MIXTURES OF LATENT VARIABLE MODELS FOR DENSITY ESTIMATION AND CLASSIFICATION

Perry Moerland [*]

IDIAP–RR 00-25

SUBMITTED FOR PUBLICATION

[*] e-mail: Perry.Moerland@idiap.ch

IDIAP Research Report 00-25

Mixtures of latent variable models for density estimation and classification

Perry Moerland

submitted for publication

Abstract. This paper deals with the problem of probability density estimation with the goal of finding a good probabilistic representation of the data. One of the most popular density estimation methods is the Gaussian mixture model (Gmm). A promising alternative to Gmms are the recently proposed mixtures of latent variable models. Examples of the latter are principal component analysis and factor analysis. The advantage of these models is that they are capable of representing the covariance structure with less parameters by choosing the dimension of a subspace in a suitable way. An empirical evaluation on a large number of data sets shows that mixtures of latent variable models almost always outperform various Gmms both in density estimation and Bayes classifiers.

To avoid having to choose a value for the dimension of the latent subspace by a computationally expensive search technique such as cross-validation, a Bayesian treatment of mixtures of latent variable models is proposed. This framework makes it possible to determine the appropriate dimension during training and experiments illustrate its viability.

Acknowledgements: This research has been funded by the Swiss National Science Foundation (21-45621.95 and 2100-55688.98). The author gratefully acknowledges the comments of Samy Bengio on a draft version of this paper.

1 Introduction

The focus of this paper is on density estimation using semi-parametric *mixtures* of Gaussian distributions. An important disadvantage of these Gaussian mixture models is that the issue of model complexity can only be handled in a very coarse-grained way. This is often done by placing rather crude constraints on the covariance matrices of the mixture components; for example, by taking only the variance into account, that is, assuming the covariance matrices to be diagonal.

An elegant manner of controlling model complexity of GMMs in a more flexible way is through the introduction of continuous *latent* variables $\mathbf{z} = (z_1, \ldots, z_\ell)^T$ which are related to the observed data $\mathbf{x} = (x_1, \ldots, x_d)^T$ in a probabilistic way. If $\ell < d$ this can be interpreted as a form of dimensionality reduction or feature extraction. The latent variables give a more compact description of the observed data. This idea forms the basis of factor analysis (FA) (Bartholomew and Knott 1999) and a recent probabilistic formulation of principal component analysis (PCA) (Tipping and Bishop 1999). Latent variable models can be readily included as component distributions of a mixture model and form a mixture of constrained Gaussians. The advantage of latent variable models is that they are capable of representing the covariance structure with less parameters by choosing the dimension ℓ of the latent space in a suitable way. Like for GMMs, the Expectation-Maximization (EM) algorithm can be used for learning the parameters of a mixture of latent variable models (Ghahramani and Hinton 1996) in the maximum likelihood framework.

The contribution of this paper is threefold. The first part should be read as an introduction to mixture models, latent variable models and maximum likelihood estimation of their parameter values. We start with a description of the Expectation-Maximization (EM) algorithm for maximum likelihood (ML) in the presence of *hidden* or latent variables. This forms the basis of the learning algorithms used in this paper. As a first example of latent variable models, we consider mixture models. Section 2 shows how to apply the EM algorithm to general mixture models and to Gaussian mixture models in particular. We then describe the linear latent variable models for probabilistic PCA and FA. This is followed by a derivation of the EM algorithm for learning the parameters of a mixture of factor analyzers (MFA) and a mixture of principal component analyzers (MPCA) in section 4. A detailed analysis of the computational complexity of the EM algorithm is given and it is shown that also in this respect mixtures of latent variable models are a valuable alternative to basic GMMs.

The second contribution of this paper is empirical, viz. an experimental comparison of mixtures of latent variable models and GMMs on about 20 data sets. The empirical density estimation results show that mixtures of latent variable models are often preferable both in terms of computational complexity and generalization performance to standard GMMs.

The third part addresses the issue of *model selection* in more detail. The experiments of section 5 used validation data to select values for free parameters such as the number of mixture components and the dimension of latent space ℓ. An approximate Bayesian inference technique is proposed that automatically selects the appropriate dimension of latent space for factor analysis. A recent Bayesian treatment of PCA (Bishop 1999a) is derived as a special case. It is shown that Bayesian FA is computationally more complex than Bayesian PCA but that it remains tractable when implemented carefully. Both methods can be readily extended to mixtures of FA and PCA. We conclude section 6 with a series of experiments on toy data and with the first experimental results on a large set of real-world benchmarks. These experiments show the viability of the Bayesian approach as a way of avoiding overfitting and selecting the dimensionality (or dimensionalities when dealing with mixture models) of latent space.

We end with an experimental evaluation of mixture models in Bayes classifiers where the class-conditional densities are modeled by GMMs, MPCAs, and MFAs. Bayesian MPCAs and MFAs are again shown to offer a viable way of selecting the dimensionality of latent space. This is especially interesting for Bayes classifiers with mixture models because the Bayesian framework enables the model to select a different dimensionality for each mixture component in each class-conditional density.

2 Expectation-Maximization Algorithm and Mixture Models

Suppose that we have unlabeled data $D = \{\mathbf{x}^n\}$ which we assume to be generated from a probability density function $p(\mathbf{x}|\boldsymbol{\theta})$ with parameters $\boldsymbol{\theta}$. We follow a *maximum likelihood* (for example, (Duda and Hart 1973)) approach to estimate the parameters of $p(\mathbf{x}|\boldsymbol{\theta})$ given a set of N examples $\{\mathbf{x}^n\}$. We make the standard assumption that the examples are drawn independently from the same distribution which allows us to factorize the joint probability $p(\{\mathbf{x}^n\}|\boldsymbol{\theta})$:

$$\mathcal{L}(\boldsymbol{\theta}) = p(\{\mathbf{x}^n\}|\boldsymbol{\theta}) = \prod_n p(\mathbf{x}^n|\boldsymbol{\theta}).$$

This function is defined as the *likelihood* of $\boldsymbol{\theta}$ with respect to the data $\{\mathbf{x}^n\}$. Maximum likelihood estimation consists of finding values for $\boldsymbol{\theta}$ which maximize $\mathcal{L}(\boldsymbol{\theta})$ or, equivalently, the log-likelihood $\ln \mathcal{L}(\boldsymbol{\theta})$. An alternative approach would be to consider not only the single most likely parameter values but to treat them as random variables, which is the so-called Bayesian approach. We will come back to this and its advantages in section 6.

2.1 Expectation-Maximization Algorithm

A more general approach is to assume that we also have unobserved or *hidden* variables $\mathbf{z}$ that help modeling the observed data $\mathbf{x}$. The hidden variables can, for example, be discrete component labels which represent imaginary class labels for the observed data. In fact, this is a way to define mixture models, as we will see later on. The log-likelihood of the observed data is obtained by marginalizing over the hidden variables. For the moment, we assume the hidden variables to be discrete and marginalization boils down to applying the sum rule on the log-likelihood:

$$\ln \mathcal{L}(\boldsymbol{\theta}) = \sum_n \ln \sum_{\mathbf{z}} p(\mathbf{x}^n, \mathbf{z}|\boldsymbol{\theta}). \tag{1}$$

An elegant and general way to optimize the log-likelihood in the presence of hidden variables is the EM algorithm. EM can be motivated from the following rewriting of the log-likelihood which stems from the work of Neal and Hinton (1999) and which holds for any distribution Q over the hidden variables:

$$\begin{aligned}\ln \mathcal{L}(\boldsymbol{\theta}) &= \sum_{n,\mathbf{z}} Q(\mathbf{z}|\mathbf{x}^n,\boldsymbol{\theta}) \ln \frac{p(\mathbf{x}^n,\mathbf{z}|\boldsymbol{\theta})}{Q(\mathbf{z}|\mathbf{x}^n,\boldsymbol{\theta})} - \sum_{n,\mathbf{z}} Q(\mathbf{z}|\mathbf{x}^n,\boldsymbol{\theta}) \ln \frac{p(\mathbf{z}|\mathbf{x}^n,\boldsymbol{\theta})}{Q(\mathbf{z}|\mathbf{x}^n,\boldsymbol{\theta})} \qquad (2)\\ &= \mathcal{L}(Q,\boldsymbol{\theta}) + \mathrm{KL}(Q||p) \;:=\; \text{free energy} \;+\; \text{Kullback-Leibler divergence.}\end{aligned}$$

Since the Kullback-Leibler divergence is non-negative (Cover and Thomas 1991), the free energy $\mathcal{L}(Q,\boldsymbol{\theta})$ is a lower bound of the log-likelihood. This forms the basis of a generalized EM algorithm that performs coordinate ascent in the lower bound $\mathcal{L}(Q,\boldsymbol{\theta})$. It is an iterative two-step procedure consisting of a "E (expectation) step" which increases $\mathcal{L}(Q,\boldsymbol{\theta})$ with respect to Q and a "M (maximization) step" which increases $\mathcal{L}(Q,\boldsymbol{\theta})$ with respect to the parameters $\boldsymbol{\theta}$ (Neal and Hinton 1999). It leads to the standard EM algorithm (Dempster et al. 1977) if no restrictions are imposed upon Q. $Q(\mathbf{z}|\mathbf{x}^n,\boldsymbol{\theta})$ can then be chosen equal to the true posterior $p(\mathbf{z}|\mathbf{x}^n,\boldsymbol{\theta})$ which makes the Kullback-Leibler divergence after the E-step equal to zero to give the tightest possible lower bound $\mathcal{L}(Q,\boldsymbol{\theta})$. At each cycle the EM algorithm is guaranteed to increase the log-likelihood unless it is already at a local maximum (Neal and Hinton 1999).

This modern view is indeed equivalent to the classical EM algorithm (Dempster et al. 1977) where the M-step reads:

$$\boldsymbol{\theta} := \underset{\boldsymbol{\theta}'}{\operatorname{argmax}} \;\text{ expected complete log-likelihood } \;= \underset{\boldsymbol{\theta}'}{\operatorname{argmax}} \sum_{n,\mathbf{z}} p(\mathbf{z}|\mathbf{x}^n,\boldsymbol{\theta}) \ln p(\mathbf{x}^n,\mathbf{z}|\boldsymbol{\theta}'), \tag{3}$$

see, for example, (Moerland 2000a) for a more detailed discussion.

The EM algorithm can be applied to a variety of models with hidden variables. We start with one of the simplest of such models: mixture models.

2.2 Mixture Models

A basic approach for density modeling is the parametric approach in which we choose a specific probability density function and maximize its likelihood. While this method is often simple, it is also sensitive to model mismatch: obviously it is difficult to know in advance which density function will fit the data best. Moreover, the parametric approach for analytically simple densities such as those belonging to the exponential family (Duda and Hart 1973) is restricted to modeling unimodal distributions. If the data has several modes or clusters, the resulting model will be poor.

A more flexible model can be obtained by considering mixtures of m simple *component* densities $p_j(\mathbf{x}|\boldsymbol{\theta}_j)$:

$$p(\mathbf{x}|\boldsymbol{\theta},\boldsymbol{\alpha}) = \sum_{j=1}^{m} \alpha_j p_j(\mathbf{x}|\boldsymbol{\theta}_j), \tag{4}$$

where the α_j are the *mixing coefficients* which are non-negative and sum to one. This guarantees that $p(\mathbf{x})$ is a valid density function. The parameters of the mixture model are $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_m)$ and $\boldsymbol{\alpha}$. The hidden variable of a mixture model is a discrete component label, $z = 1 \ldots m$, such that $p(\mathbf{x}, z = j|\boldsymbol{\theta}) = P(z = j)p(\mathbf{x}|z = j, \boldsymbol{\theta}) = \alpha_j p_j(\mathbf{x}|\boldsymbol{\theta}_j)$.

The mixture model error function which we want to minimize is the negative log-likelihood of the mixture density (4) on the training data $\{\mathbf{x}^n\}$:

$$E(\boldsymbol{\theta}) = -\sum_n \ln \sum_{j=1}^{m} \alpha_j p_j(\mathbf{x}^n|\boldsymbol{\theta}_j). \tag{5}$$

Note that, in general, this error function has multiple local minima of different values and that the EM algorithm only guarantees converge towards one of them. Specific choices for the component densities $p_j(\mathbf{x})$ and a derivation of the corresponding instantiations of the EM algorithm are described later on. Part of the EM algorithm, however, is independent of the choice of the component densities, viz. the E-step and the optimization of the mixing coefficients in the M-step. This is briefly explained in the remainder of this section, the reader is referred to, for example, (Moerland 2000a) for further details.

As we saw, the core of the EM algorithm is the maximization of the expected complete log-likelihood (3) or equivalently the minimization of its negation, with respect to the parameters of the mixture model. This gives the so-called (expected) *complete error function*:

$$\mathcal{E}(E_c) = -\sum_n \sum_{j=1}^{m} p(z = j|\mathbf{x}^n, \boldsymbol{\theta}) \ln p(\mathbf{x}^n, z = j|\boldsymbol{\theta}'_j) = -\sum_n \sum_{j=1}^{m} h_j(\mathbf{x}^n) \ln \alpha'_j p_j(\mathbf{x}^n|\boldsymbol{\theta}'_j), \tag{6}$$

denoting the posterior of the hidden variables $p(z = j|\mathbf{x}^n, \boldsymbol{\theta})$ as $h_j(\mathbf{x}^n)$. The E-step then involves the estimation of this posterior and follows from an application of Bayes' rule (leaving the dependence on the parameters $\boldsymbol{\theta}$ implicit):

$$\text{E-step:} \qquad h_j(\mathbf{x}^n) = \frac{\alpha_j p_j(\mathbf{x}^n)}{\sum\limits_{j=1}^{m} \alpha_j p_j(\mathbf{x}^n)}. \tag{7}$$

The M-step consists of the minimization of (6) in which the mixture coefficients α'_j can be optimized independently using Lagrange multipliers to satisfy the constraint $\sum_j \alpha'_j = 1$. This gives:

$$\text{M-step:} \qquad \alpha'_j = \frac{1}{N} \sum_n h_j(\mathbf{x}^n), \tag{8}$$

where N is the number of patterns in the training set $\{\mathbf{x}^n\}$.

2.3 Gaussian Mixture Models

Gaussian mixture models are a standard tool for density estimation and are described in many textbooks (for example, (McLachlan and Basford 1988; Titterington et al. 1985)). A GMM is defined as a mixture model (4) with component distributions which are multivariate Gaussians with a $d \times d$ symmetric and positive-definite covariance matrix $\mathbf{\Sigma}_j$ and $d \times 1$ mean $\boldsymbol{\mu}_j$:

$$p_j(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \mathbf{\Sigma}_j) = \frac{1}{|\mathbf{\Sigma}_j|^{1/2}(2\pi)^{d/2}} \exp\left\{ -\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_j)^T \mathbf{\Sigma}_j^{-1} (\mathbf{x} - \boldsymbol{\mu}_j) \right\}. \tag{9}$$

The parameters of a GMM can be determined by maximum likelihood estimation with the EM algorithm within the general framework for mixture models described in the previous section. This already gives us the E-step (7) and the estimation of the mixing coefficients (8). The updates in the M-step for the parameters $\mathbf{\Sigma}_j$ and $\boldsymbol{\mu}_j$ of the component densities can be found by minimizing the complete error function (6) and are summarized in the following instantiation of the EM algorithm (for example, (Duda and Hart 1973, Chapter 6)):

E-step: Estimation of the posteriors, for all j:

$$h_j(\mathbf{x}^n) = \frac{\alpha_j p_j(\mathbf{x}^n)}{\sum\limits_{j=1}^{m} \alpha_j p_j(\mathbf{x}^n)}.$$

M-step: Re-estimation of the parameters of the GMM (where the new parameter values are denoted with a prime) for all j:

$$\alpha_j' = \frac{1}{N}\sum_n h_j(\mathbf{x}^n) \quad , \quad \boldsymbol{\mu}_j' = \frac{\sum_n h_j(\mathbf{x}^n)\mathbf{x}^n}{\sum_n h_j(\mathbf{x}^n)} \quad , \quad \mathbf{\Sigma}_j' = \frac{\sum_n h_j(\mathbf{x}^n)(\mathbf{x}^n - \boldsymbol{\mu}_j{}')(\mathbf{x}^n - \boldsymbol{\mu}_j{}')^T}{\sum_n h_j(\mathbf{x}^n)}.$$

The updates for the mean $\boldsymbol{\mu}_j$ and the covariance matrix $\mathbf{\Sigma}_j$ correspond to weighted (by $h_j(\mathbf{x}^n)$) versions of the sample mean and the sample covariance matrix.

GMMs with full covariance matrices have several disadvantages, all related to the fact that a full covariance matrix contains $d(d+1)/2$ free parameters (the factor of $1/2$ is due to symmetry) which becomes unwieldy for high-dimensional data. Consequently, in each M-step the update of each of the $\mathbf{\Sigma}_j$ has a complexity of $O(d^2N)$. Moreover, the update of the posteriors in the E-step and the evaluation of the likelihood function require calculating the inverse and determinant of the $d \times d$ matrices $\mathbf{\Sigma}_j$; these operations are $O(d^3)$. Regarding the positive definiteness of the estimate $\mathbf{\Sigma}_j'$, one can observe that this constraint is satisfied if the number of data points is large enough with respect to the dimension of data space: the matrix has to be full rank, that is, a necessary condition is $N > d$. This is especially restrictive in the mixture model context since the number of data points associated with a specific component can be a small subset of the total number of data points. Therefore, in order to obtain well-determined estimates and to avoid overfitting on the training data, a large training set is often needed.

A standard remedy for the problem of badly determined parameters and overfitting when data is scarce is using regularization, that is, an extra penalty term in the error function which encourages smoother estimations. We will use a penalized likelihood approach which has been proposed for GMMs by Ormoneit and Tresp (1998). This requires only some additional factors in the M-step update of the covariance matrix and is numerically more stable:

$$\mathbf{\Sigma}_j' = \frac{\left\{\sum_n h_j(\mathbf{x}^n)(\mathbf{x}^n - \boldsymbol{\mu}_j')(\mathbf{x}^n - \boldsymbol{\mu}_j')^T\right\} + \beta \mathbf{I}_d}{\left\{\sum_n h_j(\mathbf{x}^n)\right\} + 1}. \tag{10}$$

Note that this approach requires tuning the β parameter on validation data.

Another natural way of dealing with scarce data is to limit the number of free parameters in a GMM by imposing constraints on the form of the covariance matrix. Three evident possibilities are covariance matrices which are either:

- spherical: $\Sigma_j = \sigma_j^2 \mathbf{I}_d$. A single parameter for the whole covariance structure which leads to an unflexible model.
- diagonal: with all off-diagonal elements equal to zero. This requires only d parameters but leads to a model in which the axes of the Gaussians are aligned with the data axes: it does not capture correlation amongst the variables.
- tied: parameters of the covariance matrices are tied across the component densities (Bellegarda and Nahamoo 1990). One of the simplest examples is to have one covariance matrix common to all Gaussian components.

Spherical and diagonal covariance matrices limit the computational complexity of the update of each covariance structure in the M-step to $O(dN)$. Moreover, also the inverse and the determinant are at most $O(d)$. We will refer to these models as spherical, diagonal, tied, and full GMMs respectively.

The constraints on the form of the covariance structure introduced above might seem natural, they are also quite restrictive. Modeling power can be increased by adding components to the mixture but that seems to be begging the question. Is there a way to design models which cover the big gap between having a diagonal covariance matrix (d parameters) and a full covariance matrix ($d(d+1)/2$ parameters)? In the next section, it is shown how the introduction of continuous hidden or *latent* variables can lead to more flexible models which smoothly fill up this gap. We first treat the case of simple parametric modeling and extend it to mixture models in section 4.

3 Linear Latent Variable Models

The problem of GMMs with a full covariance matrix described in the previous section is their huge number of free parameters for high-dimensional data. How could we use hidden variables to control the number of parameters? A possible answer is the introduction of latent variables $\mathbf{z} = (z_1, z_2, \ldots, z_\ell)^T$ in a space of dimension $\ell \le d-1$ and specifying probabilistically how latent and observed variables are related. This can be interpreted as a form of dimensionality reduction or feature extraction. Since our starting point is a GMM, we assume that the latent space is $\mathbb{R}^\ell$ and the data space is $\mathbb{R}^d$. The latent variables now being continuous, marginalizing over them is done by integrating:

$$p(\mathbf{x}^n|\boldsymbol{\theta}) = \int_{\mathbf{z}} p(\mathbf{x}^n, \mathbf{z}|\boldsymbol{\theta}) d\mathbf{z} = \int_{\mathbf{z}} p(\mathbf{x}^n|\mathbf{z}, \boldsymbol{\theta}) p(\mathbf{z}) d\mathbf{z}. \tag{11}$$

To keep this integration tractable, we limit ourselves to linear mappings and Gaussian distributions. We define $p(\mathbf{x}^n|\mathbf{z}, \boldsymbol{\theta})$ through the following mapping from latent space to data space, together with a Gaussian prior $p(\mathbf{z})$ for the latent variables:

$$\mathbf{x} = \mathbf{W}\mathbf{z} + \boldsymbol{\mu} + \boldsymbol{\varepsilon} \qquad \text{with} \qquad \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_\ell) \quad , \quad \boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{R}), \tag{12}$$

with model parameters $\mathbf{W}, \mathbf{R},$ and $\boldsymbol{\mu}$. The idea behind the model is illustrated in Figure 1. The prior distribution over the latent variables is a simple Gaussian ball (left-hand part of Figure 1) in latent space. A $d \times \ell$ generative or *factor loading* matrix $\mathbf{W}$ maps the latent space into data space. The effect is to stretch and translate the Gaussian ball in data space (right-hand part of Figure 1) resulting in a sort of ℓ-dimensional pancake in d-dimensional space; the pancake can also be translated over $\boldsymbol{\mu}$. To get a real manifold in data space, the pancake is finally convolved in data space with a Gaussian noise distribution $p(\boldsymbol{\varepsilon})$ with $d \times d$ covariance matrix $\mathbf{R}$ which is independent of $\mathbf{z}$ (Roweis and Ghahramani 1999). This can be interpreted as a generative model in which the hidden variables model the causes for the observed data.

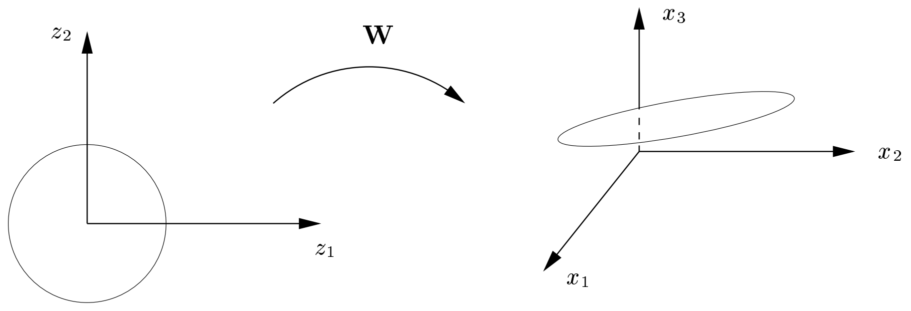

Figure 1: A generative model from a latent space of dimension 2 to a data space of dimension 3.

Due to the restrictions imposed upon the form of the mapping and the priors, everything stays entirely in the Gaussian domain. The conditional distribution of the latent variables given the observed variables is:[1]

$$\mathbf{x}|\mathbf{z} \sim \mathcal{N}(\mathbf{W}\mathbf{z} + \boldsymbol{\mu}, \mathbf{R}), \tag{13}$$

and its convolution (11) with the Gaussian prior $p(\mathbf{z})$ can be performed analytically. This gives the distribution of the observed data, which is also Gaussian:[2]

$$\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{M}) \qquad \text{with} \qquad \mathbf{M} = \mathbf{R} + \mathbf{W}\mathbf{W}^T. \tag{14}$$

But for the moment, we have not gained anything, since the covariance matrix of the observation noise $\mathbf{R}$ is unrestricted and the number of parameters is still $O(d^2)$. All second-order information in the data could be modeled by choosing $\mathbf{R}$ equal to the sample covariance matrix. The simplest way of restricting $\mathbf{R}$ is to make the observed data independent given the latent data, that is (see (13)):

- $\mathbf{R} = \sigma^2 \mathbf{I}_d$: the latent variable model is called probabilistic principal component analysis (Ppca) (Tipping and Bishop 1999) or sensible principal component analysis (Roweis 1998; Roweis and Ghahramani 1999). This terminology has been chosen while with $\sigma^2 \to 0$ conventional Pca is recovered (and even a stronger statement can be made as we will see below).
- $\mathbf{R} \sim$ diagonal matrix: the latent variable model is standard factor analysis (Everitt 1984). In this case, the hidden variables $\mathbf{z}$ are often called *factors*.

These restrictions imply that a linear latent variable model can be viewed as a way of capturing the covariance structure of the d-dimensional observed data through $\mathbf{M}$ (14) with at most $d(\ell + 1)$ parameters. This might be interpreted as a kind of *discrete* regularization in which one can tune the complexity of the model by choosing the dimension of latent space ℓ. In this way, one can cover the whole spectrum of covariance matrices with $O(d)$ parameters to covariance matrices with $O(d^2)$ parameters in a more or less smooth manner. A central issue in the use of linear latent variable models is that of choosing an appropriate value for ℓ. We will see that this problem can be addressed by a Bayesian treatment in section 6.

Estimation What about maximum likelihood estimation for linear latent variable models? The log-likelihood of (14) is:

$$\mathcal{L}(\boldsymbol{\mu}, \mathbf{W}, \mathbf{R}) = \sum_n \ln[\mathcal{N}(\boldsymbol{\mu}, \mathbf{M})] = -\frac{N}{2}\{d \ln(2\pi) + \ln|\mathbf{M}| + \mathrm{tr}(\mathbf{M}^{-1}\mathbf{S})\}, \tag{15}$$

[1]With $\mathbf{x}|\mathbf{z} \sim \ldots$ as a shorthand for $p(\mathbf{x}|\mathbf{z}) = \ldots$

[2]This follows from the fact that the sum of two independent Gaussian distributed quantities is also Gaussian distributed with as mean the sum of the means and as covariance matrix the sum of the covariance matrices. $\mathbf{x}$ is the sum of $\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$ and (the linear transformation of a Gaussian vector) $\mathbf{W}\mathbf{z} + \boldsymbol{\mu} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{W}\mathbf{W}^T)$.

where $\mathbf{S}$ is the $d \times d$ sample covariance matrix:

$$\mathbf{S} = \frac{1}{N} \sum_n (\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^T. \tag{16}$$

The maximum likelihood estimate of the mean $\boldsymbol{\mu}$ is simply the sample mean of the data: $\boldsymbol{\mu}_{\mathrm{ML}} = \frac{1}{N} \sum_n \mathbf{x}^n$. The log-likelihood (15) can be maximized with respect to the parameters $\mathbf{W}$ and $\mathbf{R}$ by the EM algorithm. For factor analysis, this instantiation of EM was derived by Rubin and Thayer (1982) and we come back to this in the next section. The special case of PPCA with $\mathbf{R} = \sigma^2 \mathbf{I}_d$ was dealt with by Roweis (1998) and Tipping and Bishop (1999). Tipping and Bishop actually showed that for the PPCA model a closed-form solution for the global maximum of the likelihood exists and that this is the only stable maximum of the likelihood surface. This solution involves an eigendecomposition of the sample covariance matrix $\mathbf{S}$ and the maximum likelihood estimate for the generative matrix $\mathbf{W}$ spans the ℓ-dimensional *principal* subspace of the data:

$$\mathbf{W}_{\mathrm{ML}} = \mathbf{U}_\ell (\Lambda_\ell - \sigma^2 \mathbf{I}_\ell)^{1/2} \mathbf{V}, \tag{17}$$

where Λ_ℓ is a $\ell \times \ell$ diagonal matrix with the ℓ largest eigenvalues λ_i of $\mathbf{S}$ on the diagonal. $\mathbf{U}_\ell$ is a $d \times \ell$ matrix containing in its columns the corresponding *principal* eigenvectors of $\mathbf{S}$, and $\mathbf{V}$ is an arbitrary $\ell \times \ell$ rotation matrix ($\mathbf{V}^{-1} = \mathbf{V}^T$).[3]

This clearly shows the strong relation with standard principal component analysis, a popular technique for dimensionality reduction (Jollife 1986). PCA is based on a linear projection onto the principal subspace spanned by the (orthonormal) principal eigenvectors $\mathbf{U}_\ell$ of the sample covariance matrix $\mathbf{S}$. The principal component projection is the orthogonal projection which minimizes the squared reconstruction error. The probabilistic model of PPCA has several advantages with respect to standard PCA such as the availability of a proper density model with likelihood scores for model comparison.

The maximum likelihood estimate for the observation noise variance $\mathbf{R} = \sigma^2 \mathbf{I}_d$ can also be found in closed form (Tipping and Bishop 1999):

$$\sigma^2_{\mathrm{ML}} = \frac{1}{d - \ell} \sum_{i=\ell+1}^{d} \lambda_i, \tag{18}$$

and this can be interpreted as the average variance lost per dimension which has been left out.

Probabilistic PCA versus factor analysis The reader might wonder about the consequences of the seemingly small difference in the choice of the noise model $\mathbf{R}$ which is spherical for PPCA and diagonal for FA. PPCA can have difficulties in separating information from noise as illustrated by the toy example of Figure 2. In this example, the second coordinate x_2 is much noisier than the others. Factor analysis with its diagonal noise covariance $\mathbf{R}$ succeeds in modeling this noisy component and finds the correlation between the third and the fourth coordinate x_3, x_4. PPCA on the other hand, is confused by the high variance on the second coordinate x_2 and finds a correlation between dimensions 2–4 and an estimate of the noise that averages over the remaining noise in the data. Loosely speaking, principal component analysis pays attention to both variance and covariance, whereas factor analysis looks only at covariance.

4 Mixtures of Factor Analyzers and PCAs

One important advantage of the linear latent variable models described in section 3 is that they define a proper probability model which can be extended to a mixture model. The mixture model (4) is then a linear combination of component distributions (14):

$$p_j(\mathbf{x} | \boldsymbol{\mu}_j, \mathbf{R}_j, \mathbf{W}_j) \sim \mathcal{N}(\boldsymbol{\mu}_j, \mathbf{R}_j + \mathbf{W}_j \mathbf{W}_j^T), \tag{19}$$

[3]This arbitrary rotation implies that the number of free parameters for PPCA is $d\ell + 1 - \ell(\ell - 1)/2$.

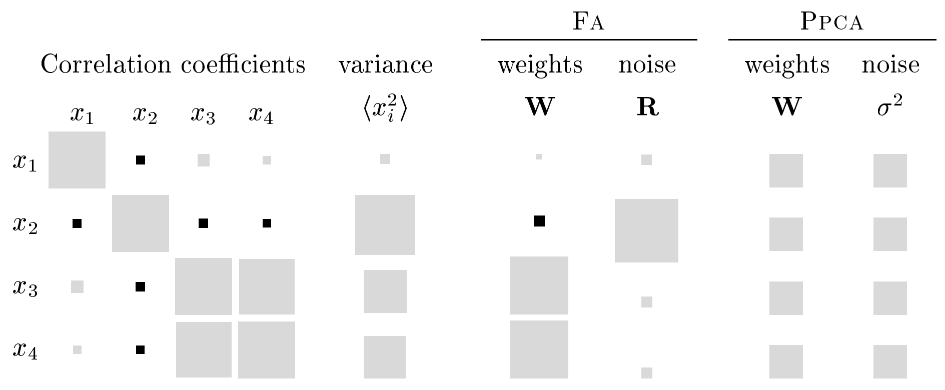

Figure 2: A toy data set of 1000 patterns in four dimensions (x_1, x_2, x_3, x_4). The first dimension has been generated from a Gaussian having small variance, while the second dimension has been generated from a Gaussian with high variance. The first two dimensions are independent of the other dimensions. The third and fourth dimension are highly correlated and have been generated from the same values (Gaussian with medium-valued variance) corrupted by small Gaussian noise. This can be seen from the Hinton diagrams with gray for positive and black for negative values; the size of the squares is proportional to the value of the corresponding parameter. The first two diagrams represent the correlation coefficients and the variance of the data. This data was used to train a FA and a PPCA model with one factor ($\ell = 1$). The other Hinton diagrams shown correspond to the estimated 4×1 factor loading $\mathbf{W}$ and the 4×1 diagonal of the estimated noise covariance $\mathbf{R}$ for both models.

with separate parameters for each of the component distributions. With spherical noise covariance $\mathbf{R}_j$, the model is called a mixture of principal component analyzers (Tipping and Bishop 1999) and with a diagonal $\mathbf{R}_j$, it is called a mixture of factor analyzers (Ghahramani and Hinton 1996). These mixtures can be interpreted as a mixture of constrained Gaussians in which the number of parameters can be controlled through the dimension of the latent space ℓ without putting too strong constraints on the flexibility of the model, that is, on the form of the covariance matrix.

In Appendix A, a detailed derivation of an EM algorithm for mixtures of factor analyzers is given (Algorithm 1). An EM algorithm for MFAs was originally presented by Ghahramani and Hinton (1996) but the one derived here gives a nicer separation of concerns by staying close to Tipping and Bishop's algorithm for MPCAs. The maximum likelihood estimates for the parameters of the mixture model are determined by dividing each EM iteration into two stages. The idea of this two-stage approach is to take into account the hidden variables indicating the component labels first and consider the latent variables $\mathbf{z}$ only in the second stage. This allows us to easily build upon the general framework for mixture models described in section 2.2. Convergence of the two-stage approach is still guaranteed since the second stage decreases the complete error function of the first stage. Thus, it corresponds to a generalized EM algorithm.

The EM algorithm for a mixture of factor analyzers consists of iteratively performing:

1. E-step (first stage): compute posteriors $h_j(\mathbf{x})$ for the component labels of the mixture (35);

2. M-step (first stage): reestimate the mixing coefficients α_j (37) and means $\boldsymbol{\mu}_j$ (38);

3. E-step (second stage): compute the first moments $\langle \mathbf{z}_j^n \rangle$ (47) and second moments $\langle \mathbf{z}_j^n (\mathbf{z}_j^n)^T \rangle$ (48) of the posterior distribution of latent variable $\mathbf{z}$ conditioned on the observed data point $\mathbf{x}^n$, using the new estimates for $\boldsymbol{\mu}_j$;

4. M-step (second stage): reestimate the generative matrices $\mathbf{W}_j$ (53) and noise covariances $\mathbf{R}_j$ (54).

Algorithm 1 EM algorithm for MFAs

Require:

- A training set of N examples: $\{\mathbf{x}^1, \ldots, \mathbf{x}^N\}$.
- Error function: $E = -\sum_n \ln \sum_{j=1}^m \alpha_j p_j(\mathbf{x}^n)$ with $p_j(\mathbf{x}) \sim \mathcal{N}(\boldsymbol{\mu}_j, \mathbf{R}_j + \mathbf{W}_j\mathbf{W}_j^T)$
- $\mathbf{W}_j : d{\times}\ell$ $\mathbf{R}_j : d{\times}d$ and diagonal $\mathbf{N}_j : \ell{\times}\ell$ $\langle \mathbf{z}_j^n \rangle : \ell{\times}1$ $\langle \mathbf{z}_j^n(\mathbf{z}_j^n)^T \rangle : \ell{\times}\ell$

```
loop
  {First stage: E-step}
  for $j := 1$ to $m$ do
    for $n := 1$ to $N$ do
      $h_j(\mathbf{x}^n) := \frac{\alpha_j p_j(\mathbf{x}^n)}{\sum_{j=1}^{m} \alpha_j p_j(\mathbf{x}^n)}$
    end for
  end for
  {First stage: M-step for $\boldsymbol{\alpha}$ and $\boldsymbol{\mu}$}
  for $j := 1$ to $m$ do
    $\alpha_j := \frac{1}{N}\sum_n h_j(\mathbf{x}^n)$
    $\boldsymbol{\mu}_j := \frac{\sum_n h_j(\mathbf{x}^n)\mathbf{x}^n}{\sum_n h_j(\mathbf{x}^n)}$
  end for
  {Second stage: E-step}
  for $j := 1$ to $m$ do
    $\mathbf{N}_j := \mathbf{I}_\ell + \mathbf{W}_j^T\mathbf{R}_j^{-1}\mathbf{W}_j$
    for $n := 1$ to $N$ do
      $\langle \mathbf{z}_j^n \rangle := \mathbf{N}_j^{-1}\mathbf{W}_j^T\mathbf{R}_j^{-1}(\mathbf{x}^n - \boldsymbol{\mu}_j)$
      $\langle \mathbf{z}_j^n(\mathbf{z}_j^n)^T \rangle := \mathbf{N}_j^{-1} + \langle \mathbf{z}_j^n \rangle\langle \mathbf{z}_j^n \rangle^T$
    end for
  end for
  {Second stage: M-step for $\mathbf{W}$ and $\mathbf{R}$}
  for $j := 1$ to $m$ do
    $\mathbf{W}_j := \left[\sum_n h_j(\mathbf{x}^n)(\mathbf{x}^n - \boldsymbol{\mu}_j)\langle \mathbf{z}_j^n \rangle^T\right]\left[\sum_n h_j(\mathbf{x}^n)\langle \mathbf{z}_j^n(\mathbf{z}_j^n)^T \rangle\right]^{-1}$
    $\mathbf{R}_j := \frac{1}{\sum_n h_j(\mathbf{x}^n)}\operatorname{diag}\left[\sum_n h_j(\mathbf{x}^n)\left\{(\mathbf{x}^n - \boldsymbol{\mu}_j) - \mathbf{W}_j\langle \mathbf{z}_j^n \rangle\right\}(\mathbf{x}^n - \boldsymbol{\mu}_j)^T\right]$
  end for
  {Stage 2 minimizes $\mathcal{E}(\hat{E}_c)$ (49) and, therefore decreases $\mathcal{E}(E_c)$ (36) which decreases the error
  function $E$}
end loop
```

Computational Complexity We know that a mixture of latent variable models offers an efficient alternative to full GMMs in terms of the number of parameters, but what about its computational complexity? The most intricate part is the calculation of the posteriors $h_j(\mathbf{x}^n)$ which requires the evaluation of the component densities:

$$p_j(\mathbf{x}) = \mathcal{N}(\boldsymbol{\mu}_j, \mathbf{R}_j + \mathbf{W}_j\mathbf{W}_j^T) = \frac{1}{|\mathbf{M}_j|^{1/2}(2\pi)^{d/2}} \exp\left\{-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_j)^T\mathbf{M}_j^{-1}(\mathbf{x} - \boldsymbol{\mu}_j)\right\}. \quad (20)$$

This seems to involve calculating the inverse and determinant of the $d{\times}d$ matrix $\mathbf{M}_j$. However, $\mathbf{M}_j^{-1}$ can be rewritten using the matrix inversion lemma (see (43)):

$$\mathbf{M}_j^{-1} = \mathbf{R}_j^{-1} - \mathbf{R}_j^{-1}\mathbf{W}_j\mathbf{N}_j^{-1}\mathbf{W}_j^T\mathbf{R}_j^{-1},$$

which only requires the $O(\ell^3)$ inversion of a $\ell\times\ell$ matrix and the easy inversion of the diagonal matrix $\mathbf{R}_j$. Actually, just calculating $\mathbf{N}_j = \mathbf{I}_\ell + \mathbf{W}_j^T\mathbf{R}_j^{-1}\mathbf{W}_j$ is a more complex $O(\ell^2 d)$ operation. When substituting this in (20) care should be taken in choosing an efficient parsing:

$$(\mathbf{x}-\boldsymbol{\mu}_j)^T\mathbf{M}_j^{-1}(\mathbf{x}-\boldsymbol{\mu}_j) = (\mathbf{x}-\boldsymbol{\mu}_j)^T\mathbf{R}_j^{-1}(\mathbf{x}-\boldsymbol{\mu}_j) - \left\{(\mathbf{x}-\boldsymbol{\mu}_j)^T\mathbf{R}_j^{-1}\mathbf{W}_j\right\}\mathbf{N}_j^{-1}\left\{\mathbf{W}_j^T\mathbf{R}_j^{-1}(\mathbf{x}-\boldsymbol{\mu}_j)\right\}.$$

This is an $O(m\ell dN)$ operation since it has to be done for each data point and each mixture component. Furthermore, the determinant $|\mathbf{M}_j|$ can be simplified with a determinant factoring lemma (see (Moerland 2000b)):

$$|\mathbf{M}_j| = |\mathbf{R}_j + \mathbf{W}_j\mathbf{W}_j^T| = |\mathbf{R}_j||\mathbf{I}_\ell + \mathbf{W}_j^T\mathbf{R}_j^{-1}\mathbf{W}_j| = |\mathbf{R}_j||\mathbf{N}_j|,$$

which only involves a determinant of a $\ell\times\ell$ matrix, which is $O(\ell^3)$, and the simple determinant of the diagonal matrix $\mathbf{R}_j$. It is easy to verify that the other operations are of the same order of complexity as the ones considered until now. This is especially so for the update of $\mathbf{R}_j$ which does not require the calculation of the sample covariance but only the variance for each coordinate, because of the "diag" operator. The computational complexity of the EM algorithm for MFAs with m components, therefore, is $O(m\ell^2 d)+O(m\ell dN)$ in each iteration. Recall from section 2.3 that the complexity of EM for full GMMs is $O(md^3)+O(md^2N)$. MFAs give a speed-up of $(d/\ell)^2$ on the first term and (d/ℓ) on the second term. Both in terms of the number of parameters and of computational complexity, MFAs smoothly cover the range between diagonal and full covariance matrices in a GMM. With respect to storage complexity, Algorithm 1 seems to require storing explicitly a $\ell\times\ell$ matrix $\langle\, \mathbf{z}_j^n(\mathbf{z}_j^n)^T\,\rangle$ for each pattern. This can be avoided since these second moments are only used as sufficient statistics in the update for the generative matrix (53).

The transformation of the algorithm for MFAs to one for MPCAs is quite easy: substituting $\mathbf{R}_j = \sigma_j^2$ in Algorithm 1 and some algebra leads to an algorithm which is identical to the one of Tipping and Bishop (1999). The reader is referred to (Moerland 2000a) for a detailed derivation of the transformation. The computational complexity of this algorithm is also $O(m\ell^2 d)+O(m\ell dN)$.

5 Experiments: Density Estimation

This section is dedicated to an experimental comparison of standard GMMs and mixtures of latent variable models and illustrates the merit of the latter in density modeling (Moerland 1999). Recent literature contains various experimental results for density modeling with mixtures of latent variable models, on toy data sets and some isolated real-world problems. Here we mention a comparison of MFAs with various other methods (standard PCA, factor analysis, generative topographic mapping) on the specific problem of density modeling of electropalatographic data (Carreira-Perpiñán and Renals 1998). MPCAs and MFAs have also been applied to handwritten digit recognition with a mixture model for each class in a Bayes classifier (Hinton et al. 1997; Tipping and Bishop 1999).

The goal of the experiments described here is to provide a more extensive comparison of GMMs and mixtures of latent variable models on a large set of benchmarks. For the moment, we consider only density estimation per se but we will come back to the inclusion of mixture models in Bayes classifiers later on (section 7).

5.1 Experimental Set-Up

The density estimation experiments were mostly performed on data sets out of the Irvine repository (Blake et al. 1998). An overview of the main characteristics of the different data sets is given in Table 1. As can be seen from this table, the benchmarks largely differ in dimension and number of patterns and cover a wide spectrum of data sets which one might encounter in practice. We limited ourselves to data sets for classification problems; of course, for the current experiments only the input space of the data sets plays a role and the class labels are not taken into consideration. The NIST data set is a subset of NIST Special Database 3 of handwritten digits (Garris and Wilkinson 1992). Data has been size-normalized, deskewed, centered, and smoothed to obtain 16×16 images with their values scaled to

Data set	# attributes		# classes	# examples		# attr.
	discr.	cont.		train	test	
banana	-	2	2	803	-	
cancer	-	10	2	699	-	10
dermatology	-	34	6	366	-	34
glass	-	9	6	214	-	
heart	-	13	2	303	-	13
ionosphere	-	34	2	351	-	
iris	-	4	3	150	-	
letter	-	16	26	20000	-	
NIST	-	256	10	15025	4975	
optical	-	64	10	3823	-	
pen	-	16	10	7494	-	
satimage	-	36	6	4435	2000	
segmentation	-	19	7	2310	-	
sonar	-	60	2	208	-	
soybean	35	-	19	683	-	134
twos	-	256	10	1948	-	
vowel	-	10	11	990	-	
waveform	-	21	3	600	-	
waveform-noise	-	40	3	600	-	

Table 1: Properties of the data sets used in the experiments. The last column gives the number of attributes after pre-processing missing data (if any).

the interval $[0, 1]$. The training set consists of 15,025 digits from 140 writers and the test set of 4,975 digits from 48 writers (not overlapping with training set). The twos data set consists of the set of digits "two" out of the NIST data. A script for generating the banana data has been made available by Gunnar Rätsch at `http://www.first.gmd.de/~raetsch/data/banana.txt`. It is a two-dimensional toy problem with two classes generated from several nonlinearly transformed Gaussian and uniform blobs (Rätsch, Onoda, and Müller 1998).

The raw data was pre-processed in various ways. First of all, each of the ordinal and real-valued attributes has been normalized to have zero mean and unit standard deviation on the training data. For soybean, some of the attributes are categorical and these were mapped to a 1-of-c coding, thus increasing the number of attributes (see the last column of Table 1). Finally, for four data sets some of the attributes are missing for several patterns. For ordinal and real-valued inputs, the missing value was replaced by zero (the mean value after normalization) and for categorical inputs an extra bit was added to the 1-of-c coding to encode a missing value.

GMMs were used with various choices for the covariance matrix: spherical, diagonal, full, and full but tied across the mixture components. From the class of mixtures of latent variable models both MFAs and MPCAs were used. The training of all mixture models consisted of an initialization phase and 15 iterations of the EM algorithm. This number of iterations of the EM algorithm was chosen because it very often turned out to be sufficient for approaching a local minimum on the training set. The initialization of each of the mixture models used k-means clustering (Bishop 1995) to determine the means. The mixing coefficients were computed from the proportion of examples closest to each cluster mean.

Covariance matrices for the initialization of the GMMs were calculated based on the sample covariance of the points associated with (that is, closest to) the corresponding means. The penalized likelihood approach described in section 2.3 was used for training tied and full GMMs. The value of the regularization parameter in (10) was varied $\beta \in \{0, 0.005, 0.01, 0.03, 0.05, 0.1\}$. Furthermore, a small

Mixture	train	test	5x2cv
Full	22.4(0.18)	26.1(0.34)	
Spherical	25.4(0.08)	25.7(0.28)	
Diagonal	24.9(0.10)	25.3(0.26)	$<$
MPCA-1	24.6(0.14)	25.2(0.31)	
MPCA-3	24.0(0.15)	25.1(0.27)	
MFA-3	23.5(0.17)	24.4(0.24)	$<$
MFA-1	23.8(0.19)	24.3(0.25)	$<$

Mixture	train	test	5x2cv
Full	45.4(0.27)	60.1(0.69)	
MPCA-3	51.5(0.14)	53.7(0.50)	$<$
Spherical	52.8(0.10)	53.4(0.48)	$<$
MPCA-1	52.3(0.12)	53.4(0.48)	
Diagonal	51.5(0.13)	52.4(0.48)	$<$
MFA-3	50.0(0.19)	51.9(0.50)	$<$
MFA-1	50.7(0.19)	51.7(0.51)	$<$

Table 2: Density modeling on waveform (left) and waveform-noise (right) with 3 mixture components. Scores are in average negative log-likelihood. The entries are the averages of the negative log-likelihood per data point over 10 cross-validated simulations with the standard deviation in parentheses. A $<$-sign indicates whether the score on the test set is significantly better (95% on the 5x2cv F test) than the one on the previous row. MFA-ℓ and MPCA-ℓ denote a mixture of latent variable models with ℓ factors.

threshold was imposed upon the singular values of the covariance matrices to avoid non-singularities and mixture components which collapse to covering only a single data point. The initialization of the generative matrices $\mathbf{W}_j$ of the mixtures of latent variable models was done using (17) via a few iterations of EM for PCA on the points associated with the corresponding means (Roweis 1998). The noise models $\mathbf{R}_j$ were initialized using the variance lost in the PCA projections found for each cluster using (18). For the MFAs, we decided to use a single noise model $\mathbf{R}$ which is tied across the mixture components. This single diagonal matrix corresponds to the intuitive idea of data corrupted by sensor noise which should not vary across the mixture components. A simple form of regularization was used by imposing a small threshold of 0.0001 upon the values of $\mathbf{R}_j$.

The 5x2cv F test (Alpaydin 1999; Dietterich 1998) has been used on all data sets for testing whether a difference in performance between the methods was statistically significant. In this test, five replications of twofold cross-validation are performed: each training set and each test set comprises 50% of the data. This was also done for the satimage data which, in principle, comes with a fixed division in a training and a test set (see Table 1). In each round of cross-validation, one third of the training data was set aside for validation purposes (respecting the class distributions). The results on the validation sets were used to select the best model for each type of mixture model on each data set. The free parameters varied for each type of model were the number of mixture components, the regularization parameter β for full GMMs, and, for MPCAs and MFAs, the dimension of latent space.

5.2 Experiments: Toy Data

As a first test, experiments were performed on two artificial classification problems with code for generating the data at the Irvine repository (Blake, Keogh, and Merz 1998): waveform and waveform-noise (the last two rows of Table 1). The waveform data is generated according to:

$$\begin{aligned} x_i &= uh_1(i) + (1-u)h_2(i) + \varepsilon_i \qquad \text{Class 1} \\ x_i &= uh_1(i) + (1-u)h_3(i) + \varepsilon_i \qquad \text{Class 2} \\ x_i &= uh_2(i) + (1-u)h_3(i) + \varepsilon_i \qquad \text{Class 3,} \end{aligned}$$

where $i = 1, 2, \ldots 21$, u is a random variable, uniformly distributed on $(0,1)$, $\varepsilon_i \sim \mathcal{N}(0, \mathbf{I})$, and the h_i are shifted triangular waveforms: $h_1(i) = \max(6 - |i - 11|, 0), h_2(i) = h_1(i-4)$, and $h_3(i) = h_1(i+4)$. For waveform-noise, the 19 additional attributes are all noise attributes with mean 0 and variance 1.

The results on waveform and waveform-noise are in Table 2. The GMM with a full covariance matrix obtains the lowest negative log-likelihood on the training set but the worst score on the test set: the model is overfitting the data due to its many parameters. For all other models the score on the test is only slightly worse than the score on the training set which suggests the absence of overfitting. On

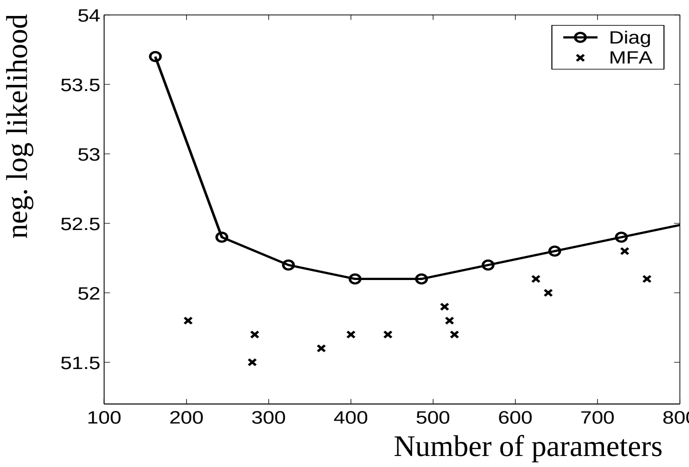

Figure 3: Comparison of a Gmm with diagonal covariance matrices and a Mfa on waveform-noise: the average negative log-likelihood on the test set versus the number of parameters.

waveform, the best results are obtained with the mixtures of latent variable models. It is especially worth noting that the Mfa model with only one factor performs much better than the Gmm with a diagonal covariance matrix which has about the same number of free parameters. This shows that the axial alignment constraint of the diagonal model is not appropriate in this case. On waveform-noise, the best results are obtained with Mfas, but Mpcas are not performing that good. This is most likely due to the fact that the 19 additional attributes for this data set are just white noise and Mfas can separately model correlations and variance ($\mathbf{R}_j$ is diagonal), whereas Mpcas cannot ($\mathbf{R}_j$ is isotropic).

It is interesting to investigate the influence of the number of parameters in a mixture model. Figure 3 illustrates that for a fixed number of parameters, Gmms with diagonal covariance matrices, varying the number of mixture components from 2 to 10, were always outperformed by Mfas, varying the number of factors and mixture components, on waveform-noise.

5.3 Experiments: Real-World Data

Do the results for mixtures of latent variable models on both artificial data sets carry over to real-world data? Experiments have been performed on the data sets listed in Table 1. The results are shown in Table 3, where the best method and the ones that are not significantly worse (95% on the 5x2cv F test) are set underlined. Note that on the dermatology and glass data, the results do not allow any conclusion about the relative performance of the different models.

For most data sets the number of mixture components tried, had at least five different values ranging from one to a value depending on the complexity of the problem. The dimension of latent space for Mpcas and Mfas was also varied and was of course upper-bounded by the dimension of data space.

The last row in Table 3 gives an immediate idea of the global performance of the different models by specifying the total number of underlined scores for each model class. This number of wins shows that Mfas are the best density estimators. They are outperformed on only three data sets (cancer, ionosphere, and vowel) out of 18. With respect to the spherical Gmms, the score of only 1 win illustrates that they are too constrained to model the data. From the results, one can also indirectly conclude that using full covariance matrices makes the model sensitive to overfitting: the best model has only one mixture component in about half of the cases. This effect can be reduced by tying the covariance matrices across the mixture components but this also considerably reduces modeling power; there is only one win for tied Gmms. When data is plentiful, however, full Gmms might still be among the best models. Diagonal Gmms are performing quite well on a few data sets; this is most apparent on

	GMM					
Data	spherical	diagonal	tied	full	MPCA	MFA
banana	$2.7(0.04)^{6}$	$\underline{2.6}(0.10)^{6}$	$2.7(0.04)^{6}$	$\underline{2.6}(0.08)^{6}$	$\underline{2.6}(0.08)^{6,1}$	$\underline{2.6}(0.10)^{6,1}$
cancer	$7.3(1.09)^{6}$	$\underline{1.5}(0.97)^{6}$	$10.6(1.26)^{6}$	$3.9(1.56)^{4}$	$\underline{3.7}(1.43)^{6,5}$	$6.1(0.98)^{2,5}$
dermatology	$39.1(0.92)^{6}$	$36.5(4.94)^{4}$	$39.7(1.98)^{4}$	$40.7(1.25)^{1}$	$37.9(0.66)^{6,1}$	$33.4(9.42)^{4,5}$
glass	$10.0(1.48)^{4}$	$11.0(9.88)^{4}$	$11.9(7.07)^{4}$	$11.7(4.13)^{1}$	$11.1(3.62)^{4,1}$	$13.1(4.93)^{1,5}$
heart	$17.9(0.26)^{6}$	$\underline{16.4}(0.74)^{2}$	$\underline{14.8}(0.54)^{4}$	$18.2(0.26)^{1}$	$18.0(0.24)^{2,1}$	$\underline{14.5}(1.38)^{6,1}$
ionosphere	$31.9(2.20)^{8}$	$\underline{27.2}(1.53)^{8}$	$60.1(7.13)^{8}$	$60.8(7.28)^{1}$	$\underline{29.4}(2.20)^{8,1}$	$37.5(1.72)^{1,5}$
iris	$4.0(0.31)^{4}$	$3.4(0.23)^{2}$	$2.8(0.24)^{4}$	$\underline{2.5}(0.21)^{2}$	$\underline{2.6}(0.31)^{2,3}$	$\underline{2.7}(0.25)^{2,2}$
letter	$18.3(0.11)^{20}$	$16.0(0.18)^{20}$	$15.9(0.07)^{20}$	$\underline{11.1}(0.26)^{20}$	$12.0(0.14)^{20,15}$	$\underline{11.6}(0.50)^{20,15}$
optical	$66.3(1.01)^{20}$	$\underline{-11.6}(1.48)^{20}$	$47.9(3.46)^{20}$	$4.3(3.01)^{6}$	$46.1(6.55)^{20,15}$	$\underline{-12.5}(1.71)^{10,15}$
pen	$13.8(0.17)^{20}$	$7.7(0.28)^{20}$	$10.4(0.15)^{20}$	$3.2(0.39)^{20}$	$3.9(0.35)^{20,15}$	$\underline{-4.7}(0.74)^{20,15}$
satimage	$13.2(0.37)^{20}$	$11.5(0.24)^{20}$	$-0.6(0.19)^{20}$	$\underline{-4.8}(0.23)^{6}$	$-3.1(0.43)^{10,15}$	$\underline{-4.4}(0.36)^{6,15}$
segmentation	$13.3(0.65)^{10}$	$-6.0(2.36)^{10}$	$-7.2(1.35)^{8}$	$\underline{-19.8}(2.72)^{10}$	$\underline{-7.6}(7.69)^{10,8}$	$\underline{-20.6}(4.66)^{10,8}$
sonar	$\underline{76.7}(1.92)^{4}$	$79.0(1.97)^{4}$	$162.7(10.91)^{1}$	$162.1(7.38)^{1}$	$\underline{73.5}(3.11)^{1,5}$	$\underline{68.9}(4.54)^{1,15}$
soybean	$16.2(2.23)^{6}$	$-167.0(9.76)^{6}$	$\underline{-214.4}(7.74)^{4}$	$\underline{-230.3}(9.05)^{2}$	$-13.2(9.40)^{4,8}$	$\underline{-196.7}(6.85)^{6,8}$
twos	$-109.1(1.99)^{20}$	$-186.7(3.00)^{20}$	$-295.2(1.88)^{1}$	$-295.1(2.20)^{1}$	$-286.7(3.82)^{4,15}$	$-\underline{323.6}(7.19)^{4,15}$
vowel	$12.4(0.14)^{11}$	$12.3(0.14)^{11}$	$11.7(0.18)^{11}$	$\underline{11.3}(0.24)^{11}$	$11.5(0.31)^{11,3}$	$11.5(0.39)^{11,5}$
waveform	$25.3(0.30)^{6}$	$24.9(0.28)^{6}$	$24.9(0.23)^{1}$	$24.9(0.25)^{1}$	$24.8(0.27)^{1,10}$	$\underline{24.3}(0.25)^{4,1}$
waveform-noise	$53.5(0.49)^{4}$	$52.5(0.49)^{4}$	$54.1(0.53)^{1}$	$54.1(0.51)^{1}$	$53.5(0.46)^{2,1}$	$\underline{51.6}(0.51)^{1,3}$
wins	1	5	2	7	6	13

Table 3: Input density modeling with GMMs, MFAs, and MPCAs. Scores are in average negative log-likelihood on the test set and are the average over 10 experiments in a 5x2cv F test set-up. The standard deviation is given between parentheses. Each score corresponds to the best model out of the particular class of models as evaluated on a validation set. The first superscript indicates the number of mixture components in the "best". The second superscript for mixtures of latent variable models specifies the dimension of latent space. The underlined scores are the ones that do not pass the 5x2cv F test with 95% confidence when compared with the model having the best score.

the optical data which is hand-written digit data with discretized attributes. Some of its attributes are zero throughout the whole data set and a diagonal GMM is well-suited for capturing this kind of absence of noise. It might also explain why the scores of the MPCAs are not as good as with MFAs: the single variance parameter for each mixture component makes it more difficult for MPCAs to separate information and noise. This is illustrated by the mediocre performance on the optical data. MPCAs do not succeed in dealing with the varying amounts of noise in the different attributes.

We can conclude that mixtures of latent variable models and especially MFAs, are indeed a flexible alternative to standard Gaussian mixture models, the complexity of which can be tuned by varying the dimension of the latent space. Moreover, their computational complexity also scales favorably with the dimension of latent space and places them on a spectrum going from diagonal GMMs to full GMMs.

6 Bayesian Methods for Latent Variable Models

One of the limitations of the maximum likelihood framework to which we have adhered until now is that it does not take into account model complexity. Therefore, we selected the number of mixture components and factors by measuring the performance of each model on validation data. Actually, using mixtures of latent variable models with the *same* dimension of latent space for each of the mixture components is simply a way of keeping this discrete model search more tractable. One would certainly expect to improve performance by allowing each of the mixture components to have its proper dimensionality. In this section, we present a novel Bayesian treatment of factor analysis which can automatically determine the effective dimension of latent space within the evidence framework of MacKay (1991). The resulting algorithm is an extension of the previously derived EM algorithm. Bishop's (1999a) Bayesian PCA is derived as a special case of Bayesian FA.

6.1 From Regularization to the Evidence Framework

An approach to control the effective complexity of a model is the use of *regularization* which involves the addition of a term to the error function in order to encourage simpler mappings. This can be used as a means to still obtain good generalization with models that are actually too complex for the data. The regularized error function thus consists of a data term E_D which we assume to be the negative log-likelihood as before and a penalty term E_P:

$$E = -\ln p(D|\boldsymbol{\theta}) + \gamma E_P(\boldsymbol{\theta}). \tag{21}$$

The parameter γ controls the influence of the penalty term on the total error function. A well-known example of regularization is known as *weight decay* or *ridge regression*: $E_P(\boldsymbol{\theta}) = \frac{1}{2}||\boldsymbol{\theta}||^2$. This regularizer favors a smoother mapping by penalizing large parameter values which are often an indication for overfitting.

The inclusion of such a regularizer simplifies the problem of model selection since we can now choose a rather complex model and rely on the penalty term to control the effective complexity. The complexity control is now embodied by the single regularization parameter γ. An appropriate value for γ could again be found by using validation data. This becomes unwieldy when more complex regularizers are used with several regularization parameters. Such a complex regularizer is exactly what we need to control the dimension of latent space in Ppca and Fa. Remember that each dimension of latent space, with $\ell = d-1$ as maximum value, corresponds to a column of the generative matrix $\mathbf{W} = (\mathbf{w}_1 \ldots \mathbf{w}_{d-1})$. Bishop (1999a), therefore, proposed to use a separate regularization term of the weight decay type for each column of $\mathbf{W}$:

$$\frac{1}{2}\sum_{i=1}^{d-1} \gamma_i ||\mathbf{w}_i||^2, \tag{22}$$

with regularization parameters $\boldsymbol{\gamma} = (\gamma_1 \ldots \gamma_{d-1})$. This means that the value of γ_i controls the norm of column $\mathbf{w}_i$ and is even able to switch it off entirely by forcing the corresponding weights to zero. This reduces the dimensionality of the model to the number of columns $\mathbf{w}_i$ which remain non-zero. This regularization term is motivated by the framework of *automatic relevance determination* (Ard) as proposed by MacKay and Neal (see, for example, MacKay 1994b). They used it for controlling the relevance of inputs of a multi-layer perceptron (Mlp). The Ard term involves $d-1$ regularization parameters and as already observed before, selecting their values using validation data would be cumbersome. A principled way of handling regularization on the training data only is the application of Bayesian inference techniques (Bishop 1995; MacKay 1991).

The essence of Bayesian inference is that instead of a single set of values for the parameters as in maximum likelihood estimation, a probability distribution over the parameter space is considered. This can be motivated by interpreting (21) entirely in probabilistic terms using Bayes' rule:

$$E = -\ln p(\boldsymbol{\theta}|D,\boldsymbol{\gamma}) = -\ln\left[\frac{p(D|\boldsymbol{\theta})p(\boldsymbol{\theta}|\boldsymbol{\gamma})}{p(D|\boldsymbol{\gamma})}\right] \propto -\ln p(D|\boldsymbol{\theta}) - \ln p(\boldsymbol{\theta}|\boldsymbol{\gamma}). \tag{23}$$

The weight decay regularizer $\frac{1}{2}\gamma||\boldsymbol{\theta}||^2$ then corresponds directly to a so-called *prior* distribution for the parameters:

$$p(\boldsymbol{\theta}|\boldsymbol{\gamma}) = \mathcal{N}(\boldsymbol{\theta}|\mathbf{0}, 1/\gamma) \propto \exp(-\frac{\gamma||\boldsymbol{\theta}||^2}{2}).$$

Thus, minimization of the regularized error function (21) is equivalent to finding the *most probable* parameter values $\boldsymbol{\theta}_{\mathrm{MP}}$ maximizing the posterior distribution $p(\boldsymbol{\theta}|D,\boldsymbol{\gamma})$. This is a nice probabilistic interpretation of the error function but the real strength of the Bayesian framework only emerges when going to a higher level of inference.

At the second level of inference, we do not consider the regularization parameters (or *hyperparameters*) fixed but also try to adapt them within the Bayesian framework. We start with the posterior distribution of the hyperparameters and turn the Bayesian crank:

$$p(\boldsymbol{\gamma}|D) = \frac{p(D|\boldsymbol{\gamma})p(\boldsymbol{\gamma})}{p(D)}. \tag{24}$$

The data-dependent term $p(D|\boldsymbol{\gamma})$ is the normalizing constant of the posterior $p(\boldsymbol{\theta}|D, \boldsymbol{\gamma})$ on the previous level of inference (23) and it is called the *evidence* for the hyperparameters $\boldsymbol{\gamma}$. It is assumed that we only have very weak prior knowledge about the hyperparameters and that we do not consider the *hyperprior* $p(\boldsymbol{\gamma})$. We now take the evidence (MacKay 1991) or type-II maximum likelihood (Berger 1985, page 99) approach and determine the most probable values for $\boldsymbol{\gamma}$ maximizing the posterior $p(\boldsymbol{\gamma}|D)$.[4] Ignoring the hyperprior and the denominator of (24) which does not depend on $\boldsymbol{\gamma}$, this boils down to maximizing the evidence $p(D|\boldsymbol{\gamma})$.

Maximizing the evidence can be done by expressing it in terms of the already defined likelihood and prior (using the sum and the product rule):

$$\text{evidence} = p(D|\boldsymbol{\gamma}) = \int p(D|\boldsymbol{\theta}, \boldsymbol{\gamma})p(\boldsymbol{\theta}|\boldsymbol{\gamma})d\boldsymbol{\theta} = \int p(D|\boldsymbol{\theta})p(\boldsymbol{\theta}|\boldsymbol{\gamma})d\boldsymbol{\theta}.$$

The above integral can sometimes be handled analytically, for example when both likelihood and prior are Gaussian and quadratic in the parameters $\boldsymbol{\theta}$. In general, however, it will require Monte Carlo methods (Neal 1996) or well-chosen approximations. Again we follow in the footsteps of MacKay (1991) and approximate the integrand by a Gaussian distribution around the most probable parameter values $\boldsymbol{\theta}_{\text{MP}}$. This leads to an analytical solution for the most probable hyperparameters maximizing the evidence. In the case of a single weight decay regularizer it gives the following solution (MacKay 1991):

$$\gamma := \frac{\delta}{||\boldsymbol{\theta}_{\text{MP}}||^2} \qquad \text{with} \qquad \delta = k - \gamma_i \text{tr}(\mathbf{H}^{-1}),$$

where k is the total number of parameters in $\boldsymbol{\theta}$ and $\mathbf{H}$ is the Hessian matrix given by the second derivatives of $-\ln p(\boldsymbol{\theta}|D)$ (evaluated at $\boldsymbol{\theta}_{\text{MP}}$); δ is generally referred to as the number of well-determined parameters and $0 \leq \delta \leq k$. In what follows, we will assume that all parameters are well-determined by the data and replace δ with the number of parameters k. This way we do not need to calculate and store the Hessian matrix which can become cumbersome when having hundreds of parameters (as is often our case). This approximation is called cheap and cheerful (MacKay 1994a).

The above is readily generalized from the case of a single weight decay regularizer to the more complicated ARD regularizer (22). This regularizer can also be interpreted as the logarithm of a prior distribution on the parameters with each hyperparameter γ_i controlling one of the columns of $\mathbf{W}$:

$$p(\mathbf{W}|\boldsymbol{\gamma}) = \prod_{i=1}^{d-1} \left(\frac{\gamma_i}{2\pi}\right)^{d/2} \exp\left(-\frac{1}{2}\gamma_i||\mathbf{w}_i||^2\right). \tag{25}$$

Each of the regularization parameters γ_i can be re-estimated separately (MacKay 1991) and using the cheap and cheerful approximation this gives:

$$\gamma_i := \frac{d}{||\mathbf{w}_i||^2}, \tag{26}$$

where we have used that each $\mathbf{w}_i$ is a vector of d parameters. The consequence of the above re-estimation formula is that columns $\mathbf{w}_i$, for which there is insufficient support from the data will be driven to zero, with the corresponding $\gamma_i \to \infty$. The un-used dimensions are switched off completely. The effective or underlying dimensionality of the model is defined as the number of columns $\mathbf{w}_i$ which remain non-zero.

A practical implementation of the evidence framework using an ARD regularizer (22), (25) consists of iteratively performing:

[4] A fully Bayesian treatment would use the entire distribution and not only the most probable hyperparameters; the distribution should then be integrated over in later stages.

1 Minimize a regularized error function (23): $E = -\ln p(D|\mathbf{W}) + \frac{1}{2}\sum_{i=1}^{d-1}\gamma_i ||\mathbf{w}_i||^2$.

2 Re-estimate the regularization parameters $\boldsymbol{\gamma}$ using (26): $\gamma_i := d/||\mathbf{w}_i||^2$.

Actually, in practice one often does not perform step 1 until convergence to a minimum; just a few iterations of the optimization algorithm suffice.

We now apply this framework to Fa and Ppca. Due to the approximation made, step 2 is straightforward. The regularization term in step 1, however, requires changing the Em algorithm for these models.

6.2 Bayesian Factor Analysis and PCA

We limit ourselves to the case of a single latent variable model in order to simplify the notation, but the extension to mixture models is easy. As before, we start with factor analysis and Ppca is dealt with as a special case.

Factor Analysis The log-likelihood maximized by the Em algorithm for factor analysis in the single component case was (15):

$$\mathcal{L}(\boldsymbol{\mu}, \mathbf{W}, \mathbf{R}) = -\frac{N}{2}\{d\ln(2\pi) + \ln|\mathbf{M}| + \mathrm{tr}(\mathbf{M}^{-1}\mathbf{S})\}.$$

The error function to be minimized including the Ard regularization term then becomes:

$$E = -\mathcal{L}(\boldsymbol{\mu}, \mathbf{W}, \mathbf{R}) + \frac{1}{2}\sum_{i=1}^{d-1}\gamma_i||\mathbf{w}_i||^2.$$

As one can see by checking the various steps of the Em algorithm for Mfas derived in Appendix A, the only part that changes is the estimation of $\mathbf{W}$ in the second stage M-step. All we need is the expected complete error function at that stage (51), while at the same time restricting ourselves to just one mixture component and adding the regularization term:

$$\begin{aligned}\mathcal{E}(\hat{E}_c) \quad = \quad & -\sum_n \Big[-\frac{1}{2}\ln|\mathbf{R}| - \frac{1}{2}(\mathbf{x}^n - \boldsymbol{\mu})^T\mathbf{R}^{-1}(\mathbf{x}^n - \boldsymbol{\mu}) + (\mathbf{x}^n - \boldsymbol{\mu})^T\mathbf{R}^{-1}\mathbf{W}\langle\, \mathbf{z}^n \,\rangle \\ & -\frac{1}{2}\mathrm{tr}\left\{\mathbf{W}^T\mathbf{R}^{-1}\mathbf{W}\langle\, \mathbf{z}^n(\mathbf{z}^n)^T\,\rangle\right\}\Big] + \frac{1}{2}\sum_{i=1}^{d-1}\gamma_i||\mathbf{w}_i||^2. \end{aligned} \tag{27}$$

We take the partial derivative with respect to $\mathbf{W}$ and put it to zero (which is just (52) plus the regularization term):

$$\frac{\partial\mathcal{E}(\hat{E}_c)}{\partial\mathbf{W}} = -\sum_n\left\{\mathbf{R}^{-1}(\mathbf{x}^n - \boldsymbol{\mu})\langle\,\mathbf{z}^n\,\rangle^T - \mathbf{R}^{-1}\mathbf{W}\langle\,\mathbf{z}^n(\mathbf{z}^n)^T\,\rangle\right\} + \frac{1}{2}\frac{\partial\sum_{i=1}^{d-1}\gamma_i||\mathbf{w}_i||^2}{\partial\mathbf{W}} = 0. \tag{28}$$

The last term can be written in matrix form with $\boldsymbol{\Gamma} = \mathrm{diag}(\boldsymbol{\gamma})$; using trace rotation, i.e. $\mathrm{tr}(\mathbf{XY}\cdots\mathbf{Z}) = \mathrm{tr}(\mathbf{ZXY}\cdots) = \cdots = \mathrm{tr}(\mathbf{Z}\cdots\mathbf{XY})$ and the diagonality of $\boldsymbol{\Gamma}$, we have:

$$\frac{\partial\sum_{i=1}^{d-1}\gamma_i||\mathbf{w}_i||^2}{\partial\mathbf{W}} = \frac{\partial\mathrm{tr}\left(\boldsymbol{\Gamma}\mathbf{W}^T\mathbf{W}\right)}{\partial\mathbf{W}} = 2\mathbf{W}\boldsymbol{\Gamma}.$$

Substituting this equation in (28) and introducing the shorthands for the sufficient statistics $\mathbf{X} = \sum_n(\mathbf{x}^n - \boldsymbol{\mu})\langle\,\mathbf{z}^n\,\rangle^T$ and $\mathbf{Z} = \sum_n\langle\,\mathbf{z}^n(\mathbf{z}^n)^T\,\rangle$ gives:

$$0 = -\mathbf{R}^{-1}\mathbf{X} + \mathbf{R}^{-1}\mathbf{WZ} + \mathbf{W}\boldsymbol{\Gamma}.$$

Multiplying both sides by $\mathbf{R}$:

$$\mathbf{X} = \mathbf{WZ} + \mathbf{RW\Gamma}. \tag{29}$$

The additional term due to regularization does complicate things, since there is no straightforward way of solving this system as before (where we had (53) $\mathbf{W} = \mathbf{XZ}^{-1}$). Our goal is to factor out $\mathbf{W}$ and this requires some further manipulation. We consider the equivalent system using the "vec" operator which stacks the columns of a matrix and applying it to both sides:

$$\text{vec}(\mathbf{X}) = \text{vec}(\mathbf{WZ}) + \text{vec}(\mathbf{RW\Gamma}).$$

Using $\text{vec}(\mathbf{ABC}) = (\mathbf{C}^T \otimes \mathbf{A})\text{vec}(\mathbf{B})$ (Golub and Van Loan 1996, page 180), $\text{vec}(\mathbf{W})$ can be factored out of the first term:

$$\text{vec}(\mathbf{X}) = (\mathbf{Z}^T \otimes \mathbf{I}_d)\text{vec}(\mathbf{W}) + \text{vec}(\mathbf{RW\Gamma}),$$

where $\otimes$ denotes the Kronecker product. Both $\mathbf{R}$ and $\mathbf{\Gamma} = \text{diag}(\boldsymbol{\gamma})$ are diagonal and defining $\mathbf{r}$ as the main diagonal of $\mathbf{R}$:

$$\text{vec}(\mathbf{X}) = (\mathbf{Z}^T \otimes \mathbf{I}_d)\text{vec}(\mathbf{W}) + \text{vec}\{(\mathbf{r}\boldsymbol{\gamma}^T) \circ \mathbf{W}\},$$

where $\circ$ denotes the element-wise (or Hadamard) matrix product. The element-wise product can be eliminated using that $\text{vec}(\mathbf{A} \circ \mathbf{B}) = \text{diag}\{\text{vec}(\mathbf{A})\}\text{vec}(\mathbf{B})$:

$$\text{vec}(\mathbf{X}) = [\mathbf{Z}^T \otimes \mathbf{I}_d + \text{diag}\{\text{vec}(\mathbf{r}\boldsymbol{\gamma}^T)\}]\text{vec}(\mathbf{W}). \tag{30}$$

At last, we have obtained a linear system for which we can find $\mathbf{W}$ with standard numerical methods. However, the matrix between square brackets is of size $d\ell \times d\ell$ and seemingly solving the system is $O(d^3\ell^3)$. Luckily, because of the sparseness of the identity matrix $\mathbf{I}_d$, it has at most $d\ell^2$ non-zero elements: the matrix is tiled with ℓ^2 diagonal matrices of size $d \times d$. Permuting rows and columns, it can actually be transformed into a matrix with d matrices of size $\ell \times \ell$ on the main diagonal. Such *banded* systems can be solved with complexity $O(d\ell^3)$ (Golub and Van Loan 1996, page 153).

Principal Component Analysis For PPCA the inclusion of the regularization term requires only a simple modification of the standard EM algorithm. We go back to the system we started with (29) and substitute the spherical noise covariance matrix of PPCA $\mathbf{R} = \sigma^2\mathbf{I}_d$:

$$\mathbf{X} = \mathbf{WZ} + \sigma^2\mathbf{W\Gamma}.$$

Now it is straightforward to factor out $\mathbf{W}$:

$$\mathbf{X} = \mathbf{W}(\mathbf{Z} + \sigma^2\mathbf{\Gamma})$$

and substituting the definitions of $\mathbf{X}$ and $\mathbf{Z}$, the estimate for $\mathbf{W}$ becomes (Bishop 1999a):

$$\mathbf{W} = \left[\sum_n (\mathbf{x}^n - \boldsymbol{\mu})\langle\, \mathbf{z}^n \,\rangle^T\right] \left[\sum_n \langle\, \mathbf{z}^n(\mathbf{z}^n)^T \,\rangle + \sigma^2\mathbf{\Gamma}\right]^{-1}. \tag{31}$$

Comparing this with the update for $\mathbf{W}$ in Algorithm 1 for MFAs, we see that this only introduced an additional term $\sigma^2\mathbf{\Gamma}$. This means that for PPCA and MPCA, the regularization term does not change computational complexity and storage requirements.

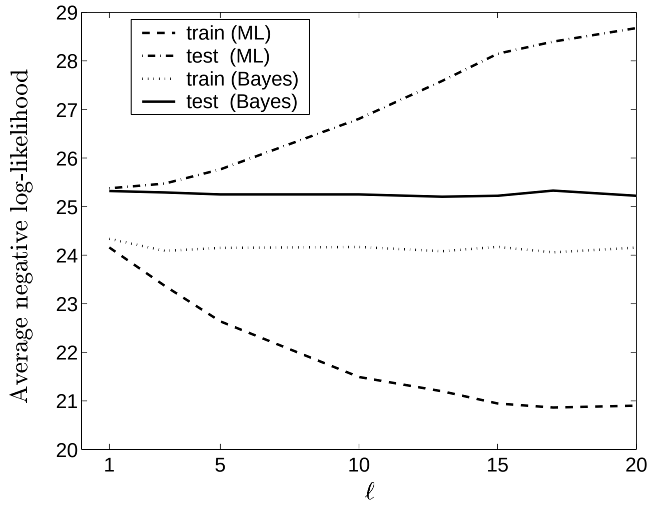

Figure 4: Comparison of a MPCA trained with maximum likelihood and a MPCA trained in a Bayesian framework on waveform: the average negative log-likelihood versus the dimension of latent space ℓ with a mixture of 4 PCAs. Results are the average of a 5x2cv experiment.

6.3 Experiments: Toy Data

We performed density estimation on the waveform data of section 5.2 with a 4-component MPCA using both ML and Bayesian estimation. A mixture of maximum likelihood PCAs starts overfitting the training data when the dimension of latent space is increased (Figure 4). A Bayesian mixture of PCAs resolves this problem and both training and test error stay on the same level when increasing ℓ. This can be explained by the fact that Bayesian estimation successfully determines the underlying dimensionality for each of the mixture components. In this experiment, in average 2-3 dimensions per component are retained and the other columns of the generative matrices are put to zero.

6.4 Experiments: Real-World Data

We now go back to the real-world data sets considered in section 5.3. A series of experiments has been performed in which the best MPCAs and MFAs found with maximum likelihood (Table 3) are compared with their Bayesian cousins. The set-up of the experiments for training the Bayesian MPCAs and MFAs is almost the same as the framework described in section 5.1. The difference lies in the estimation of the parameters for which 40 iterations of EM were performed with the new updates for the generative matrices: (31) for MPCAs and (30) for MFAs. After each fifth iteration of the EM algorithm, the hyperparameters $\boldsymbol{\gamma}$ were re-estimated (26).

The results for MPCAs are in Table 4. For convenience, the results for the best ML MPCAs found in Table 3 have been copied here together with the corresponding number of mixture components and the dimension of latent space. The Bayesian MPCAs had the same number of mixture components as their ML counterparts and a latent dimension of $\ell = d - 1$ (except for the soybean and twos data to reduce computational complexity). When comparing the negative log-likelihood scores of the ML and the Bayesian models, one observes that Bayesian estimation performs significantly better in five cases and never significantly worse. For optical, segmentation, soybean, and twos, this is due to the fact that in the ML experiments, we did not include values of ℓ which are high enough. Choosing the dimension of latent space equal to the average value found by Bayesian MPCA, the scores obtained with ML come closer to the ones for Bayesian estimation although they are still slightly worse.

The average selected dimensionality over all mixture components with Bayesian MPCAs is given in the last column of Table 4. This value is often close to the value found on validation data by ML given

		ML MPCA		Bayesian MPCA	
Data	# components	likelihood	# factors	likelihood	# factors
banana	6	2.6(0.08)	1	2.6(0.09)	0.6/1
cancer	6	3.7(1.43)	5	3.0(1.31)	3.7/9
dermatology	6	37.9(0.66)	1	36.5(1.49)	3.9/33
glass	4	11.1(3.62)	1	10.1(2.56)	4.5/8
heart	2	18.0(0.24)	1	17.6(0.90)	3.5/12
ionosphere	8	29.4(2.20)	1	26.2(1.42)	2.2/33
iris	2	2.6(0.31)	3	2.8(0.52)	2.5/3
letter	20	12.0(0.14)	15	$\underline{11.7}$(0.16)	12.2/15
optical	20	46.1(6.55)	15	$\underline{26.8}$(5.72)	45.6/63
pen	20	3.9(0.35)	15	3.8(0.38)	11.4/15
satimage	10	−3.1(0.43)	15	−3.6(0.23)	14.8/35
segmentation	10	−7.6(7.69)	8	−17.9(5.33)	10.4/18
sonar	1	73.5(3.11)	5	$\underline{71.2}$(3.38)	16.4/59
soybean	4	−13.2(9.40)	8	$\underline{-116.0}$(17.83)	21.3/50
twos	4	−286.7(3.82)	15	$\underline{-550.7}$(8.91)	63.2/70
vowel	11	11.5(0.31)	3	11.2(0.25)	3.8/9
waveform	1	24.8(0.27)	10	24.9(0.34)	7.9/20
waveform-noise	2	53.5(0.46)	1	53.5(0.47)	3.9/39

Table 4: A comparison of input density modeling with MPCAs trained with maximum likelihood and in the Bayesian framework. Scores are in average negative log-likelihood on the test set and are the average over 10 experiments in a 5x2cv F test set-up. The standard deviation is given between parentheses. The ML score corresponds to the best MPCA model as evaluated on a validation set. The second column (# components) indicates the number of mixture components in the "best" model. The fourth column specifies the dimension of latent space of the best model. The Bayesian approach has been applied to a MPCA with the same number of components (as indicated by the second column) and with a number of factors $\ell = d-1$. The final column gives the average number of factors selected by the ARD prior in the Bayesian approach versus ℓ. The underlined scores are significantly better on a 5x2cv F test with 95% confidence.

in the fourth column. Bayesian estimation is a useful alternative to cross-validation indeed. Moreover, for MPCAs it almost comes for free when using the cheap and cheerful approximation to the evidence scheme.

The results for MFAs are in Table 5. The set-up was as with the Bayesian MPCAs and again the ML scores have been copied from Table 3. The likelihood scores are also quite comparable, with Bayesian estimation performing significantly better in two cases and worse on optical and soybean. A possible explanation for the latter might be that in this particular case, the cheap and cheerful approximation breaks down due to the rather high dimension of data space and latent space. An indication is that scores comparable to the ML score are obtained when estimating Bayesian MFAs for optical and soybean with $\ell = 20$. The average selected dimensionality over all mixture components with Bayesian MFAs in the last column of Table 5 is again quite close to the value selected on validation data by ML given in the fourth column. Thus, also for MFAs the Bayesian approach is a viable alternative to costly cross-validation in a ML framework.

6.5 Discussion

The price one has to pay for the Bayesian approach for FA is that even when using the cheap and cheerful approximation, it is computationally more expensive than the standard EM algorithm. Solving

		ML MFA		Bayesian MFA	
Data	# components	likelihood	# factors	likelihood	# factors
banana	6	2.6(0.10)	1	2.7(0.05)	0.5/1
cancer	2	6.1(0.98)	5	$\underline{4.7}$(1.11)	5.3/9
dermatology	4	33.4(9.42)	5	30.8(6.25)	6.4/33
glass	1	13.1(4.93)	5	11.3(4.01)	8.0/8
heart	6	14.5(1.38)	1	13.3(3.58)	2.1/12
ionosphere	1	37.5(1.72)	5	40.6(3.59)	21.0/33
iris	2	2.7(0.25)	2	2.6(0.28)	2.0/3
letter	20	11.6(0.50)	15	12.1(0.14)	11.8/15
optical	10	$\underline{-12.5}$(1.71)	15	1.3(2.52)	52.1/63
pen	20	−4.7(0.74)	15	−4.4(0.48)	11.7/15
satimage	6	−4.4(0.36)	15	−4.2(0.24)	16.5/35
segmentation	10	−20.6(4.66)	8	−20.5(4.38)	8.5/18
sonar	1	68.9(4.54)	15	67.8(2.51)	15.0/59
soybean	6	$\underline{-196.7}$(6.85)	8	−160.4(9.22)	5.6/50
twos	4	323.6(7.19)	15	$\underline{-545.6}$(7.58)	61.0/70
vowel	11	11.5(0.39)	5	$\underline{11.2}$(0.27)	3.6/9
waveform	4	24.3(0.25)	1	24.4(0.24)	1.1/20
waveform-noise	1	51.6(0.51)	3	51.6(0.53)	2.3/39

Table 5: A comparison of input density modeling with MFAs trained with maximum likelihood and in the Bayesian framework. See Table 4 for additional details.

the linear system (30) is $O(d\ell^3)$, which is cumbersome for high-dimensional data and when choosing $\ell = d-1$. Luckily, this is in general also the case in which we do not expect a high effective dimensionality of latent space. Thus, it would not harm to follow the Bayesian approach but with a lower value of ℓ. This might also resolve some of the problems of the cheap and cheerful approximation since many parameters are likely not to be well-determined for high values of ℓ.

Recently, alternative Bayesian treatments for PCA (Bishop 1999b) and MFAs (Ghahramani and Beal 2000) have been proposed. Both take a *variational* approach to Bayesian inference. This exploits the same idea that we encountered in the derivation of the generalized EM Algorithm in order to approximate the posterior of the hidden variables. For Bayesian inference the parameters $\boldsymbol{\theta}$ are considered hidden and the posterior distribution is $p(\boldsymbol{\theta}|D)$. As we saw, integrating over this distribution is in general not feasible and we used a Gaussian approximation around the most probable parameter values $\boldsymbol{\theta}_{\mathrm{MP}}$. The choice of a Gaussian is quite arbitrary and mainly motivated by analytical simplicity. The variational approach introduces an approximating distribution $Q(\boldsymbol{\theta}|D)$ in order to make the integration tractable. Following the same reasoning as in the derivation of the EM algorithm, this leads to the idea of maximizing a lower bound of the evidence $p(D)$ (Jordan et al. 1999). It turns out that in many cases one only needs to assume that the approximating Q factorizes in a specific way, to keep the maximization tractable. The variational maximization determines the specific functional form of Q given this factorization and the priors on the parameters $\boldsymbol{\theta}$.

Variational PCA (Bishop 1999b) not only includes the ARD prior on the weights $\mathbf{W}$ but also priors for the other parameters of the model. The first moments of the optimal Q distributions for W and the hyperparameters $\boldsymbol{\gamma}$ turn out to be very similar to the ones found above with our approximations. Ghahramani and Beal (2000) went one step further and proposed to use variational inference for MFAs. Their procedure cannot only automatically determine the dimensionality for each of the mixture components but also the number of components. While pruning components fits nicely in the variational framework, component birth is done in a heuristic way by splitting existing components. Performance on a number of toy examples is promising. The first moments of the optimal Q distributions for $\mathbf{W}$

and the hyperparameters γ are again almost identical to the ones found above.

It has also been proposed to use an annealing scheme for performing model selection with MPCAs (Meinicke and Ritter 1999). The disadvantage of this approach is that at its heart it repeatedly uses an EM algorithm on full GMMs and an eigendecomposition of weighted covariance matrices. This becomes computationally expensive for high-dimensional data and mixtures of latent variable models were introduced to avoid exactly this!

7 Bayes Classifiers with Mixture Models

A classic approach for a classification problem with C classes $\{\mathcal{C}_k\}$ is based on rewriting the conditional density with Bayes' rule:

$$P(\mathcal{C}_k|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_k)P(\mathcal{C}_k)}{p(\mathbf{x})} = \frac{p(\mathbf{x}|\mathcal{C}_k)P(\mathcal{C}_k)}{\sum_j p(\mathbf{x}|\mathcal{C}_j)P(\mathcal{C}_j)} = \frac{\text{class-conditional} \times \text{prior}}{\text{normalization}} \tag{32}$$

and modeling the *prior* $P(\mathcal{C}_k)$ and the *class-conditional* densities $p(\mathbf{x}|\mathcal{C}_k)$. The prior represents the probability that an arbitrary example out of our data belongs to class $\mathcal{C}_k$. The class-conditional distribution models the density of the data belonging to class $\mathcal{C}_k$ and we can use the density models of the previous sections in this context. The probability of making an error when classifying an example $\mathbf{x}$ is minimized by Bayes' decision rule of assigning it to the class with the largest posterior probability (Duda and Hart 1973):

$$\mathbf{x} \text{ is assigned to } \mathcal{C}_k \qquad \Leftrightarrow \qquad P(\mathcal{C}_k|\mathbf{x}) \geq P(\mathcal{C}_j|\mathbf{x}) \quad \text{for all } j \neq k.$$

We first report on the results of a series of experiments with mixture models for estimating the class-conditional densities in a Bayes classifier. That is, each class conditional density is modeled by a separate mixture model is:

$$p(\mathbf{x}|\mathcal{C}_k) = \sum_i \alpha_{ki} p_{ki}(\mathbf{x}),$$

and the corresponding Bayes decision rule:

$$\mathbf{x} \text{ is assigned to } \mathcal{C}_k \qquad \Leftrightarrow \qquad P(\mathcal{C}_k)\sum_i \alpha_{ki} p_{ki}(\mathbf{x}) \geq P(\mathcal{C}_j)\sum_i \alpha_{ji} p_{ji}(\mathbf{x}) \quad \text{for all } j \neq k. \tag{33}$$

Oddly enough, the use of mixture models, as opposed to simple probability distributions, in Bayes classifiers is not that widespread in the machine learning community, but see for example (Hastie and Tibshirani 1996; Hastie, Tibshirani, and Buja 1999; Kambhatla and Leen 1995).

7.1 Experimental Set-Up

The experiments were set up in the same way as those for density estimation with mixture models described in section 5.1. The main difference is that we now take the class labels into account and form separate training sets for each class on which to train the mixture models. These separate mixture models were initialized and trained with the EM algorithm as in section 5.1.

The NIST and satimage data come with a fixed division in a training and a test set (see Table 1) and McNemar's test was used to determine whether the difference in performance between two methods was statistically significant (Dietterich 1998). This test involves only a single training run. For all other data sets the 5x2cv F test was used with five replications of twofold cross-validation. In each round one third of the training data was set aside for validation purposes (respecting the class distributions) on the entire Bayes classifier. The results on the validation data were used to select the best model for each type of mixture model on a data set. For most data sets, the number of mixture components tried for each class-conditional density had at least five different values ranging from 1 to a value depending on the complexity of the problem. The dimension of latent space for MPCAs and MFAs was also varied, upper-bounded by the dimension of data space.

	GMM					
Data	spherical	diagonal	tied	full	MPCA	MFA
banana	$89.2(1.79)^{5}$	$\underline{90.9}(1.19)^{5}$	$88.8(1.74)^{5}$	$\underline{93.0}(0.60)^{5}$	$\underline{93.3}(1.80)^{5,1}$	$\underline{93.0}(0.87)^{5,1}$
cancer	$\underline{96.1}(1.22)^{3}$	$95.8(0.82)^{2}$	$\underline{94.7}(1.08)^{2}$	$94.3(1.32)^{3}$	$95.2(1.10)^{2,3}$	$95.0(1.38)^{1,1}$
dermatology	$\underline{95.7}(1.03)^{1}$	$93.7(2.26)^{2}$	$93.1(1.57)^{2}$	$92.4(2.37)^{2}$	$94.9(1.31)^{1,1}$	$94.1(2.02)^{1,3}$
glass	$61.5(3.87)^{3}$	$63.0(4.43)^{3}$	$62.0(3.55)^{5}$	$61.8(4.29)^{3}$	$56.9(4.75)^{5,1}$	$63.5(4.38)^{2,1}$
heart	$\underline{81.7}(1.95)^{1}$	$\underline{81.7}(2.44)^{1}$	$73.0(2.72)^{5}$	$76.5(3.22)^{1}$	$\underline{81.1}(2.33)^{1,1}$	$\underline{81.9}(3.31)^{1,1}$
ionosphere	$90.7(1.33)^{5}$	$89.5(1.49)^{2}$	$87.6(3.05)^{2}$	$85.5(2.64)^{1}$	$\underline{93.9}(1.33)^{1,3}$	$\underline{93.3}(1.28)^{1,3}$
iris	$92.9(3.14)^{3}$	$93.5(2.42)^{1}$	$\underline{95.6}(1.89)^{3}$	$\underline{95.8}(2.47)^{2}$	$\underline{96.5}(1.79)^{1,3}$	$\underline{96.9}(1.99)^{1,1}$
letter	$84.9(0.35)^{20}$	$86.9(0.72)^{20}$	$\underline{95.0}(0.31)^{20}$	$94.2(0.36)^{5}$	$92.0(0.59)^{10,5}$	$94.1(0.27)^{10,5}$
NIST	94.2^{20}	95.7^{40}	95.0^{3}	96.7^{10}	$\underline{97.1}^{10,5}$	$\underline{97.3}^{20,10}$
optical	$93.8(0.61)^{10}$	$93.1(0.67)^{5}$	$\underline{96.2}(0.49)^{1}$	$\underline{96.2}(0.45)^{1}$	$\underline{96.5}(0.74)^{2,15}$	$95.5(0.79)^{3,10}$
pen	$97.1(0.50)^{10}$	$96.6(0.35)^{10}$	$98.5(0.23)^{10}$	$\underline{99.2}(0.15)^{10}$	$98.3(0.21)^{5,3}$	$98.5(0.20)^{10,5}$
satimage	86.1^{10}	87.6^{10}	87.8^{10}	87.0^{5}	$\underline{88.9}^{10,5}$	$87.2^{10,3}$
segmentation	$89.1(0.76)^{10}$	$89.7(3.95)^{10}$	$\underline{91.8}(1.39)^{10}$	$\underline{92.9}(1.32)^{5}$	$89.2(2.12)^{5,5}$	$\underline{91.6}(0.95)^{10,3}$
sonar	$\underline{75.8}(3.93)^{5}$	$\underline{74.4}(6.03)^{2}$	$69.5(2.44)^{5}$	$\underline{77.1}(5.51)^{5}$	$\underline{78.1}(4.10)^{1,3}$	$\underline{81.6}(6.32)^{2,3}$
soybean	$85.3(3.15)^{2}$	$90.7(2.02)^{2}$	$\underline{93.0}(1.09)^{2}$	$\underline{92.1}(0.90)^{1}$	$89.0(1.73)^{1,5}$	$\underline{92.9}(1.82)^{1,1}$
vowel	$72.7(2.58)^{5}$	$75.6(2.11)^{5}$	$81.8(1.96)^{5}$	$\underline{87.9}(2.84)^{5}$	$78.6(2.19)^{2,3}$	$85.2(2.51)^{2,5}$
waveform	$\underline{84.1}(1.21)^{2}$	$\underline{81.8}(2.37)^{3}$	$76.8(2.70)^{2}$	$75.3(1.61)^{1}$	$\underline{80.1}(2.55)^{3,1}$	$\underline{83.2}(1.75)^{2,1}$
waveform-noise	$\underline{79.9}(2.71)^{1}$	$\underline{79.2}(2.46)^{1}$	$65.8(2.35)^{1}$	$66.2(2.76)^{1}$	$77.8(2.14)^{1,1}$	$\underline{80.6}(1.48)^{1,1}$
wins	6	5	6	8	9	10

Table 6: Results of the experiments with Bayes classifiers using different mixture models for modeling the class-conditional densities. Scores are in percentage of correct classification on the test set and are the average over 10 experiments in a 5x2cv F test framework or a single run for NIST and satimage. The standard deviation is given between parentheses. Each of the scores corresponds to the best model out of the particular class of models as selected on validation data. The first superscript indicates the number of mixture components of each class conditional density in the best model. For mixtures of latent variable models, the second superscript specifies the dimension of latent space. The underlined scores are the ones that do not pass the 5x2cv F test or McNemar's test with 90% confidence when compared with the model having the best score.

In each Bayes classifier, the mixture models for each class-conditional density were chosen to be identical, that is, of the same type, with the same number of components and, for the latent variable models, with the same dimension of latent space. Classification of the examples was based on the Bayes decision rule (33). The priors $P(\mathcal{C}_k)$ were estimated on the basis of the training set as the proportion of samples of class $\mathcal{C}_k$.

Note that diagonal GMMs and MFAs are prone to a particular kind of overfitting for some of the data sets which have many zero-valued attributes. These two models are well-suited for capturing this kind of absence of noise. While this might seem suitable for density estimation, it often leads to weakly performing Bayes classifiers. This can be explained by the fact that most class-conditional densities capture the presence of zero values and this reduces their discriminative power. As a remedy, a simple form of regularization was used by imposing a small threshold of 0.01 upon the values of $\mathbf{R}_j$ for MFAs and upon the variance parameters for a diagonal GMM; this was done for the dermatology, NIST, optical, and soybean data sets.

7.2 Experiments: Real-World Data

The results of the experiments with Bayes classifiers are listed in Table 6, where the best method and the ones that are not significantly worse (90% on the 5x2cv F test or McNemar's test) are set underlined. The last row in Table 6 gives an idea of the global performance of the different models by specifying the total number of underlined scores for each type of model. The number of wins shows that MFAs and MPCAs provide the best Bayes classifiers though their advantage is not as clear-cut as with unsupervised density modeling. This is related to the fact that in Bayes classifiers the relation between the quantity we optimize, viz. the likelihood of the data from a given class, and the ultimate goal, viz. minimizing classification error, is obscure. It can very well happen that the Bayes classifier with highest average likelihood on the test data is not the one that minimizes the classification error.

Mixtures of latent variable models are outperformed twice by mixtures of spherical GMMs on the cancer and dermatology data. This seems to be due to overfitting since the mean classification error on the training set was consistently higher with the more complex mixture models. Spherical and diagonal GMMs give satisfactory results on small data sets since data can be very scarce in this case with only few examples for each class. The mixtures of latent variable models are also outperformed three times by mixtures of tied or full GMMs. These include data sets for which data is plentiful such as letter and pen on which full GMMs already showed good performance in density estimation (Table 3).

It is also interesting to note that in most cases, the best mixture model has several components and performs better than a single component model of the same type. A general remark on these Bayes classifiers is that their classification error often is comparable with other state-of-the-art classifiers. A nice example is the score (95% correct) obtained on letter with a mixture of 20 tied GMMs for each of the 26 classes. The best result obtained in the StatLog project (Michie, Spiegelhalter, and Taylor 1994) in which a host of statistical and machine learning were compared, was 93.6%. This score has been improved to around 98% (Schwenk and Bengio 1998) only very recently with boosting methods such as AdaBoost.

The results on the NIST handwritten digit data are satisfactory as well: the best score of 97.3% was obtained by modeling each of the ten digit classes with a 10-component 10-factor MFA. This is comparable to the best scores we obtained with large MLPs.

7.3 Experiments: MNIST Data

We also performed experiments on the MNIST data set, a large collection of handwritten digits which can be obtained from (LeCun 2000). It comes as 28×28 grey level images in a training set of 60,000 examples and a test set of 10,000 examples. Each of the 8-bit pixel values is scaled in the interval $[0, 1]$. The MNIST data set as it is available from (LeCun 2000) is already normalized and centered. However, it is still highly diverse in writing style, line thickness, slant, and arc size. This makes it a challenging test bed for learning algorithms. It also has the advantage of being widely used; this enables the comparison with many other classifiers.

The classifiers constructed are again Bayes classifiers using mixture models; the set-up of the experiments is similar to the scheme described in section 7.1. A simple form of regularization was used by imposing a small threshold of 0.01 upon the values of $\mathbf{R}_j$ for mixtures of latent variable models and upon the variance parameters for a diagonal GMM. Both GMMs and mixtures of latent variable models were used for modeling each of the ten classes in a Bayes classifier. Mixtures models were trained varying the number of mixture components $m \in \{10, 20, 30, 40, 50, 100, 200\}$ and the dimension of latent space for MPCAs and MFAs $\ell \in \{10, 20, 30, 40\}$ (not all possible combinations were tried though). All classifiers have been trained only once and the corresponding results on the 10,000 example test set are given in Table 7. McNemar's test was used to determine whether the difference in performance between two classifiers was statistically significant. These results were obtained with subsampled 16×16 images in order to reduce the computational complexity.

As one can see, the GMMs are again not flexible enough to model the data. Even when increasing the number of mixture components to 200, performance is still significantly worse than for the other models. Mixtures of latent variable models show good performance on the MNIST data set, also when compared with previous results using other classifiers, as we will see later on in this section. This does not come as a surprise as already in (Hinton et al. 1997; Tipping and Bishop 1999) good results are obtained with MPCAs and MFAs on a smaller data set of handwritten digits. Hinton et al. (1997) proposed the use of mixtures of latent variable models since digit images are supposed to lie on a

		Gmm			
m	ℓ	spherical	diagonal	Mpca	Mfa
10	10	92.0	92.6	97.9	97.9
	20			98.2	98.0
	30			98.3	98.3
20	10	93.6	93.8	98.0	97.7
	20			98.3	98.3
	30			98.3	98.3
30	10	94.2	94.7	98.1	97.9
	30			$\underline{98.5}$	98.4
40	20	94.4	94.9	$\underline{98.5}$	$\underline{98.4}$
	40			$\underline{98.5}$	$\underline{98.5}$
50	20	95.0	95.2	$\underline{98.4}$	$\underline{98.4}$
100		95.8	95.9		
200		95.8	96.5		

Table 7: Results of the experiments with Bayes classifiers on the MNIST data set. Scores are in percentage of correct classification on the test set. The underlined scores are best with 90% confidence using McNemar's test. m indicates the number of mixture components or the number of experts. ℓ indicates the dimension of latent space for the mixtures of latent variable models.

non-linear, lower-dimensional, and smooth manifold. Using a mixture model takes into account the non-linearity of the manifold, with the mixture components capturing different writing styles and significant transformations. Each mixture component corresponds to a *linear* latent variable model and captures local invariances induced by small transformations.

The different writing styles and transformations captured by a Mpca are illustrated in Figure 5. Each digit class was modeled by a mixture of 20 Ppcas with a dimension of latent space $\ell = 20$. Figure 5 shows the means of each mixture model. Examples of different writing styles can be seen in the presence of twos with and without curl and of several sevens with an additional bar. Variations in line thickness have also been taken into account; see for example, the first and second center for 1. The means incorporate the variations in slant of the digits as well.

A priori one might have expected Mfas to outperform Mpcas, given the more flexible diagonal noise model of factor analyzers. We can see two possible explanations for the fact that their performance is quite similar. Firstly, the normalization of the digit images during pre-processing might lessen the need of a diagonal noise model. Secondly, as in the previous sections, the noise model in a Mfa was assumed to be tied across the mixture components. Maybe this is too strong a constraint and better results might be obtained by not tying the noise model. Similar performance with Mpcas and Mfas on a smaller handwritten digit data set was also observed in (Hinton et al. 1997).

Bayes classifiers with mixtures of latent variable models are among the best classifiers on the MNIST data, although their performance is not yet state-of-the-art. The best off-the-shelf classifier is a support vector machine with a polynomial kernel (1% error). Other classifiers which perform better incorporate, in one way or another, prior knowledge of the problem of handwritten digit recognition. We expect that better results can also be obtained with mixtures of latent variable by some additional pre-processing. A simple approach would be to straighten up the slanted digits. This has proved fruitful for other classifiers. In fact, the best result with a multi-layer perceptron has been obtained in this way and reduced the misclassification rate to 1.6% (LeCun et al. 1995). A more ambitious step would be to incorporate transformation invariance directly into the latent variable model (Jojic and Frey 2000).

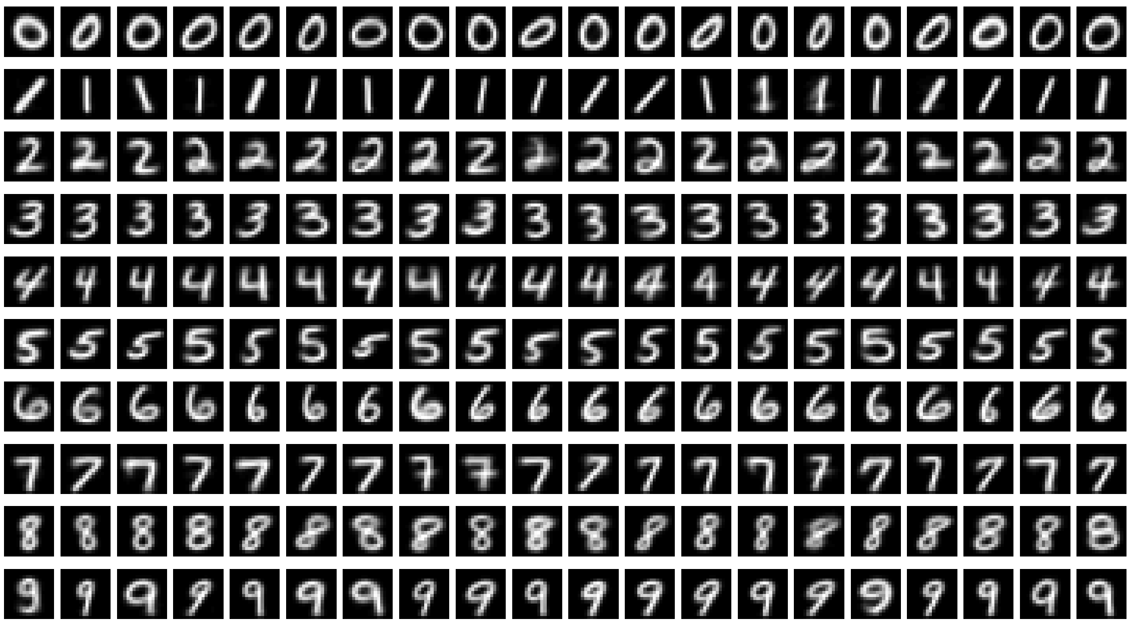

Figure 5: Means of a MPCA with 20 mixture components (and 20 factors) for each digit class of the MNIST data set.

7.4 Experiments: Bayesian MPCAs and MFAs

In section 6, we saw that it is possible to train mixtures of latent variable models in a Bayesian framework such that the appropriate dimensions of latent space are determined automatically. This can be especially interesting in the context of Bayes classifiers since in the above experiments we assumed that the number of factors is equal for all mixture components and for each class. This was necessary to keep model selection on validation data tractable. One would expect that within the Bayesian framework results can be obtained which are at least as good since the Bayesian framework enables us to search a larger model space. Note that ideally, we would also like to automatically select the number of mixture components for each class. This is another free parameter which was assumed to be equal for all class-conditional densities in the above experiments. A heuristic Bayesian way to select the number of components has been proposed recently for MFAs (Ghahramani and Beal 2000). In the experiments we did not pursue this idea and fixed the number of mixture components.

The set-up of the experiments is almost the same as the framework described in section 7.1. The main difference is that for each mixture model 40 iterations of EM were performed with the "Bayesian" updates for the generative matrices: (31) for MPCAs and (30) for MFAs. After each fifth iteration of the EM algorithm, the hyperparameters γ were re-estimated (26).

The results for Bayes classifiers with MPCAs are listed in Table 8. For convenience, the results for the best ML mixtures found in Table 6 have been copied here together with the number of mixture components and the dimension of latent space. The Bayesian MPCAs had the same number of mixture components as their ML counterparts and a latent dimension of $\ell = d$–1. The results for Bayes classifiers with MFAs are in Table 9. The set-up was as with the Bayesian MPCAs and again the ML scores have been copied from Table 6.

For the high-dimensional NIST, optical, sonar, and soybean data sets, a lower value of ℓ was chosen both for MPCAs and MFAs. This was done partly to save computation time and partly to avoid problems with the cheap and cheerful approximation already outlined in section 6. We observed that on the optical and sonar data performance deteriorated when taking $\ell = d - 1$. In this case, the value of ℓ was again selected on validation data, which is not very satisfactory since this is exactly what we

		ML MPCA		Bayesian MPCA	
Data	# components	classification (%)	# factors	classification (%)	# factors
banana	5	93.3(1.80)	1	92.5(1.71)	0.7/1
cancer	2	95.2(1.10)	3	95.7(0.71)	4.1/9
dermatology	1	94.9(1.31)	1	95.8(1.73)	5.2/33
glass	5	56.9(4.75)	1	59.1(3.71)	0.1/8
heart	1	81.1(2.33)	1	81.2(3.47)	2.3/12
ionosphere	1	93.9(1.33)	3	89.3(3.22)	20.0/33
iris	1	96.5(1.79)	3	96.0(3.07)	2.0/3
letter	10	92.0(0.59)	5	$9\underline{3.1}$(0.49)	5.8/15
NIST	10	97.1	5	97.3	18.5/20
optical	2	96.5(0.74)	15	97.1(0.53)	12.9/15
pen	5	98.3(0.21)	3	$\underline{98.7}$(0.25)	9/15
satimage	10	88.9	5	88.8	9.4/35
segmentation	5	89.2(2.12)	5	87.8(3.04)	4.8/18
sonar	1	78.1(4.10)	3	78.5(3.60)	10.0/30
soybean	1	89.0(2.55)	5	86.4(2.54)	0/50
vowel	2	78.6(2.19)	3	78.9(2.78)	3.6/9
waveform	3	80.1(2.55)	1	$\underline{83.2}$(2.48)	2.5/20
waveform-noise	1	77.8(2.14)	1	79.4(2.27)	2.5/39

Table 8: A comparison of Bayes classifiers with MPCAs trained with maximum likelihood and in the Bayesian framework of section 6. Scores are in percentage of correct classification on the test set and are the average over 10 experiments in a 5x2cv F test set-up or a single run for NIST and satimage. The standard deviation is given between parentheses. The ML score corresponds to the best MPCA model as evaluated on validation data. The second column (# components) indicates the number of mixture components in the "best" model. The fourth column specifies the dimension of latent space of the best model. The Bayesian approach has been applied to a MPCA with the same number of components (as indicated by the second column) and with a number of factors $\ell = d-1$. The final column gives the mean number of factors selected by the ARD prior in the Bayesian approach versus ℓ. The underlined scores are significantly better on a 5x2cv F test with 90% confidence.

wanted to avoid in the Bayesian framework! Luckily it is only an upper bound on the dimensions of latent space that can be selected in the Bayesian framework and difference in performance is small as long as ℓ is not close to $d-1$.

When comparing the classification scores of the ML and the Bayesian models, one observes that for Bayesian estimation the performance is slightly better. It is only outperformed once by ML estimation for a MFA on the vowel data. This does not show the improvement we were hoping for by having a more fine-grained model search. However, Bayesian estimation is certainly a useful and efficient approach since cross-validation is only needed for selecting the number of mixture components. The average selected dimensionality over all mixture components and all classes with Bayesian mixtures is given in the last column of Tables 8 and 9. This value is often quite close to the value selected on validation data with ML, especially for MFAs. With MPCAs the selected number of factors is in general higher than with ML.

8 Conclusions and Discussion

Mixtures of latent variable models were demonstrated to consistently outperform Gaussian mixture models on 18 real-world data sets when applied to the problem of density estimation. This holds both in terms of generalization performance and computational complexity. A disadvantage of mixtures of

		ML MFA		Bayesian MFA	
Data	# components	classification (%)	# factors	classification (%)	# factors
banana	5	93.0(0.87)	1	93.1(0.77)	0.7/1
cancer	1	95.0(1.38)	1	94.9(1.07)	4.2/9
dermatology	1	94.1(2.02)	3	95.3(1.51)	0/33
glass	2	63.5(4.38)	1	63.8(4.04)	0.7/8
heart	1	81.9(3.31)	1	81.0(3.57)	2/12
ionosphere	1	93.3(1.28)	3	92.1(2.71)	19.5/33
iris	1	96.9(1.99)	1	96.9(2.74)	1.3/3
letter	10	94.1(0.27)	5	93.9(0.31)	4.4/15
NIST	20	97.3	10	97.6	5.4/30
optical	3	95.5(0.79)	10	95.8(0.46)	5.4/15
pen	10	98.5(0.20)	5	98.3(0.39)	5.4/15
satimage	10	87.2	3	89.4	8.1/35
segmentation	10	91.6(0.95)	3	90.9(1.24)	3.8/18
sonar	2	81.6(6.32)	3	83.6(1.72)	6.4/30
soybean	1	92.9(1.82)	1	92.7(0.87)	0/50
vowel	2	85.2(2.51)	5	83.1(2.49)	2.7/9
waveform	2	83.2(1.75)	1	84.3(1.48)	0.8/20
waveform-noise	1	80.6(1.48)	1	81.1(1.68)	1.0/39

Table 9: A comparison of Bayes classifiers with MFAs trained with maximum likelihood and in the Bayesian framework of section 6. See Table 8 for more details.

latent variable models is that one has to choose extra free parameters, viz. the dimensions of latent space. This was first done using validation data, but it would be far more attractive if the appropriate dimensions could be determined automatically. It was shown that a Bayesian procedure can be used for this purpose without much computational burden. Bayesian PCA (Bishop 1999a) was derived as a special case of Bayesian FA. A series of experiments on toy and real-world data illustrated that the method is capable of finding the appropriate dimensions of latent space during training. Results obtained are at least as good as with maximum likelihood while avoiding a discrete model search.

Mixture models were used as class-conditional densities in a Bayes classifier. Also in this case, mixtures of latent variable models perform better than GMMs and Bayesian inference again proved to be effective. On the MNIST data set results were obtained which are close to the best state-of-the-art classifiers. Mixtures of latent variable models are likely to become an alternative to GMMs in other contexts, for example as an output density in a hidden Markov model (Saul and Rahim 1998).

The above mixture models are mixtures of Gaussians whether constrained or not. This implies that they are most suitable for modeling continuous variables. Real-world data, however, often contains binary and categorical attributes mixed with continuous ones. Discrete attributes can be handled by *non-linear* latent variable models known as latent trait models in the field of statistics (Bartholomew and Knott 1999). The non-linear models loose the analytical tractability of the Gaussian ones but a variational approximation can be used to give an efficient algorithm, as was shown recently for the binary case (Tipping 1999). It would be interesting to apply these techniques to mixed data and develop mixtures of non-linear latent variable models.

A Derivation of EM Algorithm for Mixtures of Factor Analyzers

A mixture of latent variable models takes the following form:

$$p(\mathbf{x}) = \sum_{j=1}^{m} \alpha_j p_j(\mathbf{x}), \qquad \text{with} \qquad p_j(\mathbf{x}) \sim \mathcal{N}(\boldsymbol{\mu}_j, \mathbf{R}_j + \mathbf{W}_j\mathbf{W}_j^T), \tag{34}$$

where $p_j(\mathbf{x})$ is a single latent variable model (14) and the α_j are the mixing coefficients. The error function to be minimized is the negative log-likelihood (5):

$$E(\boldsymbol{\theta}) = -\sum_n \ln \sum_{j=1}^{m} \alpha_j p_j(\mathbf{x}^n|\boldsymbol{\mu}_j, \mathbf{R}_j, \mathbf{W}_j).$$

The maximum likelihood estimates for the parameters of this mixture model can be determined with a two-stage generalized EM procedure (Tipping and Bishop 1999). The idea of this two-stage approach is to take into account the hidden variables indicating the component labels first and consider the latent variables $\{\mathbf{z}\}$ only in the second stage. It might be helpful to download (Moerland 2000b) which lists the matrix and probability identities used in this appendix.

First stage: E-step Recall that the E-step for general mixture models involves estimating the responsibilities of component j for each data point (7):

$$h_j(\mathbf{x}^n) = \frac{\alpha_j p_j(\mathbf{x}^n)}{\sum_{j=1}^{m} \alpha_j p_j(\mathbf{x}^n)}. \tag{35}$$

First stage: M-step The M-step then consists of minimizing the expected complete error function (6) with respect to the parameters of the mixture model (with $'$ denoting the new parameter values):

$$\mathcal{E}(E_c) = -\sum_n \sum_{j=1}^{m} h_j(\mathbf{x}^n) \ln \left\{ \alpha'_j p_j(\mathbf{x}^n|\boldsymbol{\mu}'_j, \mathbf{R}'_j, \mathbf{W}'_j) \right\}. \tag{36}$$

As we know, the updates of the mixing coefficients are independent of the choice of the component densities (8):

$$\alpha'_j = \frac{1}{N} \sum_n h_j(\mathbf{x}^n). \tag{37}$$

In this first stage, we only update the means $\boldsymbol{\mu}_j$, the other parameters are dealt with in the second stage. Using (34) and the definition of a multivariate Gaussian (9), the complete error function (36) can be written as:

$$\mathcal{E}(E_c) = -\sum_n \sum_{j=1}^{m} h_j(\mathbf{x}^n) \ln \left[\alpha'_j (2\pi)^{-d/2} |\mathbf{M}'_j|^{-1/2} \exp\left\{ -\frac{1}{2}(\mathbf{x}^n - \boldsymbol{\mu}'_j)^T (\mathbf{M}'_j)^{-1} (\mathbf{x}^n - \boldsymbol{\mu}'_j) \right\} \right],$$

where the covariance matrix for the latent variable model is $\mathbf{M}'_j = \mathbf{R}'_j + \mathbf{W}'_j(\mathbf{W}'_j)^T$. For the means of the component densities, the partial derivative of the complete error function is:

$$\partial \left[-\sum_n \sum_{i=1}^{m} h_i(\mathbf{x}^n) \left\{ -\frac{1}{2}(\mathbf{x}^n - \boldsymbol{\mu}'_i)^T (\mathbf{M}'_i)^{-1} (\mathbf{x}^n - \boldsymbol{\mu}'_i) \right\} \right] / \partial \boldsymbol{\mu}'_j = -\sum_n h_j(\mathbf{x}^n)(\mathbf{M}'_j)^{-1}(\mathbf{x}^n - \boldsymbol{\mu}'_j).$$

Setting this partial derivative to zero, we obtain the new estimate for the means:

$$\boldsymbol{\mu}'_j = \frac{\sum_n h_j(\mathbf{x}^n)\mathbf{x}^n}{\sum_n h_j(\mathbf{x}^n)}. \tag{38}$$

This concludes the first stage of the EM algorithm for a mixture of latent variable models. What still needs to be done is to find the updates of the parameters in $\mathbf{M}_j$, that is, the weight matrices $\mathbf{W}_j$ and the local noise models $\mathbf{R}_j$. This is done with a second stage of the EM algorithm within the M-step of the first stage. Decreasing and not minimizing the expected complete error function (36) in the M-step is sufficient for convergence to a local minimum. We will do exactly that in this second stage by introducing the latent variables $\mathbf{z}$ in the complete error function of the first stage (36) and performing one iteration of the EM algorithm to update $\mathbf{W}_j$ and $\mathbf{R}_j$. This is guaranteed to decrease $\mathcal{E}(E_c)$ and will, therefore, increase the likelihood of the entire mixture model.

Second stage: E-step We start by introducing the continuous latent variables $\mathbf{z}$ in (36):

$$\mathcal{E}(E_c) = -\sum_n \sum_{j=1}^{m} h_j(\mathbf{x}^n) \ln \left[\int_{\mathbf{z}} \alpha'_j p_j(\mathbf{x}^n, \mathbf{z}|\boldsymbol{\mu}'_j, \mathbf{R}_j, \mathbf{W}_j)\, d\mathbf{z} \right]. \tag{39}$$

EM can take care of the integral inside the logarithm through estimation of the posterior of the latent variables $p_j(\mathbf{z}|\mathbf{x}^n, \boldsymbol{\theta})$. The posterior can be found by observing that the joint distribution of the latent and observed variables for component j is also Gaussian distributed (according to the definition of the latent variable model (12) and (14)):

$$\begin{bmatrix}\mathbf{z}\\ \mathbf{x}\end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix}\mathbf{0}\\ \boldsymbol{\mu}'_j\end{bmatrix}, \begin{bmatrix}\mathbf{I}_\ell & \mathbf{W}_j^T \\ \mathbf{W}_j & \mathbf{R}_j + \mathbf{W}_j\mathbf{W}_j^T\end{bmatrix}\right).$$

The posterior can be determined directly from this joint distribution and is also Gaussian (for example, (Moerland 2000b)):

$$p_j(\mathbf{z}|\mathbf{x}^n, \boldsymbol{\mu}'_j, \mathbf{R}_j, \mathbf{W}_j) = \mathcal{N}(\mathbf{W}_j^T\mathbf{M}_j^{-1}(\mathbf{x}^n - \boldsymbol{\mu}'_j), \mathbf{I} - \mathbf{W}_j^T\mathbf{M}_j^{-1}\mathbf{W}_j). \tag{40}$$

The Gaussian posterior is fully characterized by its first and second moments:

$$\langle\, \mathbf{z}_j^n \,\rangle = \mathbf{W}_j^T\mathbf{M}_j^{-1}(\mathbf{x}^n - \boldsymbol{\mu}'_j) \tag{41}$$

$$\langle\, \mathbf{z}_j^n(\mathbf{z}_j^n)^T \,\rangle = \mathbf{I} - \mathbf{W}_j^T\mathbf{M}_j^{-1}\mathbf{W}_j + \langle\, \mathbf{z}_j^n \,\rangle\langle\, \mathbf{z}_j^n \,\rangle^T. \tag{42}$$

These two equations seem to require the inversion of the $d\times d$ matrix $\mathbf{M}_j$, but they can be simplified using the matrix inversion lemma (Golub and Van Loan 1996, page 50):

$$\mathbf{M}^{-1} = (\mathbf{W}\mathbf{W}^T + \mathbf{R})^{-1} = \mathbf{R}^{-1} - \mathbf{R}^{-1}\mathbf{W}(\mathbf{I}_\ell + \mathbf{W}^T\mathbf{R}^{-1}\mathbf{W})^{-1}\mathbf{W}^T\mathbf{R}^{-1} = \mathbf{R}^{-1} - \mathbf{R}^{-1}\mathbf{W}\mathbf{N}^{-1}\mathbf{W}^T\mathbf{R}^{-1}, \tag{43}$$

with $\ell\times\ell$ matrix:

$$\mathbf{N} = \mathbf{I}_\ell + \mathbf{W}^T\mathbf{R}^{-1}\mathbf{W}.$$

We see that inverting $\mathbf{M}$ actually only requires the inversion of a $\ell\times\ell$ matrix $\mathbf{N}$ and taking the inverse of the noise covariance matrix $\mathbf{R}$ which is easy because of the diagonality constraint. Further simplification is possible with the following straightforward rewriting:

$$\begin{tikzcd}[row sep=tiny] & \mathbf{N}^{-1} + \mathbf{W}^T\mathbf{R}^{-1}\mathbf{W}\mathbf{N}^{-1} \\ \mathbf{I}_\ell = \mathbf{N}\mathbf{N}^{-1} = \arrow[ur] \arrow[dr] & \\ & \mathbf{N}^{-1} + \mathbf{N}^{-1}\mathbf{W}^T\mathbf{R}^{-1}\mathbf{W} \end{tikzcd} \tag{44}$$

Thus, we can transform the first two factors in (41) using (43) and (44):

$$\mathbf{W}^T\mathbf{M}^{-1} = \mathbf{W}^T\mathbf{R}^{-1} - \mathbf{W}^T\mathbf{R}^{-1}\mathbf{W}\mathbf{N}^{-1}\mathbf{W}^T\mathbf{R}^{-1} = (\mathbf{I}_\ell - \mathbf{W}^T\mathbf{R}^{-1}\mathbf{W}\mathbf{N}^{-1})\mathbf{W}^T\mathbf{R}^{-1} = \mathbf{N}^{-1}\mathbf{W}^T\mathbf{R}^{-1}. \tag{45}$$

The first two terms of (42) can be written as (using (44)):

$$\mathbf{I}_\ell - \mathbf{W}^T\mathbf{M}^{-1}\mathbf{W} = \mathbf{I}_\ell - \mathbf{N}^{-1}\mathbf{W}^T\mathbf{R}^{-1}\mathbf{W} = \mathbf{N}^{-1}. \tag{46}$$

Substituting (45) and (46) in the moments (41) and (42) gives:

$$\begin{aligned} \langle\, \mathbf{z}_j^n \,\rangle &= \mathbf{N}_j^{-1}\mathbf{W}_j^T\mathbf{R}_j^{-1}(\mathbf{x}^n - \boldsymbol{\mu}_j') && (47)\\ \langle\, \mathbf{z}_j^n(\mathbf{z}_j^n)^T \,\rangle &= \mathbf{N}_j^{-1} + \langle\, \mathbf{z}_j^n \,\rangle\langle\, \mathbf{z}_j^n \,\rangle^T. && (48) \end{aligned}$$

Having fully characterized the posterior of the latent variables, this terminates the E-step of the second stage.

Second stage: M-step For the second-stage M-step, the expected (with respect to the posteriors of the latent variables) complete error function corresponding to (39) is:

$$\mathcal{E}(\hat{E}_c) = -\sum_n \sum_{j=1}^m h_j(\mathbf{x}^n) \int_{\mathbf{z}} p_j(\mathbf{z}|\mathbf{x}^n, \boldsymbol{\theta}) \left\{ \ln \alpha_j' p_j(\mathbf{x}^n, \mathbf{z}|\boldsymbol{\mu}_j', \mathbf{R}_j', \mathbf{W}_j') \right\} d\mathbf{z}.$$

This can be written in a more compact way:

$$\mathcal{E}(\hat{E}_c) = -\sum_n \sum_{j=1}^m h_j(\mathbf{x}^n)\langle\, \ln \alpha_j' p_j(\mathbf{x}^n, \mathbf{z}|\boldsymbol{\mu}_j', \mathbf{R}_j', \mathbf{W}_j') \,\rangle, \tag{49}$$

where $\langle\, . \,\rangle$ denotes the expectation with respect to the posterior distribution of $\mathbf{z}$. The joint probability distribution is (using (13) and the definition of a multivariate Gaussian (9)):

$$p_j(\mathbf{x}^n, \mathbf{z}) = p_j(\mathbf{x}^n|\mathbf{z})p_j(\mathbf{z}) = (2\pi)^{-d/2}|\mathbf{R}_j'|^{-1/2} \exp\left\{ -\frac{1}{2}(a_j^n)^T(\mathbf{R}_j')^{-1}a_j^n \right\} (2\pi)^{-\ell/2} \exp\left\{ -\frac{1}{2}\mathbf{z}^T\mathbf{z} \right\},$$

where

$$a_j^n = \mathbf{x}^n - \mathbf{W}_j'\mathbf{z} - \boldsymbol{\mu}_j'.$$

Rewriting (49) using the above expression for the joint distribution and collecting irrelevant constants which do not depend on $\mathbf{W}_j'$ and $\mathbf{R}_j'$, gives:

$$\begin{aligned} \mathcal{E}(\hat{E}_c) \;=\; & -\sum_n \sum_{j=1}^m h_j(\mathbf{x}^n)\Big\{ -\frac{1}{2}\ln|\mathbf{R}_j'| - \frac{1}{2}\big\langle\, (\mathbf{x}^n - \mathbf{W}_j'\mathbf{z}_j^n - \boldsymbol{\mu}_j')^T(\mathbf{R}_j')^{-1}(\mathbf{x}^n - \mathbf{W}_j'\mathbf{z}_j^n - \boldsymbol{\mu}_j') \,\big\rangle \Big\} \\ & + \text{constant}. \end{aligned} \tag{50}$$

Factoring out the last term on the first line (with the trace operator "tr" to get the appropriate moments):

$$\begin{aligned} & -\frac{1}{2}\Big\{ (\mathbf{x}^n - \boldsymbol{\mu}_j')^T(\mathbf{R}_j')^{-1}(\mathbf{x}^n - \boldsymbol{\mu}_j') - 2(\mathbf{x}^n - \boldsymbol{\mu}_j')^T(\mathbf{R}_j')^{-1}\mathbf{W}_j'\langle\, \mathbf{z}_j^n \,\rangle \\ & + \big\langle\, (\mathbf{z}_j^n)^T(\mathbf{W}_j')^T(\mathbf{R}_j')^{-1}\mathbf{W}_j'\mathbf{z}_j^n \,\big\rangle \Big\} \\ = \; & -\frac{1}{2}\Big[(\mathbf{x}^n - \boldsymbol{\mu}_j')^T(\mathbf{R}_j')^{-1}(\mathbf{x}^n - \boldsymbol{\mu}_j') - 2(\mathbf{x}^n - \boldsymbol{\mu}_j')^T(\mathbf{R}_j')^{-1}\mathbf{W}_j'\langle\, \mathbf{z}_j^n \,\rangle \\ & + \text{tr}\left\{ (\mathbf{W}_j')^T(\mathbf{R}_j')^{-1}\mathbf{W}_j'\langle\, \mathbf{z}_j^n(\mathbf{z}_j^n)^T \,\rangle \right\} \Big]. \end{aligned}$$

The second stage M-step then requires minimizing the complete error function (substituting the above expression in (50)):

$$\begin{aligned}\mathcal{E}(\hat{E}_c) &= -\sum_n \sum_{j=1}^m h_j(\mathbf{x}^n)\Big[-\frac{1}{2}\ln|\mathbf{R}'_j| - \frac{1}{2}(\mathbf{x}^n-\boldsymbol{\mu}'_j)^T(\mathbf{R}'_j)^{-1}(\mathbf{x}^n-\boldsymbol{\mu}'_j) \\ &\quad +(\mathbf{x}^n-\boldsymbol{\mu}'_j)^T(\mathbf{R}'_j)^{-1}\mathbf{W}'_j\langle\,\mathbf{z}^n_j\,\rangle - \frac{1}{2}\mathrm{tr}\left\{(\mathbf{W}'_j)^T(\mathbf{R}'_j)^{-1}\mathbf{W}'_j\langle\,\mathbf{z}^n_j(\mathbf{z}^n_j)^T\,\rangle\right\}\Big] \qquad (51)\end{aligned}$$

with respect to the parameters which have not yet been optimized in the first stage of the EM algorithm: the generative matrices $\mathbf{W}_j$ and the noise covariance $\mathbf{R}_j$. For $\mathbf{W}_j$, the partial derivative is (this requires some derivatives of traces and scalar forms listed in (Moerland 2000b)):

$$\frac{\partial\mathcal{E}(\hat{E}_c)}{\partial\mathbf{W}'_j} = -\sum_n h_j(\mathbf{x}^n)\left\{(\mathbf{R}'_j)^{-1}(\mathbf{x}^n-\boldsymbol{\mu}'_j)\langle\,\mathbf{z}^n_j\,\rangle^T - (\mathbf{R}'_j)^{-1}\mathbf{W}'_j\langle\,\mathbf{z}^n_j(\mathbf{z}^n_j)^T\,\rangle\right\} = 0, \qquad (52)$$

which has the following solution:

$$\mathbf{W}'_j = \left[\sum_n h_j(\mathbf{x}^n)(\mathbf{x}^n-\boldsymbol{\mu}'_j)\langle\,\mathbf{z}^n_j\,\rangle^T\right]\left[\sum_n h_j(\mathbf{x}^n)\langle\,\mathbf{z}^n_j(\mathbf{z}^n_j)^T\,\rangle\right]^{-1}. \qquad (53)$$

Of course, in a practical implementation one should not take the inverse directly but use, for example, a Cholesky decomposition of $\sum_n h_j(\mathbf{x}^n)\langle\,\mathbf{z}^n_j(\mathbf{z}^n_j)^T\,\rangle$ to solve the corresponding weighted least-squares problem.

For noise covariance $\mathbf{R}_j$, the partial derivative of the complete error function (51) with respect to its inverse is:

$$\begin{aligned}\frac{\partial\mathcal{E}(\hat{E}_c)}{\partial(\mathbf{R}'_j)^{-1}} &= -\sum_n h_j(\mathbf{x}^n)\Big\{\frac{1}{2}\mathbf{R}'_j - \frac{1}{2}(\mathbf{x}^n-\boldsymbol{\mu}'_j)(\mathbf{x}^n-\boldsymbol{\mu}'_j)^T + \mathbf{W}'_j\langle\,\mathbf{z}^n_j\,\rangle(\mathbf{x}-\boldsymbol{\mu}'_j)^T \\ &\quad -\frac{1}{2}\mathbf{W}'_j\langle\,\mathbf{z}^n_j(\mathbf{z}^n_j)^T\,\rangle(\mathbf{W}'_j)^T\Big\} = 0.\end{aligned}$$

Using the new estimate (53) for $\mathbf{W}'_j$ and the diagonal constraint on the noise model $\mathbf{R}_j$ (Rubin and Thayer 1982) we obtain:

$$\mathbf{R}'_j = \frac{1}{\sum_n h_j(\mathbf{x}^n)}\mathrm{diag}\left[\sum_n h_j(\mathbf{x}^n)\left\{(\mathbf{x}^n-\boldsymbol{\mu}'_j)-\mathbf{W}'_j\langle\,\mathbf{z}^n_j\,\rangle\right\}(\mathbf{x}^n-\boldsymbol{\mu}'_j)^T\right]. \qquad (54)$$

And this ends the derivation of the EM algorithm for MFAs (Algorithm 1).

Notation

Upper-case bold letters denote matrices, for example $\mathbf{R}$. Vectors are denoted by lower-case bold letters, for example $\mathbf{x}$, and are column vectors. Subscripting is used for indexing; thus, x_i denotes the ith element of vector $\mathbf{x}$ and R_{ij} the element in the ith row and jth column of matrix $\mathbf{R}$.

$\cdot^T$	transpose (of a vector or a matrix), thus $\mathbf{x}^T = (x_1, x_2, \ldots, x_d)$
$\cdot \sim \cdot$	"distributed according to"
$\cdot \propto \cdot$	"proportional to"
$\langle\cdot\rangle$	expectation of a random variable
$\lvert\cdot\rvert$	determinant
$\lvert\lvert\cdot\rvert\rvert$	length of a vector
$\cdot \otimes \cdot$	Kronecker product of two matrices
$\cdot \circ \cdot$	Element-wise product of two matrices
$\mathbf{0}$	all-zero matrix
α_j	mixing coefficients
$\boldsymbol{\gamma}$	vector of all hyperparameters
δ	the number of well-determined parameters
$\boldsymbol{\mu}$	mean of a multivariate Gaussian distribution
$\boldsymbol{\theta}$	parameters of a model
$\boldsymbol{\Sigma}$	covariance matrix of a multivariate Gaussian distribution
σ^2	variance (of a Gaussian distribution)
C	number of classes
$\mathcal{C}_k$	class k
D	training set
d	dimension of data space
$\mathrm{diag}(\cdot)$	function from vectors to matrices which puts the vector on the main diagonal
$\mathrm{diag}(\cdot)$	also a function from matrices to matrices which puts the off-diagonal elements to zero
$E(\cdot)$	error function
$\mathcal{E}(\cdot)$	expectation of a random variable
$h_j(\cdot)$	posterior probabilities in a mixture model
$\mathbf{I}$	identity matrix
$\mathbf{I}_k$	identity matrix of size $k \times k$
$\mathcal{L}(\cdot)$	likelihood function
ℓ	dimension of latent space
$\ln(\cdot)$	logarithm to base e
m	number of mixture components
N	number of training examples
$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	multivariate Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$
$P(\cdot)$	probability
$p(\cdot)$	probability density function
$\mathbf{R}$	factor analysis noise covariance matrix
$\mathbb{R}$	the set of real numbers
$\mathbf{S}$	sample covariance matrix
$\mathrm{tr}(\cdot)$	trace operator
$\mathbf{W}$	factor loading matrix
$\mathrm{vec}(\cdot)$	stacks the columns of a matrix into one vector
$\mathbf{x}^n$	input pattern n
$\mathbf{z}$	missing or latent variables

References

Alpaydin, E. (1999, November). Combined 5x2cv F test for comparing supervised classification learning algorithms. *Neural Computation 11*(8), 1885–1892.

Bartholomew, D. J. and M. Knott (1999). *Latent Variable Models and Factor Analysis* (2nd ed.). London: Arnold.

Bellegarda, J. and D. Nahamoo (1990). Tied mixture continuous parameter modeling for speech recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing 38*, 2033–2045.

Berger, J. O. (1985). *Statistical Decision Theory and Bayesian Analysis* (2nd ed.). New York: Springer-Verlag.

Bishop, C. M. (1995). *Neural Networks for Pattern Recognition.* Oxford: Oxford University Press.

Bishop, C. M. (1999a). Bayesian PCA. In M. S. Kearns, S. A. Solla, and D. A. Cohn (Eds.), *Advances in Neural Information Processing Systems*, Volume 11, pp. 382–388. Cambridge, MA: MIT Press.

Bishop, C. M. (1999b). Variational principal components. In *Proceedings of the Ninth International Conference on Artificial Neural Networks (ICANN'99)*, Volume 1, pp. 509–514. London: IEE.

Blake, C., E. Keogh, and C. J. Merz (1998). UCI repository of machine learning databases. Irvine: University of California, Department of Information and Computer Sciences. `www.ics.uci.edu/~mlearn/MLRepository.html`.

Carreira-Perpiñán, M. Á. and S. J. Renals (1998, December). Dimensionality reduction of electropalatographic data using latent variable models. *Speech Communication 26*(4), 259–282.

Cover, T. M. and J. A. Thomas (1991). *Elements of Information Theory.* New York: John Wiley & Sons.

Dempster, A. P., N. M. Laird, and D. B. Rubin (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, B 39*(1), 1–38.

Dietterich, T. G. (1998). Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation 10*(7), 1895–1923.

Duda, R. O. and P. E. Hart (1973). *Pattern Classification and Scene Analysis.* New York: John Wiley & Sons.

Everitt, B. S. (1984). *An Introduction to Latent Variable Models.* London: Chapman and Hill.

Garris, M. D. and R. D. Wilkinson (1992, February). NIST special database 3: handwritten segmented characters. Technical report, National Insitute of Standards and Technology.

Ghahramani, Z. and M. J. Beal (2000). Variational inference for Bayesian mixture of factor analysers. In S. A. Solla, T. K. Leen, and K.-R. Müller (Eds.), *Advances in Neural Information Processing Systems*, Volume 12. Cambridge, MA: MIT Press. `http://www.gatsby.ucl.ac.uk/~zoubin/papers/nips99.ps.gz`.

Ghahramani, Z. and G. E. Hinton (1996). The EM algorithm for mixtures of factor analyzers. Technical Report CRG-TR-96-1, University of Toronto. `ftp://ftp.cs.toronto.edu/pub/zoubin/tr-96-1.ps.gz`.

Golub, G. H. and C. F. Van Loan (1996). *Matrix Computations* (3rd ed.). Baltimore: The Johns Hopkins University Press.

Hastie, T. and R. Tibshirani (1996). Discriminant analysis by Gaussian mixtures. *Journal of the Royal Statistical Society (Series B) 58*, 155–176.

Hastie, T., R. Tibshirani, and A. Buja (1999). Flexible discriminant and mixture models. In J. Kay and D. Titterington (Eds.), *Statistics and Neural Networks.* Oxford: Oxford University Press.

Hinton, G. E., P. Dayan, and M. Revow (1997). Modelling the manifolds of images of handwritten digits. *IEEE Transactions on Neural Networks 8*(1), 65–74.

Jojic, N. and B. J. Frey (2000). Topographic transformation as a discrete latent variable. In S. A. Solla, T. K. Leen, and K.-R. Müller (Eds.), *Advances in Neural Information Processing Systems*, Volume 12. Cambridge, MA: MIT Press. `http://www.cs.toronto.edu/~frey/papers/ttdlv-nips99.ps.Z`.

Jollife, I. T. (1986). *Principal Component Analysis*. New York: Springer-Verlag.

Jordan, M. I. (Ed.) (1999). *Learning in Graphical Models*. Adaptive Computation and Machine Learning. Cambridge, MA: MIT Press.

Jordan, M. I., Z. Ghahramani, T. S. Jaakkola, and L. Saul (1999). An introduction to variational methods for graphical models. See Jordan (1999), pp. 105–161.

Kambhatla, N. and T. K. Leen (1995). Classifying with Gaussian mixtures and clusters. In G. Tesauro, D. Touretzky, and T. Leen (Eds.), *Advances in Neural Information Processing Systems*, Volume 7, pp. 681–688. Cambridge, MA: MIT Press.

LeCun, Y. (2000). The MNIST database of handwritten digits. `http://www.research.att.com/~yann/ocr/mnist/index.html`.

LeCun, Y., L. D. Jackel, L. Bottou, C. Cortes, J. S. Denker, H. Drucker, I. Guyon, U. A. Muller, E. Sackinger, P. Simard, and V. Vapnik (1995). Learning algorithms for classification: A comparison on handwritten digit recognition. In J. H. Oh, C. Kwon, and S. Cho (Eds.), *Neural Networks: The Statistical Mechanics Perspective*, pp. 261–276. Singapore: World Scientific.

MacKay, D. J. C. (1991). *Bayesian Methods for Adaptive Models*. Ph. D. thesis, California Institute of Technology. `wol.ra.phy.cam.ac.uk/mackay/README.html`.

MacKay, D. J. C. (1994a). Bayesian methods for backpropagation networks. In E. Domany, J. L. van Hemmen, and K. Schulten (Eds.), *Models of Neural Networks III*, Chapter 6. New York: Springer-Verlag.

MacKay, D. J. C. (1994b). Bayesian non-linear modeling for the energy prediction competition. *ASHRAE Transactions 100*(2), 1053–1062.

McLachlan, G. J. and K. E. Basford (1988). *Mixture Models: Inference and Applications to Clustering*. New York: Marcel Dekker, Inc.

Meinicke, P. and H. Ritter (1999). Local PCA learning with resolution-dependent mixtures of Gaussians. In *Proceedings of the Ninth International Conference on Artificial Neural Networks (ICANN'99)*, pp. 497–502. London: IEE. Extended version: `http://www.techfak.uni-bielefeld.de/gk/papers/meinick99.html`.

Michie, D., D. J. Spiegelhalter, and C. C. Taylor (Eds.) (1994). *Machine Learning, Neural and Statistical Classification*. New York: Ellis Horwood.

Moerland, P. (1999). A comparison of mixture models for density estimation. In *Proceedings of the Ninth International Conference on Artificial Neural Networks (ICANN'99)*, Volume 1, pp. 25–30. London: IEE.

Moerland, P. (2000a, June). *Mixture Models for Unsupervised and Supervised Learning*. Ph. D. thesis, École Polytechnique Fédérale de Lausanne, Computer Science Department, Lausanne, Switzerland. `ftp://ftp.idiap.ch/pub/reports/2000/rr00-18.ps.gz`.

Moerland, P. (2000b). Some matrix and probability identities. `ftp://ftp.idiap.ch/pub/papers/neural/moerland.cheat.ps.gz`.

Neal, R. M. (1996). *Bayesian Learning for Neural Networks*. Number 118 in Lecture Notes in Statistics. New York: Springer-Verlag.

Neal, R. M. and G. E. Hinton (1999). A view of the EM algorithm that justifies incremental, sparse and other variants. See Jordan (1999), pp. 355–368.

Ormoneit, D. and V. Tresp (1998). Averaging, maximum penalized likelihood and Bayesian estimation for improving Gaussian mixture probability density estimates. *IEEE Transactions on Neural Networks 9*(4), 639–650.

Rätsch, G., T. Onoda, and K. R. Müller (1998). Soft margins for AdaBoost. *Machine Learning*. To appear: `www.first.gmd.de/~raetsch/ps/Neurocolt_Margin.ps.gz`.

Roweis, S. (1998). EM algorithms for PCA and SPCA. In M. I. Jordan, M. J. Kearns, and S. A. Solla (Eds.), *Advances in Neural Information Processing Systems*, Volume 10, pp. 626–632. Cambridge, MA: MIT Press.

Roweis, S. and Z. Ghahramani (1999). A unifying review of linear Gaussian models. *Neural Computation 11*(2), 305–345.

Rubin, D. B. and D. T. Thayer (1982). EM algorithms for ML factor analysis. *Psychometrika 47*(1), 69–76.

Saul, L. and M. Rahim (1998). Modeling acoustic correlations by factor analysis. In M. I. Jordan, M. J. Kearns, and S. A. Solla (Eds.), *Advances in Neural Information Processing Systems*, Volume 10, pp. 749–755. Cambridge, MA: MIT Press.

Schwenk, H. and Y. Bengio (1998). Training methods for adaptive boosting of neural networks. In M. I. Jordan, M. J. Kearns, and S. A. Solla (Eds.), *Advances in Neural Information Processing Systems*, Volume 10, pp. 647–653. Cambridge, MA: MIT Press.

Tipping, M. E. (1999). Probabilistic visualisation of high-dimensional binary data. In M. S. Kearns, S. A. Solla, and D. A. Cohn (Eds.), *Advances in Neural Information Processing Systems*, Volume 11, pp. 592–598. Cambridge, MA: MIT Press.

Tipping, M. E. and C. M. Bishop (1999, February). Mixtures of probabilistic principal component analysers. *Neural Computation 11*(2), 443–482.

Titterington, D., A. F. M. Smith, and U. E. Makov (1985). *Statistical Analysis of Finite Mixture Distributions*. Chichester: John Wiley & Sons.