The Pennsylvania State University

The Graduate School

College of Engineering

JOINT PARSIMONIOUS MODELING AND MODEL ORDER SELECTION FOR MULTIVARIATE GAUSSIAN MIXTURES

A Thesis in

Electrical Engineering

by

Scott C. Markley

Submitted in Partial Fulfillment

of the Requirements

for the Degree of

Master of Science

May 2009

The thesis of Scott C. Markley was reviewed and approved* by the following:

David J. Miller

Professor of Electrical Engineering

Thesis Advisor

Ji-Woong Lee

Assistant Professor of Electrical Engineering

W. Kenneth Jenkins

Professor of Electrical

Head of the Department of Electrical Engineering

*Signatures are on file in the Graduate School.

Abstract

Multivariate Gaussian mixtures are widely used in science and engineering for density estimation, model-based data clustering, and statistical classification. A difficult problem, of special interest for clustering, is estimating the model order, *i.e.* the number of mixture components. Use of full covariance matrices, with number of parameters quadratic in the feature dimension, entails high model complexity, and thus may underestimate order, while naive Bayes mixtures may introduce model bias and lead to order overestimates. We develop a parsimonious modeling and model order selection method for multivariate Gaussian mixtures which allows for and *optimizes over* parameter tying configurations across mixture components applied to each individual parameter, including the covariates. We derive a generalized Expectation-Maximization algorithm for (BIC-based) penalized likelihood minimization. This algorithm, coupled with sequential model order reduction, forms our joint learning and model selection method. Our method searches over a rich space of models with different (data representation, model complexity) tradeoffs and, consistent with minimizing BIC, achieves fine-grained matching of model complexity to the amount of available data. We have found our method to be effective and largely robust in learning accurate model orders and parameter-tying structures for simulated ground-truth mixtures. We also compared against naive Bayes and standard full-covariance Gaussian mixtures for several

different criteria: i) accuracy in estimating the number of (ground-truth) components; ii) test set log-likelihood; iii) unsupervised (and semisupervised) classification accuracy; and iv) accuracy when class-conditional mixtures are used in a plug-in Bayes classifier. The results bear out that our parsimonious mixtures and coupled learning give improved accuracy with respect to each of these performance measures.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

In a multivariate Gaussian mixture model (GMM) for a random vector $\underline{X} = (X_1, X_2, \ldots, X_d)$, the density function takes the form

$$p_{\underline{X}}(\underline{x}|\Theta(M)) = \sum_{k=1}^{M} \alpha_k p_{\underline{X}|k}(\underline{x}|\theta_k), \tag{1.1}$$

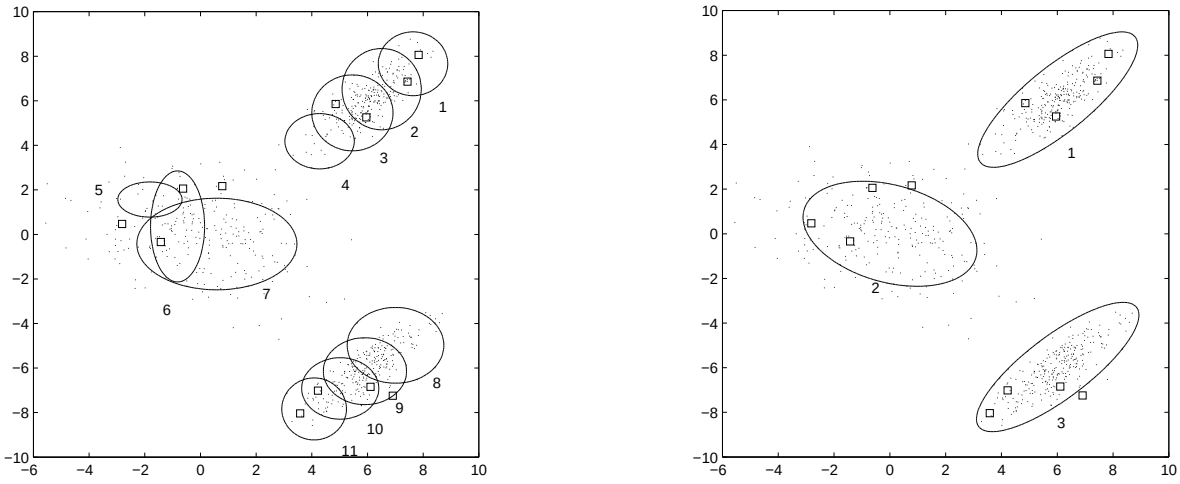
where $\{\alpha_k\}$ are the mixing proportions ($\alpha_k \geq 0$ and $\sum_{k=1}^{M} \alpha_k = 1$), $\theta_k = (\underline{\mu}_k, \Sigma_k)$ is the (mean vector, covariance matrix) specifying the k-th component, $\Theta(M) = \{\theta_k, \alpha_k, k = 1, \ldots, M\}$, and $p(\cdot)$ denotes a density function. GMMs have diverse applications, including density estimation and associated outlier detection *e.g.* [1], supervised statistical classification [2], as well as most prominent application to unsupervised (also semisupervised [3],[4]) clustering [5],[2]. For supervised classifier design, a mixture model is used to fit a (class-conditional) density function to data measurements that putatively have been stochastically generated based on an (unknown) *multimodal* density function. The resulting class-conditional models are then used in a plug-in Bayes classification rule [2]. For this application, model order selection (estimating the number of mixture components) and model structure selection (determining *e.g.* whether components

use diagonal covariance matrices, full matrices, or covariance structures of intermediate complexity [5]) may be helpful for maximizing classification accuracy. They are also useful for limiting computation and/or memory storage associated with classifying or evaluating the data likelihood for new (test) samples. Toward this end, one seeks the minimum number of components (or parameters) needed to accurately fit training (or validation) data.

While limiting computation/memory is of practical import, model order and structure selection are most compelling, and in fact at the heart of the problem, when mixtures are used as a *model-based* approach [5] to unsupervised clustering. Here, one is concerned with identifying the (*a priori* unknown) underlying groups or classes comprising a given data set: i) the number of groups; ii) a model for each; and iii) the hard/soft membership of each data sample in each group. In this context, mixture model order selection estimates the number of clusters in the data set and structure selection estimates a cluster's shape (and *e.g.* whether features are statistically independent, given the cluster of origin).

Model order and model structure selection are intimately intertwined, as illustrated in Figure 1.1. Here, the data were generated based on three bivariate Gaussians. When naive Bayes GMMs (*i.e.*, with diagonal covariances) are learned, application of the widely used *Bayesian Information Criterion* (BIC) [6] leads to gross overestimation of the true order (11 components are estimated). In a semisupervised learning context [3],[4], the undesireable consequence (also depicted in Fig. 1.1) may be that some mixture components will not "own" any labeled samples, which may make it difficult to classify samples belonging to these components. By contrast, when GMMs with full (unconstrained) covariances are learned, BIC selects the true model order, three.

GMMs with full covariances offer the most flexible data representation. However, they also require estimation of $d(d-1)/2$ free parameters per component. When the data sample $\mathcal{X} = \{\underline{x}_1, \ldots, \underline{x}_N\}$ is small relative to d, there may be insufficient samples to accurately estimate all these parameters. There is also potential for estimating singular covariance matrices [7], especially



(a) Naive Bayes

(b) Full Covariance

Figure 1.1. Bivariate Gaussian data overlayed with contours at component density value equal to 0.002 for learned Naive Bayes and Full Covariance Gaussian models. Squares indicate labeled examples. Naive Bayes components 2, 4, 7, and 8 contain no labeled examples when points are assigned to components based on a maximum likelihood decision

for small N. Finally, consider model order selection criteria such as BIC:

$$BIC = \frac{|\Theta(M)|}{2} \log N - \log \prod_{i=1}^{N} p_{\underline{X}}(\underline{x}_i|\Theta(M)), \tag{1.2}$$

where $|\Theta(M)|$ is the number of free parameters in an M-component model. If N is small, the likelihood benefit (second term) associated with a full-covariance model may be greatly outweighed by the model complexity (first term) required to achieve it. In this case, BIC minimization, as well as other penalty function based model selection methods, *e.g.* [8],[9], may *grossly* underestimate model order, as was previously demonstrated in [10] for modeling text documents. A variety of strategies have been taken to address model order and structure selection for GMMs and for mixtures in general, as we next review.

Since the number of model parameters typically grows superlinearly with feature dimension, one strategy for reducing model complexity is to apply dimensionality reduction prior to clustering. Feature transformations such as PCA/SVD are often applied for dimensionality reduction,

e.g. [11], but may lose interpretability with respect to the original set of features. Alternatively, unsupervised feature *selection* methods have been explored in recent years [12],[13],[15],[10]. This problem is inherently formidable. Even *supervised* feature selection is challenging [14], mainly due to the vast set of candidate feature subsets, exponential in d, which precludes exhaustive search. However, in the supervised case one at least has knowledge of the number of classes present, and with labeled examples of each. One can thus readily apply supervised class discriminability measures (error rate, class entropy, mutual information, signal-to-noise ratio, and other criteria [14]) to evaluate candidate feature subsets. By contrast, in the unsupervised case, there is a more insidious "chicken and egg" problem. "Good features" are cluster(class)-dependent, but "good clusters" are also *feature*-dependent – without feature reduction that is successful in removing many irrelevant/noisy features, clustering methods are likely to find poor solutions that do not partition the data set into compact, well-separated clusters and that fail to capture the ground-truth cluster structure. This is particularly true when the data is high-dimensional [10].

Unsupervised feature selection methods can be categorized into those performing *explicit* feature selection and those performing *implicit* selection. [12] wrapped explicit feature selection (forward feature growing or backward feature elimination [2]) around maximum likelihood-based learning of Gaussian mixture models. The authors recognized that standard criteria for evaluating candidate clustering solutions are biased with respect to the number of features – data likelihood tends to decrease with d, while cluster separability measures based on within and between-cluster scatter matrices tend to increase with d. A "cross-projection normalization" criterion was thus proposed to mitigate this bias and allow "fair" comparison of solutions with both different numbers of features and clusters.

One heuristic aspect of [12] is the use of different learning objectives for the clustering and feature selection steps. Implicit feature selection methods [15],[10], by contrast, perform mixture modeling on the *full* feature space to optimize a single objective function. Although these methods explicitly model all features, they achieve *effective* feature selection by optimizing over the choice

of whether a feature is modeled by a *cluster-specific* distribution or by a "background" distribution that is commonly accessible by *all* the mixture components. Background modeling is also used in [13]. A feature that is modeled using the background mechanism by *all* clusters does not significantly contribute to cluster determination and has been, *in effect*, removed. The use of background models is a special type of parameter sharing which can greatly reduce the number of free model parameters and thus help to mitigate the problem of model order underestimation [10]. [13] applied a wrapper-based method similar to [12] for document clustering, selecting both which features (words) are cluster-specific and selecting the clustering solution so as to maximize the Bayesian integrated likelihood. Instead of making hard decisions on whether features use a component-specific or background representation, [15] introduced soft feature "saliency" weights $\in [0, 1]$ and used an alternative structure and order selection criterion. While [12], [13], and [15] all *tied* the (explicit or implicit) feature space across all mixture components, [10] allowed a more flexible representation, with each cluster freely choosing its own feature subset to be modeled in a component-specific fashion (with the complement feature subset modeled using background distributions). Thus, each cluster/component will in general have its own distinct feature space. [10] developed a learning framework achieving joint minimization of the BIC criterion with respect to all model parameters – the number of components, each component's feature space, and all component parameters. This method was demonstrated in [10] to achieve better modeling accuracy than [15] and was also shown to give promising model order selection results for high dimensions (documents, with d as large as 20,000).

However, a significant limitation of [10],[13], and [15] (but not of [12]) is that these methods were only developed for naive Bayes mixtures, *i.e.* those for which the joint density function, conditioned on the mixture component, factors as a product over individual feature densities – in the Gaussian case, these methods restrict covariance matrices to be diagonal. Model order and structure selection for GMMs with more flexible covariance structures has been considered in several past works. [5] wrote the covariance matrix in the form $\Sigma_k = \lambda_k D_k A_k D_k^T$, with λ_k a

scalar volume parameter, A_K a diagonal matrix with entries proportional to the eigenvalues of Σ_k, and with matrix D_k determining the principal component directions. The authors applied BIC for model selection and evaluated a small set of special case covariance structures, for varying M: i) $\Sigma_k = \lambda_k I$ (spherical, volume unconstrained); ii) $\Sigma_k = \Sigma = \lambda DAD^T$ (tied across all components); iii) $\Sigma_k = \lambda_k D_k A_k D_k^T$ (unconstrained for each component); iv) $\Sigma_k = \lambda D_k A D_k^T$ (orientation unconstrained); v) $\Sigma_k = \lambda_k D_k A D_k$ (orientation and volume unconstrained). An alternative, latent variable model that gives more flexible covariance parameterizations is the *mixture of factors analyzers* (MFA) [16]. MFA assumes the following stochastic data generation under mixture component k: $\underline{X} = \underline{\mu}_k + \Lambda_k \underline{Z} + \underline{N}$, where $\underline{N}$ is multivariate Gaussian with covariance matrix Ψ, $\underline{Z}$, the *factor vector*, is l-dimensional ($l < d$) multivariate Gaussian with zero-mean and identity covariance matrix, and with Λ_k the "factor loading matrix". $\underline{X}$ is thus multivariate Gaussian under component k with mean μ_k and covariance matrix $\Sigma_k = \Lambda_k \Lambda_k^T + \Psi$. If Ψ is a scaled identity matrix, this model is equivalent to mixture of principal component analyzers (MPCA) [17]. MFA (and MPCA) allow varying the effective dimensionality of each mixture component and, thus, the number of free parameters specifying covariance matrices, by varying the factor dimension l. The matrix Λ_k is $d \times l$, and when l is varied up from zero, the matrix Σ_k achieves full degrees of freedom once l exceeds $(d+1)/2$. Thus, degrees of freedom in Σ_k are varied in steps of size d (with each added or deleted column of Λ_k) in the MFA model. Another parameterized GMM approach is the extended maximum likelihood linear transformation (EMLLT) [18], proposed for speech recognition, which expresses the inverse covariance (precision) matrix as a linear combination of outer products. In principle, this model allows *finest-grain* variation of the number of free covariance parameters, *i.e.* variation from d up to $d(d+1)/2$ parameters in steps of size *one*. However, [18] assumed this parameterized covariance structure is tied across all mixture components. Moreover, there was no principled model order selection in [18] (reasonable in a speech recognition context, where the number of components may be heuristically chosen).

Bayesian approaches have recently been widely applied for both learning and model order and structure selection in mixtures, *e.g.* [21],[19],[20],[16]. A Bayesian framework is the proper approach for incorporating actual prior knowledge on distributions of parameter values. Even when such knowledge is unavailable, Bayesian learning can help to avoid singular regions of parameter space and can compensate for an inadequate training sample [21], achieving parameter smoothing and penalizing "outlier" parameter estimates. A Markov chain Monte Carlo (MCMC) learning approach that allows estimation of the number of components in GMMs is developed in [20]. However, this method assumes diagonal covariances. Alternatively, [16] derived a variational lower bound to the Bayesian evidence and maximized this lower bound in estimating MFA models. Their approach estimates both the number of components and the number of factors (and hence the covariance structure) used by each component.

In this paper, alternatively, model order selection in GMMs is performed consistent with minimization of BIC. On the one hand, BIC is based on the Laplace approximation to Bayesian integration [22], which is cruder than the variational approximation to Bayesian evidence considered in [16]. On the other hand, [16] optimized over MFA models, which only allow variation in covariance degrees of freedom in steps of size d, whereas our method allows finest-grain variation in complexity of the covariance structure for every component, from zero degrees of freedom up to $d((d+1)/2$ in steps of size one. Moreover, in [16], the prior distribution over the mixture model parameters factors as a product over each component, *i.e.*, there is an implicit assumption that the model parameters for different components are independently generated. In this work, parameters are not assumed to be independently generated. In fact, statistical dependencies across components are exploited in order to achieve efficient description (coding) of parameter tying structures across components. This efficient parameter coding allows more mixture components to be included in the model for small sample sizes, which helps avert the problem of model order underestimation – our approach achieves more accurate model order selection than standard methods for small sample sizes. Our work represents a significant extension of the parsi-

monious modeling framework from [10], which was restricted to mixture models with naive Bayes structure. Here we develop a parsimonious modeling and model order and structure selection method for multivariate Gaussian mixtures which allows for and *optimizes over* parameter tying configurations across mixture components applied to *every* individual parameter, including all covariates. We derive a generalized Expectation-Maximization algorithm for (BIC-based) penalized likelihood minimization. This algorithm, coupled with sequential model order reduction, forms our joint learning and model selection. Our method searches over a rich space of models with different (data representation, complexity) tradeoffs and, consistent with minimizing BIC, achieves fine-grained matching of model complexity to the amount of available data. In Chapter 2 our method is developed. Chapter 3 presents experimental results. The paper is then summarized in Chapter 4.

Chapter 2

Penalized Likelihood Minimization for Learning and Model Order Selection

2.1 Parsimonious Multivariate Gaussian Mixtures

Consider the mixture density function in (1.1). Each component density is assumed to be multivariate Gaussian, *i.e.*

$$p[\underline{x}|\tilde{\theta}_j] = \frac{1}{(2\pi)^{\frac{d}{2}}|\tilde{\mathbf{\Sigma}}_j|^{\frac{1}{2}}} \exp\left\{-\frac{1}{2}(\underline{x}-\underline{\tilde{\mu}}_j)^{\mathrm{T}}\tilde{\mathbf{\Sigma}}_j^{-1}(\underline{x}-\underline{\tilde{\mu}}_j)\right\}, \tag{2.1}$$

where $\underline{\tilde{\mu}}_j$ and $\tilde{\mathbf{\Sigma}}_j$ represent the mean vector and covariance matrix for component j [1]. In a *standard* multivariate Gaussian mixture, each component uses a freely chosen value for every mean and covariance parameter. Thus, $|\theta(M)| = (d(d+1)/2+d)M + M - 1$, with $d(d+1)/2$ covariate free parameters per component[2]. However, as discussed in chapter 1, in order to allow a rich set of data representation/model complexity tradeoffs, we consider *structured* Gaussian mixtures that allow a particular form of parameter tying wherein, for each component, for *every* scalar (mean and covariance) parameter, there is the choice of using *either* a component-specific

[1] The tilde notation is used to reflect possible parameter tying, as will be explicated shortly.
[2] There are $M-1$ component mass free parameters.

parameter value or an alternative value shared for possible use by every component. To represent whether component j uses a specific or the shared value for the mean of scalar feature X_k, the binary parameter $u_{jk} \in \{0, 1\}$ is introduced, with $u_{jk} = 1$ if a component-specific value is used and $u_{jk} = 0$ otherwise. Similarly, binary parameters $\{v_{jk}\}$ on variances and $\{w_{jkl}\}$ on off-diagonal covariates are introduced. The full set of parameters usable by component j in our model is thus: $\tilde{\theta}_j = \{\{\mu_{jk}\}, \{\sigma^2_{jk}\}, \{\sigma_{jkl}\}, \{\mu_{sk}\}, \{\sigma^2_{sk}\}, \{\sigma_{skl}\}, \{u_{jk}\}, \{v_{jk}\}, \{w_{jkl}\}\}$, with the first three subsets the component-specific means, variances, and covariates, and the next three subsets the *shared* means, variances, and covariates, respectively. The total parameter set at order M is $\theta(M) = \{\{\tilde{\theta}_j\}, \{\alpha_j\}\}$. Based on the binary switch values, the mean vector $\underline{\tilde{\mu}}_j$ and covariance matrix $\tilde{\mathbf{\Sigma}}_j$ used in (2.1) are given by:

$$\begin{aligned} \tilde{\mu}_{jk} &= u_{jk}\ \mu_{jk} + (1 - u_{jk})\ \mu_{sk} \\ \tilde{\sigma}^2_{jk} &= v_{jk}\ \sigma^2_{jk} + (1 - v_{jk})\ \sigma^2_{sk} \\ \tilde{\sigma}_{jkl} &= w_{jkl}\ \sigma_{jkl} + (1 - w_{jkl})\ \sigma_{skl}. \end{aligned} \tag{2.2}$$

In the sequel, $\tilde{\sigma}^{jkl}$ will refer to terms of the *inverse* covariance matrix $\tilde{\mathbf{\Sigma}}_j^{-1}$.

2.2 Bayesian Information Criterion for Learning and Model Selection

This paper extends previous work [10], where a method for learning mixture models with *naive Bayes* form was developed. As in [10], we propose to learn models and select the best model order (number of components) consistent with minimization of the *Bayesian Information Criterion/Minimum Description Length* (BIC/MDL). BIC is a theoretically grounded estimator [6] that is widely used for mixture modeling [5],[7]. In [10], it was shown for naive Bayes mixtures that, so long as the mixture model affords a wide range of model parsimony/data fitting tradeoffs (achieved by allowing flexible degrees of parameter sharing), BIC minimization yields reasonable model order estimates even for very high feature dimensions and small sample sizes (as seen,

e.g. for clustering text document databases). By contrast, model order selection applied to standard mixtures, which can only vary complexity by changing the number of components, *grossly* underestimates the true order in high dimensions [10].

The BIC/MDL cost (1.2) can be rewritten generically in the form:

$$BIC(\Theta(M), \mathcal{X}) = CC(M, \Theta(M)) - \sum_{i=1}^{N} \log\Big(\sum_{j=1}^{M} \alpha_j p[\underline{x}_i|\tilde{\theta}_j]\Big), \tag{2.3}$$

with $CC(\cdot)$ the number of bits describing the model. In [10], a simple yet efficient coding scheme for $\Theta(M)$ was devised, with associated model codelength $CC(\cdot)$. Here, this scheme is extended to address the multivariate Gaussian case. In [10], feature *distribution* sharing was employed – for each component, there was one binary parameter associated with each feature, indicating that *all* parameters specifying the feature's distribution (*e.g.*, the mean and variance) either use i) component-specific values or ii) shared values, available to all the mixture components. Here, instead, we invoke *parameter* sharing, with a distinct binary parameter (switch) associated with every variable: means, variances, and covariates, as indicated in (2.2). The overall model complexity can thus be additively decomposed into four terms, the first to specify the M component masses[3], with the other three corresponding to each variable type (mean, variance, covariance):

$$CC(M, \Theta(M)) = \frac{M-1}{2} \log N + CC_u(\{u_{jk}\}, \{\tilde{\mu}_{jk}\}) + CC_v(\{v_{jk}\}, \{\tilde{\sigma}^2_{jk}\}) + CC_w(\{w_{jkl}\}, \{\tilde{\sigma}_{jkl}\}). \tag{2.4}$$

To specify the terms in (2.4) we must add up the description lengths for every real-valued scalar parameter, and for all binary switch variables. We focus on efficient coding of these latter variables. A naive choice would be to estimate the codelength for all the switch variables as $M(d(d-1)/2 + 2d)$ bits, *i.e.* assuming all configurations of switch variables are equally likely. However, such a coding scheme is inefficient if many parameters are either purely component-specific, or purely shared, across all components. We would expect the latter to apply for small

[3]There are only $M-1$ free parameters since the masses must sum to one, with a $\frac{1}{2}\log(N)$ description length for each scalar, real-valued parameter in the BIC framework.

sample sizes. Thus, we devise a coding scheme that separately considers, for each parameter, the cases where i) all components use a shared value; ii) all components use a component-specific value; iii) some components use the shared value and some use component-specific values [10], with high codelength efficiency achieved in cases i) and ii). Since the same coding scheme is applied for the means, variances, and covariates, the cost terms $CC_u()$, $CC_v()$, and $CC_w()$ have identical forms; thus, to specify the complete model complexity $CC(M, \Theta(M))$ we need only explicitly discuss coding of the means, with this development also specifying the form of $CC_v()$ and $CC_w()$.

In our scheme, we first use $d \log_2(3)$ bits to specify, for all d mean parameters, which case applies (i), ii), or iii)). Then, for each scalar mean, based on the relevant case, more information is needed to specify the parameter values across components. Consider the means $\{\tilde{\mu}_{jk}, j = 1, \ldots, M\}$ associated with feature k. Under case i) ($\sum_j u_{jk} = 0$), only the shared parameter value needs to be described, at cost $\frac{1}{2}\log(N)$ bits. Under case ii) ($\sum_j u_{jk} = M$), we must describe all M values, at cost $\frac{M}{2}\log(N)$. For case iii), M bits specify the switch value for each component, $\frac{1}{2}(\sum_{j=1}^{M} u_{jk}) log(N)$ bits describe the component-specific means, and $\frac{1}{2} log(N)$ bits specify the shared mean. The overall code length for the means is thus:

$$\begin{aligned} CC_u(\{u_{jk}\}, \{\tilde{\mu}_{jk}\}) =& d \log 3 + \sum_{k=1}^{d} \Bigg[F_1(\mathbf{u_k}) \frac{1}{2} \log N + F_2(\mathbf{u_k}) \frac{M}{2} \log N + \\ & F_3(\mathbf{u_k}) \Bigg(\frac{1}{2} \log N + \sum_{j=1}^{M} u_{jk} \frac{1}{2} \log N + M \log 2 \Bigg) \Bigg], \end{aligned} \tag{2.5}$$

where we have applied the set definition $\mathbf{u_k} \equiv \{u_{jk}, j = 1, \ldots, M\}$, and where

$$F_1(\mathbf{u_k}) = \begin{cases} 1 & \text{if } \sum_{j=1}^{M} u_{jk} = 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.6}$$

$$F_2(\mathbf{u_k}) = \begin{cases} 1 & \text{if } \sum_{j=1}^{M} u_{jk} = M \\ 0 & \text{otherwise} \end{cases} \tag{2.7}$$

$$F_3(\mathbf{u_k}) = \begin{cases} 1 & \text{if } 0 < \sum_{j=1}^{M} u_{jk} < M \\ 0 & \text{otherwise} \end{cases}$$

The $d\log(3)$ term does not depend on M or any of the switch variables and can be removed. The complexity cost for the variances $CC_v(\cdot)$ has an identical form. The complexity cost of the covariate parameters also has the same form, except with $\frac{d(d-1)}{2}$ parameters of this type, rather than d. Ignoring terms independent of M and the switch variables, the *overall model complexity* is thus the sum (2.4), based on $CC_u(M, \{u_{jk}\})$ with the $d\log(3)$ term removed,

$$\begin{aligned} CC_v(M, \{v_{jk}\}) = &\sum_{k=1}^{d} \Bigg[\big(F_1(\mathbf{v_k}) + F_3(\mathbf{v_k})\big) \frac{1}{2} \log N + F_2(\mathbf{v_k}) \frac{M}{2} \log N + \\ &F_3(\mathbf{v_k}) \sum_{j=1}^{M} \left(v_{jk} \frac{1}{2} \log N \right) + F_3(\mathbf{v_k}) M \log 2 \Bigg], \quad \text{and} \end{aligned} \tag{2.8}$$

$$\begin{aligned} CC_w(M, \{w_{jkl}\}) = &\sum_{k=1}^{d-1} \sum_{l=k+1}^{d} \Bigg[\big(F_1(\mathbf{w_{kl}}) + F_3(\mathbf{w_{kl}})\big) \frac{1}{2} \log N + F_2(\mathbf{w_{kl}}) \frac{M}{2} \log N + \\ &F_3(\mathbf{w_{kl}}) \sum_{j=1}^{M} \left(w_{jkl} \frac{1}{2} \log N \right) + F_3(\mathbf{w_{kl}}) M \log 2 \Bigg]. \end{aligned} \tag{2.9}$$

2.3 Learning the Model

For an M-component mixture, there are $2^{M(d(d-1)/2+2d)}$ global shared-parameter configurations, representing a large set of model complexity/data fitting tradeoffs. In this section, we develop a generalized Expectation-Maximization (GEM) algorithm [23] to locally minimize BIC over these configurations and over all other parameters in $\Theta(M)$, given fixed M. This minimization at fixed M is coupled (in Section 2.5) with model order reduction to jointly optimize over both model orders and the parameters at a given order. The reader interested in learning the basic theory behind Expectation Maximization algorithms is referred to [26] and [27].

In the M-step of the EM algorithm, one chooses all parameter values to globally minimize the expected penalized log-likelihood [24]. For some models (as is the case here), it is intractable to achieve this joint optimization over the full parameter set. However, it may still be possible to

break the parameter set into subsets such that, keeping all other parameters fixed, one can either globally minimize with respect to a given subset or, at least, produce parameter updates for this subset with guaranteed monotonic descent in the expected penalized log-likelihood. One can then define cyclical minimization over the subsets, revisiting each parameter subset in turn given values for all other subsets held fixed. Algorithms based on these parameter subset optimizations are called *generalized* EM algorithms [23],[25]. The GEM approach is required for optimizing over our parameter set, $\Theta(M)$, due in particular to nonlinear dependencies involving the covariate and switch parameters. For our GEM algorithm, we partition $\Theta(M)$ into parameter subsets of each type and apply a suitable optimization method for each subset. Specifically, for each *iteration* of our GEM algorithm, we perform the following:

1. a standard E-step, given all parameter values fixed;
2. a generalized M-step which sequentially optimizes over each variable type given parameters of other types fixed:
   (a) a closed form M-step for updating $\{\alpha_j\}$ given the E-step result;
   (b) generalized M-step updates for the component-specific means, then the component-specific variances, followed by gradient descent for component-specific covariates;
   (c) generalized M-step for shared means and gradient descent for shared variances and covariates;
   (d) generalized M-step updates, in turn, for the switch variable subsets $\{u_{jk}\}$,$\{v_{jk}\}$,$\{w_{jkl}\}$.

Each parameter update within our GEM iteration is a descent step in BIC (2.3) [24]. The overall algorithm thus descends in BIC at fixed order, M. We next specify each GEM iteration step.

2.3.1 E-Step

We treat as *hidden data* within EM [24] the (exclusive) component of origin for each data point, $Z_{ij} \in \{0, 1\}$, where $Z_{ij} = 1$ if $\underline{x_i}$ was generated by component j. Then, based on the current

parameters, we compute the expected hidden data values via Bayes rule:

$$E[Z_{ij}|\mathcal{X};\Theta(M)] = P[Z_{ij}=1|\underline{x_i};\Theta(M)] = \frac{P[Z_{ij}=1]p[\underline{x_i}|Z_{ij}=1;\tilde{\theta}_j]}{p[\underline{x_i}|\Theta(M)]} = \frac{\alpha_j p[\underline{x_i}|\tilde{\theta}_j]}{\sum_{m=1}^{M}\alpha_m p[\underline{x_i}|\tilde{\theta}_m]}. \tag{2.10}$$

Given these probabilities, the expected complete (penalized) log likelihood is:

$$E[BIC_c(\Theta(M);\mathcal{X})] = CC(M,\Theta(M)) - \sum_{i=1}^{N}\sum_{j=1}^{M}\left[P[Z_{ij}=1|\underline{x_i}]\cdot\left(\log\alpha_j + \log p[\underline{x_i}|\tilde{\theta}_j]\right)\right]. \tag{2.11}$$

We next develop generalized M-step updates for $\Theta(M)$ which are strictly nonincreasing in (2.11). The theory behind the EM algorithm [24] ensures that these updates (and, *moreover*, alternating E-steps and generalized M-step updates, comprising iterations) are also nonincreasing in (1.2), the BIC cost which is our true objective function.

2.4 Generalized M-Step

In the sequel, in defining our GEM steps, updated parameters will be denoted using either a subscript or superscript 'new', with the notation for current (*fixed*) parameter values left unmodified.

2.4.1 Component-specific Mass Updates

Setting partial derivatives of (2.11) with respect to the $\{\alpha_j\}$ equal to zero, and constraining the sum of the masses to equal one, we arrive at (standard) closed form M-step updates for the component masses:

$$\alpha_j^{\text{new}} = \frac{1}{N}\sum_{i=1}^{N} P[Z_{ij}=1|\underline{x_i}] \qquad \forall j \tag{2.12}$$

2.4.2 Component-specific Mean Updates

After inserting (2.1) into (2.11), if we likewise take partial derivatives with respect to $\{\mu_{jk}\}$ and set to zero, we find the necessary optimality conditions given below:

$$\mu_{jk} = \frac{\sum_{i=1}^{N} P[Z_{ij}=1|\underline{x_i}] \left[\tilde{\sigma}^{jkk}\ x_{ik} - \sum_{\substack{l=1 \\ l\neq k}}^{d} \tilde{\sigma}^{jkl}\ (x_{il} - \tilde{\mu}_{jl}) \right]}{\sum_{i=1}^{N} P[Z_{ij}=1|\underline{x_i}]\ \tilde{\sigma}^{jkk}} \quad \forall j,k. \tag{2.13}$$

Since these conditions do not decouple the μ_{jk} variables[4] (with dependence also on the inverse covariates), we *further* partition the mean variables into subsets across components, *i.e.* $\{\{\mu_{jk}, j=1,\ldots,M\}, k=1,\ldots,d\}$, with sequential (global) optimization performed in turn over each mean parameter subset, for increasing k, *i.e.* the updates:

$$\mu_{jk}^{\text{new}} = \frac{\sum_{i=1}^{N} P[Z_{ij}=1|\underline{x_i}] \left[\tilde{\sigma}^{jkk}\ x_{ik} - \sum_{l<k} \tilde{\sigma}^{jkl}\ (x_{il} - u_{jl}\mu_{jl}^{\text{new}} - (1-u_{jl})\mu_{sl}) - \sum_{l>k} \tilde{\sigma}^{jkl}\ (x_{il} - \tilde{\mu}_{jl}) \right]}{\sum_{i=1}^{N} P[Z_{ij}=1|\underline{x_i}]\ \tilde{\sigma}^{jkk}} \quad \forall j,k. \tag{2.14}$$

Note that for computing μ_{jk}^{new}, we are using all newly updated (component-specific) means μ_{jl}^{new}, $l < k$. Note further that, as indicated in (2.14), we apply updates to *all* component-specific parameters during every GM-step even though some may not be in use due to the state of the switches, *e.g.* if $u_{jk} = 0$, then $\tilde{\mu}_{jk}$, and therefore (2.11), has no dependence on μ_{jk}. It is not *immediately* necessary to update this parameter. However, subsequently, we will optimize the $\{u_{jk}\}$ switches given all other parameters fixed. In this step, the updated component-specific parameter, given by (2.14), will be needed in order to best evaluate the optimal binary state of u_{jk}. Thus, even if $u_{jk} = 0$ currently, we still update μ_{jk} to anticipate possible switching to $u_{jk} = 1$.

[4] Note in particular that $\tilde{\mu}_{jl} = u_{jl}\ \mu_{jl} + (1-u_{jl})\ \mu_{sl}$.

2.4.3 Component-specific Variance Updates

The component-specific variance GEM step closely resembles the mean step above, with the same parameter subset partitioning across components, with the effect of updates for one parameter subset "propagated" to the next visited subset, and with updates of *all* variance parameters, irrespective of the values of the switch variables, in order to "anticipate" subsequent optimization of switches on the variance variables. The variance updates, given below, are derived in Appendix A:

$$\sigma^2_{jk_{\text{new}}} = \frac{\sum_{i=1}^{N} P[Z_{ij}=1|\underline{x_i}] \left[\sum_{p=1}^{d}\sum_{q=1}^{d}(x_{ip}-\tilde{\mu}^{\text{new}}_{jp})(x_{iq}-\tilde{\mu}^{\text{new}}_{jq})\tilde{\sigma}^{jkp}_{\text{new}}\tilde{\sigma}^{jkq}_{\text{new}} - \tilde{\sigma}^{jkk}_{\text{new}} \sum_{\substack{p=1 \\ p\neq k}}^{d} \tilde{\sigma}^{jkp}_{\text{new}}\tilde{\sigma}_{jkp} \right]}{\left(\tilde{\sigma}^{jkk}_{\text{new}}\right)^2 \sum_{i=1}^{N} P[Z_{ij}=1|\underline{x_i}]} \quad \forall j,k. \tag{2.15}$$

Some clarification of (2.15) is required. First, in the above equation, if a mean parameter is currently component-specific, we are defining $\tilde{\mu}^{\text{new}}_{jp} = \mu^{\text{new}}_{jp}$, whereas if the parameter is currently shared, we are defining $\tilde{\mu}^{\text{new}}_{jp} = \mu_{sp}$. Second, we must clarify what is meant by $\tilde{\sigma}^{jkq}_{\text{new}}$ in (2.15). Consider the update of a variance parameter subset, $\{\sigma^2_{jk}, j = 1, \ldots, M\}$. For every component j with $v_{jk} = 1$, this parameter subset update effects a change to the component's covariance matrix $\tilde{\mathbf{\Sigma}}_j$, and thus to the inverse covariance matrix $\tilde{\mathbf{\Sigma}}^{-1}_j$. To reflect this, we must recompute the inverse covariance matrix for every affected component, in-between variance subset optimizations, in order to use the proper inverse covariate values (as indicated by the "new" subscript) for updating the next parameter subset $(k+1)$. Since each variance update makes a rank-one modification to a covariance matrix, efficient ($O(d^2)$, rather than $O(d^3)$) covariance matrix inversion can be done, based on the well-known Sherman-Morrison-Woodbury identity (See Appendix B for details). Note also that for the first subset ($k = 1$), the "new" inverse covariate values are just the current values.

2.4.4 Component-specific Covariates

Unfortunately, no closed form update exists for component-specific covariance parameters, even considering parameter subsets of size one. Thus, as a GM step for these parameters, we perform gradient descent, based on the partial derivative form given below:

$$\frac{\partial}{\partial \sigma_{jkl}} E[BIC_c(\Theta(M), \chi)] = \sum_{i=1}^{N} P[Z_{ij} = 1 | \underline{x_i}] \left[\tilde{\sigma}^{jkl} - \frac{1}{2} \sum_{p=1}^{d} \sum_{q=1}^{d} (x_{ip} - \tilde{\mu}_{jp})(x_{iq} - \tilde{\mu}_{jq})(\tilde{\sigma}^{jpk} \tilde{\sigma}^{jql} + \tilde{\sigma}^{jpl} \tilde{\sigma}^{jqk}) \right].$$

The covariate parameters are partitioned into $d(d-1)/2$ subsets of size M, each containing a single covariate taken across all components. Gradient descent is performed sequentially over these parameter subsets, with covariance matrix inversions needed both between parameter subset optimizations and within parameter subset optimizations, after each gradient step. Again, efficient ($O(d^2)$) matrix inversion can be performed after each covariate update, using a simple algorithm described in Appendix B.

2.4.5 Shared Mean Updates

The shared mean parameters can also be updated in closed form, sequentially, as given below:

$$\mu_{sk}^{new} = \frac{\sum_{i=1}^{N} \sum_{j=1}^{M} (1 - u_{jk}) P[Z_{ij} = 1 | \underline{x_i}] \left[x_{ik} - \sum_{l<k} \frac{\tilde{\sigma}^{jkl}}{\tilde{\sigma}^{jkk}} (x_{il} - u_{jl}\mu_{jl} - (1 - u_{jl})\mu_{sl}^{\text{new}}) - \sum_{l>k} \frac{\tilde{\sigma}^{jkl}}{\tilde{\sigma}^{jkk}} (x_{il} - \tilde{\mu}_{jl}) \right]}{\sum_{i=1}^{N} \sum_{j=1}^{M} (1 - u_{jk}) P[Z_{ij} = 1 | \underline{x_i}]}$$

$$\forall k \;\; s.t. \sum_{j} v_{jk} < M. \tag{2.16}$$

Again, the updated shared mean μ_{sk}^{new} is used in computing μ_{sl}^{new} for $l > k$. Note also that if $\sum_{j=1}^{M} u_{jk} = M$, there is no dependence in (2.11) on μ_{sk}. However, unlike the case of a component-

specific parameter, it is *not* straightforward to choose μ_{sk} to optimally anticipate possible switching within $\mathbf{u_k}$ – the number of such switch configurations that use the shared value is $2^M - 1$, each with its *own* optimal μ_{sk} value. It clearly becomes impractical to evaluate all these anticipatory shared values as M increases. We suggest a practical alternative to this in Section 2.4.8.

2.4.6 Shared Covariate Updates

Unfortunately, there is no closed form update for shared variance and shared covariance parameters. Thus, we again invoke gradient descent to update these parameters, based on the partial derivatives given below:

$$\begin{aligned}
&\frac{\partial}{\partial \sigma_{sk}^2} E[BIC_c(\Theta(M), \chi)] = \\
&\quad \sum_{i=1}^{N}\sum_{j=1}^{M}(1 - v_{jk})P[Z_{ij} = 1|\underline{x_i}]\left[\frac{\tilde{\sigma}^{jkk}}{2} - \frac{1}{2}\sum_{p=1}^{d}\sum_{q=1}^{d}(x_{ip} - \tilde{\mu}_{jp})(x_{iq} - \tilde{\mu}_{jq})(\tilde{\sigma}^{jpk}\tilde{\sigma}^{jqk})\right] \\
&\frac{\partial}{\partial \sigma_{skl}} E[BIC_c(\Theta(M), \chi)] = \\
&\quad \sum_{i=1}^{N}\sum_{j=1}^{M}(1 - w_{jkl})P[Z_{ij} = 1|\underline{x_i}]\left[\tilde{\sigma}^{jkl} - \frac{1}{2}\sum_{p=1}^{d}\sum_{q=1}^{d}(x_{ip} - \tilde{\mu}_{jp})(x_{iq} - \tilde{\mu}_{jq})(\tilde{\sigma}^{jpk}\tilde{\sigma}^{jql} + \tilde{\sigma}^{jpl}\tilde{\sigma}^{jqk})\right].
\end{aligned} \tag{2.17}$$

Consistent with our above discussion on shared means, we only perform gradient descent on σ_{sk}^2 if $\sum_j v_{jk} < M$ and on σ_{skl} if $\sum_j w_{jkl} < M$. For the cases where shared variances or covariates are not currently being used, our practically viable approach to updating these shared parameters is described in Section 2.4.8.

2.4.7 Optimizing the switch parameters

Now, fixing the component-specific and shared parameters, we would like to determine the switch configuration which minimizes (2.11). Unfortunately, with $2^{\frac{1}{2}M(3d+d^2)}$ possible configurations, it is infeasible to exhaustively search for the globally optimal solution. However, following [10], we are able to apply GEM updates for these parameters, ensuring monotonic descent in BIC, in

an efficient manner by in turn optimizing over the parameter subsets: $\mathbf{u_k}, k = 1, \ldots, d$, $\mathbf{v_k}, k = 1, \ldots, d$, and $\mathbf{w_{kl}}$, $\forall k, l$, while holding all other parameters in $\Theta(M)$ fixed.

To optimize a particular parameter subset, we calculate the expected complete data BIC cost of the best configuration for each of the three cases (i) all switches on; ii) all switches off; iii) some switches on and some off) and choose the one with least cost. In this way we achieve global minimization over all possible configurations for this parameter *subset* (given all other parameters fixed). For both cases i) and ii) there is only one switch configuration to evaluate. For case iii), there are $2^M - 2$ possible switch configurations. To find the optimal switch settings for case iii), we note that (2.11) can be additively decomposed over terms which each depend on only one member of the switch subset. For example, given $F_3(\mathbf{u_k}) = 1$ (signifying case iii)), we can write:

$$E[BIC_c(\Theta(M), \mathcal{X}] = C_0 + \sum_{j=1}^{M} \left[\left(u_{jk} \frac{1}{2} \log N \right) - \sum_{i=1}^{N} \left(P[Z_{ij} = 1 | \underline{x_i}] \cdot \log p[\underline{x_i} | \tilde{\theta}_j] \right) \right], \qquad (2.18)$$

where C_0 is a term that has no dependence on variables from $\mathbf{u_k}$, and where we note that each term inside the sum over j has dependence on $\mathbf{u_k}$ *only through* the single switch u_{jk}. Thus, the optimal switch configuration for case iii) is found by individually choosing $u_{jk} = u^*_{jk} \forall j$, where:

$$u^*_{jk} = \begin{cases} 1 \text{ if } \left(\frac{1}{2} \log N \right) - \sum_{i=1}^{N} \left(P[Z_{ij} = 1 | \underline{x_i}] \cdot \log p[\underline{x_i} | \tilde{\theta}_j] \right) \Big|_{u_{jk}=1} < - \sum_{i=1}^{N} \left(P[Z_{ij} = 1 | \underline{x_i}] \cdot \log p[\underline{x_i} | \tilde{\theta}_j] \right) \Big|_{u_{jk}=0} \\ 0 \text{ otherwise} \end{cases} \qquad (2.19)$$

Then, calculating the expected BIC cost for case iii) by plugging our optimal switch values above into (2.11), and comparing this value with the costs from cases i) and ii), we choose the subset configuration which minimizes (2.11). That is, the optimal switch configuration must now be either $u_{jk} = 1 \forall j$, $u_{jk} = 0 \forall j$, or $u_{jk} = u^*_{jk} \forall j$. Whichever one provides the least BIC cost is chosen. This process is sequentially performed, in turn, on every subset $\mathbf{u_k}, k = 1, \ldots, d$, $\mathbf{v_k}, k = 1, \ldots, d$, and $\mathbf{w_{kl}} \forall k, l$ applying the same discrete optimization strategy. The EM framework is pivotal to this switching optimization method – the expected complete data BIC cost is additive over terms

that have dependence on only a *single* switch within any switch variable subset (*e.g.*, $\mathbf{u_k}$) that we jointly optimize. By contrast, the BIC cost itself has complex dependence on all variables within a given switch variable subset.

2.4.8 Unused Shared Parameter Updates

As discussed earlier, updates for shared parameters, *e.g.* shared means, are well-defined so long as $\sum_j u_{jk} < M$. If $\sum_j u_{jk} = M$, unlike the case of unused component-specific parameters, there does not exist a simple, single shared parameter update which can optimally predict future switching. Instead, we follow the heuristic method of [10] to search for an improved shared parameter and switch configuration. For each shared parameter $p \in \{\mu_{sk}, \sigma^2_{sk}, \sigma_{skl}\}$ and its corresponding switch subset $\mathbf{s} = \mathbf{u_k}$,$\mathbf{v_k}$, or $\mathbf{w_{kl}}$ for which $\sum_j s_j = M$, we perform the following steps for each $j \in \{1, \ldots, M\}$:

1. Trial-set p to its corresponding component specific parameter, μ_{jk}, σ^2_{jk}, or σ_{jkl}.

2. Determine the optimal $\mathbf{s}$ configuration (see Section 2.4.7) given these trial-updated parameters.

3. If $\sum_j s_j < M$, update p via its GEM update rule described earlier, evaluate the new cost (2.11), and record this trial's BIC cost and parameter value p.

After trial-setting the shared parameter based on each single component-specific parameter as described above, we choose from the ($\leq M$) candidate settings the one that yields the lowest BIC cost. If none of the trial-settings gives a configuration with $\sum_j s_j < M$, we simply keep our current parameter value p.

2.5 Model Order Reduction

The previous subsections described how we minimize BIC for a given model order M. To also optimize over M, we first set the model order to $M = M_{max}$ (chosen to overestimate the number

of components). The GEM algorithm is then run to minimize BIC at this maximum order. Next, we remove one component and relearn the model at the reduced order. This procedure is repeated, reducing the model all the way down to a single component while recording the BIC cost at each order. The final selected model (and associated order) is the one with smallest BIC cost.

Various methods exist for choosing which component to delete during the model order reduction step. In [28], the authors *trial*-delete each component, relearn M candidate reduced-order models, and retain the reduced model with least cost. However, since our learning is computationally complex, trial-deletions are a time-consuming luxury. Instead, we follow [9] and [10], simply deleting the component with least mass.

2.6 Implementation Details

We preprocess the data by normalizing to zero mean and unit variance. Starting estimates for the $\{\alpha_j\}$ and $\{\mu_{jk}\}$ parameters are then initialized using the K-means algorithm with $K = M_{max}$ initial centers randomly chosen from the data points. Component-specific variances and covariances are initialized as the maximum likelihood estimates based on the clusters resulting from K-means. Initial shared means are set to zero and shared variances are set to 1 (the maximum likelihood values for the entire normalized data set). At all times during the optimization, variances are constrained to be at least 0.1 to avoid singularities.

For the gradient descent minimizations, a momentum term was added, with coefficient 0.75 (weighting the previous parameter update). Due to much larger gradient magnitudes for shared covariates, different learning rates were chosen for the component-specific and shared covariate (and variance) gradient rules and used on all tested data sets – 10^{-3} for component-specific parameters and 3×10^{-5} for shared parameters. For each (trial) gradient step taken, positive definiteness of the relevant covariance matrices is checked. If a matrix is no longer positive definite, the step length is halved and another trial gradient step is taken. Gradient steps are also

only retained if they lead to decreases in the BIC cost. Gradient descent for a given parameter is terminated if the change in parameter value from one step to the next falls below 10^{-7}. Finally, updates for unused shared parameters are performed quite infrequently – roughly three times, spaced out during optimization at a given model order, M. The overall algorithm was deemed converged at a given order when the change in BIC cost from one iteration to the next was negligible.

2.6.1 Computational Complexity

The greatest computation is required for updating covariances, with the associated matrix inversions. The complexity for updating all covariate parameters for all components is bounded by $O(Md^4 + MNd^2)$, with the first term associated with $O(Md^2)$ matrix inversions and the second term due to gradient descent updates.

2.6.2 Avoiding singular components

Several heuristics were also introduced to avoid the well-known problem of producing singular covariance matrix estimates [7]. First, at each order M, the model is initially forced to use a shared representation for all covariate parameters by fixing $w_{jkl} = 0 \ \forall j, k, l$. This ensures that good shared covariate models are available at the outset at each model order. Second, we only allow a component-specific covariate to switch on ($w_{jkl} = 1$) if the component has "sufficient" mass, considering the sample size relative to the feature dimensionality, *i.e.* only if $\alpha_j \cdot N > K * d$, with K a chosen threshold. In addition, the shared parameter, σ_{skl} is only updated as part of the GM step if $\sum_{j=1}^{M}(1 - w_{jkl})\alpha_j \cdot N > K * d$. In all our tests, the threshold value $K = 2.25$ was used as this was found to avoid singularities and provide less variance in the learned models without compromising our method for relatively large datasets (for which singular covariances are less of an issue). The complexity cost term which describes covariances must also be accordingly modified when these heuristic rules are active. In particular, assuming a coding scheme where

component masses are transmitted to the decoder first, the decoder can infer whether or not covariate switching is allowed for a given component j, based on the component's mass. If it is not allowed, there is no need to describe the parameters $\{w_{jkl} \forall k, l\}$. However, updated shared covariate parameters $\sigma_{skl} \forall k, l$ must be specified to the decoder in this case since component j (if not other components) will necessarily use shared representations for all covariates.

Chapter 3

Experimental Results

Experimental results comparing our method to two others are presented. The first is a method which uses switching structure and learning steps identical to our proposed method for means and variances, but while assuming diagonal covariances (all covariates zero), *i.e.* the naive Bayes assumption. This is essentially the Gaussian version of [10], except that separate switches are used on mean and variance parameters, instead of a single switch controlling both. Our method is also compared to standard multivariate Gaussian models learned via EM, with model order selection minimizing BIC.

3.1 Experimental Setup

Our models are learned in a completely unsupervised manner, *i.e.* without labeled examples, and without knowledge of the number of ground truth classes. However, if available, this information is used when evaluating the performance of the learned model. The unsupervised learning models are evaluated using three performance criteria:

1) accuracy in estimating the number of components

2) test set log-likelihood

3) test set classification accuracy

After model learning, test set classification accuracy is calculated by the following steps:

Step 1) assign each point of the test set to a component based on a maximum likelihood decision.

Step 2) hard-associate each component to the ground truth class from which the component owns the most data.

Step 3) measure the fraction of the data points assigned to components with the correct class label.

In addition to unsupervised learning, we also applied our method to create plug-in Bayes classifiers. Here, the labels of the training data are used to split the data into separate subsets, one per class, with a class-conditional GMM then learned for each class and used in a plug-in Bayes classification rule. The classifiers are evaluated for test set classification accuracy.

3.2 Synthetic Data

Syntheic Data sets of $N = 3000$ points were generated, each with $d = 7$, based on a 3-component GMM with equal component masses and with other parameters specified as:

$$\mu_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \qquad \Sigma_1 = \begin{bmatrix} 4 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 2 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -\frac{3}{4} & -\frac{4}{5} \\ 0 & 0 & 0 & 0 & -\frac{3}{4} & 1 & \frac{3}{4} \\ 0 & 0 & 0 & 0 & -\frac{4}{5} & \frac{3}{4} & 1 \end{bmatrix} \tag{3.1}$$

$$\mu_2 = \begin{bmatrix} 3 \\ -3 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \qquad \Sigma_2 = \begin{bmatrix} 1 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -\frac{3}{4} & -\frac{4}{5} \\ 0 & 0 & 0 & 0 & -\frac{3}{4} & 1 & \frac{3}{4} \\ 0 & 0 & 0 & 0 & -\frac{4}{5} & \frac{3}{4} & 1 \end{bmatrix} \tag{3.2}$$

$$\mu_3 = \begin{bmatrix} 3 \\ 3 \\ 3 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \qquad \Sigma_3 = \begin{bmatrix} 1 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -\frac{3}{4} & -\frac{4}{5} \\ 0 & 0 & 0 & 0 & -\frac{3}{4} & 1 & \frac{3}{4} \\ 0 & 0 & 0 & 0 & -\frac{4}{5} & \frac{3}{4} & 1 \end{bmatrix} \tag{3.3}$$

Note that features 4,5,6, and 7 are uninformative (all parameters for these features common across components). Components 2 and 3 share a mean for feature 1 and all variance and covariance parameters. Component 1 shares a mean for feature 3 with component 2, and has unique values for the variances of features 1 and 2, and for the covariates between features 1 and 2 and between features 2 and 3. All other parameters of the ground-truth components are shared.

Twenty different realizations of the above data were created, and for each realization our model was run for twenty different random initializations, using all the training data with $M_{max} = 20$. For *each* of the 400 runs, the ground-truth order of 3 was chosen. Only 6 of the 400 runs (1.5%) were able to capture the exact sharing structure, with all 105 switches matching the ground-truth sharing structure. However, only small deviations in switch structure were found on the other

runs. In total, evaluating all 105 switches for each of the 400 runs, 96.08% of the switches matched the ground-truth sharing structure.

Next, our method was compared to the naive Bayes and standard multivariate Gaussian methods. Each (of the 20) realizations of the synthetic data was split into training and test sets. For each of the three methods, for each data realization, models were learned starting from each of the 20 initializations, using 5%, 10%, and 50% of the data as a training set. Test set accuracy in classifying samples to ground-truth components and test set log-likelihood were calculated using the remaining portion of the data set, with the results averaged over all 400 (realization, initialization) trials. The results are summarized in Table 3.1.

Table 3.1. Selected model order, test set accuracy and log-likelihood comparisons for synthetic data of 3000 data points generated by 3 ground truth classes with feature dimension of 7

Data Set	Method	M^*		% test set acc.		test set log-likelihood	
		mean	(std dev)	mean	(std dev)	mean	(std dev)
50% train	proposed method	3.00	(0.00)	97.10	(0.33)	-10292.2	(31.6)
	NB with sharing	15.5	(2.58)	93.39	(1.03)	-11033.6	(75.7)
	std. Mult. Gaussian	3.00	(0.00)	96.82	(0.25)	-10360.7	(73.9)
10% train	proposed method	3.10	(0.31)	96.16	(6.37)	-18873.7	(110.9)
	NB with sharing	6.80	(0.75)	86.89	(3.35)	-20738.6	(252.4)
	std. Mult. Gaussian	2.30	(0.45)	75.08	(13.71)	-19448.6	(242.1)
5% train	proposed method	3.05	(0.22)	95.34	(1.11)	-20469.1	(154.9)
	NB with sharing	3.80	(0.87)	71.02	(9.77)	-22828.8	(342.2)
	std. Mult. Gaussian	1.60	(0.49)	52.96	(15.91)	-21657.1	(376.3)

Our method found the ground truth model order of 3 on nearly all runs, for all training sizes tested. The standard multivariate Gaussian method achieves similar results when the number of training points is large (50 % case), but regularly underestimates the order as the number of training points is lowered. As a result, test set accuracy and log-likelihood suffer severely at 10% and 5% training. Meanwhile, the naive Bayes model always performed significantly worse, attempting to model data which (ground-truth) includes large nonzero covariances. When the training data is large, the naive Bayes method chooses much higher orders in an attempt to well-fit the data (similar to Figure 1). Even at these high orders, test accuracy and log-likelihood are not comparable to performance of the other two multivariate models.

3.3 Real Data

We measured performance of our method and the two comparison methods on 6 real-world datasets from the UCI machine learning repository. For each dataset, the data was randomly split 10 times into training and test sets. Training sets were created from 50% of the overall data points, except on **magic**, where only 2.5% of the 19020 points are used for training. Performance was evaluated using our three criteria, and means and standard deviations are reported for all results. The datasets, in the order shown in Table 3.2, are: Wine, Statlog (Image Segmentation), Breast Cancer Wisconsin (Diagnostic), MAGIC Gamma Telescope, Statlog (Landsat Satellite) training set, and Connectionist Bench (Sonar, Mines vs. Rocks).

Table 3.2. Selected model order, test set accuracy and log-likelihood comparisons for 6 UCI Machine Learning Repository data sets. N represents the number of total data points, d the feature space dimensionality, and c the number of labeled classes in the data.

Data Set	Method	M^*		% test set acc.		test set log-likelihood	
		mean	(std dev)	mean	(std dev)	mean	(std dev)
wine	proposed method	3.60	(1.20)	93.15	(6.14)	-1372.1	(89.5)
N=178	NB with sharing	3.50	(0.59)	92.5	(3.11)	-1396.5	(51.2)
d=13, c=3	std. Mult. Gaussian	2.00	(0.00)	61.12	(8.68)	-1551.1	(100.3)
image	proposed method	24.40	(2.62)	82.60	(1.49)	4074.6	(716.7)
N=2320	NB with sharing	25.80	(2.57)	81.97	(2.11)	-2471.4	(427.5)
d=18, c=7	std. Mult. Gaussian	8.00	(1.83)	64.52	(3.54)	3487.0	(786.6)
wdbc	proposed method	5.20	(0.95)	92.24	(2.94)	-3008.4	(244.4)
N=569	NB with sharing	10.20	(1.43)	91.44	(1.78)	-7193.6	(327.3)
d=30, c=2	std. Mult. Gaussian	2.00	(0.00)	86.18	(11.14)	-3936.6	(442.5)
magic	proposed method	6.20	(0.75)	76.00	(1.09)	-131267.3	(2054.4)
N=19020	NB with sharing	8.58	(0.82)	75.10	(1.21)	-151911.8	(1763.9)
d=10, c=2	std. Mult. Gaussian	3.90	(0.70)	75.94	(1.10)	-138643.9	(2574.1)
landsat	proposed method	12.10	(1.22)	65.15	(5.06)	8971.8	(663.4)
N=6435	NB with sharing	47.90	(1.42)	65.22	(2.24)	-26416.7	(672.5)
d=36, c=6	std. Mult. Gaussian	3.70	(0.46)	51.39	(7.06)	8312.8	(793.9)
sonar	proposed method	5.80	(1.73)	66.20	(7.42)	-8169.2	(334.1)
N=208	NB with sharing	5.50	(1.37)	64.51	(4.45)	-8107.8	(280.4)
d=60, c=2	std. Mult. Gaussian	1.00	(0.00)	53.37	(3.08)	-10474.3	(1276.9)

Several trends are apparent. First, as one would expect, our method tends to produce order estimates which are higher than the standard multivariate Gaussian models, and lower than the naive Bayes models. As was witnessed for the synthetic data sets, BIC based order selection breaks down for the standard multivariate Gaussian method when not enough data is available to overcome the large complexity cost associated with using many components. The other two

methods use parameter sharing in order to provide needed flexibility when data is scarce relative to the dimensionality of the data. In many cases, the naive Bayes method chooses an order much higher than our method, which may be required for a naive Bayes Gaussian model to accurately describe data with significant covariance structure.

When evaluating trends in test set accuracy, we first must consider the effects of the models' much different order selection choices. First, it is no surprise that the accuracy and likelihood of the standard multivariate Gaussian model are considerably lower when an order is chosen below the number of classes in the data set. Even if the order is comparable to the number of classes, a class may require several components to achieve accurate class-conditional modeling. In table 3.2, there are clear cases where low model order selection leads to poor accuracy for the standard multivariate Gaussian model. Second, we must recognize that a naive Bayes model with many more components may in some cases be an equally strong classifier even for data with highly correlated features. However, a slight improvement in test set classification accuracy for our method over naive Bayes is an observed trend on these real world datasets. Our method also exhibits significantly higher test set log-likelihood than the naive Bayes models. As with our other performance criteria, the standard multivariate Gaussian method found lower test set log-likelihoods than our method, much lower for small data sets (sonar, wine).

The same datasets were also used to create plug-in Bayes classifiers as described in Section 3.1. Results are summarized in table 3.3. Again, we found the naive Bayes method to provide only slightly worse classification than our proposed method. The standard multivariate Gaussian also exhibited good performance on most data sets, except sonar.

Table 3.3. Test set classification accuracy for 6 UCI Machine Learning Repository data sets when used to create a Plug-in Bayes classifier. N represents the number of total data points, d the feature space dimensionality, and c the number of labeled classes in the data.

Data Set	Method	Plug-in Bayes test % acc. mean	(std dev)
wine	proposed method	96.20	(1.14)
N=178	NB with sharing	96.18	(0.72)
d=13, c=3	std. Mult. Gaussian	94.83	(3.64)
image	proposed method	93.31	(1.11)
N=2320	NB with sharing	93.16	(1.15)
d=18, c=7	std. Mult. Gaussian	91.08	(1.60)
wdbc	proposed method	94.70	(0.90)
N=569	NB with sharing	94.60	(0.78)
d=30, c=2	std. Mult. Gaussian	94.39	(0.81)
magic	proposed method	82.49	(1.66)
N=19020	NB with sharing	80.64	(1.51)
d=10, c=2	std. Mult. Gaussian	83.35	(1.23)
landsat	proposed method	87.17	(0.46)
N=6435	NB with sharing	88.03	(0.54)
d=36, c=6	std. Mult. Gaussian	87.52	(0.41)
sonar	proposed method	78.46	(3.21)
N=208	NB with sharing	77.98	(2.86)
d=60, c=2	std. Mult. Gaussian	64.23	(3.73)

Chapter 4

Conclusion

We have developed a parsimonious modeling and model order and structure selection method for multivariate Gaussian mixtures which allows for and optimizes over a rich set of tradeoffs between data representation and model complexity. Our objective was to learn the most accurate models (including accurate model order estimation) in the face of limited available data samples. This paper significantly extends [10], which was only suitable for parsimonious modeling of naive Bayes Gaussian (and non-Gaussian) mixtures. We have demonstrated that our GEM algorithm for (BIC-based) penalized likelihood minimization is effective and largely robust in learning model parameters and choosing the correct model order and model structure (tied versus component-specific parameters) for simulated data sets generated based on specified ground-truth mixtures. We also compared against parsimonious modeling of naive Bayes Gaussian mixtures [10] and against standard multivariate Gaussian mixtures (full covariances), for several evaluation criteria: i) estimating the number of components (when the ground-truth mixture is known), an important measure for data clustering; ii) test set log-likelihood, a suitable measure when the goal is density estimation; iii) unsupervised classification accuracy (on data sets where class labels are known but are only used post-learning for performance evaluation); iv) accuracy when class-conditional mixtures are learned and used in a plug-in Bayes classifier. Our parsimonious mixtures gave improved accuracy with respect to each of these performance measures, compared with naive

Bayes and full covariance Gaussian mixtures. We also demonstrated that our method can preserve crucial, class-discriminating covariance structure when the sample size is very low.

Appendix A

GEM update for component-specific variances

Here, we derive equation (2.15), which updates σ_{jk}^2 to minimize the expected BIC cost (2.11) while holding all other parameters fixed. Expanding the relevant terms of the expected BIC cost in (2.11), we begin by writing the following equation.

$$\begin{aligned} E[BIC_c(\Theta(M);\mathcal{X}] =& CC(M,\Theta(M)) - \\ & \sum_{i=1}^{N}\sum_{j=1}^{M}\left[P[Z_{ij}=1|\underline{x_i}]\cdot\left(\log\alpha_j - \frac{1}{2}\log\left[(2\pi)^d\right] - \right.\right. \\ & \left.\left.\frac{1}{2}\log\left|\tilde{\mathbf{\Sigma}}_j\right| - \frac{1}{2}\sum_{p=1}^{d}\sum_{q=1}^{d}(x_{ip}-\tilde{\mu}_{jp})(x_{iq}-\tilde{\mu}_{jq})\tilde{\sigma}^{jpq}\right)\right]. \end{aligned} \tag{A.1}$$

Next, we seek to minimize the above by taking the partial derivative with respect to σ_{jk}^2 and setting equal to zero, yielding:

$$\sum_{i=1}^{N} P[Z_{ij}=1|\underline{x_i}]\left[\frac{\tilde{\sigma}^{jkk}}{2} - \frac{1}{2}\sum_{p=1}^{d}\sum_{q=1}^{d}(x_{ip}-\tilde{\mu}_{jp})(x_{iq}-\tilde{\mu}_{jq})(\tilde{\sigma}^{jpk}\tilde{\sigma}^{jqk})\right] = 0. \tag{A.2}$$

Now, to complete our derivation, we must solve for σ_{jk}^2. Note that $\tilde{\sigma}^{jpk}$ and $\tilde{\sigma}^{jqk}$ have dependence on σ_{jk}^2 when $p \neq k$ and $q \neq k$ respectively. In order to separate σ_{jk}^2 from the other terms, we use cofactor expansion of the inverse covariance matrix terms. We define C_{jpk} as the cofactor of

term $\tilde{\sigma}_{jpk}$ of the covariance matrix and make use of the following two properties of cofactors:

$$\tilde{\sigma}^{jpk} = \frac{C_{jpk}}{|\tilde{\mathbf{\Sigma}}_j|} \tag{A.3}$$

$$|\tilde{\mathbf{\Sigma}}_j| = \sum_{\substack{p=1 \\ p\neq k}}^{d} C_{jkp}\tilde{\sigma}^{jkp} + C_{jkk}\sigma_{jk}^2 \tag{A.4}$$

Replacing all inverse covariance terms in (A.2) with the cofactor representation in (A.3) and multiplying both sides by $\frac{|\tilde{\mathbf{\Sigma}}_j|}{2}$, we arrive at:

$$\sum_{i=1}^{N} P[Z_{ij}=1|\underline{x_i}]\left[C_{jkk}|\tilde{\mathbf{\Sigma}}_j| - \sum_{p=1}^{d}\sum_{q=1}^{d}(x_{ip}-\tilde{\mu}_{jp})(x_{iq}-\tilde{\mu}_{jq})(C_{jpk}C_{jqk})\right] = 0 \tag{A.5}$$

We then solve for $|\tilde{\mathbf{\Sigma}}_j|$, since all other terms do not depend on σ_{jk}^2.

$$|\tilde{\mathbf{\Sigma}}_j| = \frac{\sum_{i=1}^{N} P[Z_{ij}=1|\underline{x_i}]\sum_{p=1}^{d}\sum_{q=1}^{d}(x_{ip}-\tilde{\mu}_{jp})(x_{iq}-\tilde{\mu}_{jq})(C_{jpk}C_{jqk})}{\sum_{i=1}^{N} P[Z_{ij}=1|\underline{x_i}]C_{jkk}} \tag{A.6}$$

Now, we use (A.4) to expand $|\tilde{\mathbf{\Sigma}}_j|$ so that we may solve for σ_{jk}^2. After some simplification we find:

$$\sigma_{jk}^2 = \frac{\sum_{i=1}^{N} P[Z_{ij}=1|\underline{x_i}]\left[\sum_{p=1}^{d}\sum_{q=1}^{d}(x_{ip}-\tilde{\mu}_{jp})(x_{iq}-\tilde{\mu}_{jq})(C_{jpk}C_{jqk}) - \sum_{\substack{p=1 \\ p\neq k}}^{d} C_{jkp}C_{jkk}\tilde{\sigma}^{jkp}\right]}{\sum_{i=1}^{N} P[Z_{ij}=1|\underline{x_i}]C_{jkk}^2}, \tag{A.7}$$

which may be put in a more useful form by multiplying by $\frac{|\tilde{\mathbf{\Sigma}}_j|^2}{|\tilde{\mathbf{\Sigma}}_j|^2}$ and making use of (A.3) to change the cofactor terms into their respective inverse terms, which are more readily available

for calculations. The final result:

$$\sigma_{jk}^2 = \frac{\sum_{i=1}^{N} P[Z_{ij}=1|\underline{x_i}] \left[\sum_{p=1}^{d} \sum_{q=1}^{d} (x_{ip}-\tilde{\mu}_{jp})(x_{iq}-\tilde{\mu}_{jq})(\tilde{\sigma}^{jkp}\tilde{\sigma}^{jkq}) - \tilde{\sigma}^{jkk} \sum_{\substack{p=1 \\ p\neq k}}^{d} \tilde{\sigma}^{jkp}\tilde{\sigma}^{jkp} \right]}{(\tilde{\sigma}_{jkk})^2 \sum_{i=1}^{N} P[Z_{ij}=1|\underline{x_i}]}, \quad \text{(A.8)}$$

leads directly to (2.15) after adding a subscript or superscript 'new' to the mean and inverse covariance matrix terms, signifying that we are using the most recent updates of these parameters in the GM-step for σ_{jk}^2.

Appendix B

Efficient Matrix Inversion

Efficient covariance matrix inversion following the modification of a variance term (or any other rank-one update) is a well known application of the Sherman-Morrison-Woodbury identity. The new matrix inverse and determinant may be calculated with the following formulae:

$$\Sigma_{new}^{-1} = (\Sigma + \Delta\sigma_k^2 \underline{e_k}\ \underline{e_k}^T)^{-1} = \Sigma^{-1} - \Sigma^{-1}\underline{e_k}(\frac{1}{\Delta\sigma_k^2} + \underline{e_k}^T\Sigma^{-1}\underline{e_k})^{-1}\underline{e_k}^T\Sigma^{-1} \tag{B.1}$$

$$\det(\Sigma_{new}) = \det(\Sigma + \Delta\sigma_k^2 \underline{e_k}\ \underline{e_k}^T) = (1 + \Delta\sigma_k^2 \underline{e_k}^T\Sigma^{-1}\underline{e_k})\det(\Sigma) \tag{B.2}$$

where Σ is the old covariance matrix, Σ_{new} is the updated covariance matrix where the k-th term on the diagonal has been modified by adding $\Delta\sigma_k^2$, and $\underline{e_k}$ is a column vector which is all zeros except for a 1 in the k-th entry. Using these equations, calculation of the determinant and inverse of Σ_{new} is an $O(d^2)$ operation when the determinant and inverse of Σ are known.

We present a similar, also $O(d^2)$, matrix inversion following the modification of a covariate term. Here, changing both terms in the symmetric matrix is a rank-two modification to the matrix. A simple, but not necessarily most efficient, approach involves two rank-one updates. We first calculate an intermediate inverse and determinant,

$$\Sigma_1^{-1} = (\Sigma + \frac{1}{2}\Delta\sigma_{pq}\underline{e_{pq}}\ \underline{e_{pq}}^T)^{-1} = \Sigma^{-1} - \Sigma^{-1}\underline{e_{pq}}(\frac{2}{\Delta\sigma_{pq}} + \underline{e_{pq}}^T\Sigma^{-1}\underline{e_{pq}})^{-1}\underline{e_{pq}}^T\Sigma^{-1} \tag{B.3}$$

$$\det(\Sigma_1) = \det(\Sigma + \frac{1}{2}\Delta\sigma_{pq}\underline{e_{pq}}\ \underline{e_{pq}}^T) = (1 + \frac{1}{2}\Delta\sigma_{pq}\underline{e_{pq}}^T\Sigma^{-1}\underline{e_{pq}})\det(\Sigma) \tag{B.4}$$

where $\underline{e_{pq}}$ is a column vector which is all zeros except for a 1 in both the p-th and q-th entries. Then, the inverse and determinant for the final updated matrix, Σ_{new}, are calculated by another rank-one update to this intermediate matrix.

$$\Sigma_{new}^{-1} = (\Sigma_1 - \frac{1}{2}\Delta\sigma_{pq}\underline{v_{pq}}\ \underline{v_{pq}}^T)^{-1} = \Sigma_1^{-1} - \Sigma_1^{-1}\underline{v_{pq}}(\frac{-2}{\Delta\sigma_{pq}} + \underline{v_{pq}}^T\Sigma_1^{-1}\underline{v_{pq}})^{-1}\underline{v_{pq}}^T\Sigma_1^{-1} \tag{B.5}$$

$$\det(\Sigma_{new}) = \det(\Sigma_1 - \frac{1}{2}\Delta\sigma_{pq}\underline{v_{pq}}\ \underline{v_{pq}}^T) = (1 - \frac{1}{2}\Delta\sigma_{pq}\underline{v_{pq}}^T\Sigma_1^{-1}\underline{v_{pq}})\det(\Sigma_1) \tag{B.6}$$

where $\underline{v_{pq}}$ is a column vector which is all zeros except for a 1 in the p-th entry and a -1 in the q-th entry (or vice-versa).

Bibliography

[1] H. Li and M. Niranjan, "Outlier detection in benchmark classification tasks", *Proc. ICASSP*, v. 5, pp. 557-560, 2006.

[2] R. O. Duda and P. E. Hart, "Pattern classification and scene analysis", *Wiley*, New York, 1973.

[3] D. J. Miller and H. Uyar,"A mixture of experts classifier with learning based on both labelled and unlabelled data", *NIPS*, vol. 9, pp. 571-577, 1997.

[4] K. Nigam, A. McCallum, S. Thrun, and T. Mitchell,"Text classification from labeled and unlabeled documents using EM", *Machine Learning*, pp. 1-34, 2000.

[5] C. Fraley and A. E. Raftery, "How many clusters ? Which clustering method ? Answers via model-based cluster analysis", *The Computer Journal*, vol. 41, no. 8, 1998.

[6] G. Schwarz, "Estimating the dimension of a model", *The Annals of Statistics*, vol. 6, no. 2, pp. 461-464, 1978.

[7] G. McLachlan and D. Peel,"Finite mixture models", *John Wiley and Sons*, New York, 2000.

[8] H. Akaike, "A new look at the stastical model identification", *IEEE Trans. Automatic Control*, vol. 19, no. 6, pp. 716-723, 1974.

[9] M. A. T. Figueiredo and A. K. Jain, "Unsupervised learning of finite mixture models", *IEEE Trans. on Patt. Anal. and Mach. Intell.*, vol. 24, pp. 381-396, 2002.

[10] M. W. Graham and D. J. Miller, "Unsupervised Learning of Parsimonious Mixtures on Large Spaces With Integrated Feature and Component Selection," *IEEE Trans. Signal Processing*, vol. 54, no. 11, pp 1289-1303, April. 2006.

[11] X. Liu, Y. Gong, W. Xu, and S. Zhu, "Document clustering with cluster refinement and model selection capabilities", *Special Interest Group on Info. Retrieval*, pp. 191-198, 2002.

[12] J. G. Dy and C. E. Brodley, "Feature selection for unsupervised learning", *Journal Mach. Learn. Research*, vol. 5, pp. 845-889, 2004.

[13] S. Vaithyanathan and B. Dom "Generalized model selection for unsupervised learning in high dimensions", *Advances in Neural Info. Proc. Systems*, vol. 11, pp. 970-976, 1999.

[14] I. Guyon, A. Elisseeff. " An introduction to variable and feature selection", *J. Machine Learning Research*, vol. 3, pp. 1157-1182, 2003.

[15] M. H. C. Law, M. A. T. Figueiredo, and A. K. Jain,"Simultaneous feature selection and clustering using mixture models", *IEEE Trans. on Patt. Anal. and Mach. Intell.*, vol. 26, no. 9, pp. 1154-1166, 2004.

[16] Z. Ghahramani and J. M. Beal, "Variational inference for Bayesian mixtures of factor analysers", *Neural Info. Proc. Systems*, vol. 12, pp. 449-455, 2000.

[17] M. E. Tipping and C. M. Bishop, "Mixtures of probabilistic principal component analyzers", *Neural Computation*, pp. 443–482, 1999.

[18] P. A. Olsen and R. A. Gopinath, "Modeling inverse covariance matrices by basis expansion", *IEEE Trans. on Speech and Audio Processsing*, vol. 12, no. 1, pp. 37 – 46, 2004.

[19] S. Richardson and P. Green, "On Bayesian analysis of mixtures with unknown number of components", *J. Royal Stat. Soc. Ser. B*, vol. 59, pp. 731–792, 1997.

[20] C. E. Rasmussen, "The infinite Gaussian mixture model", *Neural Info. Proc. Systems*,MIT Press, 2000.

[21] J.-L. Gauvain and C.-H. Lee, "Bayesian learning of Gaussian mixture densities for hidden Markov moels", *Proc. DARPA Speech and Natural Language Workshop*, pp. 272-277, 1991.

[22] A. D. Lanterman, "Schwarz, Wallace, and Rissanen: intertwining themes in theories of model selection", *Intl. Statistical Review*, vol. 69, no. 2, pp. 185-212, 2001.

[23] X. L. Meng and D. van Dyk,"The EM algorithm – an old folk-song sung to a fast new tune", *J. Royal Stat. Soc. B*, vol. 59, no. 3, pp. 511-567, 1997.

[24] A. Dempster, N. Laird, and D. Rubin,"Maximum-likelihood from incomplete data via the EM algorithm", *J. Roy. Stat. Soc., Ser. B*, vol. 39, pp. 1-38, 1977.

[25] J.A. Fessler and A.O. Hero, "Space-alternating generalized Expectation-Maximization algorithm", *IEEE Trans. on Signal Processing*, vol. 42, pp. 2664–2677, 1994.

[26] T. K. Moon, "Expectation-maximization algorithm", *IEEE Signal Proc. Magazine* vol. 13, no. 6, pp. 47-60, 1996.

[27] J. Rennie, "A short tutorial on using expectation-maximization with mixture models", http://www.ai.mit.edu/people/jrennie/writing/mixtureEM.pdf, 2004.

[28] D. J. Miller and J. Browning, "A mixture model and EM-based algorithm for class discovery, robust classification, and outlier rejection in mixed labeled/unlabeled data sets", *IEEE Trans. on Patt. Anal. and Mach. Intell.*, vol. 25, pp. 1468-1483, 2003.