Abstract

Most automatic speech recognition (ASR) systems express probability densities over sequences of acoustic feature vectors using Gaussian or Gaussian-mixture hidden Markov models. In this chapter, we explore how graphical models can help describe a variety of tied (i.e., parameter shared) and regularized Gaussian mixture systems. Unlike many previous such tied systems, however, here we allow sub-portions of the Gaussians to be tied in arbitrary ways. The space of such models includes regularized, tied, and adaptive versions of mixture conditional Gaussian models and also a regularized version of maximum-likelihood linear regression (MLLR). We derive expectation-maximization (EM) update equations and explore consequences to the training algorithm under relevant variants of the equations. In particular, we find that for certain combinations of regularization and/or tying, it is no longer the case that we may achieve a closed-form analytic solution to the EM update equations. We describe, however, a generalized EM (GEM) procedure that will still increase the likelihood and has the same fixed-points as the standard EM algorithm.

Tied and Regularized Conditional Gaussian Graphical Models for Acoustic Modeling in ASR

Jeff Bilmes
University of Washington, Seattle
Department of Electrical Engineering

`$Id: bookchapter.tex,v 1.20 2006/08/08 09:17:24 jab Exp jab`

August 8, 2006

1 Introduction

In most speech recognition systems, Gaussian and Gaussian-mixture hidden Markov models (HMMs) are used to represent density functions over sequences of acoustic observation vectors. The acoustic observations most often used are Mel-frequency cepstral coefficients (MFCCs) [35] or variants thereof — MFCCs, for example, are features that are particularly well suited to Gaussian models and work quite well in general. Moreover, when training such models, the expectation-maximization (EM) update equations are well understood [15, 49, 6].

As the need for more accurate speech recognition systems has grown, ASR systems have become more complex. Correspondingly, they possess more underlying system parameters, most often in the form of more HMM states and/or more Gaussian components within each Gaussian mixture. The dimensionality of each Gaussian component may also grow. If the amount of training data does not increase correspondingly, then a parameter optimization procedure (such as EM) that operates without parameter constraints will produce a poorly trained system. The problem can be explained using a bias-variance argument [31]: the parameter variance increases as the number of parameters grows without an increase in training data. One must thus resort either to hard constrained optimization, to regularization, or to both.

In speech recognition, constrained optimization most often takes the form of parameter sharing, where *a priori* fixed decisions are made regarding which Gaussian components should share its parameters with some other Gaussian component. Often these decisions can be made based on phonetic or other high-level knowledge. For example, in a tri-phone system, the center phone might share many of the same Gaussians if the phones are similar, regardless of the context, and often the sharing is decided in a combined knowledge- and data-driven fashion [61]. Other forms of constrained optimization are possible, including partial Gaussian parameter sharing, where a sub-portion of different Gaussian components are shared [23], or sparse Gaussians [8], where Gaussian parameters are constrained to contain a certain set of zeros.

We note that parameter sharing, however, may come at a cost since learning algorithms often assume parameter independence [32, 33, 34, 19], which intuitively means that adjusting one part of a model (either structurally or parameter-wise) will not influence another part. In this case, penalized log likelihoods will decompose into separate terms which may be optimized separately. With parameter sharing, this need not be the case. Indeed, both dynamic Bayesian networks [14, 42] and probabilistic relational models [20] can be thought of as network templates and rules, where the rules explain how a template may be expanded into a larger network, and how this larger network should share its parameters among the resulting expanded factors.

Another common procedure to avoid over-training a complex model given a fixed amount of data is regularization, which is generally applicable to regression, classification, and statistical model selection. Regularization is an example of Occams razor, which roughly states that out of all possible models that can accurately explain an observable phenomenon (i.e., in our case, the data), we should prefer the simplest one possible. There are a variety of ways of measuring simplicity, including Kolmogorov complexity, the VC-dimension, entropy, minimum description length, number of total parameters, Bayesian information criterion, or a general Bayesian prior. From a pattern recognition perspective, we desire a model that achieves a low empirical error on training data while being as simple as possible.

The trade-off between these two goals is determined using a "trade-off coefficient" which determines the regularization path [30] or the "accuracy-regularization frontier." This concept is the key behind support-vector and Kernel methods [54], where the function to minimize is a sum of a (hinge) loss function and a regularizer (which is typically the ℓ_2-norm due to its mathematical tractability). Regularization is also commonly used for training neural networks (weight decay [9]), regression (i.e., ridge-regression [31]), and splines [55].

Parameter sharing can also be seen as a form of regularization [31], as the goal is to optimize the parameters of a model that has been constrained to be "simple." In the parameter sharing case, however, simplicity corresponds roughly to the number of free parameters that are available to explain the data, but these free parameters may be pieced together in arbitrarily intricate ways. In other words, more regularization corresponds essentially to more sharing, or to reducing the number of free parameters, but collections of these parameters may combine in a large variety of ways to produce the portion of a model that governs a specific phenomenon needing to be represented (e.g., an individual phone in a speech utterance). There has not, however, been much discussion in the speech recognition literature regarding general forms of regularization as it may be applied to Gaussian mixture modeling other than what has appeared in the Bayesian adaptation area (see, for example [26, 11]). Recently, the notion of regularization has been applied to produce new forms of speaker adaptation [59, 58].

In this chapter, we explore how Gaussians, Gaussian mixtures, Gaussian mixture HMMs, and auto-regressive Gaussian mixture HMMs (what we end up calling a "generic Gaussian mixture") can benefit from the simultaneous use of fine-grained sub-Gaussian component parameter tying and also parameter regularization. We utilize aspects of graphical models only to assist in our exposition. Graphical models are a visual mathematical abstraction that are used to describe certain properties of families of probability distributions [39, 36, 47]. Different types of graphical model (e.g., directed or undirected) can be used to represent factorization properties of such distributions; factorization is the key that allows the distributive law to be used so that algorithms can be developed that are both mathematically sound and algorithmically efficient. Alternatively, many have also ascribed certain high-level meanings (such as causality) to various types of graphs [48]. There have in the past been descriptions of how graphical models can be used to describe certain aspects of speech recognition systems [5]. In this chapter, we use graphical models only to assist us in describing how Gaussian mixture HMMs can be decomposed into partially tied components.

Our main contribution is to derive general forms of EM learning equations in these models. We consider such models "acoustic" since in the field of automatic speech recognition, the Gaussian is where the rubber meets the road — specifically, we consider here only the portion of the statistical model that makes a connection between the acoustic speech signal (or better, the acoustic signal after having undergone some non-probabilistic processing, such as MFCC feature extraction [35]) and the inherent discrete processing that ultimately must correspond to word string (or other high- or meta-level) hypotheses. These models include, as described above, Gaussians and their various flavors, including sparse Gaussians, auto-regressive Gaussians and what are known as "trajectory models", where a Gaussian model regresses (or conditions) on other elements in the same or other feature streams. We treat in some detail the various forms of sub-Gaussian parameter sharing (using a tri-part decomposition of a Gaussian's parameters based on Cholesky factorization) that can arise under these scenarios. We also describe the training algorithms using a regularized EM framework. In particular, we find that for certain combinations of regularization and tying, we no longer arrive at a closed-form solution to the EM update equations. We describe, however, a generalized EM (GEM) procedure that will still increase the likelihood and has the same fixed-points as the EM algorithm.

This chapter consists of the following sections: Section 2 gives a brief background on maximum likelihood training, Gaussians, and Gaussian mixtures. Section 3 derives EM update equations for training regularized Gaussians, using a three-way decomposition of the parameters of each Gaussian component. Next, Section 4 derives EM update equations for arbitrary patterns of tying, and these equations immediately lead to adaptation algorithms for mixture conditional Gaussians, and also a regularized maximum-likelihood linear regression [40] procedure. Section 5 describes various methods for training these parameters, and explain how one method is a generalized EM, and lastly Section 6 concludes. Of particular relevance to this chapter (both conceptually and in terms of useful notation) are references [6, 5, 4].

2 Background

We begin with notation. Random variables will be depicted using capital letters (e.g., X, Y and Q), and lower case variables will be used to indicate their possible values (x, y, q). Given a set of M variables, the index set $1 \ldots M$ will be depicted using the matlab-like notation $1:M$. Given a subset of variables $A = \{a_1, a_2, \ldots, a_{|A|}\}$, where

$a_i \in 1:M$, then $X_A = \{X_{a_1}, X_{a_2}, \ldots, X_{a_{|A|}}\}$.

Many of the models we describe correspond to temporal sequences of data. We have a size-N sample of training data $\mathcal{D}^{tr} = \{x^{(i)}, q^{(i)}\}_{i=1}^{N}$, where $x^{(i)} = x_{1:T_i}^{(i)}$ is a length T_i sequence of observation vectors. Each observation vector $x_t^{(i)} = \{x_{t,1}^{(i)}, x_{t,2}^{(i)}, \ldots, x_{t,M}^{(i)}\}$ will be of length M, and may correspond to a processed acoustic frame of speech (e.g., it could be a sequence of MFCCs and their deltas). The m^{th} scalar element is denoted $x_{t,m}^{(i)}$, and in general we denote the m^{th} element of vector x as $x_{\cdot,m}$. Thus, we have that $x = x_{\cdot,1:M}$. Also, $q_{1:T_i}^{(i)}$ is a length T_i sequence of integer labels or states, so that $q_t^{(i)}$ is the t^{th} integer of the i^{th} training sequence. When unambiguous, we will often use $\{x, q\}$ as a particular training pair. Each of the training pairs are assumed to be drawn from the same unknown underlying distribution, so that $(x^{(i)}, q^{(i)}) \sim p(x, q) \quad \forall i$, where $p(x, q)$ factors according to the rules that define an HMM. We will also discuss static (non-temporal) data. In this case, we will treat $\{x^{(i)}, q\}$ as a training pair, where $x^{(i)}$ is a fixed length vector. In this case, $p(x, q)$ will be a fixed length distribution. We will express conditional independence relations between random variables using the standard notation $X \perp\!\!\!\perp Y | Z$ [13] which is read "X is independent of Y given Z."

Most speech recognition systems utilize hidden Markov models (HMMs) with mixture density functions for their observation distributions. Specifically, an HMM is a probability distribution over $2T$ variables — there are T state variables $Q_{1:T}$ that are often hidden, and T other variables that are almost always observed $X_{1:T}$. The joint distribution over these $2T$ variables factorizes as follows:

$$p(x_{1:T}, q_{1:T}) = \prod_t p(x_t|q_t)p(q_t|q_{t-1}).$$

Here, $p(q_t|q_{t-1})$ gives the probability of moving from state q_{t-1} at time $t-1$ to state q_t at time t. In this work, we will be discussing the observation distribution, $p(x_t|q_t)$ as that is where the acoustic information is typically produced as input into the HMM at time t. For example, with x_t observed, it is often the case that x_t consists of an M-dimensional MFCC feature vector for time t, although many other types feature vectors can be used as well.

In this chapter, x_t will be an continuous valued M-dimensional vector with density $p(x_t|q_t)$. This density will be a mixture model, where components of the mixture consist of Gaussian densities of some form. For example:

$$p(x_t|q_t) = \sum_\ell \alpha_{\ell q_t} \mathcal{N}(x_t; \mu_{\ell q_t}, \Sigma_{\ell q_t}),$$

where $\mathcal{N}(x; \mu, \Sigma)$ indicates that x has an M-dimensional Gaussian density, and where $\alpha_{\ell q_t}$ is the mixture component weight. An M-dimensional Gaussian density has the form

$$p(x) = \mathcal{N}(x; \mu, \Sigma) = |2\pi\Sigma|^{-1/2} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)},$$

where μ is an M-dimensional mean vector, and Σ is an $M \times M$ covariance matrix. We will use $K = \Sigma^{-1}$ to refer to the inverse covariance (or the concentration) matrix of the density.

It is possible to view a Gaussian density as either a Bayesian network [47] or as a Markov random field — the property of Gaussians being represented by various forms of graphical model were described in [5], and are very briefly summarized here. First, zeros in the covariance matrix correspond to marginal independence assumptions. For example, given the constraint $\Sigma_{ij} = 0$ for a particular i, j of the concentration matrix, then in the marginal we have $p(x_i, x_j) = p(x_i)p(x_j)$ or that $X_i \perp\!\!\!\perp X_j$. This property of Gaussians is quite well known. Second, zeros in the concentration matrix correspond to conditional independence assumptions. In particular, if $K_{ij} = 0$ then $X_i \perp\!\!\!\perp X_j | X_{\{1:M\} \setminus \{i,j\}}$. In other words, zeros in the concentration matrix correspond to the pairwise Markov property of undirected graphical models [39]. Since the Gaussian is always positive, the Hammersley-Clifford theorem [39] applies and so also do all the standard Markov properties on undirected graphical models, and we can view a Gaussian as an undirected graph on a set of nodes where there are edges between two nodes i and j if and only if $K_{ij} \neq 0$.

A Gaussian may also be described by a Bayesian network. Assuming a particular ordering of the elements in the variable X, if we Cholesky factorize [27, 29] the concentration matrix $K = U'DU$ where U is an upper triangular matrix with unity on the diagonal, and D is a diagonal matrix, then form $U = (I - B)$, we see that B is an upper triangular matrix with zeros on the diagonal, and the upper off-diagonal elements correspond to the edges in a Bayesian network that describes the Gaussian. For example, if $B_{ij} = 0$ with $j > i$, then in the Bayesian network form, there is no edge from node j to node i. With this form of Gaussian (one that we will use extensively in this chapter), the

Gaussian distribution becomes [5]:

$$p(x) = (2\pi)^{-M/2}|D|^{1/2}e^{-\frac{1}{2}(x-Bx-\tilde{\mu})^T D(x-Bx-\tilde{\mu})}.$$

We may be interested for example in training a Gaussian model where K is constrained to have a certain pattern of zeros. In fact, *Sparse Gaussians* [8] are produced by learning a Gaussian distribution that factors with respect to a particular graph. The ubiquitously utilized and easy to train diagonal covariance matrix, for example, simply corresponds to using a completely disconnected graph. Other forms of constrained Gaussians can be formed simply by utilizing a different graph (i.e., various constraints on either the covariance, concentration, or Cholesky factorized concentration matrices). But as we will see in subsequent sections, regardless of the pattern of zeros in the Cholesky factorized form, the Gaussian will be easy to estimate in a maximum likelihood sense as long as all variables are observed.

A conditional (or autoregressive) Gaussian process is essentially a type of conditional probability density. Suppose that the vector x has been bi-partitioned into two sets U, V such that $U, V \subset 1 : M$ and $U \cap V = \emptyset$. Then $p(x_U|x_V)$ is a conditional Gaussian where the mean vector x_U is a linear function of the parent variable vector x_V, but otherwise X_U is a Gaussian. In other words x_U has a different Gaussian distribution for each value x_V. This is expressed as

$$p(x_U|x_V) = |2\pi\Sigma_{U|V}|^{-1/2}e^{-\frac{1}{2}(x_U-(B_{U|V}x_V+\mu_{U|V}))^T\Sigma_{U|V}^{-1}(x_U-(B_{U|V}x_V+\mu_{U|V}))},$$

where $\Sigma_{U|V}$ is the conditional covariance matrix, which does not change with varying x_V. The conditional mean, however, does change with x_V as the affine transformation $B_{U|V}x_V + \mu_{U|V}$ is the conditional mean of x_U given x_V. This general form can arise in a number of ways. For example, after factoring a Gaussian using the chain rule of probability

$$p(x_{\cdot,1:M}) = \prod_m p(x_{\cdot,m}|x_{\cdot,1:m-1}),$$

then $p(x_{\cdot,m}|x_{\cdot,1:m-1})$ corresponds to a conditional Gaussian (in this case $U = \{m\}$ and $V = \{1, 2, \ldots, m-1\}$), and in fact the parameters associated with each factor combine together to produce the Cholesky factorization of the concentration matrix K [3].

Using conditional Gaussians, either normal Gaussian distributions or mixtures thereof can be formed over a speech observation at time t, to form $p(x_t)$. Moreover, trajectory type models can be formed over time, where the conditional variables at time t can come from the past, future, present, or all of the above relative to t. The conditional variables might even come from an entirely separate feature stream. For example, linear mixture buried Markov models (BMMs) [7, 3] can be formed by employing a mixture of conditional Gaussian distributions, where in this case we equate x_U with x_t, so x_U becomes the feature vector at the current time, and we also equate x_V with z_t, which is an L-dimensional vector of feature elements from previous times, future times, or as mentioned above even different feature streams (that may come from any time). We can denote this using sets, as $z_t \subset \{x_{t-\tau_{11}}, y_{t-\tau_{12}}, \ldots, x_{t-1}, y_{t-1}, y_t, x_{t+1}, y_{t+1}, \ldots, x_{t+\tau_{21}}, y_{t+\tau_{22}}\}$ where y_t might be a different feature stream altogether, and where τ_{ij} are arbitrary fixed integer constants that indicate how much of either the x or y stream are part of z_t. Much has been written on how conditional Gaussians and their mixtures can be used as trajectory models (examples include [50, 5]). In the most general case, z_t may also contain a constant element of unity (1) so that both the regression coefficients and the fixed Gaussian mean can all be represented by the single $M \times (L+1)$ matrix B — in other words, if we have that $z_t = [z_t' 1]^T$ indicating that the very last element of z_t is unity, then $Bz_t = B'z_t' + \mu$ where μ is the last column of B, and B' is an $M \times L$ matrix consisting of all but the last column of B. This produces a generalized conditional Gaussian of the form:

$$p(x|z) = (2\pi)^{-d/2}|D|^{1/2}e^{-\frac{1}{2}(x-Bz)^T D(x-Bz)},$$

where we assume that everything that might possibly be conditioned on is included in the large length L vector z. We note that the diagonal matrix D corresponds to the inverse conditional variance parameters of the Gaussian (in this chapter, we sometimes refer to D as just the conditional rather than the inverse conditional variances when the meaning is clear).

In general, regardless of how we form a Gaussian (either as a conditional Gaussian $p(x|z)$ or as part of a standard Gaussian that has been factored using the chain rule), we will define a set of parameters of a Gaussian as the following triple: 1) the mean μ; 2) the (conditional) variance D; and 3) the coefficients of B. The matrix B could come from the Cholesky factorized concentration matrix $K = (I - B)^T D(I - B)$, or could also be the regression coefficients on some conditional vector z, or it could come from both; we will not make a distinction between the two since they can be treated identically in the learning equations. Note, however, that in many cases B will be quite sparse — a particular sparsity pattern, then, can be viewed as a form of constrained optimization or itself a form of regularization, where fewer parameters are available in the model to adjust thus reducing parameter estimation variance. This decomposition of the Gaussian's parameters into these three portions (the mean, inverse conditional variance, and regression coefficients) will be the basis for our Gaussian parameter regularization and tying strategies described in Section 4. This treatment of the Gaussian parameters will moreover allow us to represent various training procedures for Gaussians and Gaussian mixtures in a unified way, regardless if the model consists of regular Gaussian components, various sparse Gaussians, or the conditional Gaussians used for temporal trajectory modeling (say in an HMM) or in a multi-stream model.

2.1 Maximum Likelihood Training

Given a set of training data, there are many ways to optimally adjust the parameters to be in best accordance with this data, including maximum likelihood, maximum a-posteriori (MAP), and discriminative parameter training procedures [1, 17, 16, 38, 37, 31, 57, 52]. While discriminative parameter training is most appropriate for classification tasks, maximum likelihood (ML) training is still commonly used and can sometimes perform as well or better than discriminative methods, often due to the mathematical or computational simplicity of such techniques, or to statistical estimation issues [45]. Moreover, it is often extremely easy to train a generative model in the ML-framework due to the fact that the ML objective function decomposes exactly as does the generative model (e.g., an HMM). In fact, when all variables in question X and Q are observed, then unconstrained ML training becomes quite easy, namely $p(x,q) = p(x|q)p(q)$ can often be learned simply by counting and averaging.

In the models that we consider, namely $p(x,q)$ we will assume that X is observed and Q hidden when training. The most common form of training in this situation is to use the EM algorithm [15, 6, 43]. We very briefly outline this procedure in its basic form. Assuming that all of the parameters of the model are described by the variable λ, we start with an initial estimate of the parameters, say λ^p. The goal is to optimize the next set of parameters λ using an augmented function,

$$f(\lambda) = Q(\lambda, \lambda^p) = E_{p(Q|X,\lambda^p)}[\log p(X,Q|\lambda)] = \sum_q p(q|x,\lambda^p) \log p(X,Q|\lambda).$$

This is to be compared to the "incomplete" likelihood function, consisting only of the observed variables, namely $g(\lambda) = \log p(x|\lambda)$. It can be shown that $\partial f(\lambda)/\partial\lambda|_{\lambda=\lambda^p} = \partial g(\lambda)/\partial\lambda|_{\lambda=\lambda^p}$ so the gradients of both the incomplete and the augmented likelihood functions are the same at the location of the current parameters. Moreover, it can be shown that $f(\lambda)$ is a concave lower bound on the true likelihood function at the current point, and that optimizing $f(\lambda)$ is often much easier than optimizing $g(\lambda)$, and therefore is typically preferred. The process is often as simple as finding the unique peak of $f(\lambda)$ by taking a first derivative, setting it to zero, and then solving for λ (possibly with some Lagrangian constraints to ensure we stay within the parameters' probability simplex). This process is repeated. There are also guaranteed convergence properties — once converged, we are guaranteed to have found a local maximum of the likelihood function $g(\lambda)$, which follows from the above since the derivatives are identical.

Details of the derivation of the EM update equations for Gaussian mixtures, and HMMs using Gaussian mixtures are described in [15, 43], and notation most similar to what we use in this chapter is described in [6]. In an HMM system, in order to train a Gaussian mixture HMM, one needs the quantities $p(Q_t = i, M_t = j|x_{1:T}, \lambda^p)$, the posterior probability that the Markov chain state at time t is i and the mixture component of state i at time t is j. For a plain Gaussian mixture system, the posterior needed is $p(M_t = j|x_t, \lambda^p)$, which is the posterior of the component for data instance x_t. There is a strong commonality between Gaussian mixtures and Gaussian mixture HMMs, and in fact the only difference in the EM update equations is the form of this posterior. In fact, a Gaussian mixture HMM may be seen as a high dimensional multi-variate Gaussian, specifically $p(x_{1:T}|m_{1:T}, q_{1:T})$ is just an MT-dimensional Gaussian with a block-diagonal covariance matrix. Therefore, in accordance with our unified treatment of Gaussian models as described in the previous section, we will assume what we call a "generic Gaussian mixture", where we use a simple notation p_{mt} to designate the posterior probability under the previous parameters of the current component m for time or sample t, and we will do this using the same notation. As we will see, this will greatly simplify our equations. The general scenario is shown in Figure 1. Lastly, we will let $\mathcal{Q}$ be the general auxiliary function to be optimized in EM. For the purposes of this manuscript, it is also sufficient to provide a $\mathcal{Q}$ only for a generic Gaussian mixture.

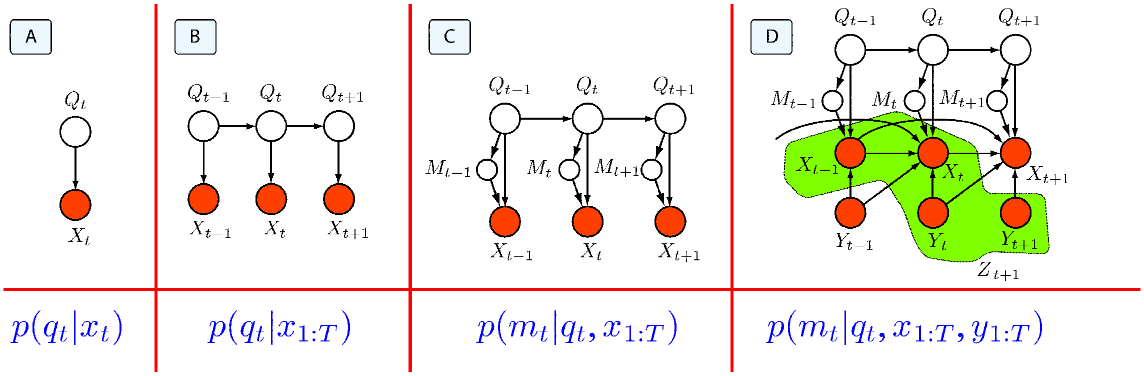

Figure 1: A variety of Gaussian-like models utilize the same form of posterior probability of the hidden variables ($q_{1:T}$ and $m_{1:T}$) given the evidence ($x_{1:T}$ or $y_{1:T}$). For each model, the posteriors are needed when training using regularized EM, and the equations for training the different models are identical up to the form of this posterior. In part A (on the left of the figure), we have a simple mixture model, with posterior form $p(q_t|x_t)$. In part B, we have the basic Gaussian HMM, and its posterior. If we use a Gaussian mixture HMM (part C), we need the posterior of the component m_t given the current hidden state q_t and the evidence. And last, in part D, we have a multi-stream Gaussian mixture auto-regressive (or trajectory) model. The main stream is $X_{1:T}$ with auxiliary stream $Y_{1:T}$, but the form of posterior needed is again still the same. In all cases, a forward-backward procedure [2] can be defined that can compute these posteriors. For our analysis, we will simply use p_{mt} to indicate the posterior of Gaussian component m at data sample (or time) t. Lastly, the part D of figure also shows that Z_t is a vector consisting of elements from X_{t-1}, X_t, Y_t, and Y_{t+1}. Our equations will use the generic notation z_t to indicate this vector, and to produce the other models it is sufficient to realize that z_t may be empty.

3 Training Regularized Gaussian Decompositions

In this section, the decomposition of a Gaussian component's parameters, as mentioned in the previous section, into the three groups, μ, D, and B, is applied during EM training of a generic Gaussian mixture. Unlike Section 4, in this section we do not yet assume there is any sub-Gaussian parameter sharing.

A desirable property of any EM update equation is that each parameter group at each EM update can be estimated independently without needing any of the other new parameters. In other words, given a collection of previous parameters $\{\mu_m^p, D_m^p, B_m^p\}_m$, then ideally the update equations of each of the new parameters $\{\mu_m, D_m, B_m\}_m$ is expressed entirely in terms of only the data and the previous parameters. Otherwise, either multiple passes through the data might be required thereby making training much more computationally expensive, or even worse there might be cyclic dependencies in the parameter updates (which we ultimately will need to deal with in this chapter as well). A simple example may be demonstrated using the estimation of the mean $\mu = \sum_t x_t$ and variance $\sigma^2 = \sum_t (x_t - \mu)^2$, where we see that the variance depends on the mean. It is well known that the variance estimate can be reformulated using additional accumulators $\sigma^2 = \sum_t x_t^2 - \left(\sum_t x_t\right)^2$, requiring only one pass through the data for both mean and variance, in this latter case accumulating both the sum and sum of squares of x_t. For certain patterns of sharing and regularization of a Gaussian's parameters, however, this will not always be possible. In any case, the general property as it will apply to our EM updates is depicted in Figure 2.

When there is no parameter sharing, the EM auxiliary function needing to be maximized becomes

$$\mathcal{Q} = \sum_m \sum_t \left(\frac{1}{2} \log(|D_m|) - \frac{1}{2}(x_t - B_m z_t)^T D_m (x_t - B_m z_t) \right) p_{mt} - \frac{\lambda}{2} \sum_m ||B_m||_F^2, \tag{1}$$

where $||A||_F = \text{tr}(AA^T)$ is the Frobenius norm of matrix A, the last term acts as an ℓ_2 regularizer on the regression coefficients, and λ is an accuracy-regularization trade-off coefficient. This form of regularization is similar to ridge-regression [31] and to weight-decay [9] in the neural-networks literature. In our case, however, we assume that there is only one such global accuracy-regularization λ coefficient for all the Gaussians in the current model, and for all scalar elements within each component. An alternative approach would employ one coefficient for each separate Gaussian component, or even one coefficient for each individual regression coefficient. In general, however, the value of such

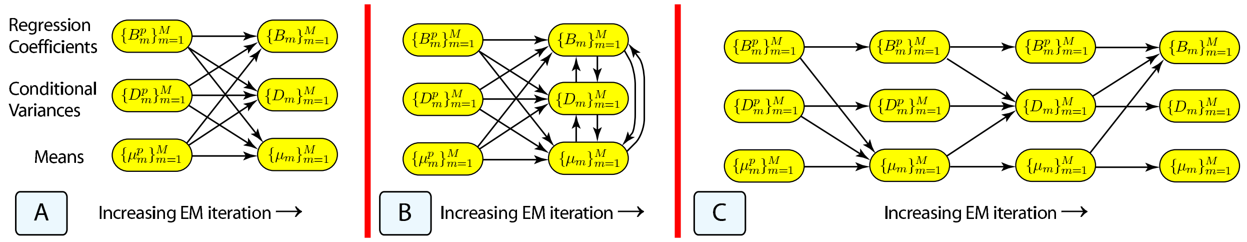

Figure 2: EM Gaussian decomposition parameter update dependencies: On the left (A), we see that each of the new parameters μ_m, B_m and D_m depend only on previous parameters μ^p_m, B^p_m and D^p_m and also on the training data set (not shown in the figure). In this case, it is possible to simultaneously update all new parameters, with only one pass through the training data. In the middle (B), the new parameters might depend on the other new parameters as well as old parameters, leading potentially to cyclic dependencies. On the right (C), we break the cycles by only updating one parameter group for each EM iteration, where each EM iteration requires a single pass through the data. This means that to update all the parameters in the model, three passes through the data are required. Still another strategy (middle figure, or B) updates all parameters simultaneously with one training data pass, but in place of each new parameter that is needed, we use the parameter from the previous EM iteration — this last step is not guaranteed to increase the likelihood of the data, although it does have the same fixed points as the left case, and may work well in practice.

coefficients can only be found using cross-validation [31] on a development or held-out data set and doing this when there are multiple accuracy-regularization coefficients would turn the problem of finding these trade-off coefficients themselves into an intractable search problem.

Equation 1 does not include any constant terms which vanish once derivatives are taken. Also note, for now we have assumed that the mean parameters are contained within the B matrix — we do this by assuming that one of the elements of z_t is always equal unity. This means also that we are regularizing in the above equation both the regression coefficients (e.g., cross-observation dependences as in a BMM) but also the means. We might not wish to regularize the means, however, so in an expansion where the means and regression coefficients are separate (see below), we might not have a regularizer on the means (we utilize separate mean and regression regularization coefficients in Section 4). Also, since there is no sharing, there is a one-to-one correspondence between every value of m and each of the D and B matrices.

Maximizing the regularizer alone $-\frac{\lambda}{2}\sum_m ||B_m||_F^2$ will force the coefficients to converge towards zero. Therefore, it is important to assume that the training data is centered, i.e., zero-mean normalized. Also, the coefficients will depend on the scaling of the data as well. Therefore, in practice, it will be important to perform both mean subtraction and variance normalization [41, 10] on the observation vectors prior to any training.

Taking derivatives of Equation 1 with respect to the B_m matrices (again, which includes the mean in the right most column of B_m), gives $\frac{\partial \mathcal{Q}}{\partial B_m} = 0$ or that

$$\sum_t D_m(x_t - B_m z_t) z_t^T p_{mt} - \lambda B_m = 0.$$

From this, we must attempt to solve for B_m, and there are several ways to proceed. One simple way next yields

$$\sum_t (x_t - B_m z_t) z_t^T p_{mt} - \lambda D_m^{-1} B_m = 0, \tag{2}$$

implying that

$$\sum_t x_t z_t^T p_{mt} - B_m \sum_t z_t z_t^T p_{mt} - \lambda D_m^{-1} B_m = 0.$$

We see that we have a matrix equation of the form $A - BC - DB = 0$ where $A = \sum_t x_t z_t^T p_{mt}$, $B = B_m$, $C = \sum_t z_t z_t^T p_{mt}$, $D = \lambda D_m^{-1}$, and the goal is to solve for B. This equation is known as the Sylvester matrix equation [25] (which is a generalization of the more well known Lyapunov equation [21] commonly used in control theory). The Sylvester equation has a unique solution if and only if there is no eigenvalue of C that is the negative of any eigenvalue of D. Therefore, since both C and D are positive definite in our case, this condition will certainly hold.

A general (but not efficient) form of solution is given by $\bar{b} = G^{-1}\bar{a}$, where $\bar{b}$ (respectively $\bar{a}$) is a vector consisting of the columns of B (resp. $-A$) stacked on top of each other, and where $G = I \otimes D + C^T \otimes I$ where $\otimes$ is the Kronecker matrix product. Much more efficient solutions are given in [25]. Also, since D is diagonal, significant additional computational efficiencies can be obtained (see Section 3.2 below).

If $\lambda = 0$, however, we have a particularly simple form, that yields:

$$B_m = \sum_t x_t z_t^T p_{mt} \left(\sum_t z_t z_t^T p_{mt} \right)^{-1}. \tag{3}$$

So if $\lambda = 0$ (no regularization), the above shows that B_m can be estimated quite easily by summing posterior weighted outer products, and that B_m does not depend on any other current parameter. This assumes of course that T is large enough so that the matrix above is invertible, but that is typical for most data sets.

With regularization $\lambda > 0$, however, the new B_m has a much more intricate relationship with the new D_m matrix, as illustrated in the middle of Figure 2.

Next, taking derivatives of Equation 1 with respect to the diagonal matrix D_m and setting $\frac{\partial \mathcal{Q}}{\partial D_m} = 0$ gives

$$\sum_t p_{mt} \left(D_m^{-1} - (x_t - B_m z_t)(x_t - B_m z_t)^T \right) = 0,$$

or that

$$D_m^{-1} = \frac{\sum_t p_{mt}(x_t - B_m z_t)(x_t - B_m z_t)^T}{\sum_t p_{mt}}.$$

In this update, we again have a cross-dependency, namely D_m is dependent on the current estimation of B_m. In this case, therefore, it appears that even if $\lambda = 0$, we have a dependence between the parameters, thus again implying multiple passes through the data on each EM iteration, first to compute B_m and next to compute D_m. The following formula, however, shows that if $\lambda = 0$, we can compute the covariance matrix using the quantities that we have and without running through the data multiple times per iteration.

$$\begin{aligned} D_m^{-1} &\propto \sum_t \left\{ p_{mt} \left(x_t x_t^T - x_t z_t^T B_m^T - B_m z_t x_t^T + B_m z_t z_t^T B_m^T \right) \right\} \\ &= \sum_t p_{mt} x_t x_t^T - \left(\sum_t p_{mt} x_t z_t^T \right) B_m^T - B_m \sum_t p_{mt} z_t x_t^T + B_m \left(\sum_t p_{mt} z_t z_t^T \right) B_m^T, \end{aligned}$$

but when $\lambda = 0$, we have

$$B_m \left(\sum_t p_{mt} z_t z_t^T \right) B_m^T = \left(\sum_t x_t z_t^T p_{mt} \right) B_m^T,$$

and therefore we get

$$D_m^{-1} \propto \sum_t p_{mt} x_t x_t^T - B_m \sum_t p_{mt} z_t x_t^T. \tag{4}$$

The equations above show that B_m and D_m can indeed be learnt simultaneously when $\lambda = 0$ without needing to make multiple passes over the data. Specifically, with only a single pass over the data, we can accumulate the three quantities $\sum_t x_t z_t^T p_{mt}$, $\sum_t z_t z_t^T p_{mt}^{-1}$, and $\sum_t p_{mt} z_t x_t^T$, which when done are sufficient for both the B_m and D_m updates in Equations 3 and 4.

Even though $\lambda = 0$ in this case, it still might appear that we have a problem in that the above equations require outer products of the form $z_t z_t^T$ and $x_t z_t^T$ for every m. Considering that there may be many hundreds or thousands of Gaussian components in a modern ASR system (corresponding to the different HMM states and mixture components therein), and that z_t could easily have many elements, these accumulators could be quite large. For example, in a BMM update equation [3], z_t consists of the collection of *all* context elements grouped together for all individual Gaussian elements, a vector that could be thousands of elements long. When $\lambda > 0$, solving the Sylvester equation would be prohibitive. A solution to this problem is given in Section 3.2, but we first discuss the relationship these models have with ridge regression.

3.1 Ridge Regression

Our procedure may in fact be seen as a multi-variate, multi-class, “soft”, and learned-covariance generalization of, ridge regression [31]. It is multivariate since we are essentially learning multiple sets of regression coefficients simultaneously. It is multi-class since the regression coefficients may correspond to entirely different Gaussians. It is “soft”, since the coefficients are being learned as part of a larger EM procedure, so all data is being essentially weighted by posterior probabilities from previous model parameters. And lastly, unlike conventional ridge regression, we must learn covariance parameters as well. Since the relation with ridge regression is important to understand, we review ridge regression in some detail, and then expand on its generalization.

Ridge regression can be explained from either a frequentist perspective (a Gaussian model with a penalized log-likelihood) or a Bayesian perspective (the maximum *a posteriori* (MAP) of β under a Gaussian model and a Gaussian prior). In either case, the goal is to learn the parameters of a linear model for predicting the scalar variable x based on a vector variable z and a vector of parameters β of the same length. We are given training samples $\mathcal{D}^{tr} = \{x^{(i)}, z^{(i)}\}_{i=1}^{N}$ from which to learn β. In the Bayesian case, we find the β that maximizes the posterior $p(\beta|\mathcal{D}^{tr})$ or equivalently that maximizes $p(\mathcal{D}^{tr}|\beta)p(\beta)$. The likelihood $p(\mathcal{D}^{tr}|\beta)$ is composed of the model $p(x|z,\beta) = \mathcal{N}(x; \beta^T z, \sigma^2)$, a conditional Gaussian with mean $\beta^T z$ and variance σ^2. As before, we assume that the last entry of z is always unity, so that we can absorb any mean parameter into the regression parameters β. We also assume a zero-mean and η^2-variance Gaussian Bayesian prior on β, meaning that $\beta \sim \mathcal{N}(0, \eta^2)$ where $p_\eta(\beta) = \mathcal{N}(0, \eta^2) = \frac{1}{\sqrt{2\pi\eta^2}} \exp\left(-\frac{1}{2}\beta^T\beta/\eta^2\right)$. The joint data and β distribution consisting of log-likelihood and prior can be written as

$$\begin{aligned} L(\beta) &= \sum_i \log p(x^{(i)}|z^{(i)}, \beta) + \log p_\eta(\beta) && (5) \\ &= -\frac{N}{2}\log(2\pi\sigma^2) - \frac{1}{2}\sum_i (x^{(i)} - z^{(i)}\beta)^2/\sigma^2 - \frac{1}{2}\log(2\pi\eta^2) - \frac{1}{2\eta^2}||\beta||^2, && (6) \end{aligned}$$

where $||\beta||$ is the ℓ_2-norm. Finding the maximum a-posteriori (MAP) value of β reduces to:

$$\operatorname*{argmin}_\beta \sum_i \frac{1}{2\sigma^2}(x^{(i)} - z^{(i)}\beta)^2 + \frac{1}{2\eta^2}||\beta||^2 = \operatorname*{argmin}_\beta \sum_i \frac{1}{2}(x^{(i)} - z^{(i)}\beta)^2 + \frac{\sigma^2}{2\eta^2}||\beta||^2.$$

If we set $\lambda = \sigma^2/\eta^2$, we get the typical form of ridge regression:

$$\operatorname*{argmin}_\beta \sum_i \frac{1}{2}(x^{(i)} - z^{(i)}\beta)^2 + \frac{\lambda}{2}||\beta||^2. \tag{7}$$

This latest form is in fact the standard frequentist starting point for ridge regression, where we have an error term (or a likelihood) along with a complexity penalty that are related by an accuracy-regularization trade-off coefficient λ.

To increase the degree of regularization, we increase λ, and as $\lambda \to \infty$, $\eta^2 \to 0$, which also takes β towards zero for any finite-length data set. λ thus gives us a single normalized parameter to determine the amount of regularization. We note that the variance of the Bayesian prior changes as a function of the data variance since $\eta^2 = \sigma^2/\lambda$. If we wanted to use a fixed Bayesian prior regardless of the data variance, ridge regression would take the form:

$$\operatorname*{argmin}_\beta \sum_i \frac{1}{2}(x^{(i)} - z^{(i)}\beta)^2 + \frac{\lambda\sigma^2}{2}||\beta||^2, \tag{8}$$

where in this case, we have an interpretation of our regularization coefficient as $\lambda = 1/\eta^2$, the inverse variance of the Bayesian prior. Of course, the interpretation of λ in Equation 7 or Equation 8 does not matter here since the regularization coefficient is usually determined using cross-validation for any given data set [31]. Note, however, that the range of values attempted for cross-validation would matter, as in the former case λ is a ratio of variances, while in the latter case λ is an inverse variance.

Suppose now that we wish to solve multiple ridge-regressions simultaneously. We can think of this as multiple separate data sets, or alternatively, we can view this as a single multivariate generalization of ridge regression, where x is now a M-dimensional vector, B is an $M \times L$ matrix of regression coefficients, where each row of B is used to predict the corresponding element of x. In either case, when we wish to use a single λ regression coefficient to control such multiple regressions simultaneously, then the interpretation of λ might start to matter. If we want one global Bayesian prior variance to apply to all regressions simultaneously, then the form of regularization should be as we have defined it in Equation 2 (for a specific value of m) where each of the variance terms are multiplied by lambda as in λD_m^{-1}. In this case, we interpret λ as the inverse of this globally shared Bayesian prior variance. If, on the other hand, we wish the Bayesian prior for each row of B_m to scale with the variance of the corresponding element in x, then Equation 2 should take the form:

$$\sum_t (x_t - B_m z_t) z_t^T p_{mt} - \lambda I B_m = 0, \tag{9}$$

where I is an appropriately sized identity matrix, in which case B_m has once again a particularly simple update equation, namely equation 3 becomes:

$$B_m = \sum_t x_t z_t^T p_{mt} \left(\sum_t z_t z_t^T p_{mt} + \lambda I \right)^{-1}. \tag{10}$$

We see that as λ gets larger, the matrix that must be inverted becomes better conditioned, and it moves B towards the zero matrix.

For the remainder of this chapter, we will assume the globally shared Bayesian prior case, but we must keep in mind that some of the equations will change if we take the interpretation shown in Equation 10.

3.2 Exploiting that D is diagonal

Significant computational simplifications can be achieved by noting that since D_m is a diagonal matrix, only the diagonal portion of the update equations need to be computed. This requires no loss of generality because of the UDU' factorization mentioned above, whereby the B matrices may contain the factored form of a full covariance matrix, or a generalized regression on z_t, or both.

We let $D_m(r)$ be the r^{th} diagonal scalar element of the diagonal matrix D_m. Also, let $B_m(r)$ be a row-vector consisting of the non-zero (i.e., existing) elements from the r^{th} row of the matrix B_m, and let $z_t(r)$ be a column vector consisting of the elements of the vector z_t that correspond to the non-zero entries in the r^{th} row of B_m. In other words, the scalar quantity $B_m(r)z_t(r)$ equals the r^{th} element of the vector $B_m z_t$ but the computation of $B_m(r)z_t(r)$ does not include the terms that are just zero. This is a slight abuse of notation as the elements contained in $z_t(r)$ depend on the B_m matrix that it is currently being multiplied with, but the meaning should be clear nevertheless. The reason this works at all is the way we formed z_t as being a subset of all possible elements that x_t might condition on lumped together as one large vector — it has been the sparsity pattern of the B_m matrices which has not only included the appropriate regression coefficients, but has also acted as a selector for appropriate elements within z_t. Note also that for $r \neq r'$, $z_t(r)$ and $z_t(r')$ might contain some of the same elements from z_t, i.e., treating $z_t(r)$ as a set rather than a vector, it is possible to have $z_t(r) \cap z_t(r') \neq \emptyset$ Finally, we let $x_t(r)$ be the r^{th} scalar element of the vector x_t. Using these definitions, we may re-write Equation 1 as:

$$\mathcal{Q} = \sum_m \sum_t \sum_r \left(\frac{1}{2} \log(D_m(r)) - \frac{1}{2} \big(x_t(r) - B_m(r) z_t(r)\big)^2 D_m(r) \right) p_{mt} - \frac{\lambda}{2} \sum_m \sum_r ||B_m(r)||^2. \tag{11}$$

It follows immediately that the update equations for the diagonal entries of the covariance matrices are:

$$\frac{1}{D_m(r)} = \frac{\sum_{t=1}^T p_{mt} \Big(x_t(r) - B_m(r) z_t(r) \Big)^2}{\sum_{t=1}^T p_{mt}} = \frac{\sum_{t=1}^T p_{mt} [x_t(r)]^2 - B_m(r) \sum_{t=1}^T p_{mt} x_t(r) z_t(r)}{\sum_{t=1}^T p_{mt}}, \tag{12}$$

where the second equality follows only when $\lambda = 0$, similar to Equation 4. Note that this is a scalar update equation for each of the diagonal elements of the matrix D_m. Depending on the sparse patterns of the matrices, the cost of the dot products $B_m(r)z_t(r)$ will be significantly smaller than before.

The update equations for the B_m matrices can also be significantly simplified. Since D_m is diagonal, the derivative of Equation 1 with respect to the r^{th} row of B_m is uncoupled with the other rows. Also, only the non-zero portions of the row are computed (recall that B_m can be quite sparse so this can be a significant savings). Taking the derivative of Equation 11 with respect to $B_m(r)$ thus yields:

$$\sum_{t=1}^T D_m(r) \big(x_t(r) - B_m(r) z_t(r) \big) z_t(r)^T p_{mt} - \lambda B_m(r) = 0.$$

This leads to the independent update equations for the non-zero portion of each row of the B matrix as follows:

$$B_m(r) = \left(\sum_{t=1}^{T} p_{mt} x_t(r) z_t(r)^T \right) \left(\sum_{t=1}^{T} p_{mt} z_t(r) z_t(r)^T + \frac{\lambda}{D_m(r)} I \right)^{-1},$$

where I is an appropriately sized identity matrix. In this case, only the outer products $z_t(r)z_t(r)^T$ and the SAXPY [27] operations $x_t(r)z_t(r)^T$ are required, yielding a significant reduction in both computational complexity and memory requirements. Also, unlike in the general case above, we see that we get a solution without needing to resort to algorithms that solve the Sylvester equation, regardless of the interpretation or value of λ. We see also again that adding the positive definite diagonal matrix $\lambda/D_m(r)I$ in the above ensures that the matrix is well conditioned (so inversion is possible). And as mentioned earlier, unlike typical ridge regression, we take here the interpretation of one global Bayesian prior variance in order to use a global λ parameter, which means we must appropriately scale by the variance $D_m(r)$.

All is not perfect, however, as when $\lambda > 0$, it appears that we once again obtain cycles in the parameter update dependencies, as in the center of Figure 2. Specifically, the equations show that both $B_m(r)$ depends on $D_m(r)$ and $D_m(r)$ depends on $B_m(r)$. Once again, we can break the cycle by using the previous parameters in place of current parameters, and make multiple passes through the data, for each pass updating only one parameter group at a time, as shown on the right in Figure 2. Thus, in each step we are still guaranteed to improve the likelihood, although convergence may not be as fast. Moreover, we clearly have the same fixed points, as when at a particular fixed point, we have that $D^p_m(r) = D_m(r)$. We can also (Figure 2 middle) attempt to update all parameters simultaneously, but such a procedure does not guarantee in all cases an improvement in likelihood on each EM iteration.

4 Training Shared Regularized Gaussian Decompositions

Normally, and as described in the previous section, each individual Gaussian component has its own unique set of parameters not shared with any other component in the model. In this section, we describe a model where some set of the Gaussian components are constrained to share a portion of their parameters with other components in the model. We wish moreover to analyze the EM update equations under such constraints. This procedure is often called parameter tying, but here the granularity of tying is somewhat finer than what is typically done in a speech recognition system. Specifically, an entire Gaussian component is commonly shared between two mixtures of Gaussians, or alternatively an entire mixture for an HMM state might be shared between two multi-state phone models in an HMM-based system [60]. Here, by contrast, each component Gaussian might share any of its μ, D, and B parameters with other components in the system. Moreover, a given component might share different groups with different other components. For example, a component's μ parameter might be shared differently than the component's D parameter. This leads to an enormous variety of parameter sharing patterns. There have been a number of other models that use partially shared Gaussian parameters, including [28, 23, 46], but this is not done in the same way described herein. In particular, we will see that Cholesky factorized covariance matrices allows us to easily combine sharing and other forms of regularization.

It is crucial to realize that sharing is a form of constrained optimization — we still seek a maximum likelihood solution, but now we are constraining the parameters to live in a space where sub-portions of Gaussian parameters are forced to be identical. It is well known that constrained optimization can be more difficult (either computationally or mathematically) than unconstrained optimization [18], and we will see that our case is no exception. We will find, however, that the update rules are still fairly simple if we use the idea we saw in the previous section of breaking cycles over EM iterations, as it then is possible to find a solution without major modifications to the previously derived equations.

In the most general case of parameter sharing, we have three functions that map from the index m of a general class to the index k of a specific extant mean, diagonal covariance, and regression coefficient used by that index. Those mappings are, respectively, $T_\mu(m)$, $T_D(m)$, and $T_B(m)$. In other words, we have that $T_\mu(m) : \{1, \ldots, M\} \rightarrow \{1, \ldots, K_\mu\}$ where M is the total number of components in the system, and K_μ is the total number of unique means in the system. An analogous definition holds for the D and B matrices, where we assume there are a total of K_D unique D matrices, and K_B total unique B matrices in the model. We also define the inverse mappings: $T_\mu^{-1}(k)$, $T_D^{-1}(k)$, and $T_B^{-1}(k)$, which refer to the set of all indices m that lead to a particular index k, that is

$$T_\mu^{-1}(k) \triangleq \{m : T_\mu(m) = k\},$$

and where the analogous definitions hold for $T_D^{-1}(k)$ and $T_B^{-1}(k)$.

Using this representation of tying via the index mapping, the auxiliary function becomes:

$$\mathcal{Q} = \sum_m \sum_t \left\{ \frac{1}{2} \log(|D_{T_D(m)}|) - \frac{1}{2}(x_t - B_{T_B(m)} z_t - \mu_{T_\mu(m)})^T D_{T_D(m)} (x_t - B_{T_B(m)} z_t - \mu_{T_\mu(m)}) \right\} p_{mt}$$
$$- \frac{\lambda}{2} \sum_{k=1}^{K_B} ||B_k||_F^2 - \frac{\eta}{2} \sum_{k=1}^{K_\mu} ||\mu_k||^2. \tag{13}$$

There are several differences between this equation and Equation 1. First, we have separated out the mean vectors from the B matrices since in this general case, there could be an entirely different tying pattern for the means and regression coefficients within B. Second, we now have two regularization coefficients, one for the regression coefficients λ, and a separate one for the shared mean pool η — note again that this form of regression only makes sense when the data have been centered or zero-mean normalized, as mentioned earlier. Other than that, we still may have a unique posterior probability p_{mt} for each m since various combinations of mean, diagonal covariance, and regression coefficient could still produce a distinct value of p_{mt} for different m and fixed t — the constrained optimization procedure requires that the posterior be filtered down into the appropriate shared component of each Gaussian.

4.1 Covariance updates

Taking derivatives and setting $\frac{\partial \mathcal{Q}}{\partial D_k} = 0$ implies that

$$\sum_t \sum_{m \in T_D^{-1}(k)} p_{mt} \left(D_k^{-1} - (x_t - B_{T_B(m)} z_t - \mu_{T_\mu(m)})(x_t - B_{T_B(m)} z_t - \mu_{T_\mu(m)})^T \right) = 0,$$

or that

$$D_k^{-1} = \frac{\sum_t \sum_{m \in T_D^{-1}(k)} (x_t - B_{T_B(m)} z_t - \mu_{T_\mu(m)})(x_t - B_{T_B(m)} z_t - \mu_{T_\mu(m)})^T p_{mt}}{\sum_t \sum_{m \in T_D^{-1}(k)} p_{mt}}.$$

In this form, it is not possible to compute the numerator without using multiple passes through the data, so we next expand the numerator in order to see what must be computed. Noting that D_k^{-1} is diagonal, and that any additive contributions to D_k^{-1} in the form of $A + A^T$ can be added as $2A$, we get:

$$D_k^{-1} \propto \sum_{m \in T_D^{-1}(k)} \left\{ \left(\sum_t x_t x_t^T p_{mt} \right) - 2 \left[\left(\sum_t x_t p_{mt} \right) \mu_{T_\mu(m)}^T + \left(\sum_t p_{mt} x_t z_t^T \right) B_{T_B(m)}^T - \mu_{T_\mu(m)} \left(\sum_t z_t^T p_{mt} \right) B_{T_B(m)}^T \right] \right.$$
$$\left. + B_{T_B(m)} \left(\sum_t z_t z_t^T p_{mt} \right) B_{T_B(m)}^T + \mu_{T_\mu(m)} \mu_{T_\mu(m)}^T \left(\sum_t p_{mt} \right). \right\} \tag{14}$$

From the above, it is particularly easy to see that each major term in the sum does not require any of the other parameters until the very end of the accumulation. For example, computing quantities such as $\sum_t x_t x_t^T p_{mt}$, $\sum_t p_{mt} x_t z_t^T$, and $\sum_t z_t p_{mt}$ can be done in one pass through the data, and at the end combined with the resulting final B matrix or mean vector for the corresponding m in producing the final numerator for the k^{th} covariance matrix.

If there are no B matrices (i.e., all the B entries are zero, which happens when we use purely diagonal covariance matrices), then we have a particularly simple form:

$$D_k^{-1} \quad \propto \quad \sum_{m \in T_D^{-1}(k)} \left\{ \left(\sum_t x_t x_t^T p_{mt} \right) - 2 \left(\sum_t x_t p_{mt} \right) \mu_{T_\mu(m)}^T + \mu_{T_\mu(m)} \mu_{T_\mu(m)}^T \left(\sum_t p_{mt} \right) \right\}. \tag{15}$$

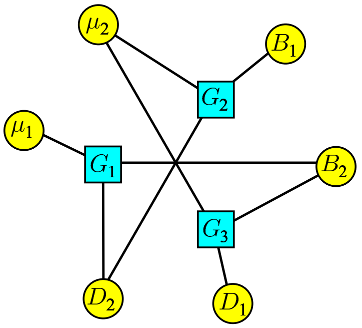

Figure 3: Sharing example. There are three Gaussian components, G_1, G_2, and G_3, with components G_2 and G_3 sharing the same mean μ_2, components G_1 and G_3 sharing the same regression matrix B_2, and G_1 and G_2 both sharing D_2. As shown in Equation 16, the EM update equation for B_2 depends on the other new parameters μ_1, μ_2, D_1, D_2, but B_1's update depends only on μ_2 and D_2.

4.2 B-matrix updates

Next, we find the update equations for the B matrices. Setting $\frac{\partial \Omega}{\partial B_k} = 0$ yields

$$\sum_{m \in T_B^{-1}(k)} D_{T_D(m)} \sum_t (x_t - B_k z_t - \mu_{T_\mu(m)}) z_t^T p_{mt} - \lambda B_k = 0,$$

or that

$$\sum_{m \in T_B^{-1}(k)} D_{T_D(m)} \left\{ \left(\sum_t x_t z_t^T p_{mt} \right) - B_k \left(\sum_t z_t z_t^T p_{mt} \right) - \mu_{T_\mu(m)} \left(\sum_t z_t^T p_{mt} \right) \right\} - \lambda B_k = 0.$$

Again, we have a similar problem in that we are trying to solve for a matrix B in an equation of the form $A - \sum_i D_i B C_i - E - \lambda B = 0$, which is a generalized form of the Sylvester equation. Note that even if we decide not to regularize so that $\lambda = 0$, then Sylvester-like equations still apply since B is still in relation with the other parameters via $A - E = \sum_i D_i B C_i$. In this case, other than the lack of a direct analytical solution for B_k, we see that it depends on other parameters from the current EM iteration, giving us the situation depicted in the center of Figure 2. In particular, B_k depends on all the other mean (μ) and variance (D) parameters that are involved in any of the components $m \in T_B^{-1}(k)$, as described in Figure 3. This means that the dependencies between current parameters as described in the center of Figure 2 will depend entirely on the tying patterns as given by the T functions, each individual parameter having a different set of dependences — in a sense, each parameter is independent of all other parameters given what might be called its variational or parameter Markov blanket.

When $\lambda > 0$ or in order not to solve the general Sylvester equation, we must once again use explicitly the fact that D_k for all k is diagonal — we may write Equation 13 in a way that is analogous to the way Equation 1 was written as Equation 11. Using again $B_k(r)$ to denote the r^{th} row of B_k, and the analogous definitions as used in Section 3.2, we immediately get:

$$B_k(r) = \left(\sum_{m \in T_B^{-1}(k)} D_{T_D(m)}(r) \left\{ \left(\sum_t x_t(r) z_t^T(r) p_{mt} \right) - \mu_{T_\mu(m)}(r) \left(\sum_t z_t^T(r) p_{mt} \right) \right\} \right)$$
$$\left(\sum_{m \in T_B^{-1}(k)} D_{T_D(m)}(r) \left(\sum_t z_t(r) z_t^T(r) p_{mt} \right) + \lambda I \right)^{-1}, \quad (16)$$

where I is an appropriately sized identity matrix. We once again see that an effect of the $\lambda > 0$ parameter ensures the invertibility of the matrix in which it is involved.

4.3 Mean updates

Since the mean parameter can have a separate tying pattern, we must treat it separately from the B matrix. We set $\frac{\partial \mathcal{Q}}{\partial \mu_k} = 0$ to get

$$\sum_{m \in T_\mu^{-1}(k)} D^{-1}_{T_D(m)} \sum_t (x_t - B_{T_B(m)} z_t - \mu_k) p_{mt} - \eta \mu_k = 0,$$

leading to

$$\sum_{m \in T_\mu^{-1}(k)} D_{T_D(m)} \left\{ \left(\sum_t x_t p_{mt} \right) - B_{T_B(m)} \left(\sum_t z_t p_{mt} \right) \right\} = \sum_{m \in T_\mu^{-1}(k)} D_{T_D(m)} \sum_t \mu_k p_{mt} + \eta \mu_k,$$

or that

$$\mu_k = \left(\sum_{m \in T_\mu^{-1}(k)} D_{T_D(m)} \left(\sum_t p_{mt} \right) + \eta I \right)^{-1} \left\{ \sum_{m \in T_\mu^{-1}(k)} D_{T_D(m)} \left[\left(\sum_t x_t p_{mt} \right) - B_{T_B(m)} \left(\sum_t z_t p_{mt} \right) \right] \right\}.$$

When there is no mean sharing (i.e., the mean is used in only one component), we get that $T_\mu^{-1}(k)$ has only one entry, and if also $\eta = 0$, the inverse covariance matrix cancels out, and we end up with a greatly simplified form for the means:

$$\mu_k = \frac{\sum_t x_t p_{mt} - B_{T_B(m)} \sum_t z_t p_{mt}}{\sum_t p_{mt}}.$$

4.4 When only the variances are tied

In certain cases, it may be that only a particular pattern of tying occurs. For example, when only the covariances are tied, a number of simplifications are possible. The auxiliary $\mathcal{Q}$ function becomes:

$$\mathcal{Q} = \sum_m \sum_t \left\{ \frac{1}{2} \log(|D_{T_D(m)}|) - \frac{1}{2} (x_t - B_m z_t - \mu_m)^T D_{T_D(m)} (x_t - B_m z_t - \mu_m) \right\} p_{mt} - f(B, \mu),$$

where $f(B, \mu)$ indicates only the regularization terms corresponding to $\{B_k : 1 \leq k \leq K_B\}$ and $\{\mu_k : 1 \leq k \leq K_\mu\}$. Because B_k and μ in this case are not tied, we can, as in Section 3, join them together in a single B matrix with the assumption that the final z has a constant of 1, and if we assume that they utilize the same regularization coefficient, we get:

$$\mathcal{Q} = \sum_m \sum_t \left\{ \frac{1}{2} \log(|D_{T_D(m)}|) - \frac{1}{2} (x_t - B_m z_t)^T D_{T_D(m)} (x_t - B_m z_t) \right\} p_{mt} - \frac{\lambda}{2} \sum_m ||B_m||_F^2.$$

As in Section 3, we set $\frac{\partial \mathcal{Q}}{\partial B_m} = 0$ to get that:

$$\sum_t D_{T_D(m)} (x_t - B_m z_t) z_t^T p_{mt} - \lambda B_m = 0,$$

or once again that:

$$\sum_t (x_t - B_m z_t) z_t^T p_{mt} - \lambda D^{-1}_{T_D(m)} B_m = 0,$$

leading to the same update equations for B_m.

To find the update equations for the tied D_k matrix, we set $\frac{\partial \mathcal{Q}}{\partial D_k} = 0$ to get:

$$D_k = \frac{\sum_t \sum_{m \in T_D^{-1}(k)} (x_t - B_m z_t)(x_t - B_m z_t)^T p_{mt}}{\sum_t \sum_{m \in T_D^{-1}(k)} p_{mt}} \tag{17}$$

$$= \frac{\sum_{m \in T_D^{-1}(k)} \left(\sum_t p_{mt} x_t x_t^T - B_m \sum_t p_{mt} z_t x_t^T \right)}{\sum_{m \in T_D^{-1}(k)} \sum_t p_{mt}}, \tag{18}$$

where again the second equality holds only when $\lambda = 0$. This equation, therefore, as in Section 3 is relatively easy to compute. Also, note that when $\lambda = 0$, this is an EM rather than a GEM procedure (see Section 5), since in each case we are maximizing rather than just increasing the auxiliary function (since there is no parameter dependence). In general, we see that the form of parameter tying and regularization can interact in complicated ways to determine how we can optimize the parameters, and this is discussed further in Section 5.

4.5 Regularized MLLR

Speaker adaptation [40, 26, 24, 44, 56] is one of the most successful parameter adjustment methods in speech recognition. In the adaptation paradigm, the training and test set distributions are not identical [58]. Moreover, there is a large training set so that the training set distribution may be well estimated, but only a small amount of adaptation data from the test distribution is available — this small test distribution data set is used to slightly adjust the parameters of a model to better represent the test set distribution. Of course, if too little restriction is placed on the amount of adjustment made to the model based on the adaptation data, then over-training can occur, and therefore adaptation must be based on a highly constrained model. In the widely used maximum-likelihood linear regression (MLLR) [40, 56], a particular form of constraint is used, namely the mean vectors in a Gaussian mixture model are subject to a transformation that consists of no more than a scale, rotation, and shift (an affine transformation).

Given our derivation of shared conditional Gaussians above, we can extend MLLR so that not only the Gaussian means are adapted, so is covariance matrix, all using non-zero mean Gaussian Bayesian priors. This is largely thanks to the Cholesky factorized covariance matrix.

In this framework, we moreover may easily adapt the regression coefficients, even if they correspond to a cross-stream conditional Gaussian. For example, the component parameters μ and B can be adapted while leaving the conditional variance parameters D fixed. This form of adaptation is seen essentially as another application of operations similar to the B matrix. In particular, suppose that for each m there is a mapping $T_A(m) : \{1, \dots, M\} \to \{1, \dots, K_A\}$ that maps from each component, say m, to a "regression class" [40], which gives an index that identifies the parameters that are to effect that component. The k^{th} adaptation parameters then are to adapt all components $\{m : m \in T_A^{-1}(k)\}$ using a similar notation as in Section 4. For each k, there is a general upper triangular (so not symmetric) adaptation coefficient matrix A_k, a shift vector b_k, and a variance scale diagonal matrix C_k. These parameters influence the model as expressed in the following adjusted auxiliary function:

$$\mathcal{Q} = \sum_m \sum_t \sum_r \left(\frac{1}{2} \log(C_{T_A(m)}(r)^2 D_m(r)) - \frac{1}{2} \big(x_t(r) - A_{T_A(m)}(r) B_m(r) z_t(r) - b_{T_A(m)}(r)\big)^2 C_{T_A(m)}(r)^2 D_m(r) \right) p_{mt}$$
$$+ \sum_k \left(-\frac{\lambda}{2} ||A_k - I||^2 - \frac{\eta}{2} ||b_k||^2 - \frac{\nu}{2} ||C_k - I||^2 \right),$$

where $A_k(r)$ (resp. $b_k(r)$, $C_k(r)$) is the r^{th} row (resp. entry, diagonal element) of matrix A (resp. vector b_k, C_k). Again, any original mean parameters are contained within $B_m(r)z_t(r)$. A_k must be upper triangular so that the form $A_k B$ is still upper diagonal with zeros along the diagonal, and this means that the Gaussian normalization constant (a function of D_m) is still valid. Also, since C_k only occurs in squared form, C_k other than being a diagonal matrix is unconstrained. As can be seen, we adapt parameters of the m^{th} component Gaussian by applying an affine transformation to the conditional parameters as in $A_{T_A(m)}(r) B_m(r) z_t(r) + b_{T_A(m)}(r)$ which becomes the new conditional mean. Also, the r^{th} variance is scaled by $C_k(r)$. Of course, to avoid over-training on the data, we assume that K_A is relatively small, and if $K_A = 1$ we have one global A, C matrix pair and b vector that influences the model by performing one joint shift and then scale and rotate of the parameters. This generalizes MLLR as well since we are not only allowing the adaptation to occur but we penalize the set of adaptation coefficients depending on its distance from the identity matrix using the Frobenius norm terms $||A_k - I||$ and $||C_k - I||$. This last regression term provides another way other than the number of regression classes that accuracy-regularization within MLLR may be controlled. This may be appropriate in cases with extremely small amounts of data, where even one (possibly sparse) A matrix would require too many parameters. Note, the approach outlined above may be seen as applying a Bayesian prior [31] on the adaptation coefficients in MLLR, but this is not the same form as MAP adaptation [26], where a Bayesian prior (in particular, a Dirichlet and Wishart prior) is applied to all of the Gaussian mixture parameters (including the mixture coefficients).

To perform adaptation, then, we can derive an EM update step very similar to the derivation in the shared parameter case in Sections 4.2 and 4.3. First, let $\bar{z}_t^m(r) = B_m(r) z_t(r)$ and $x_t^m(r) = x_t(r) - b_{T_A(m)}(r)$. Let $I(r)$ be a row-vector of zeros but with a one in the r^{th} position (i.e., $I(r)$ is the r^{th} row of the $M \times M$ identity matrix). We find $\frac{\partial \mathcal{Q}}{\partial A_k(r)}$ and $\frac{\partial \mathcal{Q}}{\partial b_k(r)}$ yielding:

$$\begin{aligned}
&\frac{\partial \mathcal{Q}}{\partial A_k(r)} \left(\sum_{m,t,r} \left(x_t^m(r) - A_{T_A(m)}(r) \bar{z}_t^m\right)^2 D_m(r) p_{mt} - \sum_{k',r'} \frac{\lambda}{2} ||A'_k(r') - I(r')||^2 \right) \\
&= \sum_t \sum_{m \in T_A^{-1}(k)} D_m(r)(x_t(r) - A_k(r) z_t^m)(z_t^m(r))^T - \lambda(A_k(r) - I(r)).
\end{aligned}$$

Solving for $A_k(r)$ gives

$$\sum_t \sum_{m \in T_A^{-1}(k)} D_m(r) x_t(r)(z_t^m(r))^T - A_k(r) \sum_t \sum_{m \in T_A^{-1}(k)} (D_m(r) I) z_t^m (z_t^m)^T - \lambda A_k(r) + \lambda I(r),$$

yielding

$$A_k(r) = \left(\sum_t \sum_{m \in T_A^{-1}(k)} D_m(r) x_t(r)(z_t^m(r))^T + \lambda I(r) \right) \left(\sum_t \sum_{m \in T_A^{-1}(k)} (D_m(r) I) z_t^m (z_t^m)^T + \lambda I \right)^{-1}.$$

We can thus see that as $\lambda \to \infty$, we have that $A_k(r) \to I(r)$, so that $A_k \to I$, thus increasing λ decreases the amount of adaptation for each regression class. Updates for C_k are also relatively straightforward, where for each $C_k(r)$ we get a quadratic form. Each EM update step can be performed one or more times and, like MLLR [40], sets of EM updates can be iteratively interleaved with decodings to potentially further improve results in the unsupervised adaptation case.

It is easy to see that the above corresponds to placing nonzero-mean Gaussian priors on each of the three components of a Gaussian as we have used throughout this chapter, a property enabled by the use of Cholesky factorization. The use of Cholesky factorization has been used before [24, 22]. Moreover, Bayesian priors have also been used for adaptation. For example, a prior may be placed on all parameters in the system [26]. Also, in maximum a-posterior linear regression (MAPLR), a matrix Gaussian prior is placed on coefficients governing the affine transformation of the model means [11, 53], or on an inner-factor diagonal of the covariance matrix [12]. A kernel ridge regression form of adaptation was also recently developed [51]. The technique presented above is different in that it does not use zero mean Gaussian priors.

5 A generalized EM procedure

A generalized EM (GEM) is a procedure where rather than maximizing the auxiliary function by adjusting the parameters at each EM iteration, we instead only increase the auxiliary function. The above procedures that have been outlined show that, depending on the regularization coefficients, and depending on the parameter tying pattern, we may not be able to form a closed form update of each of the parameters as a function only of the previous EM iterations parameters, something as shown on the left of Figure 2. This circularity in the parameter updates cannot be broken even using multiple passes through the data, as none of the new parameters can be formed without access to the other new parameters.

There have been two strategies for parameter update, however, that break the cycles. The first strategy has suggested that we can still update all parameters simultaneously if we use the previous parameters. In other words, when one parameter type is needing another from the current iteration, we substitute its value with the corresponding parameter at the previous EM iteration. This procedure, again, is guaranteed to have the same fixed points as a standard EM iteration, but we have not provided any guarantees regarding whether it increases the likelihood function, nor if it ever reaches such a fixed point.

An alternate strategy is the one suggested on the right of Figure 2. In this case, we only update one parameter set at a time. For example, we might find the optimal values only for $\{D_k : 1 \le k \le K_D\}$, use these new parameters in Equation (13). Next, we might optimize $\{B_k : 1 \le k \le K_B\}$, use these new parameters, and then last optimize $\{\mu_k : 1 \le k \le K_\mu\}$. This entire procedure can then be repeated. This alternating optimization can be done without needing to worry about cyclic parameter dependencies since none of the parameters of a given type depend on any of the other parameters of that same type in the same EM iteration (e.g., μ_k for any fixed k does not depend on any $\mu_{k'}$ for any k' in the EM current iteration).

We can easily see that this procedure is a generalized EM since each update is guaranteed not to decrease the original likelihood function — in particular, the auxiliary function, Equation (13) is concave in each of μ_k, B_k, and D_k separately, and in each case we obtain the same standard EM bounds [15, 43]. For example, computing the Hessian matrix of Equation (13) with respect to $B_k(r)$, we get:

$$\frac{\partial^2 \mathcal{Q}}{\partial B_k(r)^T \partial B_k(r)} = - \sum_{m \in T_B^{-1}(k)} D_{T_D(m)}(r) \left(\sum_t z_t(r) z_t(r)^T p_{mt} \right) - \lambda I,$$

a matrix that is clearly everywhere negative definite for all $\lambda \geq 0$ implying that $\mathcal{Q}$ is concave everywhere. Hessians with respect to the other parameters are also everywhere negative definite.

6 Conclusions

We have described the EM/GEM update equations for generic Gaussian mixture systems, which include standard Gaussian mixtures, hidden Markov models, auto-regressive HMMs, multi-stream buried Markov models, and regularized MLLR. More specifically, we have described these equations where the Gaussians are either regularized, partially parameter shared, or both.

We have seen that depending on the regularization parameters or the parameter tying pattern, the standard EM update equations might possess circular dependences, which requires modifications to the normal EM updates to break data dependency cycles. Several schemes were proposed for parameter update in this case, including when only one of each set of parameters is updated at a time. We have seen how this yields a GEM procedure that is still guaranteed to produce a local-maximum of the likelihood function. It would be interesting to relate the parameter tying mechanisms described here with Gaussian generalizations of Probabilistic Relational Models (PRMs) [20], as parameter sharing is a common feature in such models. In their case, however, parameters are typically discrete so the parameter dependency circularities do not arise. If one were to generalize PRMs to include partial Gaussian sharing as we have done above, the same procedures could potentially be adopted. Moreover, all of the above procedures would benefit from the application of ℓ_1-norm regularization [31], which would encourage sparse structure while retaining convexity in the EM update equations.

The above system of equations for tied and regularized Gaussians has been implemented as part of the graphical models toolkit (GMTK) and has been used in a wide variety of contexts. This material is based upon work supported by the National Science Foundation under Grant No. IIS-0093430, and by the Office of Naval Research under Grant No. N000140510388.

References

[1] L.R. Bahl, P.F. Brown, P.V. de Souza, and R.L. Mercer. Maximum mutual information estimation of HMM parameters for speech recognition. In *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, pages 49–52, Tokyo, Japan, December 1986.

[2] L.E. Baum and T. Petrie. Statistical inference for probabilistic functions of finite state Markov chains. *Ann. Math. Statist.*, 37(6):1554—1563, 1966.

[3] J. Bilmes. Buried markov models: A graphical modeling approach to automatic speech recognition. *Computer Speech and Language*, 17:213—231, April—July 2003.

[4] J. Bilmes and C. Bartels. Graphical model architectures for speech recognition. *IEEE Signal Processing Magazine*, 22(5):89–100, September 2005.

[5] J. A. Bilmes. Graphical models and automatic speech recognition. In R. Rosenfeld, M. Ostendorf, S. Khudanpur, and M. Johnson, editors, *Mathematical Foundations of Speech and Language Processing*. Springer-Verlag, New York, 2003.

[6] J.A. Bilmes. A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models. Technical Report TR-97-021, ICSI, 1997.

[7] J.A. Bilmes. Buried Markov models for speech recognition. In *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, Phoenix, AZ, March 1999.

[8] J.A. Bilmes. Factored sparse inverse covariance matrices. In *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, Istanbul, Turkey, 2000.

[9] C. Bishop. *Neural Networks for Pattern Recognition*. Clarendon Press, Oxford, 1995.

[10] C.-P. Chen, J. Bilmes, and D. Ellis. Speech feature smoothing for robust ASR. In *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, March 2005.

[11] C. Chesta, O. Siohan, and C.-H. Lee. Maximum a posteriori linear regression for hidden markov model adaptation. In *European Conf. on Speech Communication and Technology (Eurospeech)*, 1999.

[12] W. Chou and X. He. Maximum a posteriori linear regression (maplr) variance adaptation for continuous density hmms. In *European Conf. on Speech Communication and Technology (Eurospeech)*, pages 1513–1516, Geneva, Switzerland, 2003.

[13] A. P. Dawid. Conditional independence in statistical theory. *Journal of the Royal Statistical Society B*, 41(1):1–31, 1989.

[14] T. Dean and K. Kanazawa. Probabilistic temporal reasoning. *AAAI*, pages 524–528, 1988.

[15] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum-likelihood from incomplete data via the EM algorithm. *J. Royal Statist. Soc. Ser. B.*, 39, 1977.

[16] Y. Ephraim, A. Dembo, and L. Rabiner. A minimum discrimination information approach for HMM. *IEEE Trans. Info. Theory*, 35(5):1001–1013, September 1989.

[17] Y. Ephraim and L. Rabiner. On the relations between modeling approaches for information sources. In *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, pages 24–27, 1988.

[18] R. Fletcher. *Practical Methods of Optimization*. John Wiley & Sons, New York, NY, 1980.

[19] N. Friedman. The Bayesian structural EM algorithm. *14th Conf. on Uncertainty in Artificial Intelligence*, 1998.

[20] Nir Friedman, Lise Getoor, Daphne Koller, and Avi Pfeffer. Learning probabilistic relational models. In *IJCAI*, pages 1300–1309, 1999.

[21] Zoran Gajic. *Lyapunov matrix equation in system stability and control*. Academic Press, San Diego, 1995.

[22] M.J.F. Gales. Maximum likelihood linear transformations for hmm-based speech recognition. *Computer Speech and Language*, 12:75–98, 1998.

[23] M.J.F. Gales. Semi-tied covariance matrices for hidden Markov models. *IEEE Transactions on Speech and Audio Processing*, 7(3):272–281, May 1999.

[24] M.J.F. Gales and P.C. Woodland. Mean and variance adaptation within the mllr framework. *Computer Speech and Language*, 10, 1996.

[25] Judith D. Gardiner, Alan J. Laub, James J. Amato, and Cleve B. Moler. Solution of the sylvester matrix equation $axbt + cxdt = e$. *ACM Trans. Math. Softw.*, 18(2):223–231, 1992.

[26] J.L. Gauvain and C.H. Lee. Maximum a-posteriori estimation for multivariate gaussian mixture observations of markov chains. *IEEE Transactions on Speech and Audio Processing*, 2:291–298, 1994.

[27] G.H. Golub and C.F. Van Loan. *Matrix Computations*. Johns Hopkins, 1996.

[28] R. A. Gopinath. Maximum likelihood modeling with gaussian distributions for classification. In *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, 1998.

[29] D.A. Harville. *Matrix Algebra from a Statistician's Perspective*. Springer, 1997.

[30] T. Hastie, S. Rosset, R. Tibshirani, and J. Zhu. The entire regularization path for the support vector machine. *Journal of Machine Learning Research*, 5:1391–1415, Oct 2004.

[31] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Series in Statistics. Springer, 2001.

[32] D. Heckerman, D. Geiger, and D.M. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. Technical Report MSR-TR-94-09, Microsoft, 1994.

[33] D. Heckerman and C. Meek. Embedded bayesian network classifiers. Technical Report MSR-TR-97-06, Microsoft Research, Redmond, WA, 1997.

[34] D. Heckerman and C. Meek. Models and selection criteria for regression and classification. In *Proceedings of Thirteenth Conference on Uncertainty in Artificial Intelligence,* Providence, RI. Morgan Kaufmann, August 1997.

[35] M.J. Hunt, M. Lennig, and P. Mermelstein. Experiments in syllable-based recognition of continuous speech. *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, 1980.

[36] F.V. Jensen. *An Introduction to Bayesian Networks*. Springer, 1996.

[37] B.-H. Juang, W. Chou, and C.-H. Lee. Minimum classification error rate methods for speech recognition. *IEEE Trans. on Speech and Audio Signal Processing*, 5(3):257–265, May 1997.

[38] B-H Juang and S. Katagiri. Discriminative learning for minimum error classification. *IEEE Trans. on Signal Processing*, 40(12):3043–3054, December 1992.

[39] S.L. Lauritzen. *Graphical Models*. Oxford Science Publications, 1996.

[40] C.J. Leggetter and P.C. Woodland. Maximum likelihood linear regression for speaker adaptation of continuous density hidden Markov models. *Computer Speech and Language*, 9:171–185, 1995.

[41] Richard J. Mammone, Xiaoyu Zhang, and Ravi P. Ramachandran. Robust speaker recognition. *IEEE Signal Processing Magazine*, 13(5):58–71, September 1996.

[42] K. Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD thesis, U.C. Berkeley, Dept. of EECS, CS Division, 2002.

[43] R.M. Neal and G.E. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In M.I. Jordan, editor, *Learning in Graphical Models*, pages 355–368. Kluwer Academic Publishers, 1998.

[44] L. Neumeyer, A. Sankar, and V. Digalakis. A comparative study of speaker adaptation techniques. In *European Conf. on Speech Communication and Technology (Eurospeech)*, pages 1127–1130, Madrid, Spain, 1995.

[45] A. Ng and M. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Neural Information Processing Systems (NIPS)*, 14, Vancouver, Canada, December 2002.

[46] P. Olsen and R. Gopinath. Modeling inverse covariance matrices by basis expansion. In *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, pages 945–948, 2002.

[47] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 2nd printing edition, 1988.

[48] J. Pearl. *Causality*. Cambridge, 2000.

[49] L.R. Rabiner and B.H. Juang. An introduction to hidden Markov models. *IEEE ASSP Magazine*, 1986.

[50] S. Roweis and Z. Ghahramani. A unifying review of linear gaussian models. *Neural Computation*, 11:305–345, 1999.

[51] G. Saon. A non-linear speaker adaptation technique using kernel ridge regression. In *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, 2006.

[52] F. Sha and L. K. Saul. Large margin gaussian mixture modeling for phonetic classification and recognition. In *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, Tolouse, France, 2006.

[53] O. Siohan, T. Myrvol, and C. Lee. Structural maximum a posteriori linear regression for fast hmm adaptation. *Computer Speech and Language*, 16:5–24, 2002.

[54] V. Vapnik. *Statistical Learning Theory*. Wiley, 1998.

[55] G. Wahba. *Spline Models for Observational Data*, volume 59 of *CBMS-NSF Regional Conference Series in Applied Mathematics*. SIAM, Philadelphia, 1990.

[56] P.C. Woodland. Speaker adaptation: Techniques and challenges. In *Proc. IEEE ASRU*, 1999.

[57] P.C. Woodland and D. Povey. Large scale discriminative training for speech recognition. In *ICSA ITRW ASR2000*, 2000.

[58] X.Li and J. Bilmes. Regularized adaptation of discriminative classifiers. In *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, 2006.

[59] X.Li, J. Bilmes, and J.Malkin. Maximum margin learning and adaptation of mlp classifers. In *European Conf. on Speech Communication and Technology (Eurospeech)*, 2005.

[60] S. Young. A review of large-vocabulary continuous-speech recognition. *IEEE Signal Processing Magazine*, 13(5):45–56, September 1996.

[61] S. Young, J. Jansen, J. Odell, D. Ollason, and P. Woodland. *The HTK Book*. Entropic Labs and Cambridge University, 2.1 edition, 1990's.