# 3

# MIXTURE DENSITY ESTIMATION

In this chapter we consider mixture densities, the main building block for the dimension reduction techniques described in the following chapters. In the first section we introduce mixture densities and the expectation-maximization (EM) algorithm to estimate their parameters from data. The EM algorithm finds, from an initial parameter estimate, a sequence of parameter estimates that yield increasingly higher data log-likelihood. The algorithm is guaranteed to converge to a local maximum of the data log-likelihood as function of the parameters. However, this local maximum may yield significantly lower log-likelihood than the globally optimal parameter estimate.

The first contribution we present is a technique that is empirically found to avoid many of the poor local maxima found when using random initial parameter estimates. Our technique finds an initial parameter estimate by starting with a one-component mixture and adding components to the mixture one-by-one. In Section 3.2 we apply this technique to mixtures of Gaussian densities and in Section 3.3 to k-means clustering.

Each iteration of the EM algorithm requires a number of computations that scales linearly with the product of the number of data points and the number of mixture components, this limits its applicability in large scale applications with many data points and mixture components. In Section 3.4, we present a technique to speed-up the estimation of mixture models from large quantities of data where the amount of computation can be traded against accuracy of the algorithm. However, for any preferred accuracy the algorithm is in each step guaranteed to increase a lower bound on the data log-likelihood.

## 3.1 The EM algorithm and Gaussian mixture densities

In this section we describe the expectation-maximization (EM) algorithm for estimating the parameters of mixture densities. Parameter estimation algorithms are sometimes also referred to as 'learning algorithms' since the machinery that implements the algorithm, in a sense, 'learns' about the data by estimating the parameters. Mixture models,

a weighted sum of finitely many elements of some parametric class of component densities, form an expressive class of models for density estimation. Due to the development of automated procedures to estimate mixture models from data, applications in a wide range of fields have emerged in recent decades. Examples are density estimation, clustering, and estimating class-conditional densities in supervised learning settings. Using the EM algorithm it is relatively straightforward to apply density estimation techniques in cases where some data is missing. The missing data could be the class labels of some objects in partially supervised classification problems or the value of some features that describe the objects for which we try to find a density estimate.

### 3.1.1 Mixture densities

A mixture density (McLachlan and Peel, 2000) is defined as a weighted sum of, say $k$, component densities. The component densities are restricted to a particular parametric class of densities that is assumed to be appropriate for the data at hand or attractive for computational reasons. Let us denote by $p(\mathbf{x}; \boldsymbol{\theta}_s)$ the $s$-th component density, where $\boldsymbol{\theta}_s$ are the component parameters. We use $\pi_s$ to denote the weighing factor of the $s$-th component in the mixture. The weights must satisfy two constraints: (i) non-negativity: $\pi_s \geq 0$ and (ii) partition of unity: $\sum_{s=1}^{k} \pi_s = 1$. The weights $\pi_s$ are also known as 'mixing proportions' or 'mixing weights' and can be thought of as the probability $p(s)$ that a data sample will be drawn from mixture component $s$. A $k$ component mixture density is then defined as:

$$p(\mathbf{x}) \equiv \sum_{s=1}^{k} \pi_s p(\mathbf{x}; \boldsymbol{\theta}_s). \tag{3.1}$$

For a mixture we collectively denote all parameters with $\boldsymbol{\theta} = \{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_k, \pi_1, \ldots, \pi_k\}$. Throughout this thesis we assume that all data are identically and independently distributed (i.i.d.), and hence that the likelihood of a set of data vectors is just the product of the individual likelihoods.

One can think of a mixture density as modelling a process where first a 'source' $s$ is selected according to the multinomial distribution $\{\pi_1, \ldots, \pi_k\}$ and then a sample is drawn from the corresponding component density $p(\mathbf{x}; \boldsymbol{\theta}_s)$. Thus, the probability of selecting source $s$ and datum $\mathbf{x}$ is $\pi_s p(\mathbf{x}; \boldsymbol{\theta}_s)$. The marginal probability of selecting datum $\mathbf{x}$ is then given by (3.1). We can think of the source that generated a data vector $\mathbf{x}$ as 'missing information': we only observe $\mathbf{x}$ and do not know the generating source. The expectation-maximization algorithm, presented in the next section, can be understood in terms of iteratively estimating this missing information.

An important derived quantity is the 'posterior probability' on a mixture component given a data vector. One can think of this distribution as a distribution on which mixture component *generated* a particular data vector, i.e. "Which component density was this

data vector drawn from?" or "to which cluster does this data vector belong?". The posterior distribution on the mixture components is defined using Bayes rule:

$$p(s|\mathbf{x}) \equiv \frac{\pi_s p(\mathbf{x};\boldsymbol{\theta}_s)}{p(\mathbf{x})} = \frac{\pi_s p(\mathbf{x};\boldsymbol{\theta}_s)}{\sum_{s'} \pi_{s'} p(\mathbf{x};\boldsymbol{\theta}_{s'})}. \tag{3.2}$$

The expectation-maximization algorithm to estimate the parameters of a mixture model from data makes essential use of these posterior probabilities.

Mixture modelling is also known as semi-parametric density estimation and it can be placed in between two extremes: parametric and non-parametric density estimation. Parametric density estimation assumes the data is drawn from a density in a parametric class, say the class of Gaussian densities. The estimation problem then reduces to finding the parameters of the Gaussian that fits the data best. The assumption underlying parametric density estimation is often unrealistic but allows for very efficient parameter estimation. At the other extreme, non-parametric methods do not assume a particular form of the density from which the data is drawn. Non-parametric estimates typically take a form of a mixture density with a mixture component for every data point in the data set. The components, often referred to as 'kernels'. A well known non-parametric density estimator is the Parzen estimator (Parzen, 1962) which uses Gaussian components with mean equal to the corresponding data point and small isotropic covariance. Non-parametric estimates can implement a large class of densities. The price we have to pay is that for the evaluation of the estimator at a new point we have to evaluate all the kernels, which is computationally demanding if the estimate is based on a large data set. Mixture modelling strikes a balance between these extremes: a large class of densities can be implemented and we can evaluate the density efficiently, since only relatively few density functions have to be evaluated.

### 3.1.2 Parameter estimation with the EM algorithm

The first step when using a mixture model is to determine its architecture: a proper class of component densities and the number of component densities in the mixture. We will discuss these issues in Section 3.1.3. After these design choices have been made, we estimate the free parameters in the mixture model such that the model 'fits' our data as good as possible. The expectation-maximization algorithm is the most popular method to estimate the parameters of mixture models to a given data set.

We define "fits the data as good as possible" as "assigns maximum likelihood to the data". Hence, fitting the model to given data becomes searching for the maximum-likelihood parameters for our data in the set of probabilistic models defined by the chosen architecture. Since the logarithm is a monotone increasing function, the maximum likelihood criterion is equivalent to a maximum log-likelihood criterion and these criteria are often interchanged. Due to the i.i.d. assumption, the log-likelihood of a data set

$\mathbf{X}_N = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ can be written as:

$$\mathcal{L}(\mathbf{X}_N, \boldsymbol{\theta}) = \log p(\mathbf{X}_N; \boldsymbol{\theta}) = \log \prod_{n=1}^{N} p(\mathbf{x}_n; \boldsymbol{\theta}) = \sum_{n=1}^{N} \log p(\mathbf{x}_n; \boldsymbol{\theta}). \tag{3.3}$$

When no confusion arises, the dependence of the log-likelihood on $\mathbf{X}_N$ is not made explicit and we simply write $\mathcal{L}(\boldsymbol{\theta})$.

Finding the maximum likelihood parameters for a single component density is easy for a wide range of component densities and can often be done in closed-form. This is for example the case for Gaussian mixture components. However, if the probabilistic model is a mixture, the estimation often becomes considerably harder because the (log-)likelihood as a function of the parameters may have many local optima. Hence, some non-trivial optimization is needed to obtain good parameter estimates.

The expectation-maximization (EM) algorithm (Dempster et al., 1977) finds parameters at the local optima of the log-likelihood function given some initial parameter values. In our exposition we follow the generalized view on EM of (Neal and Hinton, 1998). The greatest advantages of the EM algorithm over other methods are (i) no parameters have to be set that influence the optimization algorithm, like e.g. the step-size for gradient based algorithms, and (ii) its ease of implementation. The biggest drawback is, as for all local-optimization methods, the sensitivity of the found solution to the initial parameter values. This sensitivity can be partially resolved by either (i) performing several runs from different initial parameter values and keeping the best, or (ii) finding the mixture parameters in a 'greedy' manner by starting with a single component mixture and adding new components one at a time (this is the topic of Section 3.2). Other parameter estimation techniques are gradient and sampling based methods, however these are not treated within this thesis.

Intuitively, the idea of the EM algorithm is to make estimates on the 'missing information' mentioned in the previous section: to which mixture component do we ascribe each data vector? The EM algorithm proceeds by (E-step) estimating to which component each data point belongs and (M-step) re-estimating the parameters on the basis of this estimation. This sounds like a chicken-and-egg problem: given the assignment of data to components we can re-estimate the components and given the components we can re-assign the data to the components. Indeed, this is a chicken-and-egg problem and we simply either start with initial parameters or with an initial assignment. However, after each iteration of EM, we are guaranteed that the re-estimated parameters give at least as high a log-likelihood as the previous parameter values.

Technically, the idea of EM is to iteratively define a lower bound on the log-likelihood and to maximize this lower bound. The lower bound is obtained by subtracting from the log-likelihood a non-negative quantity known as Kullback-Leibler (KL) divergence. KL divergence is an asymmetric dissimilarity measure between two probability distributions $p$ and $q$ from the field of information theory (Cover and Thomas, 1991), it is

defined for discrete distributions $p$ and $q$ with domain $\{1, \ldots, k\}$ as:

$$\mathcal{D}(q\|p) \equiv \sum_{s=1}^{k} q(s) \log \frac{q(s)}{p(s)} = -\mathcal{H}(q) - \sum_{s=1}^{k} q(s) \log p(s) > 0, \tag{3.4}$$

where $\mathcal{H}(\cdot)$ denotes the entropy of a distribution, an information theoretic measure of the 'uncertainty' or 'information' in a distribution. The KL divergence is defined analogously for probability density functions by replacing the sum over the domain with an integral over the domain. The KL divergence is zero if and only if the two distributions $p$ and $q$ are identical. We will use the KL divergence to measure, for each data point, how well the true posterior distribution $p(s|\mathbf{x})$ matches an approximating distribution $q$ for this data point.

Since the KL-divergence is non-negative we can bound each of the terms $\log p(\mathbf{x}_n)$ in the log-likelihood (3.3) from below by subtracting the KL-divergence $\mathcal{D}(q_n\|p(s|\mathbf{x}_n))$. Note that this bound holds for *any* distribution $q_n$ and becomes tight if and only if $q_n = p(s|\mathbf{x}_n)$. We will use $q$ to denote the set of distributions $\{q_1, \ldots, q_N\}$. Combining the bounds on the individual log-likelihoods $\log p(\mathbf{x}_n)$, the complete log-likelihood (3.3) can be bounded by:

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{n=1}^{N} \log p(\mathbf{x}_n) \geq \mathcal{F}(\boldsymbol{\theta}, q) = \sum_{n=1}^{N} \left[\log p(\mathbf{x}_n; \boldsymbol{\theta}) - \mathcal{D}(q_n\|p(s|\mathbf{x}_n))\right] \tag{3.5}$$

$$= \sum_{n=1}^{N} \left[\mathcal{H}(q_n) + \mathrm{E}_{q_n} \log \pi_s p(\mathbf{x}_n; \boldsymbol{\theta}_s)\right]. \tag{3.6}$$

Now that we have the two forms (3.5) and (3.6), iterative maximization of the lower bound $\mathcal{F}$ on the log-likelihood becomes easy. In (3.5) the only term dependent on $q_n$ is the KL divergence $\mathcal{D}(q_n\|p(s|\mathbf{x}_n))$. To maximize $\mathcal{F}$, recall that the KL divergence is zero if and only if its two arguments are identical. Therefore, to maximize $\mathcal{F}$ w.r.t. the $q_n$ we should set them as the posteriors: $q_n \leftarrow p(\cdot|\mathbf{x}_n)$. In this case the lower bound equals the data log-likelihood: $\mathcal{F} = \mathcal{L}$.

To maximize $\mathcal{F}$ w.r.t. $\boldsymbol{\theta}$, only the second term of (3.6) is of interest. The maximization is often easy since now we have to maximize a sum of single component log-likelihoods instead of a sum of mixture model log-likelihoods. Let us use the compact notation $q_{ns}$ for $q_n(s)$, then if we consider optimizing the parameters of a single component $s$ the only relevant terms in (3.6) are:

$$\sum_{n=1}^{N} q_{ns}[\log \pi_s + \log p(\mathbf{x}_n; \boldsymbol{\theta}_s)]. \tag{3.7}$$

For many component densities from the exponential family the maximization of (3.7) w.r.t. the parameters $\boldsymbol{\theta}_s$ can be done in closed form.

Summarizing, the EM algorithm consists of iteratively maximizing $\mathcal{F}$, in the E-step w.r.t. $q$ and in the M-step w.r.t. $\boldsymbol{\theta}$. Intuitively, we can see the E-step as fixing a probabilistic assignment of every data vector to mixture components. The M-step then optimizes the mixture parameters given this assignment. Since after each E-step, we have $\mathcal{F} = \mathcal{L}$ and both the E-step and the M-step do not decrease $\mathcal{F}$ we immediately have that iterative application of these steps can not decrease $\mathcal{L}$.

The $q_{ns}$ are often referred to as 'responsibilities', since they indicate for each data point which mixture component models it. With every data vector $\mathbf{x}_n$ we associate a random variable $s_n$ which can take values in $\{1, \ldots, k\}$, indicating which component generated $\mathbf{x}_n$. These variables are often called 'hidden' or 'unobserved' variables since we have no direct access to them. EM treats the mixture density log-likelihood maximization problem as a problem of dealing with missing data. Similar approaches can be taken to fit (mixture) density models to data where there truly is missing data in the sense that certain values are missing in the input vector $\mathbf{x}$.

**Generalized and variational EM.** The EM algorithm presented above can be modified in two interesting ways, which both give-up maximization of $\mathcal{F}$ and settle for increase of $\mathcal{F}$. The concession can be made either in the M-step (generalized EM) or in the E-step (variational EM). These modifications are useful when computational requirements of either step of the algorithm become intractable.

For some models it may be difficult to maximize $\mathcal{F}$ w.r.t. $\boldsymbol{\theta}$ but relatively easy to find values for $\boldsymbol{\theta}$ that increase $\mathcal{F}$. For example, it might be easy to maximize $\mathcal{F}$ w.r.t. either one of the elements of $\boldsymbol{\theta}$ but not to maximize w.r.t. all of them simultaneously. However, the convergence to local optima (or saddle points) can still be guaranteed when only increasing instead of maximizing $\mathcal{F}$ in the M-step, algorithms that do so are known as 'generalized EM' algorithms.

Above, due to the independence assumption on the data vectors it was possible to construct the EM lower bound by bounding each of the individual log-likelihoods $\log p(\mathbf{x}_n)$. This iterative bound optimization strategy can also be applied to other settings with unobserved variables. However, in some cases hidden variables are dependent on each other given the observed data. If we have $N$ hidden variables with $k$ possible values each, then in principle the distribution over hidden variables is characterized by $k^N$ numbers. In such cases, the number of summands in the expected joint log-likelihood we have to optimize in the M-step also equals $k^N$. In this case tractability can be obtained if we do not allow for general distributions $q$ over the hidden variables, but only those from a class $\mathcal{Q}$ which gives us a tractable number of summands in the expected joint log-likelihood. For example, we can use distributions over the hidden variables that factor over the different hidden variables. EM algorithms using a restricted class $\mathcal{Q}$ are known as 'variational EM' algorithms. The term refers to the 'variational parameters' that characterize the distributions $q \in \mathcal{Q}$. The variational parameters are optimized

in the E-step of the variational EM algorithm. Most often the variational approach is used to decouple dependencies present in the true distribution over hidden variables which give rise to intractability. An example of a variational EM algorithm is the mean-field algorithm used in hidden Markov random fields, which are applied to image segmentation problems (Celeux et al., 2003).

When using a variational EM algorithm we can no longer guarantee that $\mathcal{F}$ equals the log-likelihood after the E-step, since we cannot guarantee that the true posterior is in $\mathcal{Q}$. Therefore, it is also not guaranteed that the log-likelihood will increase after each EM iteration. However, we can at least still guarantee that the lower bound on the log-likelihood $\mathcal{F}$ will increase after each EM iteration. By restricting the $q$ to a specific class of distributions $\mathcal{Q}$, we effectively augment the log-likelihood objective of the optimization algorithm with a penalty term —the generally non-zero KL divergence— that measures how close the true distribution over the hidden variables is to $\mathcal{Q}$. Thus variational EM parameter estimation algorithms have a bias towards parameters that yield posterior distributions over the hidden variables similar to a member of $\mathcal{Q}$.

### 3.1.3 Model selection

To apply mixture models, two model selection issues have to be resolved: (i) how many components should be used and (ii) which class of component densities should be used. These two factors together determine the 'model structure'. These type of design choices can be compared with the selection of the number and type of hidden nodes in feed-forward neural networks. A trade-off has to be made when choosing the class of models that is used. More complex models (e.g. more mixture components) allow for modelling of more properties of the data, and in general lead to a better fit to the data. However, when using more complex models spurious artifacts of the data due to a noisy measurement system can also be captured by the model. Capturing accidental properties of the data may degrades the estimates, an effect known as 'overfitting'. Less complexity allows for more robust identification of the best model within the selected complexity class and yields a smaller risk of overfitting the data.

Throughout the previous decades several criteria have been proposed to resolve the model selection problem. However, the problem of model selection seems far from being solved. Methods for model selection can be roughly divided into four groups:

1. Methods using concepts from statistical learning theory, e.g. structural risk minimization (Vapnik, 1995). Most of these methods use worst-case guarantees of performance on future data which are derived by a statistical analysis.

2. Information theoretical methods, such as the minimum description length (MDL) principle (Rissanen, 1989). These methods are based on a data compression principle: a model is selected that allows for maximal compression of the data, where

the rate of compression is measured by the total number of bits required to encode both the model and the data (using the model).

3. Bayesian methods that use a prior distribution on model structures and parameters together with a distribution on data given models to define a posterior distribution $p(\text{model structure}|\text{data})$. To compute the required probabilities several approximation techniques have been proposed, e.g. Bayesian information criterion (Schwarz, 1978), variational Bayesian learning (Beal and Ghahramani, 2003), and a number of sampling techniques (Andrieu et al., 2003).

4. Holdout methods that keep part of the data to assess performance of models learned on the rest of the data, e.g. cross-validation (Webb, 2002).

Note that holdout methods are applicable only in cases where there is abundant data, and one can afford to ignore a part of the data for parameter estimation in order to obtain an independent test-set.

Bayesian techniques are attractive because they allow for unified treatment of model selection and parameter estimation. The application of compression based methods like MDL is sometimes problematic because the so called 'two-part' MDL code requires a particular coding scheme and it is not always clear which coding scheme should be used (Verbeek, 2000). Methods based on worst-case analysis are known to give very conservative guarantees, which limits their use.

In general, to apply a particular model selection criterion, parameter estimation has to be performed for models with different numbers of components. The algorithm for estimating the parameters of a Gaussian mixture model we present in Section 3.2 finds a parameter estimate of a $k$-component mixture by iteratively adding components, starting from a single Gaussian. Thus, this algorithm is particularly useful when the number of components has to be determined, since the model selection criterion can be applied to mixtures with $1, \ldots, k$ components as components are added to the mixture.

### 3.1.4 Gaussian mixture models

In this section we discuss the EM algorithm for Mixtures of Gaussian densities (MoG). Recall that we already saw one example of a MoG: generative topographic mapping in Section 2.2.2. We also present a hierarchy of shapes the covariance matrices can take. In the hierarchy, increasingly stringent constraints are placed on the form of the covariance matrix. Then, we discuss how some of the shapes for covariance matrices can be interpreted as linear latent variable models.

Gaussian densities are probably the most commonly used densities to model continuous valued data. The first reason for this popularity is that maximum likelihood parameter estimation can be done in closed form and only requires computation of the data mean and covariance. The second reason is that of all densities with a particular variance, the

Gaussian density has the largest entropy and therefore is the most 'vague' density in this sense. This last property motivates the use of the Gaussian as a default density when there are no reasons to assume that some other parametric density is more appropriate to model the data at hand.

A Gaussian density in a $D$-dimensional space, characterized by its mean $\boldsymbol{\mu} \in \mathbb{R}^D$ and $D \times D$ covariance matrix $\boldsymbol{\Sigma}$, is defined as:

$$\mathcal{N}(\mathbf{x}; \boldsymbol{\theta}) \equiv (2\pi)^{-D/2} |\boldsymbol{\Sigma}|^{-1/2} \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right], \tag{3.8}$$

where $\boldsymbol{\theta}$ denotes the parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ and $|\boldsymbol{\Sigma}|$ denotes the determinant of $\boldsymbol{\Sigma}$. In order for (3.8) to be a proper density, it is necessary and sufficient that the covariance matrix be positive definite. Throughout this thesis we implicitly assume that the likelihood is bounded, e.g. by restricting the parameter space such that the determinant of the covariance matrices is bounded, and hence the maximum likelihood estimator is known to exist (Lindsay, 1983). Alternatively, the imperfect precision of each measurement can be taken into account by treating each data point as a Gaussian density centered on the data point and with small but non-zero variance. We then maximize the *expected* log-likelihood, which is bounded by construction.

**The EM algorithm for mixture of Gaussian densities.** In the E-step of the EM algorithm for a MoG we compute the posterior probabilities $p(s|\mathbf{x})$ according to (3.2) and set $q_{ns} \leftarrow p(s|\mathbf{x}_n)$. In the M-step we maximize $\mathcal{F}$ w.r.t. $\boldsymbol{\theta}$, by setting the partial derivatives of $\mathcal{F}$ w.r.t. the elements of $\boldsymbol{\theta}$ to zero and taking the constraints on the $\pi_s$ into account, and find the updates:

$$\pi_s \leftarrow \frac{1}{N}\sum_{n=1}^{N} q_{ns}, \tag{3.9}$$

$$\boldsymbol{\mu}_s \leftarrow \frac{1}{N\pi_s}\sum_{n=1}^{N} q_{ns}\mathbf{x}_n, \tag{3.10}$$

$$\boldsymbol{\Sigma}_s \leftarrow \frac{1}{N\pi_s}\sum_{n=1}^{N} q_{ns}(\mathbf{x}_n-\boldsymbol{\mu}_s)(\mathbf{x}_n-\boldsymbol{\mu}_s)^\top. \tag{3.11}$$

The update of the mean uses the new mixing weight and the update of the covariance matrix uses the new mean and mixing weight.

In many practical applications of MoGs for clustering and density estimation no constraints are imposed on the covariance matrix. Using an unconstrained covariance matrix, the number of parameters of a Gaussian density grows quadratically with the data dimensionality. For example, using $16 \times 16 = 256$ pixel gray-valued images of an object as data (treating an image as a $256$ dimensional vector), we would need to estimate over

$32.000$ parameters! To reduce the number of parameters in the covariance matrix, it can be constrained to have a form that involves fewer parameters. In some applications a constrained covariance matrix can be appropriate to reflect certain assumptions about the data generating process. See the discussion below on latent variable models.

Next, we discuss several frequently used ways to constrain the covariance matrix. We will use $\mathbf{I}$ to denote the identity matrix, $\mathbf{\Psi}$ to denote a diagonal matrix, and $\mathbf{\Lambda}$ to denote a $D \times d$ matrix, with $d < D$.

**Type 1: factor analysis.** The covariance matrices in factor analysis (FA) are constrained to be of the form:

$$\mathbf{\Sigma} = \mathbf{\Psi} + \mathbf{\Lambda}\mathbf{\Lambda}^\top. \tag{3.12}$$

The $d$ columns of $\mathbf{\Lambda}$ are often referred to as the 'factor loadings'. Each column of $\mathbf{\Lambda}$ can be associated with a latent variable (see the discussion on latent variable models below). In the diagonal matrix $\mathbf{\Psi}$ the variance of each data coordinate is modelled separately, and additional variance is added in the directions spanned by the columns of $\mathbf{\Lambda}$. The number of parameters to specify the covariance matrix is $O(Dd)$. In (Ghahramani and Hinton, 1996) the EM algorithm for mixtures of factor analyzers is given. The determinant and the inverse of the covariance matrix can be efficiently computed using two identities:

$$|\mathbf{A} + \mathbf{BC}| = |\mathbf{A}| \times |\mathbf{I} + \mathbf{C}\mathbf{A}^{-1}\mathbf{B}|, \tag{3.13}$$

$$(\mathbf{A} + \mathbf{BCD})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}(\mathbf{C}^{-1} + \mathbf{D}\mathbf{A}^{-1}\mathbf{B})^{-1}\mathbf{D}\mathbf{A}^{-1}. \tag{3.14}$$

Using these identities we only need to compute inverses and determinants of $d \times d$ and diagonal matrices, rather than of full $D \times D$ matrices:

$$|\mathbf{\Psi} + \mathbf{\Lambda}\mathbf{\Lambda}^\top| = |\mathbf{\Psi}| \times |\mathbf{I} + \mathbf{\Lambda}^\top\mathbf{\Psi}^{-1}\mathbf{\Lambda}|, \tag{3.15}$$

$$(\mathbf{\Psi} + \mathbf{\Lambda}\mathbf{\Lambda}^\top)^{-1} = \mathbf{\Psi}^{-1} - \mathbf{\Psi}^{-1}\mathbf{\Lambda}(\mathbf{I} + \mathbf{\Lambda}^\top\mathbf{\Psi}^{-1}\mathbf{\Lambda})^{-1}\mathbf{\Lambda}^\top\mathbf{\Psi}^{-1}. \tag{3.16}$$

**Type 2: principal component analysis.** Here the constraints are similar to those of FA, but the matrix $\mathbf{\Psi}$ is further constrained to be a multiple of the identity matrix:

$$\mathbf{\Sigma} = \sigma^2\mathbf{I} + \mathbf{\Lambda}\mathbf{\Lambda}^\top \quad \text{with} \quad \sigma > 0. \tag{3.17}$$

A MoG where the covariance matrices are of this type is termed a 'mixture of probabilistic principal component analyzers' (Tipping and Bishop, 1999).

Principal component analysis (Jolliffe, 1986) (PCA) appears as a limiting case if we use just one mixture component and let the variance approach zero: $\sigma^2 \to 0$ (Roweis, 1998). Let $\mathbf{U}$ be the $D \times d$ matrix with the eigenvectors of the data covariance matrix with the $d$ largest eigenvalues, and $\mathbf{V}$ the diagonal matrix with the corresponding eigenvalues

on it diagonal. Then, the maximum likelihood estimate of $\mathbf{\Lambda}$ is given by $\mathbf{\Lambda} = \mathbf{U}\mathbf{V}^{\frac{1}{2}}$. For probabilistic PCA the number of parameters is also $O(Dd)$, as for FA.

To see the difference between PCA and FA note the following. Using PCA we can arbitrarily rotate the original basis of our space: the covariance matrix that is obtained for the variables corresponding to the new basis is still of the PCA type. Thus the class of PCA models is closed under rotations, which is not the case for FA. On the other hand, if some of the data dimensions are scaled, the scaled FA covariance matrix is still of the FA type. Thus the class of FA models is closed under non-uniform scaling of the variables, which is not the case for PCA. Hence if the relative scale of the variables is not considered meaningful, then an FA model may be preferred over a PCA model, and vice versa if relative scale is important.

**Type 3: k-subspaces.** Here, we further restrict the covariance matrix such that the norm of all columns of $\mathbf{\Lambda}$ is equal:

$$\mathbf{\Sigma} = \sigma^2\mathbf{I} + \mathbf{\Lambda}\mathbf{\Lambda}^\top \quad \text{with} \quad \mathbf{\Lambda}^\top\mathbf{\Lambda} = \rho^2\mathbf{I} \quad \text{and} \quad \sigma, \rho > 0. \tag{3.18}$$

This type of covariance matrix was used in (Verbeek et al., 2002b), for non-linear dimension reduction technique similar to the one described in Chapter 5. The number of parameters is again $O(Dd)$.

If we train a MoG with this type of covariance matrix with EM, then this results in an algorithm known as k-subspaces (Kambhatla and Leen, 1994; de Ridder, 2001) under the following conditions: (i) all mixing weights are equal, $\pi_s = 1/k$, and (ii) the $\sigma_s$ and $\rho_s$ are equal for all mixture components: $\sigma_s = \sigma$ and $\rho_s = \rho$. Then, if we take the limit as $\sigma \to 0$, we obtain k-subspaces. The term k-subspaces refers to the $k$ linear subspaces spanned by the columns of the $\mathbf{\Lambda}_s$. The k-subspaces algorithm, a variation on the k-means algorithm (see Section 2.1.2), iterates two steps:

1. Assign each data point to the subspace which reconstructs the data vector the best (in terms of squared distance between the data point and the projection of the data point on the subspace).
2. For every subspace compute the PCA subspace of the assigned data to get the updated subspace.

**Type 4: isotropic.** This shape of the covariance matrix is the most restricted:

$$\mathbf{\Sigma} = \sigma^2\mathbf{I} \quad \text{with} \quad \sigma > 0. \tag{3.19}$$

This model is referred to as isotropic or spherical variance since the variance is equal in all directions. It involves just the single parameter $\sigma$.

If all components share the same variance $\sigma^2$ and letting it approach zero, the posterior on the components for a given data vector tends to put all mass on the component

closest in Euclidean distance to the data vector (Bishop, 1995). This means that the E-step reduces to finding the closest component for all data vectors. The expectation in (3.6) then just counts the term for the closest component. This means for the M-step that the centers $\boldsymbol{\mu}_s$ of the components are found by simply averaging all the data vectors for which it is the closest. This simple case of the EM algorithm coincides exactly with the k-means algorithm.

Taking $\boldsymbol{\Sigma}$ diagonal, and thus letting all variables be independently distributed, is also a popular constraint yielding $O(d)$ parameters, however it is not used by any of the techniques discussed in this thesis.

**Linear latent variable models.** Three types of covariance matrix (FA, PCA and k-subspaces) can be interpreted as linear latent variable models. The idea is that the data can be thought of as being generated in a low dimensional latent space, which is linearly embedded in the higher dimensional data space. Some noise may make the observed data deviate from the embedded linear subspace.

For these models it is conceptually convenient to introduce a new, hidden variable $\mathbf{g}$ for every data vector. This variable represents the (unknown) $d$-dimensional latent coordinate of the data vector. Using this variable we can write the density for the data vectors by marginalizing over the hidden variable:

$$p(\mathbf{g}) = \mathcal{N}(\mathbf{g}; 0, \mathbf{I}) \tag{3.20}$$

$$p(\mathbf{x}|\mathbf{g}) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu} + \boldsymbol{\Lambda}\mathbf{g}, \boldsymbol{\Psi}) \tag{3.21}$$

$$p(\mathbf{x}) = \int p(\mathbf{x}|\mathbf{g})p(\mathbf{g})\mathrm{d}\mathbf{g} = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Psi} + \boldsymbol{\Lambda}\boldsymbol{\Lambda}^\top) \tag{3.22}$$

It is not necessary to introduce $\mathbf{g}$ to formulate the model, but it may be convenient to use $\mathbf{g}$ for computational reasons. For example, in (Roweis, 1998) this view is adopted to compute PCA subspaces without computing the data covariance matrix or the data inner-product matrix. The cost to compute these matrices is, respectively, $O(ND^2)$ and $O(DN^2)$. By treating the latent coordinates as hidden variables one can perform PCA with an EM algorithm that has a cost of $O(dDN)$ per iteration to find the $d$ principal components. This EM algorithm for PCA involves computing the posterior distribution on $\mathbf{g}$ given $\mathbf{x}$, which is given by:

$$p(\mathbf{g}|\mathbf{x}) = \mathcal{N}(\mathbf{g}; \mathbf{m}(\mathbf{x}), \mathbf{C}), \tag{3.23}$$

$$\mathbf{m}(\mathbf{x}) = \mathbf{C}\boldsymbol{\Lambda}^\top\boldsymbol{\Psi}^{-1}(\mathbf{x} - \boldsymbol{\mu}), \tag{3.24}$$

$$\mathbf{C}^{-1} = \mathbf{I} + \boldsymbol{\Lambda}^\top\boldsymbol{\Psi}^{-1}\boldsymbol{\Lambda}. \tag{3.25}$$

We can use linear latent variable models to reduce the dimensionality of the data by inferring the $d$-dimensional latent coordinates from the $D$-dimensional data vectors and using the found latent coordinates for further processing of the data. Another option