Mixtures of Inverse Covariances

Vincent Vanhoucke* and Ananth Sankar

Abstract

We describe a model which approximates full covariances in a Gaussian mixture while reducing significantly both the number of parameters to estimate and the computations required to evaluate the Gaussian likelihoods. In this model, the inverse covariance of each Gaussian in the mixture is expressed as a linear combination of a small set of prototype matrices that are shared across components. In addition, we demonstrate the benefits of a subspace-factored extension of this model when representing independent or near-independent product densities. We present a maximum likelihood estimation algorithm for these models, as well as a practical method for implementing it. We show through experiments performed on a variety of speech recognition tasks that this model significantly outperforms a diagonal covariance model, while using far fewer Gaussian-specific parameters. Experiments also demonstrate that a better speed/accuracy trade-off can be achieved on a real-time speech recognition system.

Index Terms

Gaussian mixture models, covariance modeling, automatic speech recognition. EDICS Category: 1-RECO

I. Introduction

A major bottleneck to the scalability of Gaussian mixture models (GMM) in high dimensions is the quadratic nature of the Gaussian covariances. In a space with as few as 20 dimensions, the number of parameters devoted to representing Gaussian covariances is already one order of magnitude greater than the number of parameters devoted to the Gaussian means. This means that, even in a space with a reasonably small dimensionality, when data assigned to a given mixture component is scarce, the covariance matrix of this component is going to be affected first by the lack of data, even when that amount of data would be sufficient to estimate the component's mixture weight and mean vector reasonably well.

It is also widely known that the maximum likelihood estimator of a covariance matrix is not well conditioned: in the limit, while a mean vector is still well defined if a single input vector is assigned to the component, the covariance matrix is singular if there are fewer input vectors than the dimensionality of the space. A good introduction to these issues can be found in [1, Chapter 1]. This lack of robustness of the covariance matrix estimator limits the scalability of a GMM in two ways:

- the number of parameters per component being large in high dimension, the number of components in the mixture can not grow larger than a small fraction of the size of the training set,
- the dimensionality of the input space which can be modeled using a GMM can not grow to large values without compromising robustness.

In addition, the evaluation of the log-likelihood of a Gaussian component amounts to computing a quadratic form in the input parameters, which requires a number of floating point operations which is quadratic in the dimensionality. This computational expense can be prohibitive for statistical models that are used in real-time applications.

However, one of the very attractive features of GMM is that they allow trading of some of the resolution at the component level for an increased number of components in the mixture. It is very common to impose some structure to the covariance matrices, thus reducing the number of parameters required to describe them, while growing the number of components in the mixture to compensate for the loss in precision. The simplest way of achieving this is to constrain the covariances to be diagonal. For a given component, this implies an assumption of independence of the feature components. However, with a larger number of components devoted to the same portion of the feature space, the mixture can still model correlations to an arbitrary precision. Constraining the covariances to be diagonal

V.Vanhoucke is a Ph.D. student in the department of Electrical Engineering at Stanford University and a member of the Speech R&D group at Nuance Communications. E-mail: vanhoucke@stanfordalumni.org. Tel: (650) 281-5352. Send correspondence to: V. Vanhoucke, Nuance Communications, 1005 Hamilton Court, Menlo Park, CA 94025.

A. Sankar is Project Manager in the Speech R&D group at Nuance Communications. E-mail: sankar@nuance.com. Tel: (650) 847-7015.

implies that the data modeled by each component is assumed to be uncorrelated. However, the joint probability of several components in the mixture does not need to be decorrelated to be modeled accurately. The benefits of this assumption lie in the fact that a diagonal covariance has a number of parameters equal to the dimensionality. This removes to a large extent all the scalability issues of the GMM by keeping the number of parameters required to model the covariance equal to the number of parameters describing the mean vector. In addition, the cost of evaluating the log-likelihood of any Gaussian in the mixture is now linear in the dimension as well.

In the context of acoustic modeling for automatic speech recognition (ASR), the assumption that feature components are decorrelated holds to some extent. Typical speech input features are weakly correlated because the final stage of the front-end processing is some form of whitening of the feature vector. This can be achieved through a Discrete Cosine Transform that approximates the Karhunen-Loève transform [2] in the case of standard Mel filter-bank cepstral coefficients (MFCCs) [3], perceptual linear prediction (PLP) [4] or RASTA/PLP [5]. Another common approach is to use linear discriminant analysis [6] on a higher-dimensional feature space to extract orthogonal features. However, explicit modeling of the correlations generally leads to better models [7], both in terms of improving recognition accuracy and reducing the size of the mixtures required to model the acoustics. Several schemes imposing weaker constraints to the covariance matrices have been proposed to improve the simple diagonal model. In general, these models belong to one of three categories:

- models factoring the Gaussians into independent subspaces, thus constraining the covariances to have a block diagonal structure [8],
- models using a transform of the feature space to better match the decorrelation assumption [9], [10], [11],
- models performing an expansion of the covariance matrix into simpler approximations [12], [13], [14], [15].

In the following, we present an efficient way of modeling and estimating covariances in a GMM through a mixture of inverse covariances (MIC). The MIC model represents each inverse covariance matrix as a linear combination of a small set of prototype matrices. While the prototype matrices are shared across components, the linear combination weights are Gaussian-specific, allowing a controlled trade-off between precision of the model and the per-component complexity. We then exploit the idea of subspace factorization to improve the model for products of independent or near-independent sources. We derive maximum likelihood estimation algorithms for these models and describe a practical implementation. An implementation of this model on a real-time ASR system shows that this method improves accuracy significantly over a standard diagonal model, and provide a significantly better speed / accuracy trade-off.

Section II describes the MIC model and related approaches. Section III applies the MIC model to acoustic modeling in ASR. Section IV details how to compute a GMM using the MIC model and describes the various complexity trade-offs. Section V describe the maximum likelihood estimation algorithms. Finally, Section VI details experimental results on a variety of ASR tasks.

II. Mixtures of Inverse Covariances

A GMM for a D-dimensional input vector $\boldsymbol{o}$, composed of M Gaussians with priors w_i, means $\boldsymbol{\mu}_i$ and covariances Σ_i can be expressed as:

$$f(\boldsymbol{o}) = \sum_{i=1}^{M} w_i \mathcal{N}(\boldsymbol{o}, \boldsymbol{\mu}_i, \Sigma_i)$$

Where:

$$\mathcal{N}(\boldsymbol{o}, \boldsymbol{\mu}_i, \Sigma_i) = \sqrt{\frac{|\Sigma_i^{-1}|}{(2\pi)^D}} e^{-\frac{1}{2}(\boldsymbol{o}-\boldsymbol{\mu}_i)^\top \Sigma_i^{-1} (\boldsymbol{o}-\boldsymbol{\mu}_i)} \tag{1}$$

Although this formulation formally treats the input as a mixture of Gaussian sources, in most cases there is only one source generating the input signal, and the various mixture components are used to model the non-Gaussianity of its distribution. As a consequence, there are strong relationships between the parameters of the various components in the mixture.

This is generally the case in acoustic modeling for ASR. The input features are weakly correlated with each other across the entire feature space, and this decorrelation can thus be expected to reflect in the covariance structure of all components. In addition, some of the feature components such as cepstral derivatives are typically computed from other components, introducing some recognizable patterns in the structure of typical covariance matrices. This last point will be discussed in more details in Section III.

Assuming that there is indeed much redundancy in the parameters of the covariance of a typical GMM, it is natural to consider compressing this information into fewer parameters that can be estimated robustly, and which will result in a more compact representation of the probability density. By treating the covariance parameters as a highly redundant input signal, techniques of lossy compression such as vector quantization (VQ) [16] can be applied to the problem.

In the MIC model, the inverse covariances[1] in the mixture are represented using a small sized codebook of prototype symmetric matrices $\Psi_k, k \in [1, K]$. In contrast with a "hard" clustering technique such as VQ, the degree of association of a given inverse covariance i with a prototype k is represented by a scalar $\lambda_{k,i} \in \mathbb{R}$, so that:

$$\Sigma_i^{-1} = \sum_{k=1}^{K} \lambda_{k,i} \Psi_k \tag{2}$$

In that respect, this formulation is similar to mixture models such as generalized additive models [17, Chapter 9] or fuzzy clustering [18, Chapter 8], which make a soft decision when associating a data sample to a mixture component. Note that unlike the case of mixtures of densities, the mixture "weights" $\lambda_{k,i}$ are not constrained to sum to one or even to be positive. In [15], we constrained the Ψ_k to be positive definite, but here we will not make that assumption in general, and will highlight the benefits of having a positivity constraint when they specifically arise. There are several arguments in favor of a soft clustering scheme as opposed to a hard one in the current context. In particular, the overall scale of typical covariance matrices can vary dramatically, and this feature is well captured in the weights of the MIC. In addition, the soft clustering model retains a number of Gaussian-specific parameters — the K weights, K being anywhere between 1 and $D(D+1)/2$, which makes it much more expressive at a controlled level of complexity. In Section IV, we will see that the computational complexity of this model is directly proportional to K.

A. Related Approaches

By imposing some additional structure onto the prototypes, many different covariance models can be expressed in the form of MIC. Consider the symmetric canonical basis of matrices $E_{i,j}$, whose elements are 0 everywhere except at locations (i, j) and (j, i) where they are 1. The unconstrained full covariance model can be expressed by having:

$$\Psi_k, k \in [1, D(D+1)/2] \equiv E_{i,j}, 1 \leq j \leq i \leq D$$

and the diagonal covariance model by having:

$$\Psi_k \equiv E_{k,k}, k \in [1, D]$$

It is clear that by relaxing these strong constraints on the structure of the prototypes, a better model can be achieved with the same number (K) of Gaussian-specific parameters. Several well-known covariance models fall under this general class. Semi-tied covariances [10] express each inverse covariance matrix Σ_i^{-1} using a diagonal inverse covariance matrix D_i and a transform A shared across Gaussians:

$$\Sigma_i^{-1} = A D_i A^\top$$

The computational benefits of this model are obvious, since it differs from a plain diagonal model by a simple transform of the feature vector. As was remarked in [13], by considering the rows $\boldsymbol{\eta}_k$ of the transform matrix A, and $d_{k,i}$ the diagonal terms of matrix D, this can be rewritten as:

$$\Sigma_i^{-1} = \sum_{k=1}^{D} d_{k,i} \boldsymbol{\eta}_k \boldsymbol{\eta}_k^\top$$

[1]Inverse covariance matrices are sometimes referred to as "precision matrices" or "concentration matrices". The choice of modeling inverse covariances as opposed to covariances is driven by the log-likelihood of a Gaussian, which has a simple expression as a function of the inverse covariance (Equation 1).

Since any rank-one matrix can be expressed uniquely as the product of a vector and its transpose, the semi-tied covariance model is an instance of the MIC model in Equation 2 with $K = D$, and with sole constraint: $\text{Rank}(\Psi_k) \equiv 1$.

Factored sparse inverse covariance matrices [9] are a more constrained model in which the transform A is an upper triangular matrix with ones along the diagonal. The main benefit of this model is that the optimization of the transform matrix in the EM framework is now linear, owing to the fact that $|\Sigma_i^{-1}| = |D_i|$ is independent of the transform. In this case, the class of prototype matrices that correspond to this model is a collection of rank-one block-diagonal matrices generated by the family of vectors:

$$\boldsymbol{\eta}'_k = [0 \ldots 0 \underbrace{1}_{k} \eta'_{k,k+1} \cdots \eta'_{k,D}]^\top$$

In [13], the extended maximum likelihood linear transform (EMLLT) model was introduced, generalizing the semi-tied approach to $K > D$. Finally, a recent publication presented the subspace of precisions and means (SPAM) model [14] which independently generalized the mixture approach to matrices of any rank.

B. Class-based Prototype Allocation

As the number of matrices to be modeled grows, the number of prototypes required to model all the covariances accurately might grow to the point of making the joint estimation of all the prototypes as well as the Gaussian likelihood evaluation computationally expensive. (See Section IV for a detailed analysis of the computational cost of these models). For more efficient modeling, one might consider using a class-based decomposition of the Gaussian mixture and allocate a distinct pool of prototypes to each class:

$$\forall i \in \mathcal{C},\ \Sigma_i^{-1} = \sum_{k=1}^{K_{\mathcal{C}}} \lambda_{k,i} \Psi_{\mathcal{C},k}$$

This limits the number of Gaussian-specific parameters to $K_{\mathcal{C}}$, while allowing the pool of prototypes to grow much larger. The determination of appropriate classes can be dictated by the problem at hand, or derived in a principled way in the same vein as classified VQ approaches [16, Section 12.5].

C. Subspace Factorization

An extremely powerful extension of the basic model is to consider the case of probability densities which, at the component level, can be assumed to be the product of independent or near-independent distributions. In this situation, the covariance matrices of the mixture components will have a block-diagonal structure. Note that we do not require the complete distribution to be (near-)independent, since the different mixture components can still model correlated events. However, the limit case of a distribution which is globally the product of independent distributions is useful to illustrate the following point: if the probability density in one of the independent subspaces bears no relationship with the density in distinct subspaces, then there is no modeling benefit at clustering the distinct subspaces jointly.

Let us thus consider a block-diagonal model, with an independent set of prototypes and bases for each sub-block. Considering L covariance sub-blocks of dimensionality D_l:

$$\Sigma_{i,l}^{-1} = \sum_{k=1}^{K_l} \lambda_{k,i,l} \Psi_{k,l}$$

The advantages of this model are multiple. First, the global estimation problem is decomposed into multiple, lower-dimensional problems that will be less expensive to solve. In addition, the cost of evaluating the log-likelihood of a subspace-factored model is lower. The last advantage is of combinatorial nature: a block-diagonal system with L subspaces and K prototypes per subspace contains implicitly K^L "full" prototypes, while only requiring $K \times L$ weights per Gaussian. As a result, for a given number of Gaussian-specific parameters, the subspace-factored model can make use of a larger collection of prototypes than its single-block counterpart.

This means that if the independence of the distinct subspaces can be assumed, there is a modeling benefit in using a block diagonal model instead of a full covariance model. This subspace decomposition method is known in coding as partitioned VQ [16, Section 12.8] which is the simplest instance of a product code. In ASR, this has been used to revive the concept of VQ-based acoustic modeling using discrete mixture hidden Markov models (DMHMM) [19], which compare well with standard hidden Markov models (HMM) which use GMM as underlying density models, and allow for a more compact representation of cepstral parameters [20]. In GMM/HMM systems, the same idea has been exploited by performing subspace clustering of the Gaussians in a mixture [21].

III. Application to Acoustic Modeling

In acoustic modeling, GMM are typically used in conjunction with HMM [22]. Each state of the HMM corresponds to a sub-phonetic unit, and the conditional probability of the acoustics, given each state, is modeled using a GMM. All of the GMMs from all the states in the HMM can be considered as a large GMM with the same component parameters, except for the mixture weights which are now state-dependent: if a Gaussian in the mixture does not correspond to a given state, then its state-dependent weight is 0, otherwise it is unchanged. This analogy extends to the estimation algorithm itself: for the purposes of estimating the state GMM parameters, the Baum-Welch estimation algorithm can actually be treated as an EM algorithm [23] over the entire pooled GMM.

In the simplest approach, the MIC model can thus be used to tie the complete set of covariances in the acoustic model. All the Gaussians are pooled into a single GMM using the priors estimated from the HMM state occupancy probabilities, and the prototypes are estimated on the entire mixture. A slightly more involved approach uses separate MIC for distinct state classes. For this purpose, state classes can be constructed in a data-driven fashion, or using linguistic knowledge derived from the sub-phonetic units associated with each state. While this leads to a significant increase in the total number of parameters in the system, the additional complexity at run time can be alleviated by only computing the prototype-dependent features for the active states at any given time during the decoding.

In order to take advantage of the subspace-factored approach, it is necessary to determine which correlation components can be discarded without any loss. The following analyzes the case of models based on MFCC [3] feature vectors, and demonstrate some non-intuitive results as to which components of the MFCC-derived covariance matrices are relevant. Section VI-D will later show experimental results validating this approach.

The global structure of a covariance matrix resulting from a MFCC input vector is described in Figure 1.

$$\begin{array}{r|c|c|c|}
\cline{2-4}
\text{Cepstrum} & a & d & f \\ \cline{2-4}
\Delta & d & b & e \\ \cline{2-4}
\Delta^2 & f & e & c \\ \cline{2-4}
\end{array}$$

Fig. 1. Structure of covariance matrices describing MFCC inputs. Sorting the MFCC feature vector into 3 blocks containing respectively the cepstra, first and second order derivative, the covariance matrix can be decomposed into 9 blocks. For example, block (d) models the correlations between the cepstral features and their derivatives

Each component of the matrix models distinct types of correlations, some of which can be qualified as *structural*, and others *incidental.* Structural correlations result from the way feature components are computed from each other, leading to dependencies between them. Incidental correlations are a result of the relationships between components preexisting in the data being modeled, independently of the front-end processing.

A good example of *structural* vs. *incidental* correlation occurs when building MFCC derivatives out of the cepstral coefficients. Typically, for a given input observation $\boldsymbol{o}(t)$ at time t, the derivative would be computed by applying a finite impulse response (FIR) filter onto the observation sequence such as depicted in Figure 2. The common features

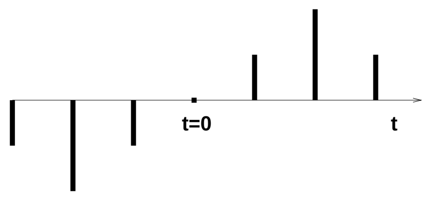

Fig. 2. Profile of a FIR filter used to compute the cepstral derivative from a sequence of observations. Note that the value of the input at $t = 0$ is not typically used in the computation, which implies that correlations between the cepstrum and its derivative will only result from time correlations in the signal itself.

of the filters used are that they estimate the value of the signal at $t < 0$ and subtract it from an estimate of the signal at $t > 0$ over a small window. Note that here the current input $\boldsymbol{o}(t)$ is not involved. As a consequence, any correlation arising between $\boldsymbol{\delta}(t)$ and $\boldsymbol{o}(t)$ would be *incidental*, i.e. would be providing information about the relationship between consecutive frames of data.

When computing the second order derivatives, a typical profile would be as depicted in Figure 3. In this case, the

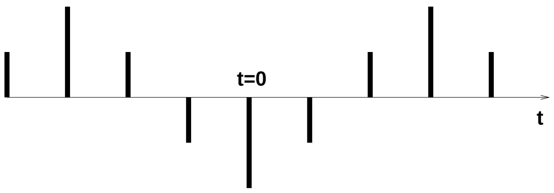

Fig. 3. Profile of a FIR filter used to compute the cepstral second derivative from a sequence of observations. Note that the value of the input at $t = 0$ is heavily weighted by this type of filter, which implies that there will be structural correlations between the cepstrum and its second derivative.

component $\boldsymbol{o}(t)$ is explicitly part of the expression of $\boldsymbol{\delta}^2(t)$, and thus there will be a structural correlation between the i^{th} MFCC component and its corresponding $\boldsymbol{\delta}^2(t)$ component. These considerations generally hold regardless of the actual implementation of the computation of the derivatives, however the exact distribution of structural correlations depends highly on the specifics of the feature extraction. Figure 4 illustrates which components of the inverse covariance matrix are structurally large in magnitude in the situation just described.

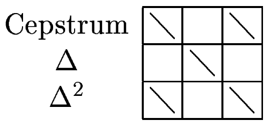

Fig. 4. Structural correlations in a typical MFCC-derived inverse covariance matrix. The large magnitude components are the result of the way the second-order derivatives are computed from the cepstral coefficients.

The importance of this distinction lies into the following observation: while *structural* correlations are usually large in magnitude, they do not provide any real information about the data, and thus modeling those will not improve the model much. On the other hand, *incidental* correlations can be smaller in magnitude, but they bring information about the data, and explicitly representing these will improve the model.

To illustrate this point, the following experiments were carried out. Several otherwise identical acoustic models were trained using different covariance structures. The error rates of recognition experiments run using these acoustic models are reported in Figure 5. The test-set is described in Section VI. Each (■) represents a block of non-zero entries in the covariance matrix, while an empty cell denotes entries that were zeroed.

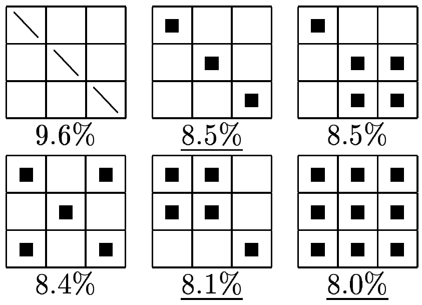

Fig. 5. Error rates for different covariance structures, ranging from diagonal (top-left) to full (bottom-right). Note that most of the gain results from modeling within-block correlations along the diagonal. Adding the block corresponding to correlations between cepstra and Δ^2, most of which are *structural*, does not improve the accuracy significantly. Introducing correlations between cepstra and Δ improves the performance by a proportionally larger amount.

From Figure 5, it is clear that modeling the correlations within blocks, i.e. incorporating the 3 blocks denoted *a*, *b* and *c* in Figure 1 into the model, is responsible for a large part of the benefits of full covariance modeling with respect to diagonal models. It is also clear that adding the correlations between cepstra and Δ^2 (block *f*), which are large in magnitude but mostly structural, does not cause a significant decrease in error rate. On the other hand, incorporating correlations between cepstra and Δ coefficients (block *d*) brings the performance of a 2 block system close to the performance of a full covariance model.

In conclusion, it appears that three classes of models are of interest for MFCC-based system. These are the models whose error rate figures are underlined in Figure 5. The first model (on the lower right of the figure) is a full-covariance model, that will be referred to as a "1-block" model. The second one is a "2-block" model, one block modeling jointly the cepstra and Δ features, and the second modeling the Δ^2. The third "3-block" model uses one block per group of features: cepstra, Δ and Δ^2. Detailed analysis of the performance of MIC applied to these models is carried out in Section VI-D.

IV. Likelihood Computation

The log-likelihood of Gaussian i for observation vector $\boldsymbol{o}$ can be written:

$$\mathcal{L}_i(\boldsymbol{o}) = c_i - \frac{1}{2}(\boldsymbol{o} - \boldsymbol{\mu}_i)^\top \Sigma_i^{-1}(\boldsymbol{o} - \boldsymbol{\mu}_i)$$

using the constant:

$$c_i = \frac{1}{2}(\log|\Sigma_i^{-1}| - D\log 2\pi)$$

When $\Sigma_i^{-1} = \sum_{k=1}^{K} \lambda_{k,i}\Psi_k$:

$$\begin{aligned}\mathcal{L}_i(\boldsymbol{o}) &= \underbrace{c_i - \frac{1}{2}\boldsymbol{\mu}_i^\top \Sigma_i^{-1}\boldsymbol{\mu}_i}_{c'_i} - \sum_{k=1}^{K} \lambda_{k,i} \underbrace{\frac{1}{2}\boldsymbol{o}^\top \Psi_k \boldsymbol{o}}_{\omega_k} \\ &\quad - \underbrace{\left[-\Sigma_i^{-1}\boldsymbol{\mu}_i\right]^\top}_{\boldsymbol{\nu}_i} \boldsymbol{o}\end{aligned}$$

The term $\frac{1}{2}\boldsymbol{\mu}_i^\top \Sigma_i^{-1}\boldsymbol{\mu}_i$ can be absorbed into the constant c'_i. The vector $\boldsymbol{\omega} : [\omega_1 \dots \omega_K]^\top$ is independent of the Gaussian and can be computed as an additional K-dimensional feature vector appended to $\boldsymbol{o}$. $\boldsymbol{\nu}_i$ is a D-dimensional Gaussian-specific vector, which leads to expressing the Gaussian computation in terms of:

- An extended feature vector: $\boldsymbol{o}' = \begin{bmatrix} \boldsymbol{o} \\ \boldsymbol{\omega} \end{bmatrix}$,
- A Gaussian-specific parameter vector: $\boldsymbol{\nu}'_i = \begin{bmatrix} \boldsymbol{\nu}_i \\ \boldsymbol{\Lambda}_i \end{bmatrix}$, with $\boldsymbol{\Lambda}_i = [\lambda_{1,i} \dots \lambda_{K,i}]^\top$.

Using this notation, the likelihood can be expressed as a scalar product between these two $K + D$ dimensional vectors:

$$\mathcal{L}_i(\boldsymbol{o}) = c'_i - {\boldsymbol{\nu}'_i}^\top \boldsymbol{o}'$$

This computation requires $D + K$ sums and products, to be compared with $2D$ for a diagonal Gaussian. Note that K can be smaller than D, in which case the Gaussians are less expensive to evaluate than in the diagonal case.

The front-end overhead is limited to the computation of $\boldsymbol{\omega}$. When the prototypes are positive definite, the quadratic form can be decomposed into its Cholesky factorization:

$$\frac{1}{2}\Psi_k = L_k L_k^\top$$

The resulting computation:

$$\boldsymbol{\xi}_k(\boldsymbol{o}) = L_k^\top \boldsymbol{o} \;\Rightarrow\; \omega_k = \boldsymbol{\xi}_k(\boldsymbol{o})^\top \boldsymbol{\xi}_k(\boldsymbol{o})$$

use on the order of $\frac{1}{2}KD^2$ multiplications.

When using a class-based approach with C classes, this overhead grows as $\frac{1}{2}CKD^2$, unless the classes to which the significant Gaussians belong can be predicted and computations can be saved. This is the case for example when using state-dependent classes in a HMM: only the $\boldsymbol{\omega}$ corresponding to active states need to be computed.

When using a subspace-factored model, the front-end overhead is reduced to:

$$\frac{1}{2}\sum_l K_l D_l^2 \;\leq\; \frac{1}{2}K[\max_l D_l]^2$$

In both cases, the log-likelihood computation itself is unaffected.

Note that, using this formulation, it is possible to perform partial evaluation of the Gaussian for the purposes of quickly pruning insignificant Gaussians in the mixture. Since:

$$\boldsymbol{\omega}^\top \boldsymbol{\Lambda}_i = \boldsymbol{o}^\top \Sigma_i^{-1} \boldsymbol{o} \geq 0$$

We have the inequality:

$$\begin{aligned} \mathcal{L}_i(\boldsymbol{o}) &= c_i' - \boldsymbol{\nu}_i{}^\top \boldsymbol{o} - \boldsymbol{\omega}^\top \boldsymbol{\Lambda}_i \\ &\leq c_i' - \boldsymbol{\nu}_i{}^\top \boldsymbol{o} \end{aligned}$$

This upper bound on the likelihood can be tested without any additional computation, prior to a full evaluation of the Gaussian, in order to determine if the Gaussian is significant or not.

It is also common [24] to weight the Gaussian log-likelihood in a mixture by a factor $\alpha \leq 1$ such that:

$$f(\boldsymbol{o}) = \sum_{i=1}^{M} w_i [\mathcal{N}(\boldsymbol{o}, \boldsymbol{\mu}_i, \Sigma_i)]^\alpha$$

This exponent typically improves the performance by reducing the dynamic range of the Gaussian scores when diagonal covariances are used. When a MIC model is used, α needs to be tuned for the particular model used. In the limit, the model approximates closely enough the full covariance, the value $\alpha = 1$ is optimal.

V. Maximum Likelihood Estimation of the Model

The sample covariance estimated from the observations $\boldsymbol{o}_t$ and priors $\gamma_{i,t}$ will be noted:

$$\bar{\Sigma}_i = \sum_t \gamma_{i,t} (\boldsymbol{o}_t - \boldsymbol{\mu}_i)(\boldsymbol{o}_t - \boldsymbol{\mu}_i)^\top \tag{3}$$

Given the independent parameters $w_i, \boldsymbol{\mu}_i$, and the sample covariance $\bar{\Sigma}_i$, the parameters of the model (Ψ, Λ), with:

$$\begin{aligned} \Psi &= \{\Psi_1, \ldots, \Psi_K\} \\ \Lambda &= \{\boldsymbol{\Lambda}_1, \ldots, \boldsymbol{\Lambda}_M\} \end{aligned}$$

can be estimated jointly using the EM algorithm [23].

Using $\boldsymbol{x}^\top A \boldsymbol{x} = \mathrm{Tr}(A \boldsymbol{x}\boldsymbol{x}^\top)$, the auxiliary function can be written:

$$\begin{aligned} Q(\Psi, \Lambda) &= \sum_{i=1}^{M} \sum_t \gamma_{i,t} \Big[\log |\Sigma_i^{-1}| \\ &\quad -(\boldsymbol{o}_t - \boldsymbol{\mu}_i)^\top \Sigma_i^{-1} (\boldsymbol{o}_t - \boldsymbol{\mu}_i)\Big] \\ &= \sum_{i=1}^{M} w_i \left[\log |\Sigma_i^{-1}| - \mathrm{Tr}\left(\Sigma_i^{-1}\bar{\Sigma}_i\right)\right] \end{aligned} \tag{4}$$

With the constraint that:

$$\Sigma_i^{-1} = \sum_k \lambda_{k,i} \Psi_k$$

A. *Casting the Problem in Terms of Convex Optimization*

Maximum-likelihood estimation of the parameters (Ψ, Λ) of the model can not be performed by a direct method. However, owing to the concavity of $\log |A|$ when A is positive definite [25], and to the linearity of the trace, both the functions $Q(\Psi|\Lambda)$ and $Q(\Lambda|\Psi)$ are concave on the domain $\Sigma_i \succ 0$ (read "the domain in which all the covariances Σ_i are positive definite"). Moreover, the domains:

- $\mathcal{L} : \Lambda \ / \ \{\forall i, \sum \lambda_{k,i}\Psi_k \succ 0\}$
- $\mathcal{P} : \Psi \ / \ \{\forall i, \sum \lambda_{k,i}\Psi_k \succ 0\}$

are both convex.

Thus, the problem of jointly estimating Ψ and Λ can be decomposed into two convex optimization problems to be solved iteratively:

$$\begin{array}{l|l} \text{Maximize } Q(\Lambda|\Psi) & \text{Maximize } Q(\Psi|\Lambda) \\ \text{Subject to } \Lambda \in \mathcal{L} & \text{Subject to } \Psi \in \mathcal{P} \end{array}$$

It is interesting to relate this approach to the classic Lloyd clustering [16, Section 6.2] and EM algorithms. The maximization of $Q(\Lambda|\Psi)$ is similar to the nearest neighbor partitioning step of Lloyd, except that the partitioning performed here is a "soft" allocation of the covariance to the various prototypes. In that respect, it is similar to the E step of the EM algorithm applied to GMM, which computes the class allocation weights for each mixture component. The maximization of $Q(\Psi|\Lambda)$, on the other hand, is akin to the centroid computation of the Lloyd algorithm or the M step of the EM algorithm, which both attempt to come up with a better set of component-dependent parameters given the fixed component allocation scheme.

Here, the distortion criterion used for both the "partitioning" and the "centroid computation" is the Q function. Up to constant terms, it is identical to the MDI criterion, which has already been used as a criterion to cluster Gaussians [26] in the design of Gaussian mixtures. In the sections that follow, we describe a succession of algorithms for reestimating the weights, initializing the weights, reestimating the prototypes and initializing the prototypes of a MIC.

B. *Reestimation of the Weights*

The weight estimation given the prototype covariances can be performed efficiently using a Newton algorithm [27]. The gradient of the auxiliary function can be computed using (see e.g. [28]):

$$\frac{\partial \log |A(x)|}{\partial x} = \mathrm{Tr}\left[A^{-1}(x)\frac{\partial A(x)}{\partial x}\right]$$

Thus:

$$\frac{\partial}{\partial \lambda_{k,i}} \log |\Sigma_i^{-1}| = \mathrm{Tr}\left(\Sigma_i \Psi_k\right)$$

Since:

$$\frac{\partial}{\partial \lambda_{k,i}} \mathrm{Tr}\left(\Sigma_i^{-1}\bar{\Sigma}_i\right) = \mathrm{Tr}\left(\Psi_k \bar{\Sigma}_i\right)$$

The gradient is:

$$\frac{\partial Q}{\partial \lambda_{k,i}} = \mathrm{Tr}\left[\Psi_k(\Sigma_i - \bar{\Sigma}_i)\right] \tag{5}$$

In the following, we will sometimes represent a symmetric matrix A in vector form — noted $\boldsymbol{A}^\star$, constructed by stacking together the diagonal $\boldsymbol{a}_0$ and the super-diagonals $\boldsymbol{a}_i, i \in [1, D-1]$ multiplied by $\sqrt{2}$:

$$\boldsymbol{A}^\star = [\boldsymbol{a}_0^\top \sqrt{2}\boldsymbol{a}_1^\top \ldots \sqrt{2}\boldsymbol{a}_{D-1}^\top]^\top$$

The $\sqrt{2}$ factor ensures that:

$$\mathrm{Tr}(AB) = \boldsymbol{A}^{\star\top}\boldsymbol{B}^\star$$

This identity maps a symmetric matrix representation and its associated Frobenius norm into a vector representation with minimal dimensionality ($D(D+1)/2$) and the more familiar L_2 norm. It is also a memory-efficient way of representing symmetric matrices which is well suited to the implementation of the reestimation algorithms. Using this convention, and denoting $P = [\mathbf{\Psi}_1^\star \ldots \mathbf{\Psi}_k^\star]$, we can write Equation 5 as:

$$\frac{\partial Q}{\partial \Lambda_i} = P^\top (\mathbf{\Sigma}_i^\star - \bar{\mathbf{\Sigma}}_i^\star) \tag{6}$$

The components of the Hessian H can be computed using the identity:

$$\frac{\partial A^{-1}(x)}{\partial x} = -A^{-1}(x) \frac{\partial A(x)}{\partial x} A^{-1}(x)$$

Which results in:

$$\begin{aligned} \frac{\partial^2 Q}{\partial \lambda_{k,i} \partial \lambda_{l,i}} &= \mathrm{Tr}\left[\Psi_k \frac{\partial \Sigma_i}{\partial \lambda_{l,i}}\right] \\ &= -\mathrm{Tr}\left[\Psi_k \Sigma_i \Psi_l \Sigma_i\right] \end{aligned}$$

Under the mild assumption of linear independence between the Ψ_k, the Hessian is invertible.

Proof: If $\{\Psi_k, k \in [1,K]\}$ is an independent family, since Σ_i is full rank, so is $\{\Psi_k \Sigma_i, k \in [1,K]\}$. Consider the $K \times D(D+1)/2$ matrix Ω_i whose k^{th} column lists all the entries of $\Psi_k \Sigma_i$ in any consistent order. The matrix Ω_i is nonsingular, and we have:

$$H_i = -\Omega_i^\top \Omega_i$$

Thus for any $\boldsymbol{X} \neq 0$:

$$\boldsymbol{X}^\top H_i \boldsymbol{X} = -(\Omega_i \boldsymbol{X})^\top (\Omega_i \boldsymbol{X}) < 0$$

and consequently H_i is negative definite, thus invertible. ■

In the unlikely case of some prototypes being linearly dependent on others, all the inverse covariances expressed as a linear combination of the prototypes can always be expressed as a function of a smaller set of independent ones, for which the Hessian will be invertible.

The optimization can be noticeably simplified by remarking that for any covariance Σ:

$$\mathbf{\Sigma}^{\star\top} \mathbf{\Sigma}^{-1\star} = \mathrm{Tr}(\Sigma \Sigma^{-1}) = D \;\; (= \text{dimensionality})$$

For $\mathbf{\Lambda}$ to be a maximum-likelihood weight vector, the gradient in Equation 6 is necessarily zero, and:

$$P^\top \mathbf{\Sigma}^\star = P^\top \bar{\mathbf{\Sigma}}^\star$$

Thus, using:

$$\mathbf{\Sigma}^{-1\star} = P\mathbf{\Lambda}$$

We have:

$$\mathbf{\Sigma}^{\star\top} P \mathbf{\Lambda} = (P^\top \bar{\mathbf{\Sigma}}^\star)^\top \mathbf{\Lambda} = D$$

This relationship defines an affine hyperplane orthogonal to:

$$\mathbf{\Lambda}_0 = D \frac{P^\top \bar{\mathbf{\Sigma}}^\star}{\|P^\top \bar{\mathbf{\Sigma}}^\star\|^2} \tag{7}$$

in which $\mathbf{\Lambda}$ is constrained to live. Denoting U a basis of the orthogonal of $P^\top \bar{\mathbf{\Sigma}}^\star$, we have:

$$\mathbf{\Lambda} = \mathbf{\Lambda}_0 + U\mathbf{\Lambda}'$$

The gradient ascent algorithm can now be performed on $\mathbf{\Lambda}' \in \mathrm{Span}(U)$, which is of dimension $K-1$, by projecting the Newton update onto $\mathrm{Span}(U)$. The Hessian can be computed easily using Equation 7 at each step of the iteration, leading to an update step:

$$\tilde{\mathbf{\Delta}} = H^{-1} P^\top (\mathbf{\Sigma}^\star - \bar{\mathbf{\Sigma}}^\star)$$

Which, projected onto $\mathrm{Span}(U)$, becomes:

$$\mathbf{\Delta} = \tilde{\mathbf{\Delta}} - \frac{\mathbf{\Lambda}_0^\top \tilde{\mathbf{\Delta}}}{\|\mathbf{\Lambda}_0\|^2} \tilde{\mathbf{\Delta}} \tag{8}$$

By concavity of $Q(\Lambda|\Psi)$, the algorithm will converge to a global maximum. The Newton update:

$$\mathbf{\Lambda} \to \mathbf{\Lambda} + \gamma \mathbf{\Delta} \tag{9}$$

converges after a few iterations. In general $\gamma = 1$, although in the first steps of the iteration it sometimes needs to be reduced to prevent intermediate estimates of $\mathbf{\Lambda}$ to step out of $\mathcal{L}$.

C. *Weight Initialization*

If the prototypes are positive definite, then $\mathbf{\Lambda}_0 \in \mathcal{L}$.

Proof: The k^{th} element of $\mathbf{\Lambda} = P^\top S^\star$ is:

$$\lambda_k = \Psi_k^{\star\top} S^\star = \mathrm{Tr}(\Psi_k S)$$

Since Ψ_k and S are positive definite symmetric matrices, $\mathrm{Tr}\,(\Psi_k S) = \lambda_k > 0$. As a consequence, $P\mathbf{\Lambda}_0 = \sum_k \lambda_k \Psi_k^\star$ is a linear combination with positive weights of positive definite matrices. From the definition of positive-definiteness:

$$\forall k,\ \boldsymbol{x}^\top \Psi_k \boldsymbol{x} > 0,\ \lambda_k > 0 \ \Rightarrow\ \sum_k \lambda_k \boldsymbol{x}^\top \Psi_k \boldsymbol{x} > 0$$

And $P\mathbf{\Lambda}_0$ is positive definite, which implies $\mathbf{\Lambda}_0 \in \mathcal{L}$. ■

We will see in Section V-E that the method that we use for generating the initial prototypes guarantees positive-definiteness, and thus $\mathbf{\Lambda}_0$ can be used to initialize the algorithm.

D. *Reestimation of the Prototypes*

In order to reestimate the prototypes given the weights, the Q function in Equation 4 has to be maximized with respect to each prototype Ψ_k. With A a symmetric matrix, using the cofactor decomposition of the determinant, we have (see e.g. [29]):

$$\frac{\partial |A|}{\partial A\rfloor_{i,j}} = \begin{cases} \mathcal{A}_{i,j} & \text{if } i = j \\ 2\mathcal{A}_{i,j} & \text{if } i \neq j \end{cases}$$

Where $\mathcal{A}_{i,j}$ are the cofactors of A, and $A\rfloor_{i,j}$ denotes the (i,j) entry of matrix A.

Similarly, if $A = \sum_k \lambda_k A_k$:

$$\frac{\partial |A|}{\partial A_k\rfloor_{i,j}} = \begin{cases} \lambda_k \mathcal{A}_{i,j} & \text{if } i = j \\ 2\lambda_k \mathcal{A}_{i,j} & \text{if } i \neq j \end{cases}$$

Thus:

$$\begin{aligned} \frac{\partial \log |A|}{\partial A_k} &= \begin{cases} \lambda_k \mathcal{A}_{i,j}/|A| & \text{if } i = j \\ 2\lambda_k \mathcal{A}_{i,j}/|A| & \text{if } i \neq j \end{cases} \\ &= \lambda_k \left[2A^{-1} - \mathrm{Diag}(A^{-1}) \right] \end{aligned}$$

Consequently:

$$\frac{\partial}{\partial \Psi_k}\sum_{i=1}^{M} w_i \log|\Sigma_i^{-1}| = \sum_{i=1}^{M} w_i\lambda_{k,i}[2\Sigma_i - \mathrm{Diag}(\Sigma_i)]$$

With A a symmetric matrix, we also have:

$$\frac{\partial \mathrm{Tr}(AB)}{\partial A\rfloor_{i,j}} = \begin{cases} B\rfloor_{i,i} & \text{if } i=j \\ B\rfloor_{j,i} + B\rfloor_{i,j} & \text{if } i \neq j \end{cases}$$

And thus:

$$\frac{\partial \mathrm{Tr}(AB)}{\partial A} = B + B^\top - \mathrm{Diag}(B)$$

We can see that if $A = \sum_k \lambda_k A_k$ and B is symmetric:

$$\frac{\partial \mathrm{Tr}(AB)}{\partial A_k} \;=\; \lambda_k\left[2B - \mathrm{Diag}(B)\right]$$

Thus:

$$\frac{\partial}{\partial \Psi_k}\sum_{i=1}^{M} w_i \mathrm{Tr}\left(\Sigma_i^{-1}\bar{\Sigma}_i\right) = \sum_{i=1}^{M} w_i\lambda_{k,i}[2\bar{\Sigma}_i - \mathrm{Diag}(\bar{\Sigma}_i)]$$

Consequently:

$$\frac{\partial Q}{\partial \Psi_k} \;=\; \sum_{i=1}^{M} w_i\lambda_{k,i}\left[2\left(\Sigma_i - \bar{\Sigma}_i\right) - \mathrm{Diag}\left(\Sigma_i - \bar{\Sigma}_i\right)\right]$$

For A symmetric:

$$2A - \mathrm{Diag}\left(A\right) \equiv 0 \Leftrightarrow A \equiv 0$$

As a consequence, we can replace the likelihood gradient by:

$$\frac{\partial Q'}{\partial \Psi_k} = \sum_{i=1}^{M} w_i\lambda_{k,i}\left(\Sigma_i - \bar{\Sigma}_i\right) \tag{10}$$

The Hessian of the auxiliary function Q' is:

$$\begin{aligned}\frac{\partial^2 Q'}{\partial \Psi_k \partial \Psi_k\rfloor_{p,q}} &= \sum_{i=1}^{M} w_i\lambda_{k,i}\frac{\partial \Sigma_i}{\partial \Psi_k\rfloor_{p,q}} \\ &= -\sum_{i=1}^{M} w_i\lambda_{k,i}^2\Sigma_i\frac{\partial \Psi_k}{\partial \Psi_k\rfloor_{p,q}}\Sigma_i\end{aligned}$$

Let's denote by $E_{i,j}$ the matrix containing all 0s except 1s at locations (i,j) and (j,i), and by $\boldsymbol{\sigma}_i^j$ the j^{th} column of Σ_i:

$$\begin{aligned}\frac{\partial^2 Q'}{\partial \Psi_k \partial \Psi_k\rfloor_{p,q}} &= -\sum_{i=1}^{M} w_i\lambda_{k,i}^2\Sigma_i E_{p,q}\Sigma_i \\ &= \frac{-1}{1+\delta_{p,q}}\sum_{i=1}^{M} w_i\lambda_{k,i}^2[\boldsymbol{\sigma}_i^p\boldsymbol{\sigma}_i^{q\top} + \boldsymbol{\sigma}_i^q\boldsymbol{\sigma}_i^{p\top}]\end{aligned} \tag{11}$$

Since there are only $D(D+1)/2$ of the D^2 entries of the prototype matrix that are independent, we need to represent the matrix in minimal form for the Hessian to be invertible. Using the notation defined in Section V-B, we can write the Newton iteration as:

$$\mathbf{\Psi}_k^\star \leftarrow \mathbf{\Psi}_k^\star + \gamma H^{-1} \sum_{i=1}^{M} w_i \lambda_{k,i} \left(\mathbf{\Sigma}_i^\star - \bar{\mathbf{\Sigma}}_i^\star \right) \tag{12}$$

Note that because of the $\sqrt{2}$ scaling factor of the off-diagonal terms of Ψ_k (represented as $\mathbf{\Psi}_k^\star$), and of Σ_i (represented as $\mathbf{\Sigma}_i^\star$), the entries of the Hessian matrix need to be scaled accordingly.

The Hessian, however, is not guaranteed to be invertible. In particular, if $2M < D$, it is always singular:

Proof: Each column of the Hessian contains in vector form the entries of the matrix:

$$C_{p,q} = -\sum_{i=1}^{M} w_i \lambda_{k,i}^2 \Sigma_i E_{p,q} \Sigma_i \quad \text{for } 1 \le q \le p \le D$$

Let's assume $2M < D$. Since Rank$(E_{p,q}) \le 2$, then Rank$(C_{p,q}) \le 2M$. Thus, the family $\Xi = \{C_{p,q}\ 1 \le q \le p \le D\}$ is contained in the space of symmetric matrices of rank smaller or equal to $2M$, which is a strict subspace of the vector space of symmetric matrices. The vector space of symmetric matrices is of dimensionality $D(D+1)/2$ ($\{E_{p,q}, 1 \le q \le p \le D\}$ is a canonical basis for it), and thus Ξ lives in a space of dimensionality strictly smaller than $D(D+1)/2$. Since the number of vectors in Ξ is $D(D+1)/2$, the family is not linearly independent, and consequently the Hessian is singular. ■

This condition is not necessary, and in most cases the number of covariances in the GMM is large enough for this bound not to be reached. A simple regularization method such as flooring of the eigenvalues will guarantee that a singularity of the Hessian matrix never causes the Newton iteration to abort.

The exact gradient and Hessian could be expensive to compute using these equations because of the potentially large number of covariances in the GMM. However both can be well estimated by adding up the contributions of a small subset of significant Gaussians. A principled way of selecting the Gaussians is to sort them by the magnitude of their relative weight in Equations 10 and 11, which are $|w_i \lambda_{k,i}|$ for the gradient and $w_i \lambda_{k,i}^2$ for the Hessian, and only accumulate the contributions of the Gaussians with the highest weight. However, it is beneficial for the overall speed of the algorithm not to have to compute the weights for all the Gaussians at each iteration before being able to make the decision whether or not to use them to reestimate the prototypes. It is thus more efficient to select at the beginning of the iterative process a set of significant Gaussians based only on w_i and only run both the weight and prototype reestimation algorithm on those. In the following experiments, less than 10% of the Gaussians were used to estimate the gradient, and less than 1% were incorporated into the Hessian. As previously, the step size γ has sometimes to be reduced to a smaller value in the first iterations to avoid stepping out of the domain $\mathcal{P}$.

E. Prototype Initialization

The initial set of prototypes can be generated by a hard clustering scheme: the M Gaussian covariances are clustered down to K initial prototypes by using the Lloyd algorithm. A Kullback-Liebler distance criterion is used, since it is a natural choice of a metric [26] between Gaussians. The distance between the Gaussian means can be ignored, since only the covariances are of interest. In addition, the variations in the scale of the prototypes — i.e. their determinant – can be normalized for, since these are captured by the weights in Equation 2:

$$\Psi_i = |\Sigma_i| \Sigma_i^{-1} \tag{13}$$

The distance measure used for clustering is thus:

$$d(\Psi_k, \Psi_l) = \mathbf{\Psi}_k^{\star\top} \mathbf{\Psi}_l^{\star -1} + \mathbf{\Psi}_l^{\star\top} \mathbf{\Psi}_k^{\star -1} \tag{14}$$

For simplicity, the centroid for each cluster is computed as the average of all the covariances allocated to this cluster. As a consequence, each centroid is guaranteed to be positive definite, which allows us to use the simple weight initialization scheme described in Section V-C. Experimentally, it has been observed that the speed of convergence of the global algorithm is much improved when such clustering is applied, as opposed to a more naive initialization scheme.

F. Implementation of the Algorithm

The implementation of the algorithm on top of a Baum-Welch reestimation algorithm is fairly straightforward (Table I). The iterative MIC reestimation scheme — steps 6 to 10 — needs to be implemented at each step of the EM reestimation, after the ML estimation of the sample mixture weights, means and covariances. In the first iteration, the prototypes can be initialized using the VQ scheme described in Table II. Note that at each EM stage, the iteration between the weight estimation (Table III) and prototype reestimation (Table IV) need only to be carried over a small subset of all the Gaussians in the mixture, since only a fraction of the covariances are used to reestimate the prototypes. In the final iteration however, the weights for all the covariances have to be reestimated.

The only implementation detail worth noting in Table IV is the two-phase approach to the prototype reestimation algorithm. In a first phase (Table IV, 1 to 8), the algorithm goes through each prototype and does one Newton update at each pass. In the second phase (Table IV, 9 to 17), the Newton iterations are repeated until the gradient is small enough. The reason for using this approach is that in the first few iterations, all the prototypes are far from the optimum. When updating a particular prototype, the first Newton steps are large in magnitude. This means that the gradient and Hessian estimates for other prototypes, which depend on every prototype in the MIC, will change dramatically at each Newton step which is taken. As a consequence, it is not beneficial to take several consecutive Newton steps in one direction since this direction will change dramatically after one cycle through the prototypes. After a few cycles, however, some prototypes will be close to their optimal, while others will still be very far from it. Cycling through the prototypes and performing one Newton update each time becomes inefficient because the algorithm keeps on updating well estimated prototypes. For this reason, the second phase optimizes the prototypes one at a time until convergence.

Table V shows typical values for the various iteration loops. These vary somewhat with the dimensionality of the problem, but the overall number of Newton updates is well within the hundreds for both the weights and prototypes, which makes the overall algorithm computationally tractable. Figure 6 shows that the likelihood increase from the iterative process typically reaches a plateau in about 6 iterations.

```
 1  generate initial GMM (without MIC)
 2  for EM iteration = 1 to N
 3     compute sufficient statistics from data and model:
       $\boldsymbol{f}_i = \sum_t \gamma_{i,t}\boldsymbol{o}_t \quad S_i = \sum_t \gamma_{i,t}\boldsymbol{o}_t\boldsymbol{o}_t^\top$
 4     compute mixture weights and means:
       $w_i = \sum_t \gamma_{i,t}/M, \quad \boldsymbol{\mu}_i = \boldsymbol{f}_i/w_i$
 5     compute sample covariances (Equation 3)
       if N == 1
 6        subset covariances based on $w_i$ (Section V-D)
 7        initialize prototypes (Section V-E and Table II)
       end if
       for iteration = 1 to P
 8        estimate weights (Sections V-B, V-C, Table III)
 9        update prototypes (Section V-D and Table IV)
       end for
10     estimate weights for all covariances
       (same as step 8)
11     update model
    end for
```

TABLE I
OVERVIEW OF THE EM ALGORITHM

VI. EXPERIMENTS

In this section, the MIC model is applied to a GMM used for acoustic modeling in a HMM-based continuous speech recognition system. A comparison against semi-tied covariances is carried out in Section VI-B. In Section VI-C, the

```
1   normalize covariance determinants (Equation 13)
2   select K initial prototypes out of the M covariances
3   for VQ iteration = 1 to Q
4     for covariance $i$ = 1 to M
5       find closest prototype $k$ (Equation 14)
6       accumulate statistics for this centroid:
        $\Phi_k = \Phi_k + \Psi_i, \quad c_k = c_k + 1$
      end for
7     reestimate centroids: $\Psi_k = \Phi_k / c_k$
8     fix empty cells (see e.g. [16])
    end for
```

TABLE II

OVERVIEW OF THE PROTOTYPES INITIALIZATION

```
1   for covariance = 1 to M
2     initialize weights (Equation 7)
      for iteration = 1 to V
3       compute gradient (Equation 6)
4       break loop if gradient$< \epsilon$
5       compute $\boldsymbol{\Delta}$ (Equation 8)
6       do $\gamma = 1$, decreasing
7         update weight vector (Equation 9)
8       while covariance not positive definite
      end for
    end for
```

TABLE III

OVERVIEW OF THE WEIGHTS REESTIMATION

accuracy gains are reported for MIC models at various levels of parametric complexity. The subspace-factored model is explored in Section VI-D, and the class-based approach in Section VI-E. Finally, the speed/accuracy trade-off is explored on a complete real-time ASR system in Section VI-F.

A. Experimental Setup

The recognition engine used is a context-dependent HMM system with 3358 triphones and tied-mixtures based on genones [30]: each state cluster shares a common set of Gaussians, while the mixture weights are state-dependent. The system has 1500 genones and 32 Gaussians per genone. The test-set is a collection of 10397 utterances of Italian telephone speech spanning several tasks, including digits, letters, proper names and command lists, with fixed task-dependent grammars for each test-set. The features are 9-dimensional MFCC with Δ and Δ^2.

The training data comprises 89000 utterances. Each model is trained using fixed HMM alignments for fair comparison. The GMM are initially trained using full or block-diagonal covariances — depending on the MIC structure used — using Gaussian splitting [31]. After the number of Gaussian per genone is reached using splitting, the sufficient statistics are collected and the MIC model trained in one iteration. For this reason, the performance results reported here are lower bounds on the accuracy that is achievable using the MIC model. Better performance would certainly be achieved by jointly optimizing the alignments and by reiterating the MIC training a few times.

The accuracy is evaluated using a sentence understanding error rate, which measures the proportion of utterances in the test-set that were interpreted incorrectly. The Gaussian exponent α (see Section IV) was globally optimized for each model on the entire collection of test-sets.

```
 1  for iteration = 1 to S
 2    for prototype = 1 to K
 3      estimate gradient (Equation 10)
 4      estimate Hessian (Equation 11)
 5      do $\gamma = 1$, decreasing
 6        update prototype (Equation 12)
 7        reestimate gradient (Equation 10)
 8      until $\gamma$ minimizing gradient is found
      end for
    end for
    do (outer loop)
 9    for prototype = 1 to K
        do (inner loop)
10        estimate gradient (Equation 10)
11        break inner loop if gradient$< \epsilon$
12        estimate Hessian (Equation 11)
13        do $\gamma = 1$, decreasing
14          update prototype (Equation 12)
15          reestimate gradient (Equation 10)
16        until $\gamma$ minimizing gradient is found
        loop
      end for
17  loop unless gradient$< \epsilon$ for all prototypes
```

TABLE IV
OVERVIEW OF THE PROTOTYPES REESTIMATION

EM iterations	$N \sim 1$ or 2
weights/prototypes optimizations	$P = 6$
VQ iterations	$Q = 4$
weight optimization	$V \sim 10$
prototype optimization: initial loop	$S = 4$
prototype optimization: outer loop	~ 4
prototype optimization: inner loop	< 10

TABLE V
TYPICAL NUMBER OF ITERATIONS

B. *Comparison against Semi-Tied Covariances*

Semi-tied covariances [10] is a very closely related model to the MIC as discussed in Section II-A. To compare the two approaches, the number of Gaussian-specific parameters in the GMM was kept constant (27) for the MIC and semi-tied models. Table VI shows the error rate on the test-set described previously. The error rate reduction using the MIC model is more than 3 times the error rate reduction obtained with semi-tied covariances.

C. *Accuracy vs. Complexity*

Figure 7 shows how the model performs as the number of Gaussian-specific parameters change. The MIC model almost matches the performance of a full-covariance system with about 45 Gaussian-specific parameters. As few as 9 parameters are sufficient for the model to match the accuracy of the diagonal covariance system.

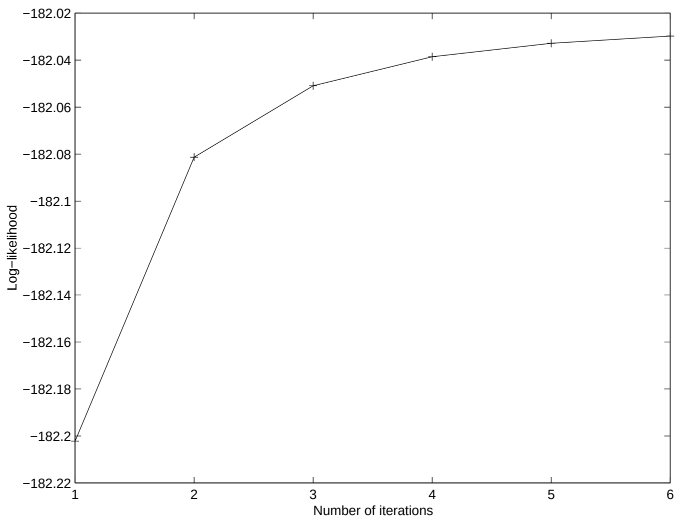

Fig. 6. Increase in the Q function as a function of the number of iterations. One iteration corresponds to running the prototype reestimation followed by the weight reestimation algorithm once. In the first iteration, the initial prototypes are computed using VQ.

Structure	Error Rate	Relative Improvement
Diagonal	9.64%	-
Semi-tied	9.24%	4.1%
MIC	8.29%	14.0%

TABLE VI

ERROR RATES ON A SET OF ITALIAN TASKS

D. Subspace-factored Approach

From the analysis in Section III, we would expect two things from a subspace-factored model using MIC:

1) In the limit of large number of Gaussian-specific parameters, the model should tend to the performance of a system where each Gaussian has a separate block-diagonal covariance (Figure 5). Thus its performance will be worse than a full covariance system.
2) In the limit of small number of Gaussian-specific parameters, the subspace-factored systems should outperform a full-covariance MIC system due to the effect of having a much larger number of effective prototypes in the system for a same number of weights.

Figure 8 shows that it is indeed the case: with 9 parameters, the 3-block system performs as well as the 1-block system, and outperforms it with only 3 parameters, while the 2-block system outperforms the 1-block system up to approximately 16 Gaussian-specific parameters. In these experiments, the number of parameters allocated to each block was kept proportional to the block size, but the allocation scheme could also be optimized.

Note that because of the front-end computations, for a given number of Gaussian-specific parameters, the computational complexity of a 3-block system will be lower than the computational complexity of a 2-block system, which in turn will be lower than the computational complexity of a 1-block system. This means that in the limit of low number of Gaussian-specific parameters, although the accuracy of a 2-block system is comparable to the accuracy of a 3-block system, the latter will be computationally more efficient.

E. Class-based Approach

Table VII compares the performance of a 2-block system with systems for which the acoustic model is partitioned into a series of phonetically-derived classes. The gains obtained from using a class-based approach are small, and do not compare well to the gains that would be obtained by increasing the number of Gaussian-specific parameters. While it is possible that the phonetic clustering used here is sub-optimal, and that a more data-driven approach would show larger gains, it is very likely that with such a large number of Gaussians in the system, the optimal set of prototypes

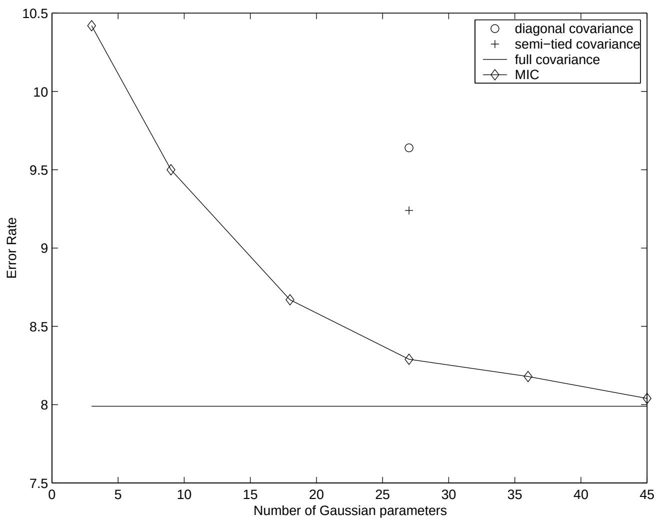

Fig. 7. Accuracy as a function of the number of Gaussian-specific parameters. The performance of the diagonal system is around 10%. As the number of Gaussian-specific parameters grows, the accuracy of the MIC approaches the accuracy of the full covariance model.

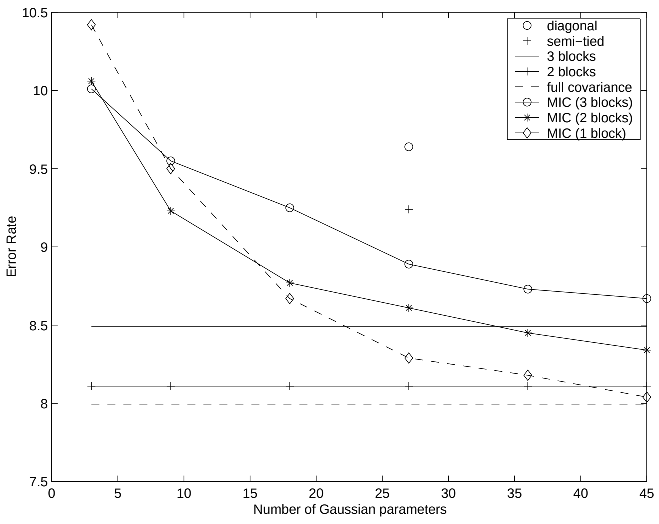

Fig. 8. Accuracy as a function of the number of Gaussian-specific parameters for the 2-block and 3-block subspace-factored approach, compared with the 1-block full covariance system.

derived for a particular phonetic class is close to the optimal for the entire GMM, and that significant accuracy benefits will only show with a much larger set of classes, which makes the approach unappealing in this context. Nevertheless, since the front-end overhead for 2-block systems is rather small, these small accuracy gains come with an extremely limited computational cost and can be of interest in contexts where the Gaussian computations dominate the front-end processing.

F. Speed vs. Accuracy

Figures 9 and 10 show how various configurations perform in real-time environments, respectively on small and large perplexity tasks. Each curve depicts the performance of a given system at various degrees of pruning in the acoustic search. By trading the number of search errors against the number of active hypotheses in the search, the accuracy of the system can be traded against its speed. Both the small and large perplexity test-sets are drawn from the Italian test-set described in Section VI-A, and contain respectively 5098 and 4612 utterances.

Because of the larger front-end overhead incurred by systems using the MIC model with full covariances (1 block), the relative slowdown on test-sets with low perplexity is much larger than the slowdown on high-perplexity test-sets.

# parameters	# classes	Error Rate
9	1	9.23%
9	3	9.14%
9	11	9.10%
27	1	8.61%
27	3	8.62%
27	11	8.48%

TABLE VII

ERROR RATES FOR 2-BLOCK SYSTEMS FOR VARIOUS NUMBERS OF CLASS-BASED MIC MODELS IN THE SYSTEM. EACH CLASS IS DERIVED BY CLUSTERING THE HMM STATES USING THEIR PHONETIC LABELS.

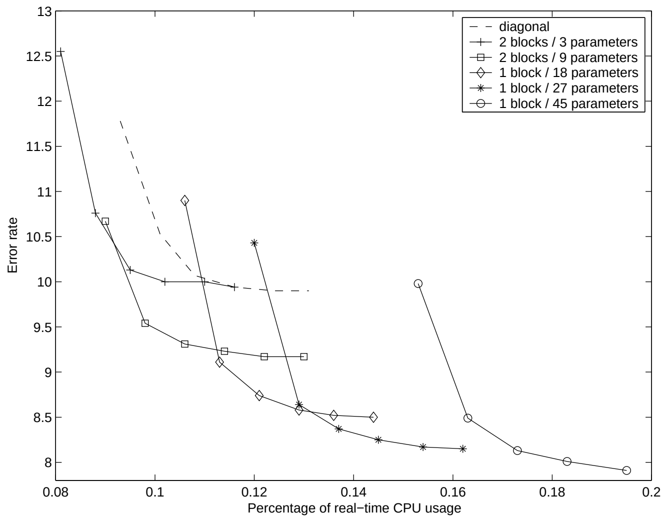

Fig. 9. Speed/Accuracy trade-off on a set of low-perplexity tasks. The error rate is plotted against the fraction of real-time CPU computations required to perform recognition.

When using models with multiple blocks, this effect is much smaller and does not appear to influence the results. Thus, the faster 2-block systems scale with the perplexity of the task approximatively in the same way as the diagonal model does. The speed improvement of a 3-block system (not plotted) compared to a 2-block system with similar complexity is never large enough to compensate for the loss in accuracy.

Typically, an optimally tuned recognizer would operate in the lower-right half of the speed/accuracy curve, close to the knee of the curve, where the efficiency of the system is maximized while not sacrificing accuracy by any significant amount. For both the small and large perplexity test-sets, the 9 parameter / 2 blocks system is the fastest model that would operate at the same level of accuracy as the baseline diagonal model at its optimal operating point. In both cases, the speed increase is about 10% at no cost in accuracy. In both cases as well, the full covariance MIC system is the most accurate at the same speed as the diagonal system at its optimal operating point. The accuracy gain without any slowdown is about 13% for the low-perplexity test-sets, and 8% for the high-perplexity test-sets.

Overall, the different model architectures allow for a wide range of operating points, and makes a system with an accuracy comparable to the accuracy of a full covariance MIC system (45 parameter / 1 block) reachable at an additional cost in computations of approximately 50%. On the same test-set, the increase of computation incurred when using a full covariance model is approximately 1100%.

VII. CONCLUSION

A low-complexity approximation to full covariance Gaussian mixture models was introduced, along with robust maximum likelihood estimation algorithms to compute the parameters of this model. A low-complexity subspace-factored approach extending that model was also introduced, and both models were applied to acoustic-modeling for

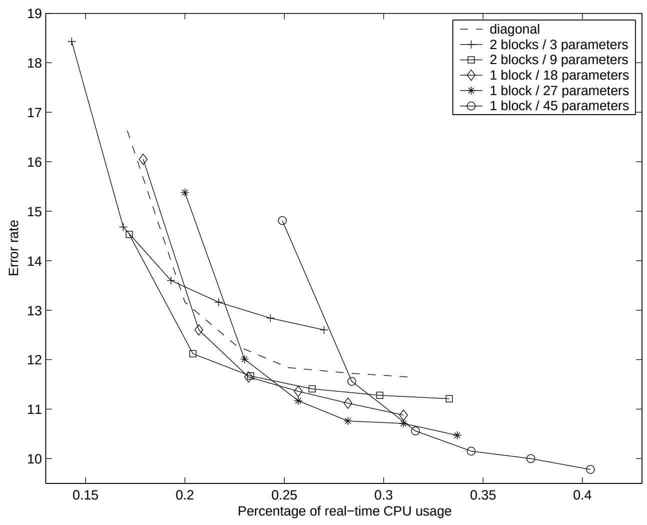

Fig. 10. Speed/Accuracy trade-off at various levels of pruning on large-perplexity tasks for the same configurations as Figure 9.

ASR. When used in the context of a GMM-based HMM acoustic model, this class of models lead to a broad range of systems which, in comparison with a standard diagonal system, can be:

- as much as 10% faster at no cost in accuracy,
- about 10% more accurate at no cost in speed,
- or about 16% more accurate at a 50% cost in speed.

Acknowledgments

The authors would like to thank M. Schuster and R. Teunen for their helpful contributions to this research. We are also grateful to Professor R. Olshen and Professor R.M. Gray for their insightful comments.

References

[1] O. Ledoit, *Essays on risk and return in the stock market*, Ph.D. thesis, Massachusetts Institute of Technology, Sloan School of Management, 1995.

[2] R. Clarke, “Relation between the Karhunen Loève and cosine transforms,” *IEE Proceedings*, vol. 128, no. 6-F, pp. 359–360, Nov. 1981.

[3] S. B. Davis and P. Mermelstein, “Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences,” *IEEE Trans. Acoustics, Speech, and Signal Processing*, vol. ASSP-28, no. 4, pp. 357, 1980.

[4] H. Hermansky, “Perceptual linear predictive (PLP) analysis of speech,” *Acoust. Soc. Amer.*, vol. 87, no. 4, pp. 1738–1752, 1990.

[5] H. Hermansky and N. Morgan, “Rasta processing of speech,” *IEEE Trans. Acoustics, Speech, and Signal Processing*, vol. 2, no. 4, pp. 578–589, 1994.

[6] T. Eisele, R. Haeb-Umbach, and D. Langmann, “A comparative study of linear feature transformation techniques for automatic speech recognition,” *Proceedings of ICSLP 96*, 1996.

[7] A. Ljolje, “The importance of cepstral parameter correlation in speech recognition,” *Computer Speech and Language*, vol. 8, pp. 223–232, 1994.

[8] B. Doherty, S. Vaseghi, and P. McCourt, “Full covariance modelling and adaptation in sub-bands,” *Proceedings of ICASSP 2000*, vol. 2, pp. 969=–972, 2000.

[9] J.A. Bilmes, “Factored sparse inverse covariance matrices,” *Proceedings of ICASSP 00*, 2000.

[10] M.J.F. Gales, “Semi-tied covariance matrices for hidden Markov models,” *IEEE Transactions on Speech and Audio Processing*, 1999.

[11] S. Chen and R. A. Gopinath, “Gaussianization,” *Proceedings of NIPS 2000*, 2000.

[12] R. A. Gopinath, B. Ramabhadran, and S. Dharanipragada, “Factor analysis invariant to linear transformations of data,” *Proceedings of ICSLP ’98*, 1998.

[13] P. Olsen and R. Gopinath, “Modeling inverse covariance matrices by basis expansion,” *Proceedings of ICASSP 02*, 2002.

[14] S. Axelrod, R. Gopinath, and P. Olsen, “Modeling with a subspace constraint on inverse covariance matrices,” *Proceedings of ICSLP 02*, 2002.

[15] V. Vanhoucke and A. Sankar, “Mixtures of inverse covariances,” in *Proceedings of ICASSP’03 (to appear)*, 2003.

[16] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*, Kluwer Academic Publishers, 1992.

[17] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, Springer-Verlag, 2001.

[18] M. Berthold and D. Hand (Editors), *Intelligent Data Analysis, An Introduction*, Springer-Verlag, 1999.

[19] V. Digalakis, S. Tsakalidis, C. Harizakis, and L. Neumeyer, “Efficient speech recognition using subvector quantization and discrete-mixture HMM,” *Computer Speech and Language*, vol. 14, no. 1, pp. 33–46, Jan. 2000.

[20] V. Digalakis, L. Neumeyer, and M. Perakakis, “Product-code vector quantization of cepstral parameters for speech recognition over the www,” *Proceedings of ICSLP 98*, 1998.

[21] B. Mak and E. Bocchieri, “Direct training of subspace distribution clustering hidden Markov model.,” *IEEE Transactions on Speech and Audio Processing*, vol. 9, no. 4, pp. 378–387, May 2001.

[22] X. D. Huang, Y. Ariki, and M. A. Jack, *Hidden Markov Models for Speech Recognition*, Edinburgh University Press, Edinburgh, 1990.

[23] A.P. Dempster, N.M. Laird, and D.B. Rubin, “Maximum likelihood from incomplete data via the EM algorithm,” *Journal of the Royal Statistical Society*, vol. 39, no. 1, pp. 1–38, 1977.

[24] V. Digalakis and H. Murveit, “Genones: Optimizing the degree of mixture-tying in a large-vocabulary hmm-based speech recognizer,” *Proceedings of ICASSP94*, vol. I, pp. 537–540, 1994.

[25] S. Boyd and L. Vandenberghe, *Convex Optimization*, draft available on the web, http://www.stanford.edu/˜boyd/cvxbook.html, 2003.

[26] R.M. Gray, “Gauss mixtures quantization: clustering Gauss mixtures,” in *Proceedings of the Math Sciences Research Institute Workshop on Nonlinear Estimation and Classification, Mar. 17–29, 2002*, D. D. Denison, M. H. Hansen, C. C. Holmes, B. Mallick, and B. Yu, Eds., http://ee-www.stanford.edu/ gray/msri.pdf, 2003, pp. 189–212, Springer, New York, 1003.

[27] E.K.P. Chong and S.H. Zak, *An Introduction to Optimization, Second Edition*, Wiley-Interscience Series in Discrete Mathematics and Optimization. John Wiley & Sons, Inc., 2001.

[28] S. Boyd and L. El Ghaoui, “Method of centers for minimizing generalized eigenvalues,” *Linear Algebra and Applications, special issue on Numerical Linear Algebra Methods in Control, Signals and Systems*, vol. 188, pp. 63–111, 1993.

[29] J.A. Bilmes, “A gentle tutorial of the EM algorithm and its applications to parameter estimattion for Gaussian mixture and HMM,” Tech. Rep., UC Berkley, http://www.cs.ucr.edu/˜stelo/cs260/bilmes98gentle.pdf, 1998.

[30] V. Digalakis, P. Monaco, and H. Murveit, “Genones: Generalized mixture tying in continuous hidden Markov model-based speech recognizers,” *IEEE Transactions on Speech and Audio Processing*, vol. 4, no. 4, pp. 281–289, 1996.

[31] A. Sankar, “Robust HMM estimation with Gaussian merging-splitting and tied-transform HMMs,” in *Proceedings of ICSLP98*, 1998.