technique is often used in conjunction with Fisher-Rao Linear Discriminant Analysis (LDA) technique [10]–[12] for feature selection. In this technique, one starts with an "unprojected" acoustic data vector x with some large number d of components. (Such a vector at a given time t is typically obtained by concatenating transforms of the log-spectral coefficients of the waveforms for times near t.) One

then obtains acoustic vectors x_1 in some lower dimension d_1 by a linear projection of x by some $d_1 \times d$ matrix U_1. The original formulation of LDA is motivated by an attempt to choose x_1 to consist of features that are most likely to have power to discriminate between states. Subsequently, Campbell [13] showed that the the projection matrix U_1 may be calculated by maximizing the likelihood of a collection of Gaussian distributions which are tied in directions complementary to x_1 and which all have equal covariance matrices. The Heteroscedastic Linear Discriminant Analysis (HLDA) models of [14], [15] generalize Campbell's models for LDA by allowing the covariance matrices in the x_1 directions to be Gaussian dependent.

In this paper we consider state dependent distributions which are Gaussian mixture models that have the constraint that, when written as exponential models, the weight vectors are constrained to lie in an affine subspace shared by all Gaussians in the system. The precise definition of these subspace constrained Gaussian mixture models (SCGMMs) appears in section II. By imposing various conditions on the form of the constraining subspace, both the EMLLT and HLDA models arise as special cases. In section III, we summarize the computational cost for these and other special cases, including, in order of increasing generality: "affine EMLLT" models which are like EMLLT models but allow for a constant matrix (i.e. affine basepoint) to be added to the form (1) of the precision matrix; precision constrained Gaussian mixture (PCGMM) models, where the means are allowed to be arbitrary, but the precision matrices of all the Gaussians are tied to belong to the same affine subspace (of the space of symmetric matrices); and SPAM models, where separate affine subspace constraints are placed on the precisions and means (or, more precisely, on the linear weights). In each case, the computational cost for the large number of Gaussians typical in modern systems is dominated by the number of Gaussians times the dimension of the subspace to which the exponential model weights are constrained. By varying the subspace dimension, the developer is free to choose their preferred tradeoff between computational cost and modeling power. This property is one of the advantages of constraining the exponential model weights rather than the means or covariances directly.

In section IV of this paper (with some additional detail in appendix II), we provide algorithms for training the parameters of models of the types discussed above using the maximum likelihood criterion. Although this training determines the continuous model parameters, it still leave many discrete choices undetermined, including: which type of model to use, what subspace dimensions to use, and how many Gaussian should be contained in the GMM for each state. These choices provides the developer a rather rich landscape of models within which to find a model that provides a preferred tradeoff between computational complexity and recognition performance. The experimental results of sections V and VI explore the landscape of possible models. The results show that one can indeed obtain an improved complexity/performance curve as each generalization is introduced.

Many of the theoretical and experimental results in this paper have appeared in a series of papers [16]–[20] which this paper both summarizes and extends. We have tried to make this paper as self contained as possible, although we make a few references to the previous papers for technical points. The SPAM models were first mentioned in [16], although that paper focused on the special case of precision constrained GMMs. That paper showed that the precision constrained models obtained good improvements over MLLT and EMLLT models with equal per Gaussian computational cost. Subsequent work [21] also obtained significant gains with precision constrained GMMs (although the subspace basis in [21] was required to consist of *positive definite* basis matrices and the experiments there did not compare to the gains obtainable with ordinary MLLT). In [20], [22], precision constrained Gaussian mixture models with speaker adaptive training were applied to large vocabulary conversational speech recognition.

The precision constrained models in the experiments of [16], [20], [22] used the "modified Frobenius" basis (which approximates the ML basis and is described in section IV-C) for the tied parameters and trained the untied parameters using the EM algorithm with the E-step implemented using a quasi-newton search strategy with efficient line searches. The generalization of this to the algorithm described in section IV for efficient full ML training of all parameters of a precision constrained model, although implicit in [16], was first implemented in [17]. Reference [17] also introduced affine EMLLT models and hybrid models (see section III), as well as efficient methods for training all of them. For general SPAM models, reference [18] gave efficient algorithms for finding an untied basis which approximately maximizes likelihood and for full ML training of the untied parameters given a tied basis. That paper also gave the first comparison between SPAM and LDA as techniques for dimensional reduction. Full ML training of general subspace constrained Gaussian mixture models was described in [19]. Discriminative training of these models was described in [23] and will be presented in a more systematic and rigorous manner in future work [24].

II. The Model

We consider speech recognition systems consisting of the following components: a *frontend* which processes a raw input acoustic waveform into a time series of acoustic feature vectors $X = (x_1, x_2, ..., x_T)$, where x_t is a vector in $\mathbf{R}^d$ called the acoustic data vector at time frame t; a *language model* which provides a prior probability distribution $P(W)$ over possible word sequence $W = (w_1, w_2, ..., w_N)$ that the user may utter; an *acoustic model* which gives a conditional probability distribution $P(X|W)$ for the acoustic data given the word sequence; and a *search strategy* that finds the word sequence W that (approximately) maximizes the joint likelihood $P(X, W) = P(X|W)P(W)$.

An HMM based acoustic model provides a set of states $\mathcal{S}$, a probability distribution $p(S|W)$ over possible state sequences $S = (s_1, ..., s_T)$ produced by a word sequence W; and probability density functions $p(x|s)$ associated with a state $s \in \mathcal{S}$ and an acoustic vector $x \in \mathbf{R}^d$. The state sequence model $P(S|W)$ for an HMM has a particular form allowing

efficient calculation, but everything we say in this paper applies with an arbitrary state sequence model. The conditional distribution $P(X|W)$ is written as a sum over hidden state sequences $S = (s_1, ..., s_T)$ that may be associated with the word sequence W:

$$P(X|W) = \sum_S P(X|S)P(S|W) \quad (2)$$

$$P(X|S) = \prod_{t=1}^{T} p(x_t|s_t) \ . \quad (3)$$

We take the distributions $p(x|s)$ for each s to be a Gaussian mixture model

$$p(x|s) = \sum_{g \in \mathcal{G}(s)} \pi_g \mathcal{N}(x; \mu_g, \Sigma_g) \ , \quad (4)$$

where $\mathcal{G}(s)$ is the set of Gaussian associated to state s, π_g is the prior for Gaussian g, and $\mathcal{N}(x; \mu_g, \Sigma_g)$ is a Gaussian distribution with mean μ_g and covariance Σ_g.

For all the model types considered in this paper, the parameters for the Gaussians are required to obey a subspace constraint. To describe this, we rewrite an individual Gaussian distribution,

$$\mathcal{N}(x; \mu, \Sigma) = \det\left(\frac{\Sigma^{-1}}{2\pi}\right)^{1/2} e^{-1/2(x-\mu)^T\Sigma^{-1}(x-\mu)} \ , \quad (5)$$

in the form of an exponential model (as described in Appendix I). To do so, we first write the Gaussian as

$$\mathcal{N}(x; \mu, \Sigma) = e^{-1/2 x^T P x + \psi^T x + K(P,\psi)} \ , \quad (6)$$

where

$$P = \Sigma^{-1} \quad (7)$$

$$\psi = P\mu \quad (8)$$

$$2\,K(P,\psi) = -d\log 2\pi + \log\det(P) - \psi^T P^{-1}\psi \ . \quad (9)$$

The inverse covariance matrix P is called the *precision matrix* and we will refer to ψ simply as the *linear weights*. Next we define the feature vector $f(x)$ and weight vector θ, which are both column vectors of size $d(d+3)/2$:

$$f(x) = \begin{bmatrix} -1/2\,\mathrm{vec}(xx^T) \\ x \end{bmatrix} \qquad \theta = \begin{bmatrix} \mathrm{vec}(P) \\ \psi \end{bmatrix} \ . \quad (10)$$

Here, for any $d \times d$ symmetric matrix S, $\mathrm{vec}(S)$ is a column vector whose entries are the $d(d+1)/2$ upper triangular elements of S written in some fixed order, with the off diagonal elements multiplied by $\sqrt{2}$. This map is defined so as to preserve inner product, i.e. so that $\mathrm{vec}(S_1)^T\mathrm{vec}(S_2)$ equals $\mathrm{Tr}(S_1^T S_2)$ for any symmetric matrices S_1 and S_2. For convenience, we may also write column vectors as pairs, e.g. $\theta = (\mathrm{vec}(P), \psi)$.

Now we may write (6) in standard exponential model format:

$$\mathcal{N}(x; \mu, \Sigma) = e^{\theta^T f(x) + K(P,\psi)} \ . \quad (11)$$

We will interchangeably use (P, ψ) and θ as is convenient. For example, the quantity $K(P, \psi)$ above is a special case of the general definition of $K(\theta)$ given in appendix I.

We now define a Subspace Constrained Gaussian Mixture Model (SCGMM) to be a state model

$$p(x|s) = \sum_{g \in \mathcal{G}(s)} \pi_g p(x|g) \quad (12)$$

$$p(x|g) = \mathcal{N}(x; \mu_g, \Sigma_g) = e^{\theta_g^T f(x) + K(\theta_g)} \quad (13)$$

with the constraint that the θ_g belong to a common F-dimensional affine subspace of the space of all parameters. Letting $b_0 \in \mathbf{R}^{d(d+3)/2}$ be a basepoint of the affine space and B be a matrix of size $d(d+3)/2 \times F$ whose columns form a basis of the subspace, we may write:

$$\theta_g = b_0 + B\lambda_g \ . \quad (14)$$

Note that the distribution (13) with the constraint (14) may be regarded as an exponential distribution with $F+1$ dimensional features $f_B(X)$,

$$f_B(x) = [b_0 \ B]^T f(x) \ . \quad (15)$$

From this point of view, the choice of constraining subspace may be viewed as a choice of exponential model features.

III. Zoology of Models and their Complexities

In this section we give an overview of various special cases of the general Subspace Constrained Gaussian Mixture Models (SCGMMs) which were defined in the previous section. All the special cases arise by placing restrictions on the form of the constraining subspace. For each model type, the parameters are divided into the tied parameters, which define the subspace, and the untied parameters which specify both the point in the subspace and the prior probability for each Gaussian. Similarly, the cost of evaluating all of the Gaussians for a given input vector x is divided into two parts: the cost of precomputing the features $f_B(X)$, which is the number of tied parameters (up to a constant and a factor depending on precisely how we define "computation time"); and a "per Gaussian" cost that gets multiplied by the number of Gaussians. The per Gaussian cost is the time required to calculate

$$[1 \ \lambda_g^T] f_B(x) = b_0^T f(x) + \lambda_g^T (B^T f(x)) \quad (16)$$

(up to a constant term coming from the cost of adding the precomputed values of the normalizing constant $K(\theta_g)$ and exponentiating). For decoding purposes, the first term (coming from the affine shift) may be dropped because it is the same for all Gaussians and is irrelevant to distinguishing between word sequences.

Table I summarizes the precomputation cost (more precisely, the number of tied parameters not counting affine shift parameters), and the per Gaussian cost (the number of untied parameters per Gaussian, not counting the prior) for various models. The first two lines are just the cost for full covariance and diagonal covariance models in dimension d, denoted $\mathrm{FC}(d)$ and $\mathrm{DC}(d)$. Notice that neither model requires any feature precomputation and the full covariance models have a much more expensive per Gaussian cost than the diagonal models. The third line in Table I gives costs for the general subspace constrained GMM model with F dimensional feature space, denoted $\mathrm{SCGMM}(d, F)$. The dimension F here is free

to vary from 0 all the way to the full covariance value $d(d+3)/2$.

Models of type SPAM(d, D, L) are subspace precison and mean (SPAM) models which require the precision matrices and linear weights to be in D and L dimensional affine subspaces, respectively. D is free to range from 0 (or 1 if no affine shift term is included) to the full covariance value of $d(d+1)/2$ while L ranges from 0 to d. For the SPAM models, we may write:

$$P_g = S_0 + \sum_{k=1}^{D} \lambda_g^k S_k \quad (17)$$

$$\psi_g = l_0 + \sum_{k=1}^{L} \lambda_g^{k+D} l_k \ . \quad (18)$$

This corresponds to taking B in (14) to be block diagonal,

$$\theta_g = \begin{bmatrix} \mathrm{vec}(P_g) \\ \psi_g \end{bmatrix} = \begin{bmatrix} \mathrm{vec}(S_0) \\ l_0 \end{bmatrix} + \begin{bmatrix} B_{11} & 0 \\ 0 & B_{22} \end{bmatrix} \lambda_g \ , \quad (19)$$

where B_{11} is the $d(d+1)/2 \times D$ matrix whose k'th column is $\mathrm{vec}(S_k)$ and B_{22} is the $d \times L$ matrix whose k'th columns is l_k.

Gaussian mixture models with an affine subspace constraint only on the precision matrices are SPAM models with $l_0 = 0$ and B_{22} the identity matrix. These model have untied means which may be taken as independent parameters rather than the λ_g^k with $k = D+1, ..., D+d$. We call these models precision constrained Gaussian mixture models, denoted PCGMM(d, D) in the table. The affine EMLLT models EMLLT(d, D) are a special case of precision constrained models in which the basis vectors S_k for the precision subspace are required to be rank one matrices $a_k\, a_k^T$ for $k = 1, ..., D$. Ordinary EMLLT models don't allow for an S_0 (i.e. take $S_0 = 0$), and so require $D \geq d$ (otherwise it is not possible to obtain positive definite precision matrices). MLLT models are the special case of EMLLT models when $D = d$. Diagonal models are just MLLT models with $[a_1 ... a_d]$ equal to the identity matrix.

Although the precomputation cost is a small fraction of the total cost for models considered in this paper which have a large number of Gaussians, they can become a significant part of the expense if the effective number of Gaussians needed to be computed is greatly reduced (for example by the clustering method of [18] discussed in section VII). To reduce the precomputation cost while still retaining much of the performance gain that the PCGMM models have over the EMLLT models (for example), we can use the "hybrid models" Hybrid(d, D, K), which are precision constrained models in which the S_k, $k = 1, ..., D$ are constrained to belong to a subspace spanned by K rank one matrices:

$$S_k^{(hybrid)} = \sum_{l=1}^{K} u_k^l \ a_l a_l^T, \qquad k = 1, ..., D \ . \quad (20)$$

We could define a hybrid version of a general SCGMM model similarly, although for simplicity we have not included a separate line for such a model in Table I.

model	precompute	per Gaussian
DC(d)	0	$2*d$
FC(d)	0	$d(d+3)/2$
SCGMM(d, F)	$F*d(d+3)/2$	F
SPAM(d, D, L)	$D*d(d+1)/2 + L*d$	$D+L$
PCGMM(d, D)	$D*d(d+1)/2$	$D+d$
EMLLT(d, D)	$D*d$	$D+d$
Hybrid(d, D, K)	$D*K+K*d$	$D+d$
HLDA$(d, d_1)+...$	$d*d_1+...$	$0+...$

TABLE I

PRECOMPUTATION AND PER GAUSSIAN COSTS FOR VARIOUS SPECIALIZATIONS OF SUBSPACE CONSTRAINED GAUSSIAN MIXTURE MODELS.

A. HLDA type models

To explain the last model type in Table I, we define, following [25], an HLDA type model to be a class dependent Gaussian Mixture Model in which the vector $x \in \mathbf{R}^d$ is broken into two complementary pieces

$$x \mapsto \begin{bmatrix} x^1 \\ x^2 \end{bmatrix} = Ux, \quad U = \begin{bmatrix} U^1 \\ U^2 \end{bmatrix}, \quad U^i \text{ is } d_i \times d$$

and the Gaussians are tied along one of the subspaces:

$$U\mu_g = \begin{bmatrix} \mu_g^1 \\ \mu^2 \end{bmatrix}$$

$$U\Sigma_g U^T = \begin{bmatrix} \Sigma_g^{11} & 0 \\ 0 & \Sigma^{22} \end{bmatrix} .$$

These constraints are equivalent to the following constraint on the exponential model weights:

$$\psi_g = P_g \mu_g = U^T \begin{bmatrix} \psi_g^1 \\ \psi^2 \end{bmatrix} \quad (21)$$

$$P_g = U^T \begin{bmatrix} P_g^{11} & 0 \\ 0 & P^{22} \end{bmatrix} U \ , \quad (22)$$

where $\psi_g^1 = P_g^{11}\mu_g^1$, $\psi^2 = P^{22}\mu^2$, $P_g^{11} = (\Sigma_g^{11})^{-1}$, and $P^{22} = (\Sigma^{22})^{-1}$.

We call this an HLDA "type" model because one can obtain various specializations depending on what additional constraints one imposes. For example, the case with no additional constraints is the "full covariance" HLDA model considered originally in [14], [15]. As another example, one could require that the vectors ψ_g^1 and P_g^{11} obey SPAM type constraints,

$$P_g^{11} = S_0 + \sum_{k=1}^{D} \lambda_g^k S_k, \qquad \psi_g^1 = l_0 + \sum_{k=1}^{L} \lambda_g^{k+D} l_k \ . \quad (23)$$

In that case, the full model for x is a SPAM model:

$$P_g = U^T \begin{bmatrix} S_0 & 0 \\ 0 & P^{22} \end{bmatrix} U + \sum_{k=1}^{D} \lambda_g^k (U^1)^T S_k U^1 \quad (24)$$

$$\psi_g = U^T \begin{bmatrix} l_0 \\ \psi^2 \end{bmatrix} + \sum_{k=1}^{L} \lambda_g^{k+D} (U^1)^T l_k \ . \quad (25)$$

The last line of Table I indicates the costs for a model "$HLDA(d, d_1) + ...$" obtained by combining HLDA projection with some class of models in dimension d_1. For example, the combination of HLDA with a SPAM

model in the previous paragraph would be written as HLDA(d, d_1) + SPAM(d_1, D, L). The distribution $p(x|g)$ for the HLDA type models breaks up as a product of a piece $|\det(U)|\mathcal{N}(x^2; \mu^2, \Sigma^{22})$ which is independent of g and a g dependent piece $\mathcal{N}(x^1; \mu^1_g, \Sigma^{11}_g)$ which is independent of x^2. This implies that the decoding for the full model only require evaluating the model in dimension d_1 on the features $U^1 x$. In the last line of the table, the ellipsis stand in for the model in lower dimension and its costs.

B. LDA as a special case of HLDA

For completeness, we now review how the standard Linear Discriminant Analysis (LDA) projection matrix can be derived by maximum likelihood training of the special case of HLDA where the covariance matrices Σ^{11}_g are all tied to be equal to the same matrix Σ^{11}, but the model parameters are otherwise unconstrained. So

$$p(x; \mu_g, \Sigma_g) = p(U^1 x; \mu^1_g, \Sigma^{11}) p(U^2 x; \mu^2, \Sigma^{22}) |\det(U)|. \quad (26)$$

If one has a collection of labeled training data $\{x_t, g_t\}$ then parameters maximizing the total log-likelihood $\sum_t \log p(x_t|g_t)$ for a fixed U are

$$\begin{array}{rclcrcl} \mu^1_g & = & U^1 \bar{\mu}_g & & \mu^2 & = & U^2 \bar{\mu}^{tot} \\ \Sigma^{11} & = & U^1 W (U^1)^T & & \Sigma^{22} & = & U^2 T (U^2)^T, \end{array} \quad (27)$$

where $\bar{\mu}^{tot}$ is the mean of all the data points; T is the "total" covariance matrix of all the points; $\bar{\mu}_g$ is the mean of the data labeled by g; and W is the within class covariance matrix, which is just the average over all the training data of $(x_t - \bar{\mu}_g)(x_t - \bar{\mu}_g)^T$.

Plugging in the above values for mean and covariance, the log-likelihood as a function of U, after ignoring constants and dividing through by the total number of data points, is

$$\log \left(\frac{|\det(U)|}{\sqrt{|\det(U^1 W (U^1)^T)\ \det(U^2 T (U^2)^T)|}} \right) . \quad (28)$$

For a fixed d_1, the global maximum of this is obtained by taking the rows of U to be the eigenvectors of $T^{-1}W$ in order of decreasing eigenvalues.

IV. Training

In this section we focus on training the parameters Ω involved in the definition of the distribution $p(x|s)$ given a training corpus consisting of many acoustic waveforms and their corresponding word sequences, which we may consider to be concatenated into a single utterance X^* and word sequence W^*. We assume given a fixed language model $P(W)$ and state sequence model $P(S|W)$. In this paper, we will focus on maximum likelihood (ML) estimation where the goal is to maximize the overall likelihood of "generating" X^* given W^*,

$$L(\Omega) \quad = \quad P(X^*|W^*, \Omega) \ . \quad (29)$$

In [24], we will show that the same training algorithms derived here may also be employed when using other, more "discriminative" training criteria, in particular maximum mutual information estimation (MMIE), whose goal is to maximize the mutual information between X^* and W^*, as well as another training criterion which we call "error weighted training" which is like ML training, except that it gives extra weight to utterances having decoding errors.

Since the utility function $L(\Omega)$ is expensive to evaluate, requiring running through all of the training data, we follow the common procedure of using the Baum-Welch or Expectation Maximization procedure to obtain an auxiliary function Q that can be evaluated cheaply in terms of statistics collected in a single pass over the training data and whose optimization guarantees an increase of the target utility function.

Model training begins by collecting full covariance statistics with respect to some seed model. To find the constrained GMM model that best matches these statistics requires training both the tied parameters (the subspace bases) as well as the untied parameters (the weight for each Gaussian within this subspace). After finding suitable initial parameter values, our strategy is to alternate between maximizing the untied parameters (which may be performed on each Gaussian in parallel), and the tied parameters (whose optimization it is harder to parallelize). The statistics used in the Q function may be updated when convenient to provide a better lower bound on the true likelihood. Training of tied parameters is the hardest step, since it requires collecting a set of full covariance statistics for each Gaussian. Training of the untied parameters is cheaper because it only requires the statistics for the features $B^T f(x)$. An optimization problem which is "dual" to the one of finding the untied paramaters of a PCGMM is the problem of finding a matrix with maximal determinant subjects to some linear constraints. Although the discussion presented here is self-contained, the interested reader may refer to [26] and references cited therein for a thorough mathematical discussion of these dual optimization problems.

The core of the optimizations for both the tied and untied parameters uses a gradient based search technique. Such techniques maximize a function f given an initial vector x by iteratively finding a search direction v (in terms of the gradient of f at x and previous data accumulated during the optimization procedure) and performing a line search to maximize the function $g(t) = f(x + tv)$. The search direction is always chosen so as to have positive inner product with the gradient at x, so the line search is actually a search along the half-line with positive values of t. When the search space is constrained (such as the constraint in all of our searches that the precision matrices be positive definite), the line search is constrained to the segment consisting of those positive values of t for which $x+tv$ obeys the constraints. The simplest gradient base search technique is steepest ascent, in which the search direction is simply taken to be that of the gradient. However, there are many standard algorithms that have much faster convergence. We have used both the conjugate-gradient [27] and limited memory BFGS techniques [28], which produce comparable results for us in roughly comparable runtime. Both techniques implicitly use information about the Hessian of f, without actually having to compute or store the Hessian, to achieve a quadratic rate of convergence. This means that if $d^{(k)}$ is the distance to a local maximum at iteration k, then $d^{(k+1)}$ is of order $(d^{(k)})^2$.

One of the time consuming parts of the gradient search algorithms can be the multiple function evaluation required when performing the line searches. For the conjugate gradient algorithm this is particularly important since that algorithm assumes that the line search is performed to convergence, whereas searches using the BFGS technique don't assume the line search has converged and therefore don't require as many line search evaluations. In any case, we substantially speed things up by using formulas (54) and (66) below which allow for very rapid evaluation of the required function (and their derivatives) along a line of search. This same technique for speeding up the line searches also allows us to easily impose the positive definiteness constraints.

Another time consuming part of the search is the time required to get near a local maximum; it is only once one is near a local maximum that the quadratic convergence property becomes useful. We can substantially improve the time required to optimize by finding good parameter values with which to initialize our search. For the case when means are unconstrained and only the precision matrices are constrained to belong to a subspace, we explain in section IV-C how to choose an initial value for the tied subspace that maximizes a certain quadratic approximation to the overall likelihood function. This subspace can be quickly calculated and provides an excellent approximation to the genuine maximum likelihood subspace, especially when the subspace dimension is large. For a SPAM model, which imposes a subspace constraint on the linear weights as well as the precision matrices, an initial model may be obtained by fixing the precision matrices to be those of some precision constrained model obtained previously, and training constrained linear weights which are a (local) optimum of the Q function. As described in more detail in [18], the latter optimization can be performed quite readily by starting with the solution of a certain singular value decomposition and then alternating between optimizing for the tied and untied linear weight parameters (which in each case requires optimizing a simple quadratic function).

For EMLLT models where the means are unconstrained and the precision matrices are constrained in a subspace with a basis of rank one matrices $\{a_k\, a_k^T\}$, one choice of seed is to take the rank one direction $\{a_k\}$ to be the combination of the MLLT bases for a semi-tied model, i.e. to take $A = (a_1, ..., a_D)$ to be the stacking of the MLLT basis computed for each of several clusters of Gaussians. For affine EMLLT models, another choice of seed is to start with a positive definite affine basepoint S_0 and add rank one directions one at a time. The greedy addition of a new direction a_k, i.e. the maximum likelihood choice of a_k holding fixed the previous directions, is quite tractable, see [17] for more details.

The optimization for the untied parameters is fast enough so that the search starting with any initial point that obeys the positive definiteness constraint is fairly rapid. In all experiments reported here we choose the affine subspaces in such a way that there is always an obvious candidate for initial point. For completeness, Appendix II provides a practical general algorithm for finding an allowed value for the untied parameters, or else finding that no such values exists.

A. Setup for EM Training

Given a starting value Ω_0 for the parameters, the Baum-Welch or expectation maximization (EM) procedure provides an auxiliary function $Q(\Omega, \Omega_0)$ such that $Q(\Omega, \Omega_0) - Q(\Omega_0, \Omega_0)$ is a lower bound for $L(\Omega) - L(\Omega_0)$. Thus a new value of Ω which has higher likelihood than Ω_0 can be obtained by maximizing $Q(\Omega, \Omega_0)$.

The function Q is:

$$Q(\Omega, \Omega_0) = \sum_g \sum_t \gamma(t,g) \log \pi_g\, p(x_t|g;\Omega) \ . \quad (30)$$

It is convenient to write this as

$$Q(\Omega, \Omega_0) = \sum_s N(s) \left[\sum_{g;s(g)=s} \tilde{\pi}_g \log \pi_g \right] + \sum_g n(g) E_{\rho_g}(\log p(x|g,\Omega)) \ , \quad (31)$$

where,

$$\gamma(t,g) = P(g(t) = g|X^*, W^*; \Omega^0), \quad (32)$$

$$n(g) = \sum_t \gamma(t,g), \quad (33)$$

$$N(s) = \sum_{g;s(g)=s} n(g), \quad (34)$$

$$\tilde{\pi}_g = n(g)/N(s(g)), \quad (35)$$

$$\rho_g(x) = \sum_t \frac{\gamma(t,g)}{n(g)} \delta(x - x_t), \quad (36)$$

$$E_{\rho_g}(F(x)) = \int dx \rho_g(x) F(x) \quad (37)$$

$$= \frac{1}{n(g)} \sum_t \gamma(t,g) F(x_t) \ . \quad (38)$$

Here $s(g)$ is the state that Gaussian g is associated with (i.e. the state so that $g \in \mathcal{G}(s)$) and $g(t) = g$ is the indicator for presence of Gaussian g at time t. Thus, $\gamma(t,g)$ is the conditional probability of observing Gaussian g at time t given the training acoustic data and reference word scripts. The quantities $n(g)$ and $N(s)$ are the total counts for Gaussian g and state s, respectively. The quantity $\tilde{\pi}_g$ is the probability, under the old model Ω^0, of seeing Gaussian g given that the state is $s(g)$. For any function F of an acoustic feature vector, $E_{\rho_g}(F(x))$ is the expectation of F under the distribution ρ_g on $\mathbf{R}^d$ that gives sample x_t weight proportional to $\gamma(t,g)$.

To reduce the computational load, many of the models we report on here are trained with "Viterbi style" training for which $P(S|W^*)$ is non-zero only for a single state sequence S^* called the fixed alignment (obtained as the maximum likelihood state sequence from a HMM at some prior stage of processing). In this case, the weight $\gamma(t,g)$ is given by

$$\gamma(t,g) = \begin{cases} \frac{\pi_g^0\, p(x_t|g,\Omega^0)}{p(x_t|s(g),\Omega^0)} & \text{if } s_t^* = s(g) \\ 0 & \text{otherwise.} \end{cases} \quad (39)$$

The usual prior update formula, $\pi_g = \tilde{\pi}_g$, arises as the unique maximum of the first term in (31). The remainder of this section will focus on choosing the remaining parameters on optimizing the second term of (31) with respect to both the

tied and untied parameters for general subspace constrained Gaussian mixtures models and for various subcases. To this end, it is convenient to give two different expressions for $E_{\rho_g}(\log p(x|g,\Omega))$ in terms of sufficients statistics. The first expression is in terms of the parameters θ_g and the expectation $\tilde{f}_g$ of the features vector $f(x)$ under ρ_g:

$$E_{\rho_g}(\log p(x|g,\Omega)) = \tilde{f}_g^T\theta_g + K(\theta_g) \quad (40)$$
$$\tilde{f}_g = E_{\rho_g}(f(x)) \ , \quad (41)$$

where $K(\theta)$ is defined in (9). The second expression is in terms of the model mean μ_g and precision matrix P_g and sample mean $\tilde{\mu}_g$ and covariance $\tilde{\Sigma}_g$ (where the sample distribution is ρ_g):

$$E_{\rho_g}(\log p(x|g,\Omega)) = \frac{1}{2}\left(G(P_g;\Sigma_g) - d\log 2\pi \right) \quad (42)$$
$$G(P;\Sigma) = \log(\det(P)) - \mathrm{Tr}(\Sigma P) \quad (43)$$
$$\Sigma_g = \tilde{\Sigma}_g + (\mu_g - \tilde{\mu}_g)(\mu_g - \tilde{\mu}_g)^T \quad (44)$$
$$\tilde{\mu}_g = E_{\rho_g}(x) \quad (45)$$
$$\tilde{\Sigma}_g = E_{\rho_g}(x\,x^T) - \tilde{\mu}_g\,\tilde{\mu}_g^T \ . \quad (46)$$

B. Q Functions, Gradient and Line Search Speedup

In this section we gather the formulas required to evalute the Q function and it's gradients with respect to both the tied and untied parameters of the Gaussians for both the case where only the precision matrices are constrained and when an arbitrary subspace constraint on exponential model parameters is imposed. We also give the formulas we use to allow for rapidly finding the maximum of Q along a line.

1) Subspace constraint only on precision matrices: If only the precision matrices are constrained (as is the case of PCGMM and EMLLT models), it is convenient and efficient to consider the means μ_g as independent parameters, rather than the linear parameters ψ_g, because we can immediately set μ_g equal to the data mean $\tilde{\mu}_g$ which maximizes $G(P_g;\Sigma_g)$ in (42)-(46). Once we do this, Σ_g equals $\tilde{\Sigma}_g$. Having solved out for the mean, the remaining parameters to optimize for are the tied parameters $\{S_k\}$ and the untied parameters $\{\lambda_g^k, k = 1,...,D\}$ involved in the definition of the precision matrix. The Q function to optimize is

$$Q(\{\lambda_g\},\{S_k\}) = \sum_g n(g) G(P_g;\tilde{\Sigma}_g) \quad (47)$$
$$P_g = S_0 + \sum_{k=1}^{D} \lambda_g^k\, S_k = \sum_{k=0}^{D} \lambda_g^k S_k \ . \quad (48)$$

Note that the expression on the far right of (48) assumes that λ_g^0 is fixed to be 1.

To perform the gradient search, we need to calculate the gradients with respect to the relevant parameters. To begin with, we record the gradient of $G(P;\Sigma)$ with respect to P:

$$\nabla_P G(P;\Sigma) = P^{-1} - \Sigma \ . \quad (49)$$

The gradient is a matrix. The directional derivative in the direction of a variation δP of P is $\mathrm{Tr}((\delta P)^T \nabla_P G(P;\Sigma))$. The pairing of the variation and the gradient used here is the natural inner product on the vector space of matrices. Notice that if the precision matrix is allowed to be an arbitrary (full covariance) matrix, the zero gradient condition tells us that $P = \Sigma^{-1}$, as expected.

The chain rule gives us the derivative of G with respect to the untied parameters:

$$\frac{\partial}{\partial \lambda_g^k} G(P_g;\tilde{\Sigma}_g) = \mathrm{Tr}\left(S_k(P_g^{-1} - \tilde{\Sigma}_g) \right) \quad (50)$$

and the tied parameters:

$$\nabla_{S_k} G(P_g;\tilde{\Sigma}_g) = \lambda_g^k (P_g^{-1} - \tilde{\Sigma}_g) \ . \quad (51)$$

To perform the line search when optimizing for either the tied or untied parameters, we need to find those values of t for which a precision matrix of the form $P + t\delta P$ is positive definite and a way to quickly evaluate $f(t) = G(P + t\delta P;\Sigma)$ and it's derivatives along the line. To do this, we let E be the orthonormal matrix whose columns are the eigenvectors of $P^{-1/2}\delta P P^{-1/2}$ and D be the diagonal matrix whose entries are the corresponding eigenvalues. So

$$P^{-1/2}\delta P P^{-1/2} = EDE^T, \text{ and} \quad (52)$$
$$P + t\delta P = P^{1/2} E(I + tD)E^T P^{1/2} \ . \quad (53)$$

Thus $P + t\delta P$ is positive definite exactly when $I + tD$ is, i.e. when $1 + tD_{ii}$ is positive for all $i = 1,...,d$. The following formula allows fast evaluation of $f(t)$ (and it's derivatives) with an order of d operations, whereas a direct calculation requires $O(d^3)$ operations (more for higher order derivatives):

$$\delta f(t) = G(P + t\delta P;\Sigma) - G(P;\Sigma) \quad (54)$$
$$= \beta + \sum_{i=1}^{d} \log(1 + tD_{ii}) \quad (55)$$
$$\beta = \mathrm{Tr}(\Sigma\delta P) \ . \quad (56)$$

The formula above applies when the tied parameters S_k are allowed to be arbitrary symmetric matrices. For the EMLLT models, the S_k are required to be rank one matrices $S_k = a_k a_k^T$ and we need to use the chain rule to express the gradients in terms of the free parameters a_k. When doing the line search when optimizing for the a_k, the precision matrix is a quadratic function of the line search parameter. The parameter values where the determinant is zero are called "quadratic eigenvalues" [29]. They may be readily computed using an eigenvalue decomposition of a certain matrix of size $2d$. To perform efficient line search, we express the determinant of the precision matrix in terms of the "quadratic eigenvalues". See [17] for further details.

2) Generic Subspace Constraint: First we give formulas for the Q function and its gradient for a Gaussian mixture model with a generic subspace constraint on the exponential model parameters. For simplicity, we drop the affine shift parameter b_0 in the formulas. This causes no loss of generality since b_0 (appearing in (14) and (15)) may be absorbed into B with the proviso that the corresponding weight be set to 1 for all Gaussians. Putting (40) in (31), the Q functions for the tied

parameters in B and the untied parameters in $\{\lambda_g\}$ is:

$$Q(\{\lambda_g\}, B) = \sum_g n(g) H(\theta_g, \tilde{f}_g) \quad (57)$$

$$H(\theta, \tilde{f}) = K(\theta) + \tilde{f}^T \theta \quad (58)$$

$$\theta_g = B\lambda_g \ . \quad (59)$$

Recalling equations (41) and (10) involved in the definition of $\tilde{f}_g$, we will write $\tilde{f} = (-E(xx^T)/2, E(x))$.

We shall write ∇_ψ for the gradient with respect to ψ and ∇_P for the matrix giving the gradient with respect P. Thus the gradient of a function $F(\theta)$ at a point $\theta = (\mathrm{vec}(P), \psi)$ in the direction of a vector $\delta\theta = (\mathrm{vec}(\delta P), \delta\psi)$ may be written as:

$$(\delta\theta)^T \nabla_\theta F(\theta) = (\delta\psi)^T \nabla_\psi F(\theta) + \mathrm{Tr}\left(\delta P \nabla_P F(\theta)\right) \quad (60)$$

The gradient of H with respect to $\theta = (\mathrm{vec}(P), \psi)$ is:

$$\nabla_\theta H(\theta, \tilde{f}) = (\nabla_\psi H(\theta, \tilde{f}), \mathrm{vec}(\nabla_P H(\theta, \tilde{f}))) \quad (61)$$

$$\nabla_\psi H(\theta, \tilde{f}) = -P^{-1}\psi + E(x) \quad (62)$$

$$\nabla_P H(\theta, \tilde{f}) = \frac{1}{2}\left(P^{-1} + (P^{-1}\psi)(P^{-1}\psi)^T - E(xx^T)\right) \quad (63)$$

To optimize the untied parameter for an individual Gaussian, we use the chain rule to evaluate the gradient of $H(B\lambda, \tilde{f})$ with respect to λ:

$$\nabla_\lambda H(B\lambda, \tilde{f}) = B^T \nabla_\theta H(B\lambda, \tilde{f}) \ . \quad (64)$$

To optimize the tied parameters, we need the gradient of (57) with respect to the matrix B:

$$\nabla_B Q(\{\lambda_g\}, B) = \sum_g n(g) \left(\nabla_\theta H(B\lambda_g, \tilde{f}_g)\right) \lambda_g^T \quad (65)$$

The speedup of the line search for the case of a generic subspace constraint is very similar to the one we used when only the precision matrices were constrained. The line search when optimizing for the untied parameters for an individual Gaussian, requires us to find the maximum of $H(\theta_g + t\delta\theta_g, \tilde{f}_g)$, where $\theta_g = (\mathrm{vec}(P_g), \psi_g)$ equals $B\lambda_g$ and $\delta\theta_g = (\mathrm{vec}(\delta P_g), \delta\psi_g)$ equals $B\delta\lambda_g$. The search is constrained to the positive values of t for which the precision matrix $P_g + t\delta P_g$ is positive definite. For the tied parameters, we need to maximize the sum $\sum_g n(g) H(\theta_g + t\delta\theta_g, \tilde{f}_g)$ where θ_g again equals $B\lambda_g$, but now $\delta\theta_g$ has the form $\delta B\lambda_g$ and we require the precision matrices to be positive definite for all g.

To speed up the search in both the tied and untied case, we again use the expression (53) for $P + t\delta P$ in terms of the eigendecomposition (52) of $P^{-1/2}\delta P P^{-1/2}$. The values of t for which $P + t\delta P$ is positive definite are the same as before. For fast evaluation of $f(t) = H(\theta + t\delta\theta, \tilde{f})$ (and it's derivatives), we define the vectors $v = E^T P^{-1/2}\psi$ and $\delta v = E^T P^{-1/2}\delta\psi$ and the scalar $\beta = \delta\theta^T \tilde{f}$. Now we have

$$\delta f(t) = H(\theta + t\delta\theta, \tilde{f}) - H(\theta, \tilde{f}) \quad (66)$$

$$= \beta + \sum_{i=1}^{d} \log(1 + tD_{ii}) - \frac{(v + t\delta v)_i^2}{1 + tD_{ii}} + v_i^2 \quad (67)$$

Again the evaluation of the function $f(t)$ (and it's gradient) along the line requires calculating the determinant and inverse of P, which requires d^3 operations.

C. Modified Frobenius Seed for Precision Constraint

In this section we describe an efficient algorithm for finding a good "seed" bases $\{S_k\}$ for a precision constrained Gaussian mixture model (PCGMM). A naive guess for a seed, although one not obviously connected to a maximum likelihood criteria, would be to choose the $\{S_k\}$ so that the affine subspace they generate has the least total (weighted) distance to the full covariance statistics $\tilde{\Sigma}_g^{-1}$. The natural notion of distance between two matrices is the norm of their difference, where the norm of a matrix U is the Frobenius norm $||U|| = \mathrm{Tr}(U^T U)^{1/2}$. In this section, we will derive a variant of this naive guess using a certain approximation to the Q function for ML training defined in equations (47)-(48). We call this approximation the "modified Frobenius" approximation since it involves a modification of the Frobenius approximation. In the end, we describe the seed basis that maximizes this modified Frobenius approximation using principal component analysis.

To begin, we perform the Taylor expansion to quadratic order of $G(P; \Sigma) = \log\det(P) - \mathrm{Tr}(P\Sigma)$ about $P = \Sigma^{-1}$, the global maximum when P is unconstrained. The result is

$$G(P; \Sigma) \approx G(\Sigma^{-1}; \Sigma) - 0.5||P - \Sigma^{-1}||_\Sigma^2 \ . \quad (68)$$

Here the norm in the quadratic term is the matrix norm on $d \times d$ symmetric matrices coming from the inner product

$$< U, V >_\Sigma = \mathrm{Tr}(\Sigma\, U\, \Sigma\, V) \ . \quad (69)$$

Note that the norm associated with the identity matrix is the Frobenius norm. Also note that

$$||P - \Sigma^{-1}||_\Sigma^2 = \mathrm{Tr}\left((P\Sigma - \mathbb{1})(P\Sigma - \mathbb{1})\right) \ . \quad (70)$$

Dropping the constant terms, the quadratic approximation for Q in (47) is:

$$Q^{quad}(\{S_k\}, \{\lambda_g\}) = -0.5 \sum_g n(g) ||P_g - \tilde{\Sigma}_g^{-1}||_{\tilde{\Sigma}_g}^2 \quad (71)$$

The $\{S_k, \lambda_j^k\}$ that maximizes (71) may be found by a generalization of an algorithm for calculating singular value decompositions. In fact, we will reduce our computation of the seed to an actual singular value decomposition by making one further approximation. Namely, we replace the Gaussian dependent norms in (71) by the Gaussian independent norm associated to the mean covariance matrix $\bar{\Sigma} = \sum_g n(g)\tilde{\Sigma}_g / \sum_g n(g)$. The resulting function is the "modified Frobenius" approximation to the original Q function:

$$Q^{mf}(\{S_k\}, \{\lambda_g^k\}) = -0.5 \sum_g n(g) ||P_g - \tilde{\Sigma}_g^{-1}||_{\bar{\Sigma}}^2 \quad (72)$$

As with the original Q function, this function has the invariance property that it only depends on the P_g, so that a change of the basis $\{S_k\}$ that leaves fixed the affine subspace

$$\mathcal{A} = \{S_0 + \sum_{k=1}^{D} \lambda^k S_k; \lambda \in \mathbf{R}^d\} \quad (73)$$

can be compensated by a change of the weights λ_g^k.

To find the seed basis, we first optimize for the $\{\lambda_g^k\}$. Then P_g becomes the projection of $\tilde{\Sigma}_g^{-1}$ onto $\mathcal{A}$, i.e. the

point in $\mathcal{A}$ which is closest to $\tilde{\Sigma}_g^{-1}$. Note that the P_g at this stage are not guaranteed to be positive definite. This is not a problem though, because we are only interested in obtaining a reasonable starting point for the subspace $\mathcal{A}$; at a later stage positive definite precision matrices lying in $\mathcal{A}$ will be trained to maximize training likelihood.

To make the algorithm to compute the basis explicit, we define the following $d(d+1)/2$-dimensional vectors, which are representations of S_k and $\tilde{\Sigma}_g^{-1}$ under a mapping which takes the modified inner product (69) to the ordinary vector inner product:

$$s_k = \mathrm{vec}(\bar{\Sigma}^{1/2} S_k \bar{\Sigma}^{1/2}) \quad (74)$$

$$\rho_g = \mathrm{vec}(\bar{\Sigma}^{1/2} \tilde{\Sigma}_g^{-1} \bar{\Sigma}^{1/2}) \ . \quad (75)$$

For convenience, we constrain the set of s_k for $k > 0$ to be orthogonal to one another and to have unit length. Solving for the projection weights, and plugging into Q^{mf} (and multiplying by -2), we obtain the following function to minimize:

$$F(\{s_k\}) = \sum_g n(g) ||\rho_g - s_0 - \sum_{k=1}^{D} \lambda_g^k s_k||^2 \quad (76)$$

$$= \sum_g n(g) ||\rho_g - s_0||^2 - \sum_{k=1}^{D} (\lambda_g^k)^2 \quad (77)$$

$$\lambda_g^k = < s_k, \rho_g - s_0 > \ . \quad (78)$$

With a little more thought (or review of principal component analysis), the reader may convince himself or herself that the maximum is obtained when s_0 is taken to be the weighted sum of the $\{\rho_g\}$, i.e.

$$S_0 = \sum_g n(g) \Sigma_g^{-1} / \sum_g n(g) \ , \quad (79)$$

and s_k is taken to be the eigenvector of the covariance matrix of the $\{\rho_g\}$ which has the k'th largest eigenvalue (i.e. the k'th left singular vector of the matrix whose columns are the ρ_g).

As the affine subspace dimension D becomes larger and larger, the quadratic approximation (71) becomes more and more valid because the terms $P_g - \tilde{\Sigma}_g^{-1}$ that were projected away, and which we performed a Taylor expansion in, become smaller and smaller. On the other hand, in the limiting case when D becomes 0, i.e. when all precision matrices are constrained to equal S_0, the maximum likelihood value of S_0 is $\bar{\Sigma}^{-1}$. So, for small D, $\bar{\Sigma}^{-1}$ may be a preferable choice of S_0 than the one in (79).

Either of the candidates above for the affine basepoint S_0 are automatically positive definite and so provide an allowed starting point in a search for precision matrices in the subspace $\mathcal{A}$ that maximize likelihood. In order to have a search starting point even in the case of an ordinary subspace constraint on the precision matrices (no affine shift), we still take one of the basis vectors to be of the form of S_0 (and the rest to be PCA vectors).

V. Main Experiments Results

In this section we report on results of maximum likelihood training for various constrained exponential models. The experiments were all performed with the training and test set and Viterbi decoder reported on in [30] and [8], [9]. Some of the results here appear in [16]–[19]. The test set consists of 73743 words from utterances in four small vocabulary grammar based tasks (addresses, digits, command, and control). There were 22 test speakers. Data for each task was collected in a car at 3 speeds: idling, 30 mph, and 60 mph. The front-end computes standard 13-dimensional mel-frequency cepstral coefficients (MFCC) computed from 16-bit PCM sampled at 11.025KHz using a 25 ms frame window with a 15 ms shift between frames. The front-end also performs adaptive mean and energy normalization. Acoustic vectors with 117 components were obtained at each time frame by concatenating 9 consecutive frames of 13 dimensional MFCC vectors. The acoustic model had 89 phonemes and uses a decision tree to break these down further into a total of 680 context-dependent (HMM) states. All experiments use the same grammar based language models, HMM state transition probabilities, and Viterbi decoder which is passed state dependent probabilities for each frame vector which are obtained by table lookup [31] based on the ranking of probabilities obtained with a constrained Gaussian mixture model. All model training in this section was done using a fixed Viterbi alignment of 300 hours of multi-style training data (recorded in a car at various speeds and with various microphone placements). All models, except those reported on in section V-B, have a total of 10253 Gaussians distributed across the 680 states using the Bayesian Information Criterion (BIC) based on a diagonal covariance system. (See section V-B for a brief review of BIC.)

The final stage of training for all models described in this section was training the untied parameters by iterating EM until the utility function decrease was negligible. At each EM iteration, Gaussians with very low priors or nearly singular precision matrices were pruned. We do not include an affine shift in the subspace constraints unless otherwise noted.

A. Get most of full covariance gains with precision constraint

As a first step, we created LDA projection matrices based on the within class and between class full covariance statistics of the samples for each state. For 8 different values of the dimension d_1 ranging from 13 to 117, we constructed matrices LDA(d_1) which project from 117 to d_1 dimensions and we built full covariance models, FC(d_1), based on the projected vectors. In order to verify that the features obtained by the LDA projections were good, we also used the Gaussian level statistics of the models FC(117) and FC(52) to construct LDA and HLDA projection matrices (as well as a successful variant of HLDA presented in [32]). The models FC(d_1) gave WERs with less than, usually much less than, a 3.5% degradation relative to the best performing of all of the full covariance systems with the same projected dimension.

Next, we built the systems we will refer to as MLLT(d_1), which are MLLT systems for vectors produced by multiplication with LDA(d_1). (As a check on these MLLT systems, we observed that they did as well or better than the MLLT system based on features built using the diagonal version of HLDA.) We also built the systems PCGMM($d = d_1, D = d_1$), which are GMMs in dimension d_1 with unconstrained means

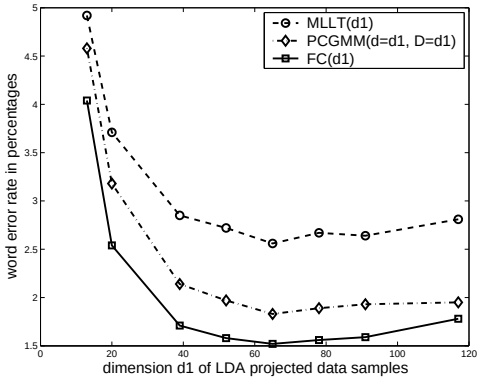

Fig. 1. WER as function of dimension showing the precision constrained models achieves signifi cant fraction of improvement from MLLT to full covariance (while maintaining the same per Gaussian cost as the MLLT systems).

and precision matrices constrained to a $D = d_1$ dimensional subspace. The subspace basis $\{S_k\}$ was obtained using the modified Frobenius approximation of section IV-C, where the statistics $\tilde{\Sigma}_g$ were taken to be those of the full covariance model FC(d_1).

Figure 1 shows that the PCGMM models achieve a significant fraction (e.g. 64% in 52 dimensions) of the total improvement possible in going from MLLT to full covariance, while maintaining the same per Gaussian computational cost as the MLLT system. The figure also shows that the optimal performance is obtained with 65 LDA features, independent of the type of precision constraint. Including more LDA features degrades performance, which we will discuss further in section V-E. Apart from section V-E, we restrict all experiments in the remainder of the section to use the 52 dimensional LDA projected features.

B. Result Obtained by Varying Model Size

One of the many possible parameters to vary in selecting an acoustic model is the number of parameters per Gaussian. We have already seen that significant improvement can be obtained with 10K Gaussians by going from MLLT models to full covariance or precision constrained models, all with the same number of Gaussians. The reader may wonder how much of that same improvement could have been obtained simply by increasing the number of Gaussians. Table II is an attempt to answer that question. It gives word error rates for MLLT and full covariance models of various sizes, all based on the features from the 52 dimensional LDA projection matrix LDA(52). The table shows that we could obtain much (but not all) of the full covariance gains using very large diagonal models, but that it requires very many parameters to do so, whereas the precision constrained approach can get similar gains with far fewer parameters.

To build the models in the table, we needed to have a way of distributing the total number of Gaussians in a model among the different context dependent states. The simplest approach is to assign the same number of Gaussians to each state. However, a somewhat more principled, but still tractable, approach based on the Bayesian Information Criterion (BIC) [33] has been shown to yield more accurate speech recognition systems [34]. In our context, we can apply BIC to determine the most likely number of Gaussian components C_s for the untied model for state s, given a fixed tied model. By performing the steepest descent approximation for the Bayesian integral giving the likelihood of C_s when the size N_s of the training data X_s for state s is very large, one can reason that the most likely value of C_s is the maximum of

$$F(C_s) \quad = \quad \log p(X_s|\Omega_s) - \frac{P}{2}\#(\Omega_s)\log(N_s) \ . \quad (80)$$

Here Ω_s is the maximum likelihood model with C_s components, $\#(\Omega_s)$ is the number of parameters that model has, and P is called the penalty weight. Performing the Gaussian approximation to the Bayesian integral around the (maximum likelihood) critical point leads to the value of $P = 1$. Including a Gaussian prior on the model parameters Ω_s results in larger values for P; so varying P provides a natural way to obtain a one parameter family of choices for the state dependent number of Gaussians C_s.

To obtain the models in Table II, we trained models which are diagonal in the MLLT basis of the model MLLT(52) described above, and which have a size that optimizes the BIC criterion function (80) for various values of the BIC penalty P. To reduce computational overhead (which was already several computer-years of time), the number of Gaussian components for lines in the table whose BIC penalty is marked with an asterix was calculated by extrapolating the number of components determined for larger BIC penalty values. The full covariance models have the same number of components for each state as the MLLT model on the same line.

The error rates for the systems with 10K Gaussians in Table II are 2.68% and 1.56%, which are quite close to the error rates of 2.67% and 1.58% for the 52-dimensional MLLT and full covariance models in Figure 1. The latter model has 10253 Gaussians which are distributed among the states using BIC at previous stage of processing. This serves as a "sanity check" that the distribution of Gaussians from the previous BIC calculation, which we use for all experiments in this section apart from those in Table II, is reasonably good.

Notice that the error rate of the full covariance model with just 10K Gaussians is significantly better than that of the MLLT model with 600K Gaussians, even though the full covariance model has only about one eighth as many parameters. The full covariance model with 60K Gaussians offers further WER improvements still.

C. Exploring Precision Constraints

Table III reports word error rates for 52 dimensional systems with unconstrained means and various constraints on the precision subspace. In all of the models the subspace for the precision matrices were computed based on the EM utility function using the statistics of the full covariance model FC(52). The final models were obtained by training the untied parameters by iterating EM on the training data to convergence. The table reports result for varying subspace dimension D, with the last row being the full covariance limit.

nGauss	BIC penalty	MLLT WER	FC WER
5000	35.180	3.48%	1.83%
10000	15.110	2.68%	1.56%
19999	6.910	2.21%	1.45%
30001	4.457	2.15%	1.42%
42993	3.000	2.00%	1.35%
60144	2.250	1.95%	1.34%
79990	1.744	1.87%	1.35%
110818	1.305	1.83%	
142622	1.000	1.74%	1.54%
149709*	1.000*	1.76%	
210666	0.770	1.73%	
255626	0.670	1.72%	
350286	0.500	1.68%	
428770*	0.354*	1.68%	
609100*	0.250*	1.65%	

TABLE II

WER OF MLLT AND FULL COVARIANCE MODELS ON THE LDA(52) FEATURES. THE NUMBER OF GAUSSIANS FOR EACH STATES WAS TRAINED USING BIC FOR THE MLLT SYSTEMS USING VARIOUS BIC PENALTY VALUES (AND SOME EXTRAPOLATION IN THE CASE OF LINES WITH AN ASTERIX).

D	EMLLT			PCGMM	
	stacked	ML	affi ne	mod. frob	ML
0.25d	–	–	3.05	2.70	2.50
0.5d	–	–	2.89	2.25	2.23
d	2.67	2.67	2.42	1.97	1.96
2d	2.35	2.04	2.06	1.75	1.75
4d	2.01	1.81	–	1.65	–
8d	1.82	1.65	–	1.64	–
26.5d	–	1.58	–	1.58	1.58

TABLE III

WER COMPARISON OF MODELS FOR LDA(52) FEATURES WHICH HAVE PRECISION MATRICES CONSTRAINED TO A SUBSPACE OF DIMENSION D. MODELS ARE: EMLLT WITH STACKED BASIS AND ML TRAINED BASIS, AFFINE EMLLT WITH ML TRAINED BASIS, PCGMM WITH BASIS FROM MODIFIED FROBENIUS ALGORITHM AND ML TRAINED BASIS.

The models in the "stacked" column in Table III are EMLLT models where the matrix A whose columns are the rank one directions $\{a_k\}$ for the precision matrices $\{a_k\, a_k^T\}$ are obtained by "stacking" MLLT matrices for various phone classes (which forces D to be a multiple of d). Preliminary experiments on the method used to generate the phone classes (manual vs data-driven) showed no significant differences in performance.

Models for the second column of results in the table have ML trained EMLLT basis, seeded by the stacked bases and trained by updating the a_k one at a time. The next column reports results for affine EMLLT models with the affine shift S_0 fixed to be the inverse of the (weighted) mean of the covariance matrices of the full covariance model and the rank one vectors a_k trained by ML.

The last two columns report results for PCGMM models where the subspace containing the precision matrices for all Gaussians (i.e. the basis of full covariance matrices $\{S_k\}$) was trained using the modified Frobenius approximation (next to last column) and maximum likelihood (last column).

We make the following observations:

- The WER goes smoothly down to the full covariance result as the basis size increases.
- Models with an arbitrary precision subspace constraint perform much better than comparable size EMLLT models (which restrict the subspace basis to consist of rank one matrices), except of course when the basis size becomes big enough that one is almost obtaining the full covariance results.
- For the EMLLT models, the ML training of bases and the "stacked" bases are the same for ordinary MLLT ($D = d$) and would be the same for full covariance D if the stack model were defined in that case, but for intermediate values we see that ML training essentially reduces the number of parameters required to model the covariance by half. On the other hand, for the general bases (PCGMM) case, maximum likelihood training does not provide any significant gain over the seed model from the modified Frobenius algorithm, unless one goes down to a basis size as small as $D = 0.25d$. (Since the gain should only go down as D increases, we didn't bother computing the ML trained PCGMM models with $D > 2d$.)
- Including an affine basepoint for the EMLLT basis leads to a significant improvement for the EMLLT models with $D = d$, but already by the time the EMLLT basis is of size $D = 2d$, including the affine basepoint yields no gain. (Since the gain should only go down as D increases, we didn't compute the affine EMLLT models for $D > 2d$.)

The final stage of training for each of the models above consisted of training the untied parameters to convergence. The initial point for that training was a model trained solely based on the statistics of the full covariance model FC(52) which may be viewed as a "compressed" version of the full covariance model. It turns out that the compressed models were in fact rather close to the final model obtained after allowing the untied parameters to "see" all of the training data. Only in cases when the subspace dimension is very low (so that the compressed model is very far from FC(52)) did the final training of the untied parameters have an appreciable effect. In those cases we might hope to gain some improvement by allowing the untied parameters to see the training data (and not just the model FC(52)). To test this, we collected full covariance EM statistics for the ML trained PCGMM model with $D = 0.25d$ and retrained both the tied an untied parameters based on those statistics. This reduced the word error rate from 2.50% to 2.47%.

D. Dimensional Reduction with SCGMM and SPAM models

The experiments so far in this section have shown the benefit of restricting precision matrices to a tied subspace. We now turn to experiments which restrict the linear weights as well. Results of these experiments are presented graphically in Figure 2 and numerically in Table IV. The table and graph give word error rates for various types of models, as a function of the number of untied parameters per Gaussian. All systems in any one row have roughly the same total number of parameters

(and computational cost) since the tied parameters contribute only a small fraction to the total. All models are built on the 52 dimensional acoustic vectors produced by LDA(52). The next paragraph describes how each of the five types of models were trained. In all cases, after the tied subspace constraints were determined, the untied parameters were trained by iterating the EM algorithm to convergence.

The models, in order of increasing accuracy are as follows. The models labeled MLLT or MLLT(d_1) are the same as appearing in Figure 1, namely MLLT model built on the LDA projected vectors in dimension d_1 for $d_1 \leq 52$. (These models may be considered models for vectors produced by LDA(52) since the projection matrix LDA(d_1) from 117 to d_1 dimensions is the composition of the projection from 117 to 52 dimensions with the projection from 52 to d_1 dimensions.) The models labeled PCGMM or PCGMM$(d = d_1, D = d_1)$ also appear in Figure 1. They are models on the LDA(d_1) vectors with a d_1-dimensional precision basis obtained by modified Frobenius training. The models SPAM$_0(d = 52, D = d_1, L = d_1)$ have the seed precision subspace that optimizes the modified Frobenius approximation and the seed linear weight subspace which optimizes the Q function when holding the precision subspace fixed. The SPAM$_{ML}$ models are obtained from the SPAM$_0$ models by further ML training of the basis (based on statistics of $FC(52)$). Finally, the SCGMM$(d = 52, F = 2 * d_1)$ models are Gaussian mixture models with the exponential weights constrained to a subspace of dimension $2 * d_1$ obtained by maximum likelihood training, seeded by the SPAM$_{ML}$ models.

We observe the following:

- The improvement of the SPAM$_{ML}$ models with ML trained basis over the seed models SPAM$_0$ becomes significant for small basis sizes.
- The SPAM$_{ML}$ models which constrain the precision and linear weights of a 52 dimensional model to dimension d_1 have significantly lower WER than the equally costly PCGMM models on the LDA projected features vectors. The latter models are, as we have already seen, better than the corresponding MLLT models.
- The general subspace constrained (SCGMM) models, yield still further improvements over the SPAM$_{ML}$ models with comparable cost.
- As one example of the combined gains, note that the model SCGMM$(52, 26)$ obtains a comparable error rate (about 2.7%) as the model MLLT(52) which has four times as many parameters per Gaussian and 3.5 times as many total parameters.

The astute reader may question whether or not the improvements obtained in allowing an arbitrary subspace constraint could have been obtained by still sticking with SPAM models in which the dimensions D of the precision and L of the linear weight subspaces are allowed to vary, as long as the total dimension $D + L$ is held fixed. Table V addresses this. It compares the general subspace constrained GMM with a 26 dimensional feature space to ML trained SPAM models, also with a 26 dimensional feature space. It is interesting to note that in this case we obtained improvement in the SPAM model by allowing more untied parameters to be invested in the precision subspace than in the linear weights subspace. However, allowing a general subspace constraint yields even better results and obviates the need to optimize over D for fixed $D + L$.

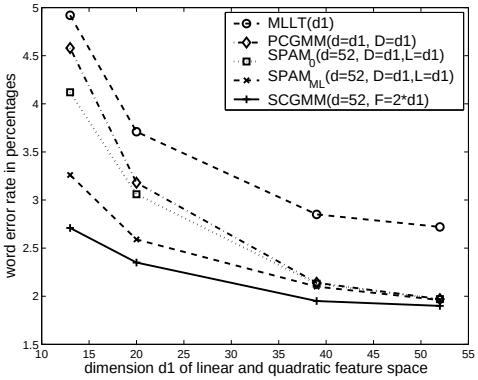

Fig. 2. Comparison of word error rates for various types of subspace constrained Gaussian mixture models as a function of half the subspace dimension (which is the dimension of both the linear and quadratic subspaces when they are separately defined). Plot shows that the more general the model and the more completely the model is trained to maximize likelihood, the better the WER becomes. This plot is graphical version of Table IV (not including first line of table).

d_1	MLLT	PCGMM	SPAM$_0$	SPAM$_{ML}$	SCGMM
8	-	-	-	4.69	3.42
13	4.92	4.58	4.12	3.26	2.71
20	3.71	3.18	3.06	2.59	2.35
39	2.85	2.14	2.13	2.10	1.95
52	2.67	1.97	1.97	1.96	1.90

TABLE IV

WER TABLE FOR: d_1-DIMENSIONAL MLLT AND PCGMM MODELS AND 52-DIMENSIONAL SPAM MODELS (WITH SEED AND ML TRAINED BASES) AND GENERAL SUBSPACE CONSTRAINED (SCGMM) MODELS. ALL MODELS HAVE SEPARATE d_1-DIMENSIONAL SUBSPACES FOR LINEAR WEIGHT AND PRECISION MATRICES, EXCEPT FOR SCGMM MODELS WHICH HAS A COMBINED $2 * d_1$ DIMENSIONAL SUBSPACE.

E. Concerning degradation in 117 *dimensions*

As we've seen already from Figure 1, the MLLT, full covariance, and precision constraint models all degrade when they model LDA projected vectors with more than about 65 features. This is despite the fact that they all use only $10K$ Gaussians and we know from the BIC experiments in section V-B that full covariance models in dimension 52 don't start to degrade significantly due to overtraining until they have at

Model Type	WER
SPAM(d=52,D=8,L=18)	3.52
SPAM(d=52,D=13,L=13)	3.26
SPAM(d=52,D=18,L=8)	3.00
SCGMM(d=52,F=26)	2.71

TABLE V

COMPARISON OF SCGMM AND SPAM MODELS WITH THE SAME TOTAL SUBSPACE DIMENSION.

least $80K$ Gaussians. So the degradation when increasing the LDA dimension up to 117 (which one might expect in the full covariance case to require about $2.25 = 117/52$ times as many data points per Gaussian as the 52 dimensional models) is probably not due to a lack of training data, but simply the fact that the extra features are not informative.

To explore this just a little bit further, we performed a couple of additional experiments. First of all, we "lifted" the model SCGMM$(d = 52, F = 26)$ to 117 dimensions (by including the LDA projection terms) and used that as a seed to obtain an ML trained model SCGMM$(117, 27)$ in 117 dimensions with a 27 dimensional subspace constraint. The error rate of 2.71% for SCGMM$(52, 26)$ degrades to 3.19% for SCGMM$(117, 26)$. On the other hand the analogous degredation in error rate in the MLLT case, from 2.67% for the model MLLT(52) of Figure 1 to 2.81% for MLLT(117) was not as bad. We conclude that maximum likelihood training of the tied parameters in going from SCGMM$(52, 26)$ to SCGMM$(117, 27)$ seems to be "learning" to look at components of the LDA projected vector which are providing noise rather than information.

On a qualitative level, there seem to be at least two contributing factors to the degradation that comes from looking at the noisy LDA directions. On the one hand, the part of the model in the noisy directions could itself be degrading system performance. On the other hand, the model in the non-noisy directions could be degraded because ML training is having it's "attention taken away" from the clean directions. To try to sort this out, we did some additional experiments. Table VI reports the results.

The first two lines of the table give the WER for the fully EM trained 52 and 117 dimensional full covariance models seen already in Figure 1. The model of the third line was the full covariance model in 117 dimensions obtained from the E-step statistics of FC(52). The model on the final line of Table VI was a 117 dimensional full covariance model with block diagonal covariance matrices where the part of the means in the first 52 dimensions were taken from FC(52), while the complementary parts were taken from the E-step statistics above.

Since the last model in the Table VI, whose Gaussians are a product of Gaussians from FC(52) with Gaussians in the noisy directions, didn't suffer from the problem of degradation of the non-noisy directions, we can conclude that the part of that model in the noisy directions is what is causing the degradation. The fact that the models in the last three lines of the table have about the same performance (in total, although their breakdowns by test condition differ more) suggests that the problem of degradation of the non-noisy part of the model is less severe.

F. Additional Complexity Reduction Techniques

We conclude with two experiments showing that the precomputation cost and number of Gaussians that need to be evaluated for a subspace constrained model can be reduced greatly without taking too much of a performance hit. We will reduce the precomputation cost by using a hybrid model as defined in (20). The number of Gaussians that we need to evaluate will be reduced by using a Gaussian clustering technique [35]. These experiments were first reported on in [18], which the reader can refer to for further details.

Model	WER
FC(52)	1.58%
FC(117)	1.83%
117 dimensional stats of $FC(52)$	1.81%
$FC(52)$ + complimentary stats	1.84%

TABLE VI
COMPARISON OF WER FOR SOME FULL COVARIANCE MODELS DESCRIBED IN SECTION V-E.

Table IV shows that the model $PCGMM(d = 39, D = 39)$ has error rate 2.14%. A comparable error rate of 2.13% is obtained from a hybrid model Hybrid$(d = 39, D = 39, K = 156)$ in which the basis matrices S_k are each a linear combination of $K = 156$ rank one matrices. The model was trained to maximize likelihood by the techniques of [17]. This comparable error rate was obtained by balancing the small degradation due to the constraint on the S_k with the small improvements due to the fact that an affine S_0 was included and the S_k were trained in a true maximum likelihood fashion. The model reduces the feature precomputation cost from $Dd(d+1)/2 \approx 30000$ to $Kd + DK \approx 12000$.

Next, we cluster the Gaussians of the hybrid model using a variant of K-means clustering that uses the Kullback-Liebler distance between Gaussians. We thus produce 1024 cluster centers, each of which is represented by a Gaussian (constrained by the hybrid model subspace) and a map from Gaussians of the hybrid model to cluster centers. When evaluating a model, we first evaluate all of the cluster center Gaussians and use the top rankings one to choose 1000 Gaussians of the hybrid model to evaluate, with the unevaluated Gaussians replaced by a simple threshold value. This causes the error rate to increase to 2.19%, but reduces the number of Gaussian evaluations required from 10253 to 2024. Although the memory cost has been increased a little bit, the evaluation time cost (gotten by multiplying by 78, the number of parameters per Gaussian) reduces from about $800K$ to about $158K$.

VI. LARGE VOCABULARY CONVERSATION SPEECH RESULTS

All of the experiments in section V were performed on a collection of small vocabulary, grammar based tasks. Although we tried to explore both low cost and high cost regions of the parameter space of models, one thing that we did not do was vary the number, 680, of context dependent states. We also did not consider speaker adaptation. Also, the test set we used was not an industry standard set. These issues are addressed in [20], where we obtained gains using precision constrained models in the unconstrained resource case on several tasks in the IBM superhuman speech corpus [22]. For completeness, we briefly summarize here some of the results obtained on the part of the superhuman corpus consisting of the switchboard

model	WER
MLLT$(d = 60)$	32.6%
PCGMM$(d = 60, D = 60)$	32.3%
PCGMM$(d = 60, D = 120)$	31.7%
PCGMM$(d = 60, D = 240)$	31.5%
PCGMM$(d = 60, D = 120)$ + FM	30.8%

TABLE VII

RESULT ON SWITCHBOARD 98 CORPUS FOR: MLLT BASELINE, PRECISION CONSTRAINED MODELS WITH SUBSPACE DIMENSION OF $D = d, 2d, 4d$, AND FMLLR+MLLR ADAPTATION OF THE $D = 2d$ MODEL. ALL MODELS HAD $217K$ GAUSSIANS. ALLOWING ANY NUMBER OF GAUSSIANS FOR AN MLLT MODEL PRODUCED AT BEST A WER OF 32.4%.

portion of the 1998 Hub 5e evaluation set, which consists of 113 minutes of audio.

We started by training baseline MLLT models of various sizes using BIC, similarly to how it was done in V-B. Training data consisted of 247 hours of Switchboard data and 17 hours of CallHome English, all released by the LDC. All models had 4440 context dependent states and use 60 dimensional data vectors obtained by LDA projection of 216 dimensional vectors, which were themselves obtained by concatenating 9 consecutive vectors of mean normalized, 24 dimensional MFCC features. Speaker adaptive VTLN warping factors [36], and FMLLR transformations [37] were trained based on an initial decoding. A trigram language model with $34K$ words was used. We took as our baseline model, the one with the BIC penalty of 1 and $217K$ Gaussians, which had an error rate of 32.6%. (The lowest WER of 32.4% was achieved with $357K$ Gaussians, while going up to $617K$ Gaussian degraded performance to 32.7%.)

We calculated modified Frobenius basis direction S_k based on the statistics of a $61K$ Gaussian full covariance model. An MLLT model with this number of Gaussians had an error rate of 35.3%, which improved to 33.4% with full covariance modeling. The precision constrained model with equal per Gaussian cost as the MLLT model (i.e. $D = 60$), had an error rate of 34.4%, which splits the difference.

Next, we trained a precision constrained model with the same $D = 60$ basis as above, but with $217K$ Gaussians as in the baseline model. This gave a very small improvement over the baseline model. However, we expect that a little more gain would have been obtained if we had trained the basis by maximum likelihood and based on the full $217K$ statistics (which would have been rather painful). We were able to gain better improvements over the MLLT baseline by increasing the precision subspace size to 120 and 240. We obtained further performance gains by MLLR and FMLLR speaker adaptation which were comparable to the performance gains when those techniques were used with MLLT models based on different features (as can be seen from Table 2 of [22]). with $D = 120$. The result of all these experiments appear in Table VII.

VII. DISCUSSION

The general picture that emerges from our experiments on both small vocabulary, grammar based tasks and large vocabulary conversational speech tasks is that the best performance is obtained by restricting to features that have information rather than noise and employing full covariance Gaussian mixture models rather than diagonal models, even diagonal models with a huge number of Gaussians. Although, the full covariance models are very expensive to evaluate and train, they can be usefully approximated by mixtures of Gaussians which place a subspace constraint on the precision matrices and linear weights of all the Gaussians. Models with larger subspace dimension have the lowest error rates. We have explored various types of specialization of the general subspace constraint to compare their performance/cost operating curves. In particular the very successful combination of LDA projection with MLLT (and in fact EMLLT) models is one specific specialization. We have found that the more general the subspace constraint and the more completely the model is trained to maximize likelihood, the better the operating curve. We have seen that these models can be combined successfully with the techniques of speaker adaptation (to reduce the error rate) and clustering (to reduce the acoustic model evaluation time).

We have tried to present a fairly thorough and self-contained discussion. However many other techniques and combinations of techniques are left to be explored. For example, we have not considered here models in which the Gaussians are clustered and there is a separate subspace restriction for each cluster, rather than a globally tied subspace.

Perhaps because we have tried to present a picture that unifies many different previous approaches, the reader may be left feeling that we have added more parameters to the already bewildering task of finding the best model given some resource constraints. One possibility is to choose the model (i.e. number of Gaussians per state, type of subspace constraint, semi-tied clustering, clustering to reduce number of Gaussian evaluations, the choice of subspace types and dimension, ...) which maximizes training likelihood. With a few additional tricks and shortcuts, this just might be practicable with the computational power available today. On the other hand, we could also try to choose a criterion other than maximum likelihood. In fact, in [23] we showed that futher gains are possible with these subspace constrained models using discriminative training techniques. We intend to explore this futher in the future [24].

APPENDIX I
DEFINITION AND SIMPLE PROPERTIES OF EXPONENTIAL MODELS

We first give some background about distributions in the exponential family. For further detail, see, for example, [38].

An exponential model $x \in \mathbf{R}^d$ associated with feature function $f : \mathbf{R}^d \mapsto \mathbf{R}^F$, $f(x) = (f_1(x), ..., f_F(x))$, and weight vector $\theta = (\theta_1, ..., \theta_F) \in \mathbf{R}^F$ is the probability density function

$$p_{(f,\theta)}(x) = e^{\theta^T f(x)}/Z(f,\theta) = e^{\theta^T f(x) + K(f,\theta)} \quad , \quad (81)$$

where

$$Z(f,\theta) = \int_x e^{\theta^T f(x)} \quad (82)$$

$$K(f,\theta) = -\log(Z(f,\theta)) \quad (83)$$

is a normalization constant and its negative logarithm. When there is no danger of ambiguity, we write $Z(\theta)$ and $K(\theta)$ as shorthand for $Z(f,\theta)$ and $K(f,\theta)$. The weight vector θ is required to belong to the "natural parameter space" $\Theta(f)$, consisting of the weight vectors for which the normalization constant is finite:

$$\Theta(f) = \{\theta; Z(f,\theta) < \infty\} \ . \tag{84}$$

Given a set of training sample $x_1, ..., x_T \in \mathbf{R}^d$, let p_{train} be the distribution on $\mathbf{R}^d$ consisting of equally weighted point masses at the x_i. Then the mean log-likelihood of the x_i under the distribution $p_{(f,\theta)}$ is:

$$L(\theta; \{x_i\}) = \frac{1}{T}\sum_i \log p_{(f,\theta)}(x_i) \tag{85}$$

$$= \theta^T E_{p_{train}}(f) + K(\theta) \tag{86}$$

Exponential distributions have several well known properties, including the following:

1) The gradient of $\log Z(\theta)$ equals the mean of f under the distribution $p_{(f,\theta)}$
2) The Hessian of $\log Z(f,\theta)$ equals the Covariance matrix of f under the distribution $p_{(f,\theta)}$.
3) $\log Z(f,\theta)$ is convex in θ.
4) $\Theta(f)$ is a convex set.
5) The mean log-likelihood function $L(\theta; \{x_i\})$ is concave in θ. The maximum likelihood weight vector $\hat{\theta}$ for the training data $\{x_i\}$ is uniquely determined by the zero gradient condition:

$$0 = E_{p_{train}}(f(x)) - E_{p_{(f,\hat{\theta})}}(f(x)) \ , \tag{87}$$

   i.e. by the condition that the training expectation of the features equals the model expectation of the features.
6) The maximum likelihood distribution $p_{(f,\hat{\theta})}$ associated with training data $\{x_i\}$ is also the unique probability density function $p(x)$ on $\mathbf{R}^d$ which maximizes the entropy $-\int_x p(x)\log p(x)$ subject to the constraint that $E_p(f(x)) = E_{p_{train}}(f(x))$.

Note that the convexity of $\log Z$ and the concavity of L follow immediately from property 2 above and the fact that the covariance matrices are positive definite. (Ok, we're ignoring the possibility of degeneracies here.) The fact that the set $\Theta(f)$ is convex follows because the function $Z(f,\theta)$ is convex in θ. This, in turns follows either from the fact that $\log Z$ is convex or directly from the facts $\exp(\theta^T f(x))$ is convex in θ and integration is linear.

APPENDIX II
INITIALIZATION OF UNTIED PARAMETERS

This appendix provides an algorithm which determines a positive definite matrix within a given affine subspace of the space of $(d \times d)$ symmetric matrices (or shows that no such point exists). That is, given the tied parameters, a basepoint S_0 and a basis $\{S_k\}_{k=1}^D$, we show how to find untied parameters $\lambda \in \mathbf{R}^D$ so that the matrix $P = S_0 + \sum_k \lambda^k S_k$ is positive definite. In the body of the paper, we have restricted ourselves to cases where the solution is obvious because either one of the S_k is positive definite or a subset of the S_k consists of rank one matrices whose directions form a basis of $\mathbf{R}^d$.

We handle the general case in this appendix with a simple adaptation of the method of alternating Projections Onto Convex Sets (POCS). The POCS theorem [39]–[41] states that if there is a point in the intersection of two closed convex sets (both contained in a suitable metric space), then the point can be found from an arbitrary initial point, by alternatively projecting on one space and then the other. The projection of a matrix P onto a convex set, is the matrix Q in the convex set that minimizes the distance to the matrix P. The distance function we use here is the one induced from the Frobenius norm. Applying POCS naively to our problem, we obtain a point in the intersection of our affine space $\mathcal{A}$ (73) and the space of non-negative matrices. Unfortunately, we could well obtain the zero matrix. Also, the algorithm does not converge in a finite number of steps. We modify the naive algorithm so as to avoid these problems by replacing the non-negativity condition by a threshold on the smallest eigenvalue.

It is simplest to first describe the algorithm to find a positive definite point in $\mathcal{A}$ in the case when the affine shift S_0 vanishes. Let $\mathcal{P}_t$ be the (closed, convex) set $\mathcal{P}_t = \{P; P \geq t\mathbb{1}\}$ of symmetric matrices whose minimum eigenvalues is greater than or equal to t. Also let $\mathcal{R}_t$ be the (closed, convex) intersection of $\mathcal{P}_t$ and $\mathcal{A}$, and let $\mathcal{P}_t^\circ$ and $\mathcal{R}_t^\circ$ be their respective interiors. Our goal is to find a point in $\mathcal{R}_0^\circ$, but it is easy to see, using the fact that $\mathcal{R}_0$ is a cone, that finding a point in $\mathcal{R}_0^\circ$ is equivalent to finding a point in $\mathcal{R}_1 \subset \mathcal{R}_0^\circ$. To find such a point, we start with an initial symmetric matrix P_0 (e.g. the inverse covariance matrix $\hat{\Sigma}^{-1}$ which we are trying to approximate) and then alternatingly project on the set $\mathcal{A}$ and $\mathcal{P}_1$ to obtain a sequence P_t, $t = 1, ..., \infty$, where P_t is in $\mathcal{A}$ for t odd and is in $\mathcal{P}_1$ for $t > 0$ even. The projection of P_{2n} onto $P_{2n+1} \in \mathcal{A}$ is a simple linear projection. The projection onto P_{2n-1} onto $P_{2n} \in \mathcal{P}_1$ is the matrix

$$P_{2n} = \sum_{i=1}^d \max(1, \epsilon_i)\ e_i e_i^T \tag{88}$$

where $\{e_i\}$ and $\{\epsilon_i\}$ are the eigenvectors and corresponding eigenvalues of P_{2n-1}. The POCS theorem guarantees that, if $\mathcal{R}_1 = \mathcal{A} \cap \mathcal{P}_1$ is non-empty, then the sequence P_t converges to a point P_∞ in $\mathcal{R}_1$. Then there exists some n such that $\|P_{2n+1} - P_\infty\|^2 < 1$ and thus P_{2n+1} is strictly positive definite. Since P_{2n+1} is also an element of $\mathcal{A}$, the algorithm converges in a finite number of steps. On the other hand, if $\mathcal{R}_1$ is empty, then the sequence $d_n = \|P_{2n+1} - P_{2n}\|$ decreases from above to a positive lower bound d_∞. In practive, the algorithm should terminate with a failure if n becomes too large, d_n is no longer decreasing significantly, and P_{2n+1} still fails to be positive definite.

To handle the case when S_0 is non-zero, we define the closed, convex sets

$$\mathcal{A}' = \{P = \sum_{k=0}^D \lambda_k S_k, \lambda \in \mathbf{R}^{D+1}, \lambda_0 \geq 1\} \tag{89}$$

$$\mathcal{R}'_t = \mathcal{A}' \cap \mathcal{P}_t \ . \tag{90}$$

By multiplying by a scalar, points in $\mathcal{R}_1'$ determine positive definite points in $\mathcal{A}$, and the converse is true as well. Similarly to above, we can use POCS to determine a point in $\mathcal{R}_1'$ by alternating between projection onto $\mathcal{P}_1$ and $\mathcal{A}'$. The projection of P_{2n} onto $P_{2n+1} \in \mathcal{A}$ is given by:

$$P_{2n+1} = \begin{cases} \sum_{k=0}^{D} \lambda_k^{D+1} S_k & \text{if } \lambda_0^{D+1} \geq 1 \\ S_0 + \sum_{k=0}^{D} \lambda_k^{D} S_k & \text{otherwise.} \end{cases} \quad (91)$$

Here $\lambda^D \in \mathbf{R}^D$ and $\lambda^{D+1} \in \mathbf{R}^{D+1}$ have components:

$$\lambda_k^D = \sum_{l=1}^{D} (C^D)_{kl}^{-1} \mathrm{Tr}\left((P_{2n} - S_0) S_k\right) \quad (92)$$

$$\lambda_k^{D+1} = \sum_{l=0}^{D} (C^{D+1})_{kl}^{-1} \mathrm{Tr}\left(P_{2n} S_k\right) \ , \quad (93)$$

where C^D is the matrix of inner products $\mathrm{Tr}(S_k S_l)$, $k, l = 1, ...D$ and C^{D+1} is the matrix of inner products for $k, l = 0, ...D$.

ACKNOWLEDGMENT

We are grateful to the members of the IBM "Superhuman team" for discussion, models, lattices, code, and all that, in particular to Brian Kingsbury, George Saon, Stan Chen, and Lidia Mangu. We also would like to thank Charles Micchelli for useful mathematical discussions.

REFERENCES

[1] D. B. Rubin and D. Thayer, "EM algorithm for ML factor analysis," *Psychometrika*, vol. 57, pp. 69–76, 1976.
[2] C. Bishop and M. Tipping, "Probabilistic principal component analysis," *J. of the Royal Statistical Society, Series B*, vol. 21, no. 3, pp. 611–622, 1999.
[3] L. K. Saul and M. G. Rahim, "Maximum likelihood and minimum classification error factor analysis for automatic speech recognition," *IEEE Transactions on Speech and Audio Processing*, vol. 8, no. 2, pp. 115–125, 1999.
[4] R. A. Gopinath, B. Ramabhadran, and S. Dharanipragada, "Factor analysis invariant to linear transformations of data," in *Proc. ICSLP*, 1998.
[5] S. Dharanipragada and K. Visweswariah, "Covariance and precision modeling in shared multiple subspaces," in *Proc. ICASSP*, 2003.
[6] R. Gopinath, "Maximum likelihood modeling with gaussian distributions for classification," in *Proc. ICASSP*, 1998.
[7] M. J. F. Gales, "Semi-tied covariance matrices for hidden Markov models," *IEEE Transactions on Speech and Audio Processing*, 1999.
[8] P. Olsen and R. A. Gopinath, "Modeling inverse covariance matrices by basis expansion," in *Proc. ICASSP*, 2002.
[9] ——, "Modeling inverse covariance matrices by basis expansion," *IEEE Transactions on Speech and Audio Processing*, to appear.
[10] R. Fisher, "The use of multiple measurements in taxonomic problems," *Ann. Eugen.*, 1936.
[11] ——, "The statisttical utilization of multiple measurements," *Ann. Eugen.*, 1938.
[12] C. Rao, *Linear Statistical Inference and Its Applications*. John Wiley and Sons, 1965.
[13] N. Campbell, "Canonical variate analysis - a general formulation," *Australian Journal of Statistics*, 1984.
[14] N. Kumar, "Investigations of silicon auditory models and generalization of linear discriminant anlysis for improved speech recognition," Ph.D. dissertation, Johns Hopkins University, March 1997.
[15] N. Kumar and A. G. Andreou, "Heteroscedastic discriminant analysis and reduced-rank HMMs for improved speech recognition," *Speech Comm.*, vol. 26, pp. 283–297, 1998.
[16] S. Axelrod, R. A. Gopinath, and P. Olsen, "Modeling with a subspace constraint on inverse covariance matrices," in *Proc. ICSLP*, 2002.
[17] K. Visweswariah, P. Olsen, R. A. Gopinath, and S. Axelrod, "Maximum likelihood training of subspaces for inverse covariance modeling," in *Proc. ICASSP*, 2003.
[18] S. Axelrod, R. A. Gopinath, P. Olsen, and K. Visweswariah, "Dimensional reduction, covariance modeling, and computational complexity in ASR systems," in *Proc. ICASSP*, 2003.
[19] K. Visweswariah, S. Axelrod, and R. Gopinath, "Acoustic modeling with mixtures of subspace constrained exponential models," in *Proc. Eurospeech*, 2003, to appear.
[20] S. Axelrod, V. Goel, B. Kingsbury, K. Visweswariah, and R. Gopinath, "Large vocabulary conversational speech recognition with a subspace constraint on inverse covariance matrices," in *Proc. Eurospeech*, 2003, to appear.
[21] V. Vanhoucke and A. Sankar, "Mixtures of inverse covariances," in *Proc. ICASSP*, 2003.
[22] B. Kingsbury, L. Mangu, G. Saon, G. Zweig, S. Axelrod, V. Goel, K. Visweswariah, and M. Picheny, "Toward domain-independent conversational speech recognition," in *Proc. Eurospeech*, 2003, to appear.
[23] V. Goel, S. Axelrod, R. Gopinath, P. Olsen, and K. Visweswariah, "Discriminative estimation of subspace precision and mean (SPAM) models," in *Proc. Eurospeech*, 2003, to appear.
[24] S. Axelrod, V. Goel, R. A. Gopinath, P. Olsen, and K. Visweswariah, "Discriminatively trained acoustic models comprised of mixtures of exponentials with a tied subspace constraint," in preparation.
[25] N. K. Goel and A. G. Andreou, "Heteroscedastic discriminant analysis and reduced-rank HMMs for improved speech recognition," *Speech Comm.*, vol. 26, pp. 283–297, 1998.
[26] L. Vandenberghe, S. Boyd, and S.-P. Wu, "Determinant maximization with linear matrix inequality constraints," *SIAM J. Matrix Anal. Applic.*, vol. 19, no. 2, pp. 499–533, April 1998.
[27] E. Polak, *Computational Methods in Optimization: A Unified Approach*. Academic Press, 1971.
[28] J. N. D.C. Liu, "On the limited memory BFGS method for large scale optimization problems," *Mathematical Programming*, vol. 45, pp. 503–528, 1989.
[29] F. Tisseur, K. Meerbergen, "The quadratic eigenvalue problem," *Society for industrial and applied mathematics*, vol. 43, no. 2, pp. 235–286, 2001.
[30] S. Deligne, E. Eide, R. Gopinath, D. Kanevsky, B. Maison, P. Olsen, H. Printz, and J. Sedivy, "Low–resource speech recognition of 500 word vocabularies," in *Proceedings of the Sixth European Conference on Speech Communication and Technology*, Aarhus, Denmark, September 2001.
[31] L.R. Bahl et al., "Performance of the IBM large vocabulary continuous speech recognition system on the ARPA Wall Street Journal task," in *Proc. ICASSP*, 1995.
[32] G. Saon, M. Padmanabhan, R. Gopinath, and S. Chen, "Maximum likelihood discriminant feature spaces," in *Proc. ICASSP*, 2000.
[33] G. Schwarz, "Estimating the dimensions of a model," *Annals of Statistics*, 1978.
[34] S. Chen and R. A. Gopinath, "Model selection in acoustic modeling," in *Proc. Eurospeech*, 1999.
[35] E. Bocchieri, "Vector quantization for efficient computation of continuous density likelihoods," in *Proc. ICASSP*, 1993.
[36] H. Hermansky, "Perceptual linear predictive (PLP) analysis of speech," *Journal of the Acoustical Society of America*, vol. 87, no. 4, pp. 1738–1752, April 1990.
[37] M. J. F. Gales, "Maximum likelihood linear transformations for HMM-based speech recognition," Cambridge University, Technical Report TR 291, 1997.
[38] L. Brown, *Fundamentals of Statistical Exponential Familes*. vol 9 of Lecture Notes - Monograph Series, Institute of Math. Stat., 1991.
[39] L. M. Bregman, "Finding the common point of convex sets by the method of successive projections," *Dokl. Akad. Nauk. USSR*, vol. 162, pp. 487–490, 1965.
[40] ——, "The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming," *USSR Computational Math. & Math. Physics*, vol. 7, no. 3, pp. 200–217, 1967.
[41] L. G. Gubin, B. T. Polyak, and E. V. Raik, "The method of projections for finding the common point of convex sets," *USSR Computational Math. & Math. Physics*, vol. 7, no. 6, pp. 1–24, 1967.