Stochastic Approximation Learning for

Mixtures of Multivariate Elliptical Distributions

Ezequiel López-Rubio

Department of Computer Languages and Computer Science

University of Málaga

Bulevar Louis Pasteur, 35. 29071 Málaga.

SPAIN

Phone: (+34) 95 213 71 55

Fax: (+34) 95 213 13 97

ezeqlr@lcc.uma.es

Abstract: Most of current approaches to mixture modeling consider mixture components from a few families of probability distributions, in particular from the Gaussian family. The reasons of these preferences can be traced to their training algorithms, typically versions of the Expectation-Maximization (EM) method. The reestimation equations needed by this method become very complex as the mixture components depart from the simplest cases. Here we propose to use a stochastic approximation method for probabilistic mixture learning. Under this method it is straightforward to train mixtures composed by a wide range of mixture components from different families. Hence, it is a flexible alternative for mixture learning. Experimental results are presented to show the probability density and missing value estimation capabilities of our proposal.

Keywords: Mixture modeling, stochastic approximation, multivariate data analysis, unsupervised learning, missing value estimation.
1 Introduction

Probabilistic mixture models are among the most widely used techniques in many fields of pattern recognition and machine learning (Frühwirth-Schnatter, 2006). At the root of this success lies the Expectation Maximization learning method and its many variants (McLachlan & Krishnan, 2008; Paalanen et al., 2006; Figueiredo & Jain, 2002), which provides maximum likelihood estimates by means of an iterative scheme. Many times the hardest task when developing an EM algorithm consists in maximizing (M step) the expected likelihood found on the E step. This depends on the particular form of the mixture component densities. The Gaussian family is by far the most employed (Langseth & Nielsen, 2005; Ingrassia & Rocci, 2007; Zeng & Cheung, 2009; Goldberger et al., 2008), but the multivariate Student-t family has also attracted attention because of its robustness against outlying data (Peel & McLachlan, 2000; Shoham, 2002; Svensén & Bishop, 2005; McLachlan et al., 2002), which can be combined with Probabilistic Principal Components Analysis (PPCA) to obtain parsimonious models (Archambeau et al, 2008). Nevertheless, the estimation of the Student-t degrees of freedom parameter is difficult, since the likelihood function has many irrelevant maxima (Liu & Rubin, 1995; Fernandez & Steel, 1999; Meng & van Dyk, 1997). For an arbitrary family of probability distributions the M step equations could not be expressed in closed form. General nonlinear optimization techniques like the BFGS method (Avriel, 2003) could be employed to maximize the likelihood in each step, but this would lead to a considerable computational load. Zribi & Ghorbel (2003) circumvent this problem by training estimators which are not proper probability density functions. In addition to this, EM can be used to train hierarchical mixtures of experts, where each expert network produces an output which is a generalized linear function of
the input sample; this architecture is associated with a decision tree which can be modelled by a probability density (Jordan & Jacobs, 1994; Jordan & Xu, 1995; Polikar, 2006; Subasi, 2007).

Variational Bayesian (VB) methods can be regarded as extensions of the EM framework, from the estimation of the single most probable value of each parameter (EM) to an approximation of the posterior distribution of the parameters and the latent variables (VB). Classic VB learning methods impose priors on the parameters of a Gaussian mixture (Attias, 1999; Constantinopoulos & Likas, 2007). Hence, they are heavily dependent on the particular properties of the Gaussian distribution. However, this framework has been recently extended to robust mixtures of Student-t distributions by expressing the Student-t as a marginalization over extra latent variables (Svensén & Bishop, 2005). Moreover, Archambeau & Verleysen (2007) have shown that it is not necessary to assume a factorized approximation of the posterior distribution on the latent indicator variables and the latent scale variables, thus obtaining a tractable algorithm with fewer assumptions.

There are some approaches to mixture model learning not based on EM. Acito et al. (2007) search for the best fitting between the empirical and the theoretical cumulative distribution functions, but this is a technique which is limited to adjusting a small number of parameters because of its computational load. In vector quantization applications, it is possible to design Gaussian mixture models by means of Lloyd clustering (Gray and Linder, 2003; Aiyer et al. 2005). Gibbs sampling has been used to learn mixtures of factors analysers (Fokoué and Titterington, 2003). Finally, supervised learning of Gaussian mixture density hidden Markov models has been carried out by Generalized Probabilistic Descent (GPD) for speech recognition purposes (Kurimo, 1997). These techniques have a limited field of application and/or only consider
Gaussian mixtures, which explains that EM remains as the most popular learning method for mixture models.

We propose a learning method which is based on the stochastic approximation framework (Kushner & Yin, 2003) to overcome these problems. Our goal is to train mixture components with arbitrary elliptical densities (Fang et al., 1987; Arnold et al., 2008; Gomez et al., 2007), also called Mahalanobis densities (Magdon-Ismail & Sill, 2008), provided that both the mean vector and the covariance matrix exist. Stochastic approximation is suited for this purpose because it recursively adjusts the relevant parameters so that some performance goal is met. It has been considered recently for other machine learning applications (Liang, 2007; Liang et al., 2007). This procedure accounts explicitly for noise corrupted samples, whose effect is averaged out (Wang et al., 1996; Chen, 1998; Andrieu, 2005).

It must be pointed out that stochastic approximation has been previously used for mixture model learning. Known approaches include stochastic EM (SEM, Celeux & Diebolt, 1992), stochastic approximation EM (SAEM, Delyon et al, 1999), and online EM (Cappe & Moulines, 2009). However, these models are fundamentally different from our approach, since they integrate the stochastic approximation within the Expectation Maximization algorithm. In particular, they employ stochastic approximation to propose an alternative to the original expectation (E) step of the classical EM algorithm, while the maximization (M) step remains the same. Hence, these methods suffer from the fact that for an arbitrary family of probability distributions the M step equations might not be expressed in closed form, just as in standard EM. In fact, most of them are commonly applied to exponential families, where the M step equations are easier to obtain. Consequently, they do not solve the problem we try to address here.
The outline of the paper is as follows. In Section 2 we present the class of mixture models that are suited for our training procedure, which we call *Mixtures of Multivariate Elliptical Distributions (MMED)*. The learning algorithm for them is considered in Section 3. A discussion of the differences among known models and our proposal is carried out in Section 4. Finally, computational results are shown and analyzed in Section 5.

2 Multivariate elliptical distributions

As explained in the introduction, our proposal is able to learn mixture models with a wide range of components. First we study the possible component densities (subsection 2.1) and then the mixtures (subsection 2.2).

2.1 Mixture components

In this subsection we specify the probability density distributions that are suited for our learning procedure. The class of mixture components we consider is that of elliptical densities (Fang et al., 1987; Arnold et al., 2008; Gomez et al., 2007), also called Mahalanobis densities (Magdon-Ismail & Sill, 2008). We impose restrictions to ensure the existence of the mean vector and the covariance matrix.

We say that a \( D \)-dimensional probability distribution is a *multivariate elliptical distribution* if and only if both the mean vector and the covariance matrix exist,

\[
\mu = E[t] 
\]

\[
C = E[(t - \mu)(t - \mu)^T] 
\]

and its probability density function (pdf) can be written as

\[
p(t) = g((t - \mu)^T C^{-1}(t - \mu)) 
\]
Mixtures of Multivariate Elliptical Distributions

where \( g \) is called the density generating function. For the sake of brevity, in the following equations we use \( \alpha \) as shorthand for the squared Mahalanobis distance:

\[
\alpha = (t - \mu)^T C^{-1} (t - \mu)
\]  

(4)

For our purposes, a family of probability distributions is a set of multivariate elliptical distributions that only differ in the particular values of some parameters of \( g \) other than \( \mu \) and \( C \). Here we consider six families. The first one is the Gaussian family, and it only has one member:

\[
g_{\text{Gaussian}}(\alpha) = (2\pi)^{D/2} |C|^{-1/2} \exp \left( -\frac{1}{2} \alpha \right)
\]  

(5)

As in the previous case, the Exponential Square Root family (Magdon-Ismail & Sill, 2008) and the Laplacian family (Eltoft et al., 2006) only have one member each:

\[
g_{\text{ExpSquareRoot}}(\alpha) = \frac{(D-1)^{(D+1)/2} \Gamma \left( \frac{D}{2} \right)}{2 \Gamma(D) \pi^{D/2} |C|^{1/2}} \exp \left( -\frac{\sqrt{D-1}}{\sqrt{\alpha}} \right)
\]  

(6)

where \( D > 1 \) and \( \Gamma \) is the gamma function, and

\[
g_{\text{Laplacian}}(\alpha) = 2(2\pi)^{-D/2} |C|^{-1/D} \left( \frac{1}{2} |C|^{2/D} \alpha \right)^{1/2-D/4} \Gamma_{(D-1)-1} \left( \sqrt{\alpha} \right)
\]  

(7)

where \( K_m(x) \) is the modified Bessel function of the second kind and order \( m \), evaluated at \( x \). The Laplacian distribution is useful for representing sparsely distributed data, with mutually dependent components.

The Student-t family (Svensén & Bishop, 2005) is made up of probability distributions which differ in the value of the degrees of freedom parameter \( \nu \geq 2 \):

\[
g_{\text{Student}}(\alpha) = \frac{\Gamma \left( \frac{\nu + D}{2} \right)}{|C|^{D/2} (\pi (\nu - 2))^{D/2} \Gamma \left( \frac{\nu}{2} \right)} \left( 1 + \frac{\alpha}{\nu - 2} \right)^{-\frac{\nu-D}{2}}
\]  

(8)
Mixtures of Multivariate Elliptical Distributions

where \( \nu > 2 \). The Student-t distributions have long tails, which make them suitable for modelling data with outliers, particularly when \( \nu \) is low. In the limit \( \nu \to \infty \), the Student-t approaches to the Gaussian distribution.

Next we consider the Uniform family of probability distributions on hyperellipsoids (Li, 1992; Sun & Farooq, 2002) with a distance parameter \( \delta^2 \):

\[
g_{\text{Uniform}}(\alpha) = \begin{cases} 
    \left( \delta^2 \pi \right)^{-D/2} |C|^{-1/2} \Gamma\left( \frac{D}{2} + 1 \right) & \text{if } \alpha \leq \delta^2 \\
    0 & \text{if } \alpha > \delta^2 
\end{cases}
\]

(9)

where \( \delta > 0 \) is the Mahalanobis radius which defines the hyperellipsoid \( H_\delta \):

\[
H_\delta = \{ t \in \mathbb{R}^D \mid (t - \mu)^T C^{-1} (t - \mu) \leq \delta^2 \}
\]

(10)

This family is better suited for clustered data, since it concentrates all the probability mass in a finite volume.

Finally we propose to use the multivariate Triangular family, which has the same \( \delta^2 \) parameter as the Uniform family. The support of a Triangular probability density function is the hyperellipsoid \( H_\delta \) and the probability density inside \( H_\delta \) is proportional to \( \delta - r \), where \( r \) is the Mahalanobis distance:

\[
r = \sqrt{\alpha} = \sqrt{(t - \mu)^T C^{-1} (t - \mu)}
\]

(11)

\[
g_{\text{Triangular}}(\alpha) = \begin{cases} 
    \frac{\delta - \sqrt{\alpha}}{2} \pi^{-D/2} \delta^{-D-1} \left( \frac{1}{D} - \frac{1}{D+1} \right)^{-1} |C|^{-1/2} \Gamma\left( \frac{D}{2} \right) & \text{if } \alpha \leq \delta^2 \\
    0 & \text{if } \alpha > \delta^2 
\end{cases}
\]

(12)

In Appendix A we prove that (12) is a proper probability density function for all parameter values \( \delta^2 > 0 \). This family has the convenient property that its support is finite, while at the same time its probability density grows as we get closer to the mean vector. If the input dimension is \( D=1 \), this family reduces to the well known univariate symmetric triangular distribution:
Mixtures of Multivariate Elliptical Distributions

\[ D = 1 \Rightarrow g_{\text{Triangular}}(\alpha) = \begin{cases} \frac{\delta - \sqrt{\alpha}}{\delta^2} [C]^{-1/2} & \text{if } \alpha \leq \delta^2 \\ 0 & \text{if } \alpha > \delta^2 \end{cases} \]  

\hspace{1cm} (13)

Other choices of pdf families are also possible, but it is required that at least one of the considered pdfs has an infinite support which spans the entire input space. Otherwise we would run into problems in presence of outliers, which would be assigned a null probability density.

2.2 Mixture models

Now we are ready to define how the probability density models considered in the previous subsection are combined to form mixtures. We propose the use of several components in the same mixture which share a common mean vector and a common covariance matrix. We call such set of components a Group of Mixture Components (GMC). Hence, a Mixture of Multivariate Elliptical Distributions (MMED) is defined as

\[ p(t) = \sum_{i=1}^{H} \pi_i p(t | i) \]  

\hspace{1cm} (14)

where each mixture component \( i \) is a multivariate elliptical distribution belonging to one and only one GMC. Here \( H \) is the number of mixture components and \( \pi_i \) are the prior probabilities or mixing proportions.

For example, we could define an MMED with \( G \) GMCs, where each GMC comprises a Gaussian and a Laplacian mixture component. The corresponding density would be

\[ p(t) = \sum_{i=1}^{G} \left( \pi_{i1} g_{\text{Gaussian}}(\alpha_i) + \pi_{i2} g_{\text{Laplacian}}(\alpha_i) \right) \]  

\hspace{1cm} (15)

where each GMC has its own mean vector \( \mu_i \) and covariance matrix \( C_i \),

\[ \forall i \in \{1,...,G\}, \quad \alpha_i = (t - \mu_i)^T C_i^{-1} (t - \mu_i) \]  

\hspace{1cm} (16)
In our experiments each group is composed of a component from each of the first three families (Gaussian, Exponential Square Root and Laplacian), and a variable number of components of the other three families (Student-t, Uniform and Triangular), where the values of the parameters are fixed and not subject to learning. This strategy saves a considerable amount of model parameters, particularly with large $D$, while at the same time it features the data representation capabilities of all the considered families.

Let $GMC_j$ be the set of the indexes of the mixture components belonging to the $j$-th group of mixture components. Then the prior probability of each GMC is computed as the sum of the prior probabilities of its mixture components:

$$\forall j \in \{1, \ldots, M\}, \quad P(GMC_j) = \sum_{i \in GMC_j} \pi_i$$  \hspace{1cm} (17)

where $M$ is the total number of GMCs. Hence we have

$$\forall i \in GMC_j, \quad P(i \mid GMC_j) = \frac{\pi_i}{\sum_{z \in GMC_j} \pi_z}$$  \hspace{1cm} (18)

This allows us to obtain the probability density function associated with the $j$-th GMC:

$$p(t \mid GMC_j) = \sum_{i \in GMC_j} P(i \mid GMC_j) p(t \mid i)$$  \hspace{1cm} (19)

Then the mixture probability density can be expressed in terms of these densities:

$$p(t) = \sum_{j=1}^{M} P(GMC_j) p(t \mid GMC_j)$$  \hspace{1cm} (20)

The posterior responsibility of mixture component $i$ for having generated a sample $t_n$ is given by:

$$R_{ni} = P(i \mid t_n) = \frac{\pi_i p(t_n \mid i)}{\sum_{z} \pi_z p(t_n \mid z)}$$  \hspace{1cm} (21)
The posterior responsibility of the $j$-th GMC for having generated a sample $t_n$ is obtained similarly:

$$P(GMC_j | t_n) = \frac{P(GMC_j)p(t_n | GMC_j)}{\sum_{z=1}^{M} P(GMC_z)p(t_n | GMC_z)}$$

(22)

There is another way to study a MMED, namely by classifying the mixture components by families. Let Gaussians, ExpSquareRoots, Laplacians, Students, Uniforms and Triangulars be the sets of mixture components belonging to each of the six considered families. These six sets of mixture components are disjoint one another, and their prior probabilities are readily computed:

$$\forall Family \in \{\text{Gaussians,...,Gammas}\}, \quad P(\text{Family}) = \sum_{i \in \text{Family}} \pi_i$$

(23)

From the above equation we can derive the probability density function associated with each family and the posterior responsibilities, just as we have done with the GMCs:

$$\forall Family \in \{\text{Gaussians,...,Triangul}ars\}, \quad P(i | \text{Family}) = \frac{\pi_i}{\sum_{z \in \text{Family}} \pi_z}$$

(24)

$$\forall Family \in \{\text{Gaussians,...,Triangul}ars\}, \quad p(t | \text{Family}) = \sum_{i \in \text{Family}} P(i | \text{Family})p(t | i)$$

(25)

$$\forall Family \in \{\text{Gaussians,...,Triangul}ars\}, \quad P(\text{Family} | t_n) = \frac{P(\text{Family})p(t_n | \text{Family})}{\sum_{Family'} P(\text{Family}')p(t_n | \text{Family}')}$$

(26)

3 Learning algorithm

The learning procedure for MMED models consists of two phases: first the model is adapted online to the input data by means of stochastic approximation, and then the irrelevant mixture components are pruned by a procedure that we call histogram analysis. Hence, the presentation of the learning method is split in four parts: first we
show how to update the mixture by stochastic approximation to adapt it to a new training sample, then we explain the initialization procedure, thirdly we present the histogram analysis to prune the trained model, and finally the algorithm is summarized as a sequence of steps.

3.1 Mixture update

When a new sample $t_n$ is presented to the mixture at time step $n$, its information is used to update the mixture components. If we are to update each mixture component $i$ with the information from sample $t_n$, an online learning method for a MMED is required. This strategy has been examined by Sato & Ishii (2000) for Gaussian PCA mixtures. Our online method generates the updated values $\pi_i(n)$, $\mu_i(n)$ and $C_i(n)$ from the old values $\pi_i(n-1)$, $\mu_i(n-1)$, $C_i(n-1)$ and the new sample $t_n$. The application of Robbins-Monro stochastic approximation algorithm yields the following update equations:

$$\pi_i(n) = (1 - \varepsilon(n))\pi_i(n-1) + \varepsilon(n)R_{ni}$$  (27)

$$m_i(n) = (1 - \varepsilon(n))m_i(n-1) + \varepsilon(n)R_{ni}t_n$$  (28)

$$\mu_i(n) = \frac{m_i(n)}{\pi_i(n)}$$  (29)

$$M_i(n) = (1 - \varepsilon(n))M_i(n-1) + \varepsilon(n)R_{ni}(t_n - \mu_i(n))(t_n - \mu_i(n))^\top$$  (30)

$$C_i(n) = \frac{M_i(n)}{\pi_i(n)}$$  (31)

Please see Appendix B for details and a convergence proof. We must remark that $m_i(n)$ and $M_i(n)$ are auxiliary variables required to update the model parameters. In the above equations, $\varepsilon(n)$ is the step size of the Robbins-Monro algorithm and is typically chosen as

$$\varepsilon(n) = \frac{1}{an + b}, \ 0 < a < 1$$  (32)
In our experiments we have found that selecting $b = \frac{1}{a}$ often yields good results because the model remains plastic long enough. That is, the model parameters are able to incorporate the information from the last samples adequately, so that the initialization has little influence in the learning process. Hence, we have used in practice

$$\varepsilon(n) = \frac{1}{\varepsilon_0 n + \frac{1}{\varepsilon_0}}, \quad 0 < \varepsilon_0 < 1$$

(33)

where $\varepsilon_0$ is set close to zero because higher values produce an excessive plasticity (variability) of the estimations.

In order to save computations and free parameters of the model, the mixture components with $\pi_i(n) < 10^{-5}$ are pruned (Paul, 1991; Duchateau et al., 1998), i.e. their a priori probabilities $\pi_i(n)$ are zeroed and no more computations are done on them. This pruning procedure is of a secondary importance, since the main pruning is done in the histogram analysis phase (Subsection 3.3).

### 3.2 Mixture initialization

The initialization of each mixture component is achieved by estimating its starting parameters from a randomly chosen set of samples, which is different for each mixture component. For each mixture component $i$ we select $K$ samples $t_n$, and we compute their mean as $\mu_i(0)$:

$$\mu_i(0) = \frac{1}{K} \sum_{n=1}^{K} t_n$$

(34)
The value of $K$ is not crucial, provided that it is higher than $D$ in order to avoid degenerate covariance matrices. We have used $K=10D$ in the experiments. Then we compute their differences with $\mu_i(0)$ to yield $C_i(0)$:

$$C_i(0) = \frac{1}{K} \sum_{n=1}^{K} \left(t_n - \mu_i(0)\right)\left(t_n - \mu_i(0)\right)^T$$  \hspace{1cm} (35)$$

The initial mixing probabilities are assumed to be equal:

$$\pi_i(0) = \frac{1}{H}$$  \hspace{1cm} (36)$$

The following auxiliary variables are estimated following their definitions (A.13) and (A.14):

$$m_i(0) = \frac{1}{HK} \sum_{n=1}^{K} t_n = \frac{1}{H} \mu_i(0)$$  \hspace{1cm} (37)$$

$$M_i(0) = \frac{1}{HK} \sum_{n=1}^{K} \left(t_n - \mu_i(0)\right)\left(t_n - \mu_i(0)\right)^T = \frac{1}{H} \frac{1}{C_i(0)}$$  \hspace{1cm} (38)$$

3.3 Histogram analysis

Classical probabilistic mixture pruning techniques involves removing some mixture components with low a priori probability (Paul, 1991; Duchateau et al., 1998). While we have already taken advantage of these techniques in Subsection 3.1, better results can be obtained by incorporating a procedure which analyzes which mixture components fit best to the features of the input data. Here we propose such a procedure, which we call histogram analysis. It is to be carried out after the stochastic approximation learning of Subsection 3.1 has been completed. This is because the training is online due to the nature of the stochastic approximation framework. This implies that some mixture components which appear to be useless at the beginning of
the training can actually be very important as new training samples are presented to the algorithm.

Each GMC is analyzed separately. Let us assume that we are analyzing the $k$-th GMC, $GMC_k$. Our aim is to compare the experimental histogram $\beta$ of the training data $t_n$ with the theoretical histograms $b_i$ which could be expected from each of the mixture components $i$ of $GMC_k$:

$$
\forall j \in \{1, ..., NBins\}, \quad \beta_j = \frac{\sum_{t_n} P(GMC_k | t_n)}{\sum_{t_n} P(GMC_k | t_n)}
$$

$$
\forall j \in \{1, ..., NBins\}, \quad V_j = \{t \in \mathbb{R}^D | \lambda_{j-1} \leq (t - \mu)^T C^{-1} (t - \mu) < \lambda_j \}
$$

$$
\forall i \in GMC_k \forall j \in \{1, ..., NBins\}, \quad b_{ij} = P(\lambda_{j-1} \leq (t - \mu)^T C^{-1} (t - \mu) < \lambda_j | i)
$$

where $NBins$ is the number of bins of the histograms, the regions $V_j$ define the bins, and the limits of the bins $\lambda_j$ are prespecified values (not subject to learning). The extreme limits must be $\lambda_0 = 0$ and $\lambda_{NBins} = +\infty$ so that the whole input space is covered by the regions $V_j$. Also, please note that the contributions to the experimental histogram of the training data $t_n$ are weighted by their probabilities $P(GMC_k | t_n)$ of having been generated by the GMC at hand, $GMC_k$. This way, the experimental histogram $\beta$ accounts for the training data associated with $GMC_k$.

Then we obtain the weighting $w$ of the mixture components which best fits to the experimental histogram:

$$
B = (b_i)_{i \in GMC_k}
$$

$$
w = \arg \min_{w \geq 0} \|\beta - BW\|
$$
As seen, equation (43) is a nonnegative least squares optimization, which can be carried out by following, for example, Lawson & Hanson (1974). Other methods could be used to increase the sparseness of the solution, such as sparse learning (Wu et al, 2006; Zhang & Zhou, 2010) and compressed sensing (Donoho, 2006; Candes & Wakin, 2008); however, in this case care must be taken so that important mixture components are not stripped out. On the other hand, it must be noted that $B$ is a matrix of constants (it does not depend on the training data) which should be precomputed before the learning starts so that no training time is lost in this task. Appendix C explains how to compute these constants $b_{ij}$.

The optimization (43) usually yields a weight vector $w$ with many zeros, since a reduced number of mixture components is enough to obtain the best fit to the experimental histogram $\beta$. This is particularly convenient, since the mixture components $i$ with $w_i = 0$ can be removed from the model. The removal is accomplished by modifying the a priori probabilities of the mixture components corresponding to $GMC_k$ so that they reflect the weighting given by $w$:

$$\forall i \in GMC_k, \quad \pi_i = \frac{w_i}{\sum_{h \in GMC_i} w_h} \quad (44)$$

The other trainable parameters (the mean vectors and the covariance matrices) are left unchanged. This procedure is repeated for all the GMCs of the model, so that all the unnecessary mixture components are pruned. As a final check, the changes are accepted only if the likelihood of the training data increases.

### 3.4 Summary

The learning algorithm for the MMED model can be summarized as follows:
1. Set the initial values $\pi_i(0)$, $\mu_i(0)$, $C_i(0)$, $m_i(0)$, and $M_i(0)$ for all mixture components $i$, as explained in Subsection 3.2.

2. Choose an input sample $t_n$ and use (21) to compute the posterior responsibilities of the mixture components $R_{ni}$.

3. For every component $i$, estimate its parameters $\pi_i(n)$, $\mu_i(n)$, $C_i(n)$, $m_i(n)$ and $M_i(n)$, by equations (27)-(31).

4. If the mixture has converged or the maximum time step $N$ has been reached, go to step 5. Otherwise, go to step 2.

5. Prune the trained model by means of histogram analysis, as explained in Subsection 3.3.

4 Discussion

The MMED model is presented here as a suitable framework for developing specialized applications. This is facilitated by several desirable features:

a) Any multivariate elliptical probability distribution can be incorporated into it, no matter how complex is its probability density function, with minimal changes to the learning algorithm. This allows including mixture components suited for different kinds of data, for example clustered data and outlier data. Cluster modeling by probability distributions is a recurring problem in pattern recognition (Cuevas et al., 2001; Volkovich et al., 2008). Outlier modeling is also pervasive in this area (Chatzis & Varvarigou, 2008; Aysal & Barner, 2006).

b) The number of model parameters is greatly reduced by means of mean and covariance sharing, particularly with large $D$. Each additional mixture component in a GMC only produces one extra parameter (its prior probability). This way, our method presents an alternative to classic Mixtures of Gaussians, where each increase in the
model complexity involves adding a Gaussian mixture component with either
\( \frac{1}{2} D^2 + \frac{1}{2} D \) new parameters (full covariance matrix) or \( DK + D + 1 - \frac{K(K-1)}{2} \) new parameters (Gaussian PPCA with \( K \) principal directions). For large dimension \( D \), this implies that Mixtures of Gaussians can only grow at the expense of fitting many free parameters, while with our proposal we can add free parameters one by one.

c) There is no need to estimate the particular parameters of each family, since a range of possible values is considered with a small computational cost. This has the advantage of alleviating the possible parameter estimation errors due to the lack of input data or inherent instabilities of the estimation procedure. For example, the estimation of the degrees of freedom parameter \( \nu \) of the Student-t family is well known in literature as a hard problem (Liu & Rubin, 1995; Fernandez & Steel, 1999; Peel & McLachlan, 2000; Liu, 1997; Meng & van Dyk, 1997). This is because the likelihood function of the Student-t can have very high spikes with little associated posterior mass (Liu & Rubin, 1995). Here our method makes a soft decision by means of learning the mixing weights \( \pi \) corresponding to different values of a particular parameter, rather than a hard decision about the ‘true’ value of the parameter.

d) The stochastic approximation algorithm works by averaging out the estimation errors (Kushner & Yin, 2003). This reduces the effects of noisy samples (Wang et al., 1996; Chen, 1998) and guarantees asymptotical convergence (see Appendix B).

There are some variants of the considered training method which we regard as possible lines of future work. As seen, mixture component pruning (Solka et al., 1998) enhances the performance of our proposal by removing those mixture components or GMCs whose contribution to the model is small. A natural extension of this procedure is the dynamic creation/destruction of mixture components and GMCs. There are
several well established criteria to select the optimal structure among several options (Graham & Miller, 2006; Fonseca & Cardoso, 2007). We consider two of them in the experimental section. In addition to this, hierarchical mixture models (Ghosh, 2006; Iwata & Hayashi, 2008) could also be developed.

5 Experimental results

We have designed four sets of experiments to show the capabilities of the MMED model. Some of them include comparisons with other probabilistic mixture models, namely the Mixtures of Probabilistic PCA (Tipping & Bishop, 1999) and the standard Mixtures of full covariance Gaussians. These two models have been called MFullCov and MPPCA, and their implementations come from the NETLAB library (Neural Computing Research Group, 2003), which correspond to Expectation-Maximization (EM) algorithms. The four sets of experiments are presented in subsections 5.1-5.4, and a final analysis of their results is carried out in subsection 5.5.

We have used MMED models with either 18 or 78 mixture components per group. For the 18 components per group models, each group was made of the following components: a Gaussian, an exponential square root and a multivariate Laplacian; five multivariate Students with $\nu=2.01, 5, 10, 100$ and $1000$; five Uniforms with $\delta^2=0.25, 0.5, 1, 2$ and $4$; and finally five Triangulars with the same values of $\delta^2$. For the 78 components per group models, we added the following components: twenty Students with $\nu=3, 4, 6, 7, 8, 9, 15, 20, 25, 30, 35, 40, 45, 50, 200, 300, 400, 500, 600$ and $700$; twenty Uniforms with $\delta=0.0001, 0.01, 0.02, 0.05, 0.1, 0.125, 0.2, 0.3, 0.75, 0.8, 1.5, 2.5, 3, 5, 8, 10, 100, 20, 50, 1000$; and twenty Triangulars with the same values of $\delta^2$. The particular choices of the parameters are not crucial, provided that the parameter ranges of each pdf family are adequately covered. The overall strategy is to have enough
Mixtures of Multivariate Elliptical Distributions

components so that the difference between a component and the next one of its family is sufficiently small. This way we ensure a small sensitivity to changes in the above described parameters. However, this should not lead us to the wrong conclusion that all the pdfs are the same. There are important differences among them, in particular if we consider the tails of the distributions for high input dimensionalities $D$, which are crucial for outlier modeling. This point is illustrated in Figures 1 and 2, where we show the probability densities with respect to the Mahalanobis distance $r$ for the 18 mixture components per group setup. As seen, each of the 18 mixture components is significantly different from all the others. Please note that the $y$-axis of the figures is in log scale, so that small distances in that axis correspond to large differences in the values of the probability density.

![Figure 1. Probability densities for the 18 mixture components per group setup and input dimensionality $D=40$.](image)

Figure 2. Detail of Figure 1. Please note the difference among the Gaussian and all the Student-t’s for high values of the Mahalanobis distance $r$ (tails of the distributions).

In which respects to the histogram analysis, we have always used $N_{Bins}=13$ and bin limits $\lambda_j=0, 0.4, 0.8, 1.1, 1.4, 1.8, 2.25, 2.5, 3, 4.5, 6, 10, 100, +\infty$. The lower $\lambda_j$ values are designed to analyze in detail the shape of the centers of the experimental distributions, while the higher values are useful to measure the thickness of the tails of the experimental distributions. Like before, the exact values of $\lambda_j$ are not crucial, provided that the centers and the tails of the distributions are adequately analyzed.

5.1 Low dimensional datasets experiments
First of all we have carried out experiments with low dimensional data to show graphically the capabilities of MMED. The MMED models have 78 components per group, and we have trained them for 100,000 steps.
We have selected a simple distribution, namely the unidimensional uniform distribution on the interval [0,1]. We have drawn 10,000 training samples from this distribution. Figure 3 shows a MMED with only one group, and Figure 4 shows a MMED with 3 groups. The target (uniform) pdf is also shown as a reference in both figures. In addition to this, the Average Negative Log-Likelihood (ANLL) has been computed on a test set with \(Z=1000\) samples:

\[
ANLL = -\frac{1}{Z} \sum_{n=1}^{Z} \log p(t_n)
\]  

(45)

The ANLL results for MMED are very close to that of the real input distribution in both cases. This is because MMED is quite flexible because of its use of differently shaped mixture components. It must be remarked that the 3 groups result (Figure 4) is very similar to the one group result (Figure 3), despite the fact that for this input distribution the two extra groups can not improve the performance, i.e. the three groups model is purportedly oversized. On the other hand, it must be pointed out that a perfect match with a single uniform pdf with unit support is not possible in practice, since the mean of the input distribution can not be exactly estimated from a finite training set. This implies that the learnt uniform would leave out of its support some of the data, which would lead to a test set ANLL\(=\infty\) (the worst possible ANLL). Hence, the fact that MMED does not output a single uniform is a proof of its robustness.
Figure 3. Learning the uniform distribution on [0,1]. MMED with one group (test ANLL=0.0185), uniform distribution (test ANLL=0).
Figure 4. Learning the uniform distribution on [0,1]. MMED with 3 groups (test ANLL= 0.0214), uniform distribution (test ANLL=0).

We have repeated the experiments for the bidimensional uniform distribution on the unit square \([0,1] \times [0,1]\). The results are shown in Figure 5; the unit square has been plotted in black as a reference. As before, the MMED models adjust tightly to the input data, which is confirmed by their test ANLL. Please note that for the unit square there is no possibility of a perfect fit, since all the MMED pdfs are ellipse shaped.

Figure 5. Learning the uniform distribution on the unit square (shown in white). MMED with 1 group (left, test ANLL=0.1971), MMED with 3 groups (right, test ANLL=0.1178). The graphical scale of log-densities is shown below.

5.2 Pdf families experiments

Next we present a series of experiments designed to gain certain insight into the relationships among the components. We are interested in the relative contributions of each family (Gaussian, Exponential Square Root, Laplacian, Student-t, Uniform and Triangular), measured as their posterior responsibilities for having generated regular
and outlying test samples. We also compare the posterior responsibilities of the components of the same family, but different parameters (for example, Student-t components with different values of $v$). In this set of experiments, we have not considered MFullCov or MPPCA because they do not use mixture components from different pdf families.

We have selected the ‘Contraceptive’ dataset from the UCI Machine Learning Repository (Asuncion & Newman, 2007), which has dimension $D=9$. The data was randomly split into 10 disjoint subsets. Then one of the subsets was used as the test set (10% of the original dataset), and the remaining nine subsets formed the training set (90%). In addition to this, we have generated an outlier test set by multiplying each component of each sample of the original test set by a random number from the interval $[-10,10]$. This time we have used a MMED architecture with 5 groups. The other parameters remain the same as in the previous subsection.
Figure 6. Posterior responsibilities of the Student-t components for an original test sample and an outlier test sample.

Table 1 shows the posterior responsibilities of each family for having generated the original and outlier test samples. It can be inferred that Student-t, Uniform and Triangular families are the best suited for the original samples, while the outliers are adequately modeled by Student-t pdfs.

Figure 6 shows the posterior responsibilities of the Student-t components \( i \) classified by its \( \nu \) parameter, that is, \( P(\text{\( i \)} \text{ is Student-t with parameter } \nu | \text{ t} ) \) versus \( \nu \). For the outlier, the posterior responsibility is concentrated in the components with \( \nu=3 \), which corresponds to a heavy tailed pdf. On the other hand, the posterior responsibility for the original sample attains its maximum at \( \nu=15 \), which is more similar to a Gaussian and less heavily tailed. Furthermore, the posterior responsibility does not vanish for \( \nu \rightarrow \infty \), i.e., the Gaussian limit. Hence, the farther the test sample is from the distribution center, the more important the components with low \( \nu \) become.

Figure 7 shows the posterior responsibilities of the Uniform components \( i \) classified by its \( \delta^2 \) parameter, that is, \( P(\text{\( i \)} \text{ is Uniform with parameter } \delta^2 | \text{ t} ) \) versus \( \delta^2 \). The posterior responsibility is zero for the components with low \( \delta^2 \) parameter, because the test sample does not belong to the region where the uniform pdf is nonzero. The first values of \( \delta^2 \) for which the original test sample belongs to the nonzero region have the highest posterior responsibility. This is because higher values of \( \delta^2 \) spread the probability mass in a higher volume. The posterior responsibilities of the Uniform components are much lower in the outlier, because the heavy tails of the Student-t components explain the outliers much better than the components of all the other families. Finally, there is a steep increase in the outlier responsibility for the mixture component with the highest \( \delta^2 \). This is because that component has been assigned a
relatively higher a priori probability in the training process, since it models the outlying training data well.

Finally, Figure 8 depicts the situation with the Triangular family. It looks similar to Figure 7, but the probability density for the Triangular components is much higher than that of the Uniform components for the original test sample (please note the log scale in the y-axis). This is because the original test sample is not far from the distribution center, and the Triangular family concentrates more probability mass near the center than the Uniform family.

![Figure 7](image_url)

**Figure 7.** Posterior responsibilities of the Uniform components for an original test sample and an outlier test sample.
Figure 8. Posterior responsibilities of the Triangular components for an original test sample and an outlier test sample.

<table>
<thead>
<tr>
<th></th>
<th>Original test set</th>
<th>Outliers test set</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P(\text{Gaussians} \mid t) )</td>
<td>0.044492 (0.002561)</td>
<td>1.8753E–152 (2.89E–153)</td>
</tr>
<tr>
<td>( P(\text{ExpSquareRoots} \mid t) )</td>
<td>3.07764E–5 (2.8162E–6)</td>
<td>2.80109E–19 (9.12E–19)</td>
</tr>
<tr>
<td>( P(\text{Laplacians} \mid t) )</td>
<td>4.23870E–49 (5.710E–49)</td>
<td>4.646250E–51 (4.18E–50)</td>
</tr>
<tr>
<td>( P(\text{Students} \mid t) )</td>
<td>0.71039 (0.12912)</td>
<td>\approx1 (2.71E–11)</td>
</tr>
<tr>
<td>( P(\text{Uniforms} \mid t) )</td>
<td>9.73497E–4 (3.1270E–4)</td>
<td>9.02927E–98 (3.11E–99)</td>
</tr>
<tr>
<td>( P(\text{Triangulars} \mid t) )</td>
<td>0.244112 (0.07167)</td>
<td>1.04704E–97 (2.01E–97)</td>
</tr>
</tbody>
</table>

Table 1. Posterior responsibilities of each family of having generated original test samples and outliers. The standard deviations are shown in parentheses.

5.3 Density estimation experiments
We have selected some standard benchmark databases of different application domains to test the probability density estimation performance of our proposal from the UCI Machine Learning Repository (Asuncion & Newman, 2007). We have considered MMED mixtures with 18 components per group, and we have tried different numbers of groups (from 1 to 5). We have trained each MMED mixture for 100,000 steps. In addition to this, MFullCov and MPPCA models with 1 to 5 mixture components have also been tested. For the MPPCA model we have run different simulations for every possible value of the number of principal components parameter $q=1,...,D-1$ (the same for all mixture components), and for each database the results corresponding to the best choice of $q$ are shown.

We have run a 10-fold cross-validation, with disjoint training, validation and test sets for each of the model setups considered above. That is, for each setup the data was randomly split into 10 disjoint subsets. Then for each of the 10 folds, one of the subsets was used as the validation set (10% of the original dataset), other subset was used as test set (10%), and the remaining eight subsets formed the training set (80%). Three performance measures have been considered. First, the Average Negative Log-Likelihood (ANLL) has been computed on the test set (we repeat the equation for convenience):

$$ANLL = -\frac{1}{Z} \sum_{n=1}^{Z} \log p(t_n)$$

where $Z$ is the number of test samples. We have also used the Akaike’s Information Criterion, AIC (Akaike, 1974):

$$AIC = 2q - 2 \sum_{n=1}^{Z} \log p(t_n)$$

where $q$ is the number of free parameters of the considered probabilistic model. Finally we have computed the Bayesian Information Criterion (BIC) or Schwarz Information Criterion (Schwarz, 1978):
Mixtures of Multivariate Elliptical Distributions

\[ BIC = q \log Z - 2 \sum_{n=1}^{Z} \log p(t_n) \]  \hspace{1cm} (48)

Please note that for the three criteria (ANLL, AIC and BIC) the model with the lower value of the measure is the one to be preferred. We have used the T-test to check the statistical significance of the advantage of the two best performing approaches for each database and performance criterion. We have considered that the difference is statistically significant if we have less than 5% probability that the difference between the means is caused by chance.

<table>
<thead>
<tr>
<th>Database</th>
<th>MMED</th>
<th>MFullCov</th>
<th>MPPCA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Glass</td>
<td>−2.833 (2.516)</td>
<td>−2.700 (0.013)</td>
<td><strong>−6.275 (1.866)</strong></td>
</tr>
<tr>
<td>Ionosphere</td>
<td>−2.431 (3.338)*</td>
<td>19.710 (8.178)</td>
<td><strong>−4.076 (5.217)</strong>*</td>
</tr>
<tr>
<td>Liver</td>
<td><strong>21.025 (0.568)</strong>*</td>
<td>21.353 (0.516)</td>
<td>21.433 (0.517)*</td>
</tr>
<tr>
<td>Pima</td>
<td><strong>24.469 (0.858)</strong></td>
<td>29.415 (0.353)</td>
<td>26.236 (0.608)</td>
</tr>
<tr>
<td>Yeast</td>
<td>−15.306 (4.533)</td>
<td>−8.022 (0.598)</td>
<td>−11.650 (0.012)</td>
</tr>
<tr>
<td>Vowel</td>
<td>7.466 (0.159)</td>
<td><strong>6.972 (0.441)</strong>*</td>
<td>7.143 (0.250)*</td>
</tr>
<tr>
<td>BalanceScale</td>
<td>5.463 (0.591)*</td>
<td><strong>4.527 (3.934)</strong>*</td>
<td>62.614 (0.131)</td>
</tr>
<tr>
<td>BreastCancWis</td>
<td><strong>8.097 (2.731)</strong></td>
<td>18.421 (0.868)</td>
<td>13.080 (1.532)</td>
</tr>
<tr>
<td>Contraceptive</td>
<td><strong>5.615 (2.468)</strong></td>
<td>11.365 (0.167)</td>
<td>11.061 (0.032)</td>
</tr>
<tr>
<td>TAE</td>
<td><strong>7.363 (1.446)</strong></td>
<td>11.526 (0.401)</td>
<td>10.382 (0.021)</td>
</tr>
<tr>
<td>Wine</td>
<td>18.892 (1.082)</td>
<td>19.063 (1.548)*</td>
<td><strong>18.831 (1.333)</strong>*</td>
</tr>
<tr>
<td>Servo</td>
<td>4.022 (1.794)*</td>
<td><strong>3.701 (5.145)</strong>*</td>
<td>5.844 (0.786)</td>
</tr>
<tr>
<td>HayesRoth</td>
<td><strong>2.799 (1.192)</strong></td>
<td>4.874 (0.609)</td>
<td>5.387 (0.447)</td>
</tr>
<tr>
<td>Haberman</td>
<td><strong>7.966 (0.665)</strong></td>
<td>8.673 (0.127)</td>
<td>9.204 (0.366)</td>
</tr>
<tr>
<td>Iris</td>
<td>1.831 (0.531)</td>
<td><strong>1.502 (0.520)</strong>*</td>
<td>1.772 (0.506)*</td>
</tr>
<tr>
<td>SpaceShuttle1</td>
<td>8.767 (1.291)*</td>
<td><strong>6.706 (1.668)</strong>*</td>
<td>8.876 (5.155)</td>
</tr>
<tr>
<td>SpaceShuttle2</td>
<td>8.459 (13.685)</td>
<td><strong>5.259 (7.296)</strong>*</td>
<td>7.776 (3.673)*</td>
</tr>
</tbody>
</table>
Table 2. ANLL (test set). The standard deviations for the 10 runs are shown in parentheses. The asterisks (*) indicate that difference between the two best performing approaches is not statistically significant for that database.

<table>
<thead>
<tr>
<th>Database</th>
<th>MMED</th>
<th>MFullCov</th>
<th>MPPCA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Glass</td>
<td>167.59 (64.27)</td>
<td>103.22 (0.10)*</td>
<td>41.38 (98.04)*</td>
</tr>
<tr>
<td>Ionosphere</td>
<td>1473.31 (263.87)</td>
<td>2718.01 (692.51)</td>
<td>712.64 (400.92)</td>
</tr>
<tr>
<td>Liver</td>
<td>1584.81 (248.59)*</td>
<td>1646.18 (321.45)*</td>
<td>1646.83 (208.23)</td>
</tr>
<tr>
<td>Pima</td>
<td>4147.34 (414.15)</td>
<td>4613.49 (367.74)</td>
<td>4606.61 (595.44)</td>
</tr>
<tr>
<td>Yeast</td>
<td>–4369.46 (1514.45)</td>
<td>–2299.21 (339.44)</td>
<td>–3017.03 (0.21)</td>
</tr>
<tr>
<td>Vowel</td>
<td>1812.66 (274.71)</td>
<td>2048.72 (175.23)</td>
<td>2063.11 (152.40)</td>
</tr>
<tr>
<td>BalanceScale</td>
<td>868.47 (112.28)*</td>
<td>704.42 (471.68)*</td>
<td>140850.3 (2851.32)</td>
</tr>
<tr>
<td>BreastCancWis</td>
<td>1594.63 (394.42)</td>
<td>2621.73 (231.67)</td>
<td>1960.32 (322.36)</td>
</tr>
<tr>
<td>Contraceptive</td>
<td>2182.13 (679.20)</td>
<td>3458.43 (277.16)</td>
<td>3767.43 (0.75)</td>
</tr>
<tr>
<td>Dermatology</td>
<td>3408.76 (302.38)*</td>
<td>3532.19 (342.17)*</td>
<td>4220.16 (758.60)</td>
</tr>
<tr>
<td>TAE</td>
<td>370.17 (88.49)*</td>
<td>390.16 (53.40)*</td>
<td>530.22 (0.00)</td>
</tr>
<tr>
<td>Wine</td>
<td>955.56 (170.46)</td>
<td>1106.22 (203.77)</td>
<td>1093.74 (139.30)</td>
</tr>
<tr>
<td>Servo</td>
<td>309.43 (72.02)*</td>
<td>221.80 (184.43)*</td>
<td>399.58 (44.32)</td>
</tr>
<tr>
<td>HayesRoth</td>
<td>178.25 (20.52)*</td>
<td>226.34 (13.87)</td>
<td>162.47 (31.84)*</td>
</tr>
<tr>
<td>Haberman</td>
<td>568.02 (119.06)</td>
<td>636.39 (13.62)</td>
<td>650.92 (97.89)</td>
</tr>
<tr>
<td>Iris</td>
<td>117.98 (28.64)</td>
<td>139.59 (28.42)</td>
<td>139.47 (19.49)</td>
</tr>
<tr>
<td>SpaceShuttle1</td>
<td>83.379 (31.089)</td>
<td>63.132 (18.002)*</td>
<td>81.782 (25.043)*</td>
</tr>
<tr>
<td>SpaceShuttle2</td>
<td>80.675 (24.658)</td>
<td>91.229 (45.757)*</td>
<td>77.304 (23.646)*</td>
</tr>
</tbody>
</table>

Table 3. AIC (test set). The standard deviations for the 10 runs are shown in parentheses. The asterisks (*) indicate that difference between the two best performing approaches is not statistically significant for that database.
The results for the considered databases are presented in Tables 2, 3 and 4. Please note that we present the results of the best performing model setup for each of the four models. The best approach for each database and performance criterion is marked in **bold**. On the other hand, CPU time results are in Figure 7.

The ANLL results (Table 2) indicate that the more flexible models (MFullCov and MMED) achieve a better adaptation to the complex features of these datasets than those that restrict the covariance matrix (MPPCA). On the other hand, when we take into account the complexity of the model by considering AIC and BIC (Tables 3 and 4), MPPCA approaches MFullCov, which has many free parameters. Nevertheless, MMED continues giving adequate results because of its parameter saving by mean vector and covariance matrix sharing. Hence, MMED can be regarded as a good compromise between model adaptability and complexity when the effects of the $O(D^2)$ growth of the covariance matrix size are noticeable. This is further discussed in subsection 5.5.

<table>
<thead>
<tr>
<th>Database</th>
<th>MMED</th>
<th>MFullCov</th>
<th>MPPCA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Glass</td>
<td>231.63 (63.01)</td>
<td>224.32 (0.12)*</td>
<td><strong>215.09 (81.28)</strong>*</td>
</tr>
<tr>
<td>Ionosphere</td>
<td>2452.90 (325.08)</td>
<td>3697.22 (791.78)</td>
<td><strong>1542.21 (367.11)</strong></td>
</tr>
<tr>
<td>Liver</td>
<td><strong>1639.56 (254.45)</strong>*</td>
<td>1776.80 (340.61)*</td>
<td>1778.34 (222.24)</td>
</tr>
<tr>
<td>Pima</td>
<td><strong>4442.70 (535.80)</strong>*</td>
<td>4718.77 (371.31)*</td>
<td>5124.89 (625.46)</td>
</tr>
<tr>
<td>Yeast</td>
<td><strong>–4069.76 (1509.93)</strong></td>
<td>–2164.37 (335.81)</td>
<td>–2406.98 (0.08)</td>
</tr>
<tr>
<td>Vowel</td>
<td>1985.83 (284.88)</td>
<td>2913.67 (208.22)</td>
<td>2902.59 (182.09)</td>
</tr>
<tr>
<td>BalanceScale</td>
<td>953.51 (110.66)*</td>
<td><strong>872.42 (468.97)</strong>*</td>
<td>144656.2 (2865.6)</td>
</tr>
<tr>
<td>BreastCancWis</td>
<td><strong>2084.66 (427.63)</strong>*</td>
<td>2743.78 (236.70)</td>
<td>2144.45 (329.99)*</td>
</tr>
<tr>
<td>Contraceptive</td>
<td>2905.05 (763.59)</td>
<td>3622.85 (281.50)</td>
<td>4151.04 (0.01)</td>
</tr>
<tr>
<td>Dermatology</td>
<td><strong>4404.59 (378.09)</strong>*</td>
<td>4521.55 (417.37)*</td>
<td>6048.81 (875.83)</td>
</tr>
<tr>
<td>TAE</td>
<td><strong>403.97 (107.41)</strong>*</td>
<td>404.97 (56.27)*</td>
<td>606.70 (0.03)</td>
</tr>
<tr>
<td>Wine</td>
<td><strong>1047.72 (192.28)</strong></td>
<td>1284.73 (260.79)</td>
<td>1276.02 (176.52)</td>
</tr>
</tbody>
</table>
Mixtures of Multivariate Elliptical Distributions

<table>
<thead>
<tr>
<th></th>
<th>Servo</th>
<th>HayesRoth</th>
<th>Haberman</th>
<th>Iris</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>329.614 (78.323)*</td>
<td>188.312 (23.125)*</td>
<td><strong>616.42 (126.85)</strong>*</td>
<td><strong>131.11 (32.00)</strong></td>
</tr>
<tr>
<td></td>
<td><strong>257.18 (188.71)</strong>*</td>
<td>247.19 (18.45)</td>
<td>684.28 (21.45)*</td>
<td>170.27 (41.38)</td>
</tr>
<tr>
<td></td>
<td>481.71 (62.72)</td>
<td><strong>168.10 (33.65)</strong>*</td>
<td>712.81 (105.75)</td>
<td>168.85 (31.44)</td>
</tr>
</tbody>
</table>

Table 4. BIC (test set). The standard deviations for the 10 runs are shown in parentheses. The asterisks (*) indicate that difference between the two best performing approaches is not statistically significant for that database.

Figure 9. Overall CPU time (hours). This includes all the considered databases, and the 10 runs for each model setup.

5.4 Missing values experiments

We have designed another set of experiments to test the ability of our approach to be trained with incomplete data and its ability to estimate missing values in test data. We have compared the same three approaches considered in the previous subsection, with the same parameter selections.

The databases and parameter selections have been the same as in the density estimation experiments. We have run 10-fold cross-validations, with disjoint training, validation and test sets. The missing value estimation performance has been tested by using a 5% of missing values in the test data, where the missing values have been
distributed uniformly among the components of the test vectors. The performance measure has been the mean squared error of the estimated (reconstructed) vector $\hat{t}_i$ with respect to the true vector $t_i$ (the real complete test data):

$$MSE = \frac{1}{Z} \sum_{n=1}^{Z} |t_i - \hat{t}_i|^2$$

(49)

As in the previous subsection, we have used the T-test to check the statistical significance of the advantage of the two best performing approaches for each database. We have considered that the difference is statistically significant if we have less than 5% probability that the difference between the means is caused by chance.

The results are presented in Table 5. We present the results of the best performing model setup for each of the four models. The best approach for each database is marked in **bold**. On the other hand, Figure 10 presents the CPU time requirements, with results similar to those of the density estimation experiments.

In this set of experiments, the missing value estimation performance is not enhanced by a more parsimonious modelling, but rather by a more exact representation of the input distribution. Hence, in the line of the ANLL results of the previous subsection, the models which capture the finer details of the input distributions (MFullCov and MMED) yield better results. Moreover, MMED has an advantage over MFullCov, since MMED is able to cope with non Gaussian input data more naturally given the wide range of probability density families it integrates.

<table>
<thead>
<tr>
<th>Database</th>
<th>MMED</th>
<th>MFullCov</th>
<th>MPPCA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Glass</td>
<td>0.00192 (0.00434)</td>
<td>0.00022 (0.00005)</td>
<td>0.06400 (0.04261)</td>
</tr>
<tr>
<td>Ionosphere</td>
<td><strong>0.00739 (0.00437)</strong></td>
<td>0.00793 (0.00417)*</td>
<td>0.01066 (0.00332)</td>
</tr>
<tr>
<td>Liver</td>
<td><strong>9.411 (14.917)</strong>*</td>
<td>28.139 (47.362)</td>
<td>10.579 (12.243)*</td>
</tr>
<tr>
<td>Pima</td>
<td><strong>39.98 (22.45)</strong>*</td>
<td>69.86 (76.20)*</td>
<td>99.85 (83.58)</td>
</tr>
</tbody>
</table>
Table 5. MSE (test set with 5% missing values). The standard deviations for the 10 runs are shown in parentheses. The asterisks (*) indicate that difference between the two best performing approaches is not statistically significant for that database.

![CPU Time (hours) chart](chart.png)

Figure 10. Overall CPU time (hours). This includes all the considered databases, and the 10 runs for each model setup.
5.5 Final analysis

The experimental results we have just presented reveal the MMED model with stochastic approximation learning as a possible alternative to other mixture learning methods. Its capability to include a wide range of elliptical densities allows it to adapt tightly to the input distribution (Subsection 5.1). In Subsection 5.2 it has been shown that we can incorporate mixture components which are members of the same family with different values of the parameters, so that it is possible to assess which are the most likely values of these parameters, without the need of selecting a single value of each parameter. Furthermore, the same study can be carried out to compare different pdf families by its responsibility for having generated the input data.

Flexibility always comes at a cost, which is model complexity. Nevertheless, the MMED model tries to minimize this effect by sharing the mean vector and the covariance matrix. This means that the \((D^2+3D)/2\) free parameters corresponding to \(\mu\) and \(C\) are not replicated on each mixture component, since they are common to all components in the same GMC. As \(D\) grows, the amount of saved free parameters increases dramatically. In fact, covariance sharing is a well known technique for reducing the number of free parameters of probabilistic models (Rosti & Gales, 2004; Bandiera et al., 2007). In subsection 5.3, this technique allows MMED to achieve good performance not only with respect to ANLL, which does not take into account model complexity, but also with respect to AIC and BIC, which provide a balance between accuracy and complexity. Finally, subsection 5.4 illustrates the suitability of our proposal to the missing value estimation problem, an issue which arises in many applications (Little & Rubin, 2002).

As seen in this section, the relative merits of the considered approaches depend on the particular application at hand. Nevertheless, our model has the capability to
accommodate very different pdf families with parsimony, which is a distinctive feature that makes it suitable for problems where the data do not fit easily to a single pdf family. The particular choice of pdf families and mixture components to be incorporated to the MMED model is an issue to be addressed on a one by one basis by the practitioner, given the expert knowledge of the problem in each case. Therefore, the choices considered here are useful to illustrate the potential of the model, but a strict adherence to them is not advisable in any way.

**Conclusions**

We have proposed a probabilistic mixture model which considers components from different families of distributions in the same mixture. A learning method has been developed for this class of mixtures, which is based on stochastic approximation. It allows training a wide class of mixture components with minimal changes to the training algorithm. Hence, the flexibility of the model is greatly increased with respect to models with components from a single family. We have presented experimental results to illustrate the novel features of our proposal, and its performance in probability density and missing value estimation applications.

**Acknowledgements**

This work has been partially supported by the Ministry of Science and Innovation of Spain under grant TIN2010-15351, project name ‘Probabilistic self organizing models for the restoration of lossy compressed images and video’.
Appendix A: Multivariate Triangular probability distribution

Next we prove that the function (12) is a proper probability density function for all parameter values $\delta^2 > 0$, i.e. it integrates to one. We must evaluate the integral of the function over the whole input space:

$$ I = \int p(t) dt = \int g_{\text{Triangular}}(\alpha) dt $$

$$ = K_{\text{Triangular}} \int (\delta - r) dt $$

(A.1)

where we note

$$ K_{\text{Triangular}} = \frac{1}{2} \pi^{-D/2} \delta^{D-1} \left( \frac{1}{D} - \frac{1}{D+1} \right)^{-1} |C|^{-1/2} \Gamma \left( \frac{D}{2} \right) $$

(A.2)

for the sake of brevity. The integral (A.1) is best computed if we change from hyperelliptical coordinates to hyperspherical coordinates centered in $\mu$; see López-Rubio (2009) for an analogous derivation. The coordinate change is

$$ y = A(t - \mu) $$

(A.3)

where $A$ is the Cholesky decomposition matrix of $C^{-1}$, and hence we have

$$ C^{-1} = A^T A $$

(A.4)

The integral now reads

$$ I = |C|^{1/2} K_{\text{Triangular}} \int (\delta - r) dy $$

(A.5)

where the Mahalanobis distance $r$ is expressed as the standard Euclidean distance in the new coordinate system:

$$ r = \sqrt{y^T y} $$

(A.6)

A hyperspherical volume element of thickness $dr$ has volume

$$ dy = \frac{2\pi^{D/2}}{\Gamma \left( \frac{D}{2} \right)} r^{D-1} dr $$

(A.7)
Hence we get

$$I = \frac{2K_{\text{triangular}} |C|^{1/2} \pi^{D/2}}{\Gamma(D/2)} \int_0^\delta (\delta - r)r^{D-1}dr$$  \hspace{1cm} (A.8)$$

On the other hand, we have:

$$\int_0^\delta (\delta - r)r^{D-1}dr = \delta^{D+1}\left(\frac{1}{D} - \frac{1}{D+1}\right)$$  \hspace{1cm} (A.9)$$

And then we substitute (A.9) into (A.8) to yield

$$I = 1$$  \hspace{1cm} (A.10)$$

as desired.

---

**Appendix B: Stochastic approximation**

This Appendix has two parts. First we apply Robbins-Monro stochastic approximation method to the MMED model, so that an update algorithm is obtained. Then we present a proposition which proves its convergence.

Let $\Theta_i = (\pi_i, \mu_i, C_i)$ be a vector comprising the parameters for mixture component $i$. Let $\varphi(\Theta_i, t)$ be an arbitrary function of $\Theta_i$ and the input sample $t$. Then we define the weighted mean of $\varphi(\Theta_i, t)$ as:

$$\langle \varphi \rangle_i = E[P(i | t)\varphi(\Theta_i, t)]$$  \hspace{1cm} (B.1)$$

This allows us to express the conditional expectation of $\varphi(\Theta, t)$ as follows:

$$\frac{\langle \varphi \rangle_i}{\langle 1 \rangle_i} = \frac{E[P(i | t)\varphi(\Theta_i, t)]}{E[P(i | t)]} = \frac{\int p(t)P(i | t)\varphi(\Theta_i, t)dt}{\pi_i}$$

$$= \int \frac{p(t)}{\pi_i} \varphi(\Theta_i, t)dt = \int p(t | i)\varphi(\Theta_i, t)dt = E[\varphi(\Theta_i, t) | i]$$  \hspace{1cm} (B.2)$$

Therefore we can rewrite the mixture parameters in terms of the weighted means $\langle \varphi \rangle_i$:
Mixtures of Multivariate Elliptical Distributions

\[ \pi_i = \frac{1}{i} \]  \hspace{1cm} (B.3)

\[ \mu_i = \frac{t_i}{1_i} \]  \hspace{1cm} (B.4)

\[ C_i = \frac{(t_i - \mu_i)(t_i - \mu_i)^T}{1_i} \]  \hspace{1cm} (B.5)

If we have \( n \) samples (finite case), the linear least squares approximation for \( \langle \varphi \rangle_i \) is:

\[ \langle \varphi \rangle_i = \frac{1}{n} \sum_{j=1}^{n} P(i \mid t_j) \varphi(\Theta, t_j) = \frac{1}{n} \sum_{j=1}^{n} R_{ji} \varphi(\Theta, t_j) \]  \hspace{1cm} (B.6)

where we should remember that the responsibility of mixture component \( i \) for having generated the sample \( t_j \) is noted

\[ R_{ji} = P(i \mid t_j) \]  \hspace{1cm} (B.7)

As \( n \to \infty \), the approximation of (B.6) converges to the exact value given by (B.1). Next we apply the Robbins-Monro stochastic approximation algorithm (see Kushner & Yin, 2003) to estimate iteratively the value of the weighted means \( \langle \varphi \rangle_i \):\n
\[ \langle \varphi \rangle_i(0) = P(i \mid t_0) \varphi(\Theta, t_0) \]  \hspace{1cm} (B.8)

\[ \langle \varphi \rangle_i(n) = \langle \varphi \rangle_i(n-1) + \varepsilon(n)[P(i \mid t_n)\varphi(\Theta, t_n) - \langle \varphi \rangle_i(n-1)] \]  \hspace{1cm} (B.9)

where \( \varepsilon(n) \) is the step size, which must satisfy the following conditions in order to guarantee convergence of the Robbins-Monro method:

\[ \varepsilon(n) > 0, \lim_{n \to \infty} \varepsilon(n) = 0, \sum_{n=1}^{\infty} \varepsilon(n) = \infty, \sum_{n=1}^{\infty} \varepsilon^2(n) < \infty \]  \hspace{1cm} (B.10)

In order to fulfill these requirements, \( \varepsilon(n) \) is typically selected as

\[ \varepsilon(n) = \frac{1}{an + b}, 0 < a < 1 \]  \hspace{1cm} (B.11)

On the other hand, equation (B.9) is more conveniently written as:
Mixtures of Multivariate Elliptical Distributions

\[
\langle \varphi \rangle_i(n) = (1 - \varepsilon(n))\langle \varphi \rangle_i(n-1) + \varepsilon(n)R_{ni}\varphi(\Theta_i, t_n)
\]  
(B.12)

Now we derive an online stochastic approximation algorithm by applying equation (B.12) to equations (B.3)-(B.5). First we need to define two auxiliary variables:

\[
m_i = \langle t \rangle_i
\]  
(B.13)

\[
M_i = \langle (t - \mu_i)(t - \mu_i)^\top \rangle_i
\]  
(B.14)

The corresponding update equations are:

\[
m_i(n) = (1 - \varepsilon(n))m_i(n-1) + \varepsilon(n)R_{ni}t_n
\]  
(B.15)

\[
M_i(n) = (1 - \varepsilon(n))M_i(n-1) + \varepsilon(n)R_{ni}(t_n - \mu_i(n))(t_n - \mu_i(n))^\top
\]  
(B.16)

Then we are ready to rewrite (B.3)-(B.5):

\[
\pi_i(n) = (1 - \varepsilon(n))\pi_i(n-1) + \varepsilon(n)R_{ni}
\]  
(B.17)

\[
\mu_i(n) = \frac{m_i(n)}{\pi_i(n)}
\]  
(B.18)

\[
C_i(n) = \frac{M_i(n)}{\pi_i(n)}
\]  
(B.19)

**Proposition:** If (B.10) holds, then the stochastic approximation algorithm of (B.15)-(B.19) converges to the exact solution of (B.3)-(B.5).

**Proof:**

The general form of the Robbins-Monro stochastic algorithm is

\[
\theta(n + 1) = \theta(n) + \varepsilon(n)Y(n)
\]  
(B.20)

where, in our case we take

\[
\theta(n) = \langle \langle \varphi \rangle_i(n) \rangle_{\theta_j}
\]  
(B.21)

That is, we include all the weighted means in the current estimation vector \( \theta(n) \). We also take

\[
Y(n) = \xi(n) - \theta(n)
\]  
(B.22)
where the new data to be incorporated into the estimation is

\[ \xi(n) = (P(i | t_n, \Theta(n))\phi(\Theta(n), t_n))_{\varphi,i} \]  

(B.23)

In the above equation, \( \Theta=(\pi_i, \mu_i, C_i) \) is the complete parameter vector of the mixture model. The goal of the stochastic algorithm is to find a root of the equation

\[ \bar{g}(\theta(n)) = 0 \]  

(B.24)

where

\[ g(\theta(n)) = (E[P(i | t, \Theta(n))\phi(\Theta(n), t)] - \theta(\varphi,i, n))_{\varphi,i} \]  

(B.25)

In order to prove the convergence of the algorithm, we are going to prove that the “noise” in the observation \( Y_n \) is a martingale difference (see Kushner & Yin, 2003). That is, there is a function \( g_n(\cdot) \) of \( \theta \) such that

\[ E[Y_n | \theta(0), Y_j, j < n] = g_n(\theta(n)) \]  

(B.26)

This is readily verified:

\[ g_n(\theta(n)) = (E[P(i | t, \Theta(n))\phi(\Theta(n), t)] - \theta(\varphi,i, n))_{\varphi,i} \]  

(B.27)

Note that \( g_n(\theta(n)) \) only depends on \( n \) and \( \theta(n) \), because \( \Theta(n) \) can be computed from \( \theta(n) \). So we have that

\[ Y_n = g_n(\theta(n)) + \delta M_n \]  

(B.28)

where \( \delta M_n \) is a martingale difference. The associated “bias” process is defined as:

\[ \beta_n = E[Y_n | \theta(0), Y_j, j < n] - g(\theta(n)) \]  

(B.29)

We can guarantee convergence by proving the following three conditions:

\[ \sup_n E[\|Y_n\|^2] < \infty \]  

(B.30)

\[ \lim_{n \to \infty} \beta_n = 0 \]  

(B.31)

\[ \forall m > 0 \forall \theta, \lim_{n \to \infty} \left| \sum_{j=n}^{n+m} g_j(\theta) - g(\theta) \right| = 0 \]  

(B.32)
Mixtures of Multivariate Elliptical Distributions

Next we examine the first condition:

\[
E[Y_n] = E[\|\zeta(n) - \theta(n)\|^2] = \sum_{\varphi,j} E[P(i | t, \Theta(n))\varphi(\Theta_i(n), t) - \langle \varphi \rangle_i(n)]^2 
\]  

(B.33)

The above value is finite if we assume that the input distribution has a compact support. This assumption should hold in practice, since real data always appear in a finite domain. Alternatively we can relax this assumption by considering that the input density decreases exponentially as \(\|t\| \to \infty\).

Now we study the second condition:

\[
\lim_{n \to \infty} \beta_n = \lim_{n \to \infty} (E[Y_n | \theta(0), Y_j, j < n] - g(\theta(n))) = \lim_{n \to \infty} (E[P(i | t, \Theta(n))\varphi(\Theta_i(n), t)] - E[P(i | t, \Theta(n))\varphi(\Theta_i(n), t)])_{\varphi,j} = 0 
\]  

(B.34)

which comes from (B.21), (B.25) and (B.27). Finally, for the third condition we have

\[
\forall \theta, \lim_{j \to \infty} (g_j(\theta) - g(\theta)) = 
\lim_{j \to \infty} (E[P(i | t, \Theta(0))\varphi(\Theta_i, t)] - E[P(i | t, \Theta(0))\varphi(\Theta_i, t)])_{\varphi,j} = 0 
\]  

(B.35)

again from (B.21), (B.25) and (B.27). In turn this means that the limit

\[
\lim_{n \to \infty} \left\| \sum_{j=1}^{m+n} \varepsilon_j (g_j(\theta) - g(\theta)) \right\| 
\]  

(B.36)

is zero \(\forall m > 0 \forall \theta\) since it is the norm of a finite sum of two factors where both are zero.

Hence we have proved that the algorithm converges to a root of equation (B.24).

At convergence we have

\[
\langle \varphi \rangle_i(n) = E[P(i | t, \Theta(n))\varphi(\Theta_i(n), t)] \quad \forall \varphi \forall i 
\]  

(B.37)

which is equivalent to the true value of the weighted mean, given by (B.1). Therefore, (B.37) is the exact solution of (B.3)-(B.5).
Appendix C: Histogram analysis

For a certain histogram bin $j$ and mixture component $i$, we are interested in finding

$$b_j = P(\lambda_{j-1} \leq (t - \mu)^T C^{-1} (t - \mu) < \lambda_j | i) \quad (C.1)$$

From (40) we know that (C.1) can be rewritten as

$$b_j = \int_{\mathcal{F}_j} p(t | i) dt \quad (C.2)$$

As in Appendix A, the integral in (C.2) is best computed if we change from hyperelliptical coordinates to hyperspherical coordinates centered in $\mu$. We apply the coordinate change in (A.3) to (C.2) to get

$$b_j = |C|^{1/2} \int_{\mathcal{F}_j} p(r | i) dy \quad (C.3)$$

where we can write $p(r | i)$ instead of $p(t | i)$ because we are only considering elliptical distributions, whose densities depend on the data $t$ only through the Mahalanobis distance $r$.

Now, from (A.6) and (A.7) we get

$$b_j = \frac{2\pi^{D/2}}{\Gamma \left( \frac{D}{2} \right) } \int_{\mathcal{F}_j} |C|^{1/2} r^{D-1} p(r | i) dr \quad (C.4)$$

Finally, the one-dimensional integral in (C.4) can be easily computed by any numerical integration method. Please note that the $|C|^{1/2}$ factor must cancel with a corresponding $|C|^{-1/2}$ factor inside $p(r | i)$; otherwise $p(r | i)$ would not be a multivariate elliptical probability density function in the first place. We can avoid the computation of the improper integral associated to the last bin $b_{i,NBins}$ by remembering that the constants $b_j$ are probabilities:
Mixtures of Multivariate Elliptical Distributions

\[ b_{i,NBins} = 1 - \sum_{j=1}^{NBins-1} b_{ij} \]  
(C.5)

References


Mixtures of Multivariate Elliptical Distributions


