Bayesian Methods for Non-Gaussian Data Modeling and Applications

Tarek Elguebaly

A Thesis

in

The Concordia Institute

for

Information Systems Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of Master of Applied Science (Quality Systems Engineering) at

Concordia University

Montréal, Québec, Canada

November 2009

Your file Votre référence
ISBN: 978-0-494-67314-0
Our file Notre référence
ISBN: 978-0-494-67314-0

Abstract

Bayesian Methods for Non-Gaussian Data Modeling and Applications

Tarek Elguebaly

Finite mixture models are among the most useful machine learning techniques and are receiving considerable attention in various applications. The use of finite mixture models in image and signal processing has proved to be of considerable interest in terms of both theoretical development and in their usefulness in several applications. In most of the applications, the Gaussian density is used in the mixture modeling of data. Although a Gaussian mixture may provide a reasonable approximation to many real-world distributions, it is certainly not always the best approximation especially in image and signal processing applications where we often deal with non-Gaussian data.

In this thesis, we propose two novel approaches that may be used in modeling non-Gaussian data. These approaches use two highly flexible distributions, the generalized Gaussian distribution (GGD) and the general Beta distribution, in order to model the data. We are motivated by the fact that these distributions are able to fit many distributional shapes and then can be considered as a useful class of flexible models to address several problems and applications involving measurements and features having well-known marked deviation from the Gaussian shape. For the mixture estimation and selection problem, researchers have demonstrated that Bayesian approaches are fully optimal. The Bayesian learning allows the incorporation of prior knowledge in a formal coherent way that avoids overfitting problems. For this reason, we adopt different Bayesian approaches in order to learn our models parameters.

First, we present a fully Bayesian approach to analyze finite generalized Gaussian mixture models which incorporate several standard mixtures, such as Laplace and Gaussian. This approach evaluates the posterior distribution and Bayes estimators using a Gibbs sampling algorithm, and selects the number of components in the mixture using the integrated likelihood. We also propose a fully Bayesian approach for finite Beta mixtures learning using a Reversible Jump Markov Chain Monte Carlo (RJMCMC) technique which simultaneously allows cluster assignments, parameters estimation, and the selection of the optimal number of clusters. We then validate the proposed methods by applying them to different image processing applications.

Acknowledgements

I owe my deepest gratitude to my supervisor, Dr. Nizar Bouguila, for his continuous support and encouragement throughout my graduate studies. It was an honor for me to work with such a wonderful advisor.

I am indebted to many of my colleagues in the lab for their helpful suggestions during my two years study.

Finally, I would like to thank my family for unconditional support throughout my studies, your endless love and care always encourage me.

Table of Contents

List of Tables

List of Figures

CHAPTER 1

Introduction

Over the last decade, technological advances have led to an explosion of enormous data size. These data pose a challenge to standard statistical methods and have received much attention recently. The importance of finding a way to model and analyze data lie in their usefulness in wide range of applications such as Bioinformatics, image processing, and computer vision. In recent years a lot of different algorithms were developed in the aim of automatically learning to recognize complex patterns, and to make intelligent decisions based on observed data. Machine learning is a scientific discipline that is concerned with the design and development of algorithms that allow computers to change behavior based on data, such as from sensor data or databases. A major focus of machine learning research is to offer a principled approach for developing and studying automatic techniques capable of learning their parameters based on training data [1–4]. Machine learning and statistical pattern recognition have seen dramatic growth over the past few years, this is due to the fact that it can be applied in diverse areas such as engineering, medicine, computer science, psychology, neuroscience, physics, and mathematics. In many statistical applications the observed data can be seen as stemming from multiple populations. It is of interest to build a generic model, which allows us to combine the samples from different populations.

Mixture models are one of the machine learning techniques receiving considerable attention in different applications. They are an interesting and flexible model family. The different uses of mixture models include for example clustering and density estimation. Moreover, mixture models have been successfully used in various kinds of tasks such as modeling failure rate data, and clustering teaching behavior. Although mixture models have been applied in different areas, they have proven particular efficiency in quality control systems.

In telephone networks, for instance, mixture models were applied in order to evaluate and monitor speech quality [5]. For software quality prediction, mixture models are used as a tool for early prediction of fault-prone program modules [6, 7]. In Bioinformatics, the analysis of DNA microarray data sets can be important in order to diagnose and discover different types of diseases, the use of mixture models can lead to a better treatment of patients in high risk [8]. Mixture models can be finite or infinite [9, 10]. In this thesis, we are only interested in finite mixture models.

1.1 Finite Mixture models

Finite mixture models assume that each component comes from a probability distribution, P, which has a given weight p_j, $j = 1, ..., M$, where the sum of the weights of all components is equal to one and M represents the total number of components. Finite mixture models can be represented by

$$P(\vec{X}|\Theta) = \sum_{j=1}^{M} p_j P(\vec{X}|\Theta_j) \tag{1}$$

where p_j ($0 \leq p_j \leq 1$ and $\sum_{j=1}^{M} p_j = 1$) are the mixing proportions and $P(\vec{X}|\Theta_j)$, is the probability density function describing component j. The symbol Θ_j, $j = 1, ..., M$, represents the different parameters vectors of the mixture components. In order to use mixture models, three main problems have to be resolved: the choice of the probability density function (PDF), parameters estimation and the selection of the number of clusters.

1.1.1 Probability Density Function Selection

The selection of the PDF to be used for modeling the data is of a crucial importance, because it affects the capability of the mixture to represent the data shape. The wrong selection of PDF may force the mixture model to increase the number of components in order to model the data (i.e overfitting). In most of the applications, the Gaussian density is used in the mixture modeling of data. As a smooth, bell-shaped distribution that can be completely characterized by its mean and its standard deviation, the Gaussian is in general used and justified for asymptotic reasons (i.e the sample is supposed to be sufficiently large) [11]. Although a Gaussian mixture may provide a reasonable approximation to many real-world distributions, it is certainly not always the best approximation especially in image and signal processing applications where we often deal with small samples [12–15]. Indeed, there are many phenomena and applications for which the Gaussian model is not realistic (for instance, it is well-known that natural image clutter is generally non-Gaussian). In this thesis, we consider the generalized Gaussian distribution (GGD) and the general Beta distribution as they can be good alternatives to the Gaussian distribution thanks to their shape flexibility which allows the modeling of a large number of non-Gaussian signals [12, 16–19].

1.1.2 Parameters Learning

Parameter learning approaches are used in order to estimate the model parameters. This problem is not straightforward and many deterministic as well as Bayesian approaches have been proposed. In deterministic approaches, parameters are assumed as fixed and unknown, and inference is founded on the likelihood of the data. Despite the fact that deterministic approaches have dominated mixture models estimation due to their small computational time, many works have demonstrated that these methods have severe problems such as convergence to local maxima and their tendency to overfitt the data [11], especially, when data are sparse or noisy. With the computational tools evolution, researchers were encouraged to implement and use Bayesian Markov Chain Monte Carlo (MCMC) methods and techniques as an alternative approach [20, 21]. Bayesian methods consider parameters to be random, and to follow different prior distributions. These distributions are used to describe our knowledge before considering the data, as for updating our prior beliefs the likelihood is used. Please refer to [11, 22] for interesting and in depth discussions about the general Bayesian theory. In this thesis, we are interested in the application of MCMC methods for the estimation of the model parameters.

1.1.3 Selection of the number of components

An important issue in mixture modeling is the selection of the number of components. The usual tradeoff in model order selection problems arises: with too many components, the mixture may overfitt the data, while a mixture with too few components may not be flexible enough to approximate the true underlying model. Lack of knowledge about the number of clusters is a challenging problem in mixture modeling and considerable efforts already have been made to investigate this important aspect. The majority of the approaches that have been proposed separate the estimation and the selection of the number of components (i.e a certain criterion should be compared for different number of clusters) (see, for instance, [9, 10] for interesting discussions and comparisons between different criteria). In this thesis, we use two different methods in order to select the number of clusters. The first method, is used to compare different values of M and finally select the one that increases the marginal likelihood of the data. For this method, we employ two criteria: the Bayesian information criterion (BIC) and the Laplace approximation. The other method takes into account the fact that both estimation and selection problems are strongly related. In order to apply this method we are using the Reversible Jump MCMC (RJMCMC) to simultaneously estimate and select mixture models parameters.

1.2 Contributions

The contributions of this thesis are as follows:

☞ **A Bayesian Approach for Finite Generalized Gaussian Mixture Models Learning:** We implement a fully Bayesian approach to analyze GGM models. Our approach evaluates the posterior distribution and Bayes estimators using a Gibbs sampling algorithm, while for model selection uses the integrated likelihood. We then validate this novel approach by applying it to different image processing applications; while comparing it to different other approaches.

☞ **A Fully Bayesian Model Based on RJMCMC and Finite Beta Mixture:**
We propose a Bayesian model founded on the RJMCMC for the General Beta distribution. Our approach is able to select and estimate finite Beta mixture models simultaneously. This was reached by treating the number of clusters as a random variable having a prior distribution. We then demonstrate its effectiveness using synthetic mixture data, real data, and image texture classification and retrieval.

1.3 Thesis Overview

The organization of this thesis is as follows:

- ❑ The first Chapter contains an introduction to finite mixture models.
- ❑ In Chapter 2, we introduce a Bayesian model based on Gibbs sampling, integrated likelihood, and finite Generalized Gaussian mixture. We investigate the effectiveness of our model by comparing it to different Bayesian and deterministic methods in various image processing applications.
- ❑ In Chapter 3, we propose a fully Bayesian algorithm for Beta mixtures learning based on the RJMCMC technique. We study the capability of our model in texture classification and retrieval while comparing it to the Gaussian mixture model.
- ❑ In Conclusions, we summarize our contributions.

CHAPTER 2

Bayesian Learning of Finite Generalized Gaussian Mixture Models on Images

This chapter presents a fully Bayesian approach to analyze finite Generalized Gaussian mixture models which incorporate several standard mixtures, widely used in signal and image processing applications, such as Laplace and Gaussian. Our work is motivated by the fact that the Generalized Gaussian Distribution (GGD) can be applied on a wide range of data due to its shape flexibility which justifies its usefulness to model the statistical behavior of multimedia signals [23]. We present a method to evaluate the posterior distribution and Bayes estimators using a Gibbs sampling algorithm. For the selection of number of components in the mixture, we use the Laplace approximation and Bayesian information criterion. We validate the proposed method by applying it to: synthetic data, real datasets, texture classification and retrieval, and image segmentation; while comparing it to different other approaches.

2.1 Introduction

Finite mixtures are a flexible and powerful probabilistic tool for modeling data [9]. Mixture models are very useful in areas where statistical modeling of data is needed such as in signal and image processing, pattern recognition, bioinformatics, computer vision, and machine learning. As noted earlier, the three main problems in mixture modeling are the choice of the probability density function (PDF), the parameters and model learning. Many studies have shown that the GGD, can be a good alternative to the Gaussian thanks to its shape flexibility which allows the modeling of a large number of non-Gaussian signals [12, 16–18]. The GGD contains the Laplacian, the Gaussian and asymptotically the uniform distribution as special cases [24] and has been used, for instance, in [14, 25] to fit subband histograms, in [26] for multiresolution transmission of high-definition video, in [27] for subband decomposition of video, in [28] for buffer control, in [29–31] for texture classification and retrieval, in [32] for denoising applications, in [33, 34] for data and image compression, in [35] for edge modeling, in [36, 37] for image thresholding, in [38, 39] for speech modeling, in [40, 41] for video and image segmentation, in [42] for SAR images statistics modeling, and in [43] for multichannel audio resynthesis.

Several approaches have been considered in the past to estimate GGD's parameters such as moment estimation [27, 44, 45], entropy matching estimation [39, 46], and maximum likelihood estimation [12, 29, 44, 47, 48]. It is noteworthy that these approaches consider a single distribution. Concerning finite mixture models parameters estimation, some deterministic approaches have been proposed in the past for the estimation of finite generalized Gaussian mixture (GGM) models parameters (see, for instance, [40, 41]). To the best of our knowledge the learning techniques that have been proposed for the GGM are deterministic and then usually excessively sensitive to noise. Thus, we propose in this chapter a novel Bayesian approach to evaluate the posterior distribution of GGM and then learn its parameters using Gibbs sampling [49] for the estimation and the integrated likelihood for the selection of the optimal number of components. To validate our learning algorithm, we compare it to three different techniques: the expectation-maximization (EM) estimation of the GGM, the EM and Bayesian approaches for the Gaussian mixture (GM) using synthetic data, real datasets, and real world applications involving texture classification and retrieval, image segmentation, biomedical image analysis, and DNA spot detection

2.2 The Finite GGM and Bayesian Estimation

2.2.1 Finite GGM Model

If the random variable $x \in \mathbb{R}$ follows a GGD with parameters μ, α and β, then the density function is given by [25, 27]:

$$p(x|\mu, \alpha, \beta) = \frac{\beta\alpha}{2\Gamma(1/\beta)} e^{-(\alpha|x-\mu|)^{\beta}} \tag{1}$$

where $\alpha = \frac{1}{\sigma}\sqrt{\frac{\Gamma(3/\beta)}{\Gamma(1/\beta)}}$, $-\infty < \mu < \infty$, $\beta > 0$, and $\alpha > 0$, and $\Gamma(.)$ is the Gamma function given by: $\Gamma(x) = \int_0^\infty t^{x-1}e^{-t}dt$, $x > 0$. μ, α and β denote the distribution mean, the inverse scale parameter, and the shape parameter, respectively. The parameter β controls the shape of the pdf. The larger the value, the flatter the pdf; and the smaller the value, the more picked the pdf. This means that β determines the decay rate of the density function (see Fig. 2.1). Note that for the

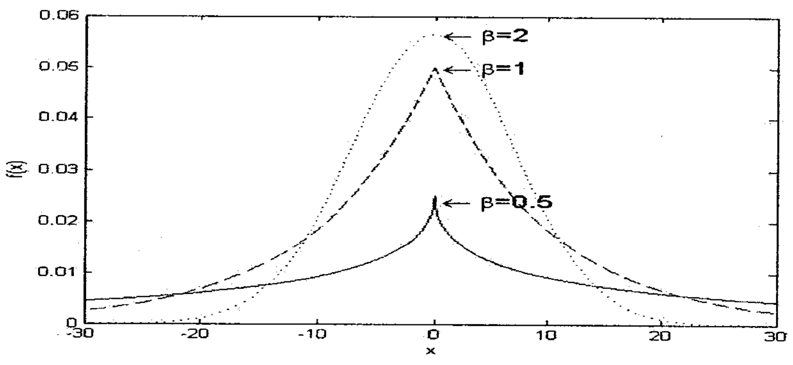


Figure 2.1: Generalized Gaussian Distributions with different values of the shape parameter.

two special cases, when $\beta = 2$ and $\beta = 1$, the GGD is reduced to the Gaussian and Laplacian distributions, respectively. If x follows a mixture of M GGDs, then

$$p(x|\Theta) = \sum_{j=1}^{M} p(x|\mu_j, \alpha_j, \beta_j)p_j \tag{2}$$

where p_j ($0 \leq p_j \leq 1$ and $\sum_{j=1}^{M} p_j = 1$) are the mixing proportions and $p(x|\mu_j, \alpha_j, \beta_j)$ is the GGD describing component j. As for the symbol $\Theta = (\xi, p)$, it refers to the entire set of parameters to be estimated, knowing that $\xi = (\mu_1, \alpha_1, \beta_1, ..., \mu_M, \alpha_M, \beta_M)$, and $p = (p_1, ..., p_M)$.

Consider N observations, $\mathcal{X} = (x_1, ..., x_N)$, the well-known approach to estimate the parameters of a mixture model is to maximize the likelihood through the expectation-maximization (EM) algorithm [50], supposing that the number of mixture components M is known. The likelihood corresponding to this case is:

$$p(\mathcal{X}|\Theta) = \prod_{i=1}^{N} \sum_{j=1}^{M} p(x_i|\xi_j) p_j \tag{3}$$

Where $\xi_j = (\mu_j, \alpha_j, \beta_j)$. For each variable x_i, let Z_i be an M-dimensional vector known by the unobserved or missing vector that indicates to which component x_i belongs, such that: Z_{ij} will be equal 1 if x_i belongs to class j or 0, otherwise. The complete-data likelihood is then:

$$p(\mathcal{X}, Z|\Theta) = \prod_{i=1}^{N} \prod_{j=1}^{M} (p(x_i|\xi_j) p_j)^{Z_{ij}} \tag{4}$$

Where $Z = \{Z_1, Z_2, ..., Z_N\}$. The EM algorithm consists of getting the mixture parameters that maximize the log-likelihood function given by:

$$L(\Theta, Z, \mathcal{X}) = \sum_{j=1}^{M} \sum_{i=1}^{N} Z_{ij} \log(p(x_i|\xi_j) p_j) \tag{5}$$

by replacing each Z_{ij} by its expectation, defined as the posterior probability that the ith observation arises from the jth component of the mixture as follows:

$$\widehat{Z}_{ij}^{(t)} = \frac{p^{(t-1)}(x_i|\xi_j^{(t-1)}) p_j^{t-1}}{\sum_{j=1}^{M} p^{(t-1)}(x_i|\xi_j^{(t-1)}) p_j^{t-1}} \tag{6}$$

where t denotes the current iteration step and $\xi_j^{(t)}$ and $p_j^{(t)}$ are the current evaluations of the parameters. The EM produces a sequence of estimates to the mixture parameters Θ^t, for $t = 0, 1, \ldots$, until a certain convergence criterion is satisfied through two different steps: the expectation and maximization. The EM algorithm consists of:

1. Initialization of the mixture paraneters.

2. E-step: Compute $\widehat{Z}_{ij}^{(t)}$ (Eq. 6) using the initialized parameters.

3. M-step: Update parameters estimates using: $\widehat{\Theta}^{(t)} = \arg\max_{\Theta} L(\Theta^{t-1}, Z, \mathcal{X})$

However, the EM has some drawbacks, like convergence to local maxima due to its dependence on the initialization step. For a detailed and interesting discussion about EM disadvantages please refer to [50]. An efficient alternative technique that we will propose in the following is the Bayesian approach which has received a lot of attention recently thanks to the evolution of Markov Chain Monte Carlo (MCMC) computational tools.

2.2.2 Bayesian Estimation of the GGM

Simulation methods like MCMC algorithms are chosen as a solution to overcome the problems of numerical methods. Generally these methods are related to the Bayesian theory, which means that they allow for probability statements to be made directly about the unknown parameters of the mixture, while taking into consideration prior or expert opinion. The goal is to get the posterior distribution $\pi(\Theta|\mathcal{X}, Z)$, by combining the prior information about the parameters, $\pi(\Theta)$, with the observed value or realization of the complete data $p(\mathcal{X}, Z|\Theta)$, which is derived from Bayes formula:

$$\pi(\Theta|\mathcal{X}, Z) = \frac{\pi(\Theta)p(\mathcal{X}, Z|\Theta)}{\int \pi(\Theta)p(\mathcal{X}, Z|\Theta)} \propto \pi(\Theta)p(\mathcal{X}, Z|\Theta) \tag{7}$$

Where $(\mathcal{X}, Z)$, is the complete data. Having the joint distribution, $\pi(\Theta)p(\mathcal{X}, Z|\Theta)$, we can deduce the posterior distribution (Eq. 7). With $\pi(\Theta|\mathcal{X}, Z)$ in hand we can simulate our model parameters Θ, rather than computing them. The Gibbs sampler is a well known simulation technique [49] and it is based on the successive simulation of Z, p, and ξ conditional on each other and on the observations which offers an efficient way to explore the parameter space. The standard Gibbs sampler for mixture models consists of:

1. Initialization: choose p^0 and ξ^0

2. Step t, for $t = 1, ...$

   (a) Generate $Z^{(t)}$ from $\pi(Z|\Theta, \mathcal{X})$.

   (b) Generate $p^{(t)}$ from $\pi(p|Z^{(t)})$.

   (c) Generate $\xi^{(t)}$ from $\pi(\xi|Z^{(t)}, \mathcal{X})$.

We simulate Z according to the posterior probability $\pi(Z|\Theta, \mathcal{X})$, chosen to be Multinomial of order one with a weight given by $\widehat{Z}_{ij}(\mathcal{M}(1; \widehat{Z}_{i1}; ...; \widehat{Z}_{iM}))$. This choice is due to two reasons, first, we know that each Z_i is a vector of zero-one indicator variables to define from which component j the x observation arises. Second, the probability that the ith observation, x_i, arises from the jth component of the mixture is given by $\widehat{Z}_{ij}$. So, we can deduce that each vector Z_i is generated by a Multinomial distribution of order one with weight given by $\widehat{Z}_{ij}$. Now to simulate p we need to get $\pi(p|Z^{(t)})$, using Bayes rule: $\pi(p|Z) = \frac{\pi(Z|p)\pi(p)}{\int \pi(Z|p)\pi(p)} \propto \pi(Z|p)\pi(p)$. This means that we need to determine $\pi(Z|p)$, and $\pi(p)$. It is well known that the vector p is defined as ($\sum_{j=1}^{M} p_j = 1$, where $p_j \geq 0$), then the commonly considered choice as a prior is the Dirichlet distribution [11, 22]:

$$\pi(p) = \frac{\Gamma(\sum_{j=1}^{M} \eta_j)}{\prod_{j=1}^{M} \Gamma(\eta_j)} \prod_{j=1}^{M} p_j^{\eta_j - 1} \tag{8}$$

Where $(\eta_1, \ldots, \eta_M)$ is the parameter vector of the Dirichlet distribution. As for $\pi(Z|p)$ we have:

$$\pi(Z|p) = \prod_{j=1}^{M} \pi(Z_i|p) = \prod_{i=1}^{N}\prod_{j=1}^{M} p_j^{Z_{ij}} = \prod_{j=1}^{M} p_j^{n_j} \tag{9}$$

Where $n_j = \sum_{i=1}^{N} \mathbb{I}_{Z_{ij=1}}$, then we can conclude that:

$$\pi(p|Z) = \pi(Z|P)\pi(p) = \frac{\Gamma(\sum_{j=1}^{M} \eta_j)}{\prod_{j=1}^{M} \Gamma(\eta_j)} \prod_{j=1}^{M} p_j^{\eta_j + n_j - 1} \propto \mathcal{D}(\eta_1 + n_1, ..., \eta_M + n_M) \tag{10}$$

where $\mathcal{D}$ denotes the Dirichlet distribution with parameters $(\eta_1 + n_1, ..., \eta_M + n_M)$. Thus, we can deduce that the Dirichlet distribution is a conjugate prior for the mixture proportions (i.e the prior and the posterior have the same form). As for the parameters ξ, we assigned independent Normal priors for the distributions means, and Gamma priors for the inverse scale and shape parameters [51,52]: $\mu_j \sim \mathcal{N}(\mu_0, \sigma_0^2)$, $\beta_j \sim \mathcal{G}(\alpha_\beta, \beta_\beta)$, $\alpha_j \sim \mathcal{G}(\alpha_\alpha, \beta_\alpha)$, where $\mathcal{N}(\mu_0, \sigma_0^2)$ is the normal distribution with mean μ_0 and variance σ_0^2, $\mathcal{G}(\alpha_\beta, \beta_\beta)$ is the gamma distribution with shape parameter α_β and rate parameter β_β. $\mu_0, \sigma_0^2, \alpha_\beta, \beta_\beta, \alpha_\alpha, \beta_\alpha$ are called the hyperparameters of the model. Having these priors in hand, the posterior distributions for μ, α, and β are (see Appendix A):

$$\pi(\mu_j|Z, \mathcal{X}) \propto \mathrm{e}^{\frac{(\mu_j-\mu_0)^2}{2\sigma_0^2} + \sum_{Z_{ij}=1}(-\alpha_j|x_i-\mu_j|)^{\beta_j}} \tag{11}$$

$$\pi(\alpha_j|Z, \mathcal{X}) \propto \alpha_j^{\alpha_\alpha-1}\mathrm{e}^{-\beta_\alpha\alpha_j}(\alpha_j)^{n_j}\mathrm{e}^{\sum_{Z_{ij}=1}(-\alpha_j|x_i-\mu_j|)^{\beta_j}} \tag{12}$$

$$\pi(\beta_j|Z, \mathcal{X}) \propto \beta_j^{\alpha_\beta-1}\mathrm{e}^{-\beta_\beta\beta_j}(\frac{\beta_j}{\Gamma(1/\beta_j)})^{n_j}\mathrm{e}^{\sum_{Z_{ij}=1}(-\alpha_j|x_i-\mu_j|)^{\beta_j}} \tag{13}$$

In this case we can notice that our posterior distributions are not in well known forms, so we cannot simulate directly from these posterior distributions. The Metropolis-Hastings (M-H) algorithm [51] offers a solution for this problem, and thus the complete algorithm is given by:

1. Initialization: choose p^0 and Θ^0
2. Step t, for $t = 1, ...$
   (a) Generate $Z_i^{(t)} \sim (\mathcal{M}(1; \widehat{Z}_{i1}; ...; \widehat{Z}_{iM}))$
   (b) Compute $n_j^{(t)} = \sum_{i=1}^N \mathbb{I}_{Z_{ij}^{(t)}}$
   (c) Generate $p^{(t)}$ from Eq.(10).
   (d) Generate $(\mu_j, \alpha_j, \beta_j)^{(t)}$ for $(j = 1, ..., M)$ from Eqs.(11), (12), and(13) using M-H algorithm.

The M-H algorithm can be summarized in 3 steps:

1. Generate $(\widetilde{\mu_j}, \widetilde{\alpha_j}, \widetilde{\beta_j}) \sim q(\mu_j, \alpha_j, \beta_j \mid \mu_j^{(t-1)}, \alpha_j^{(t-1)}, \beta_j^{(t-1)})$ and $\mathbf{U} \sim \mathcal{U}_{[0,1]}$
2. Compute $r = \frac{\pi\left(\widetilde{\mu_j},\widetilde{\alpha_j},\widetilde{\beta_j}|Z,\mathcal{X}\right)q\left(\mu_j^{(t-1)},\alpha_j^{(t-1)},\beta_j^{(t-1)}|\widetilde{\mu_j},\widetilde{\alpha_j},\widetilde{\beta_j}\right)}{\pi\left(\mu_j^{(t-1)},\alpha_j^{(t-1)},\beta_j^{(t-1)}|Z,\mathcal{X}\right)q\left(\widetilde{\mu_j},\widetilde{\alpha_j},\widetilde{\beta_j}|\mu_j^{(t-1)},\alpha_j^{(t-1)},\beta_j^{(t-1)}\right)}$

3. If $r < \mathrm{U}$ then $(\mu_j^{(t)}, \alpha_j^{(t)}, \beta_j^{(t)}) = (\widetilde{\mu}_j, \widetilde{\alpha}_j, \widetilde{\beta}_j)$ else $(\mu_j^{(t)}, \alpha_j^{(t)}, \beta_j^{(t)}) = (\mu_j^{(t-1)}, \alpha_j^{(t-1)}, \beta_j^{(t-1)})$

The major problem in this algorithm is the need to choose the proposal distribution q. To solve this problem we used the most generic random walk M-H by considering the following proposals: $\widetilde{\alpha}_j \sim \mathcal{LN}(\log(\alpha_j^{(t-1)}), \zeta^2)$, $\widetilde{\beta}_j \sim \mathcal{LN}(\log(\beta_j^{(t-1)}), \zeta^2)$, where $\mathcal{LN}$ is the log-normal distribution, since, we know that $\widetilde{\alpha}_j > 0$ and $\widetilde{\beta}_j > 0$. As for $\widetilde{\mu}_j$ we have $\widetilde{\mu}_j \sim \mathcal{N}(\mu_j^{(t-1)}, \zeta^2)$, where ζ^2 is the scale of the random walk. With these proposals the random walk M-H algorithm is composed of the following steps:

1. Generate $\widetilde{\mu}_j \sim \mathcal{N}(\mu_j^{(t-1)}, \zeta^2)$, $\widetilde{\alpha}_j \sim \mathcal{LN}(\log(\alpha_j^{(t-1)}), \zeta^2)$, $\widetilde{\beta}_j \sim \mathcal{LN}(\log(\beta_j^{(t-1)}), \zeta^2)$, and $\mathrm{U} \sim \mathcal{U}_{[0,1]}$

2. Compute

$$r_\mu = \frac{\pi(\widetilde{\mu}_j | Z, \mathcal{X}) \mathcal{N}(\mu_j^{(t-1)} | \widetilde{\mu}_j, \zeta^2)}{\pi(\mu_j^{(t-1)} \mid Z, \mathcal{X}) \mathcal{N}(\widetilde{\mu}_j | \mu_j^{(t-1)}, \zeta^2)} \tag{14}$$

$$r_\alpha = \frac{\pi(\widetilde{\alpha}_j | Z, \mathcal{X}) \mathcal{LN}(\alpha_j^{(t-1)} | \log(\widetilde{\alpha}_j), \zeta^2)}{\pi(\alpha_j^{(t-1)} \mid Z, \mathcal{X}) \mathcal{LN}(\widetilde{\alpha}_j | \log(\alpha_j^{(t-1)}), \zeta^2)} \tag{15}$$

$$r_\beta = \frac{\pi(\widetilde{\beta}_j | Z, \mathcal{X}) \mathcal{LN}(\beta_j^{(t-1)} | \log(\widetilde{\beta}_j), \zeta^2)}{\pi(\beta_j^{(t-1)} \mid Z, \mathcal{X}) \mathcal{LN}(\widetilde{\beta}_j | \log(\beta_j^{(t-1)}), \zeta^2)} \tag{16}$$

3.
   - If $r_\mu >$ u then $\mu_j^{(t)} = \widetilde{\mu}_j$, else $\mu_j^{(t)} = \mu_j^{(t-1)}$
   - If $r_\alpha >$ u then $\alpha_j^{(t)} = \widetilde{\alpha}_j$, else $\alpha_j^{(t)} = \alpha_j^{(t-1)}$
   - If $r_\beta >$ u then $\beta_j^{(t)} = \widetilde{\beta}_j$, else $\beta_j^{(t)} = \beta_j^{(t-1)}$

2.3 Experimental results

2.3.1 Design of Experiments

In this section, we apply our Bayesian GGM estimation algorithm for synthetic data, real datasets, and real applications involving texture classification and retrieval, and image segmentation. We validate our algorithm by comparing it to the EM approach, the EM and Bayesian approaches for the well-known GM. In fact, choosing a relevant model consists both of choosing its form (GGM or GM in our case) and the number of components M. We use two approaches in order to rate the ability of the tested models to fit the data or to determine the number of clusters M. The first criterion is one of the key quantities used for Bayesian hypothesis testing and model selection, the integrated or marginal likelihood defined by [53]:

$$p(\mathcal{X}|M) = \int \pi(\Theta|\mathcal{X}, M)d\Theta = \int p(\mathcal{X}|\Theta, M)\pi(\Theta|M)d\Theta \tag{17}$$

where Θ is the vector of parameters of a finite mixture model, $\pi(\Theta|M)$ is its prior density, and $p(\mathcal{X}|\Theta, M)$ is the likelihood function taking into account that the number of clusters is M. Using the Laplace approximation as in [53] we get:

$$\log(p(\mathcal{X}|M)) \approx \log(p(\mathcal{X}|\widehat{\Theta}, M)) + \log(\pi(\widehat{\Theta}|M)) + \frac{N_p}{2}\log(2\pi) + \frac{1}{2}\log(|H(\widehat{\Theta})|) \tag{18}$$

where $|H(\widehat{\Theta})|$ is the determinant of the Hessian matrix, and N_p is the number of parameters to be estimated which is equal to $(4M)$ for the GGM. We can use the Laplace-Metropolis estimator [53] which consist of finding the Metropolis estimates of $\widehat{\Theta}$ and $H(\widehat{\Theta})$. With samples of the posterior parameters simulated from the M-H in hand, we estimate $\widehat{\Theta}$ as the Θ in the sample at which the likelihood $p(\mathcal{X}|\widehat{\Theta}, M)$ achieves its maximum. For the other quantity needed $H(\widehat{\Theta})$ it is asymptotically equal to the posterior covariance matrix, and could be estimated by the sample covariance matrix of the posterior simulation outputs. The second approach is the Bayesian Information Criterion (BIC) of Schwartz [54] which is actually an approximation for the integrated likelihood criterion [53]: $BIC = \log(\mathcal{X}|M, \widehat{\Theta}_M) - \frac{N_p}{2}\log(N)$. In the following applications, we have used 5000 iteration for our Metropolis-within-Gibbs sampler (we discarded the first 800 iterations as "burn-in" and kept the rest), and our specific choices for the hypeparameters are $(\mu_0, \sigma_0^2, \alpha_\alpha, \beta_\alpha, \alpha_\beta, \beta_\beta) = (0, 1, 0.2, 2, 0.2, 2)$. As for the scale of the random walk we use it as $\zeta^2 = 0.01$ to increase the sensitivity of the random walk sampler.

Table 2.1: Parameters for the different generated data sets. N represents the number of elements in each data set. μ_j,α_j, β_j, and p_j are the real parameters. $\hat{\mu}_j$, $\hat{\alpha}_j$, $\hat{\beta}_j$, and $\hat{p}_j$ are the estimated parameters.

	j	μ_j	α_j	β_j	p_j	$\hat{\mu}_j$	$\hat{\alpha}_j$	$\hat{\beta}_j$	$\hat{p}_j$
Data 1 (N=262144)	1	100.0000	0.0778	1.7000	0.7800	100.0086	0.0704	1.7300	0.7799
	2	200.0000	0.0406	2.3000	0.2200	200.0248	0.0434	2.2900	0.2201
Data 2 (N=65536)	1	63.6283	0.0700	2.1000	0.1458	65.1100	0.0692	2.0000	0.1523
	2	104.6854	0.0603	1.9000	0.7370	106.0010	0.0658	1.9400	0.7222
	3	195.2122	0.0333	1.7000	0.1172	196.5765	0.0340	1.6600	0.1255
Data 3 (N=97344)	1	33.1256	0.0500	2.0000	0.1721	33.7487	0.0509	1.9899	0.1768
	2	95.5550	0.0450	2.4000	0.2000	95.7038	0.0434	2.4368	0.1928
	3	150.5876	0.0650	3.5000	0.3700	150.0495	0.0636	3.7778	0.3751
	4	185.9900	0.0400	3.1000	0.2579	185.3104	0.0409	3.1177	0.2552

2.3.2 Synthetic Data

This section has two main goals, first testing the effectiveness of the algorithm to: estimate the mixture parameters and to select the number of clusters. Then to illustrate the higher performance of our algorithm compared to the EM estimation. To reach our first goal we generated three datasets and applied our method to estimate the parameters and select the number of components of the associated mixture models. Table 2.1 contains the real and estimated parameters of the generated datasets. Fig. 2.2 shows the real and the estimated histograms of the three generated data sets. The integrated likelihood and the BIC calculated for different number of clusters (M=1, 2, 3, 4 and 5) of the above datasets are given in Fig. 2.3. Fig. 2.4 shows the time series plot of our

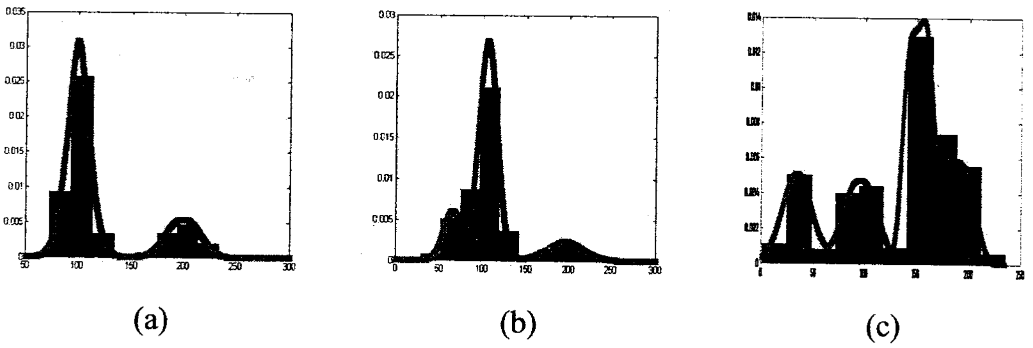
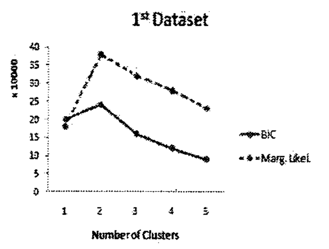

(a) (b) (c)

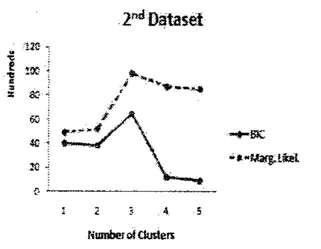
Figure 2.2: Real and estimated histograms for the three datasets. (a) First dataset with M=2, (b) Second dataset with M=3, (c) Third dataset with M=4.

Table 2.2: Parameters for the generated data set using both algorithms. N and M represent the number of elements in the data set, and the number of components, respectively. μ_j,α_j, β_j, and p_j are the real parameters. $\hat{\mu}_j^B$, $\hat{\alpha}_j^B$, $\hat{\beta}_j^B$, and $\hat{p}_j^B$ are the estimated parameters using our algorithm. $\hat{\mu}_j^{EM}$, $\hat{\alpha}_j^{EM}$, $\hat{\beta}_j^{EM}$, and $\hat{p}_j^{EM}$ are the estimated parameters using the EM.

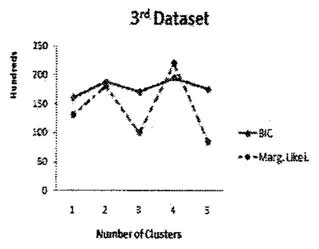
	j	μ_j	α_j	β_j	p_j	$\hat{\mu}_j^B$	$\hat{\alpha}_j^B$	$\hat{\beta}_j^B$	$\hat{p}_j^B$	$\hat{\mu}_j^{EM}$	$\hat{\alpha}_j^{EM}$	$\hat{\beta}_j^{EM}$	$\hat{p}_j^{EM}$
(N=15625)	1	40.0000	0.0381	4.5000	0.1900	40.2617	0.0400	4.4549	0.1889	50.8644	0.0343	2.8050	0.2466
(M=5)	2	75.0000	0.0655	2.5000	0.1500	74.1178	0.0620	2.4363	0.1394	101.4133	0.0404	2.0245	0.3164
	3	105.0000	0.0400	3.3000	0.2200	107.6681	0.0460	3.1363	0.2286	202.4412	0.0364	2.4100	0.4370
	4	182.0000	0.0475	3.0000	0.2500	182.2170	0.0451	3.1749	0.2514				
	5	215.0000	0.0600	3.5000	0.1900	217.3647	0.0559	3.5041	0.1963				



(a)



(b)



(c)

Figure 2.3: Marginal Likelihood and BIC values for the three datasets with different number of clusters. (a) First dataset, (b) Second dataset, (c) Third dataset.

Bayesian algorithm iterations by taking the first dataset as an example. We can see clearly that our algorithm is able to get the exact number of clusters and a very good approximation of the mixtures parameters of the generated data sets. For the second goal, we generated a dataset that has five components and applied both methods (Bayesian and EM) to check if they can approximate it effectively. For our algorithm, it was able to: recognize that the dataset is generated from five classes, and to estimate its parameters effectively. As for the EM algorithm its selection of the number of components was wrong which forces it to a false estimation of the parameters. Table 2.2 contains the real and estimated parameters of the generated dataset using both algorithms. Fig. 2.5 shows the real and the estimated histograms of the dataset using the two methods. The integrated likelihood and the BIC calculated for different number of clusters (M=1, 2, 3, 4 and 5)

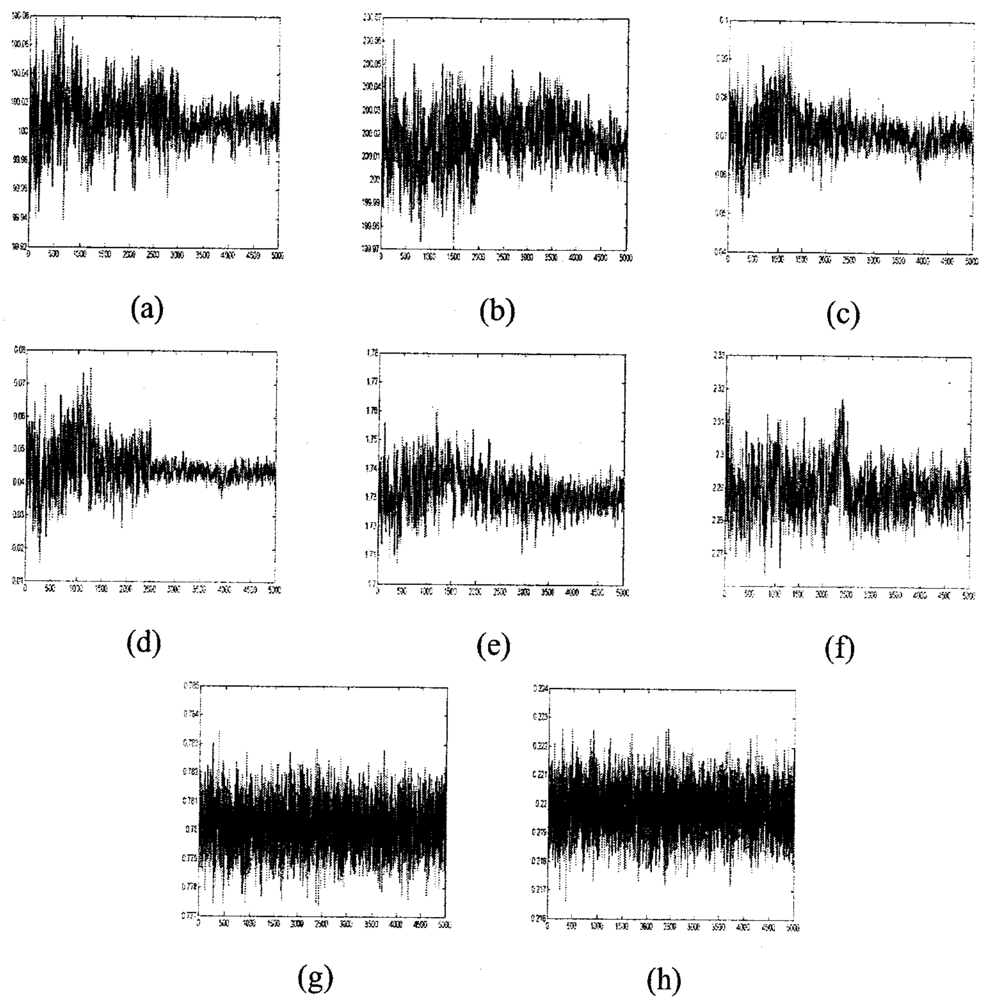
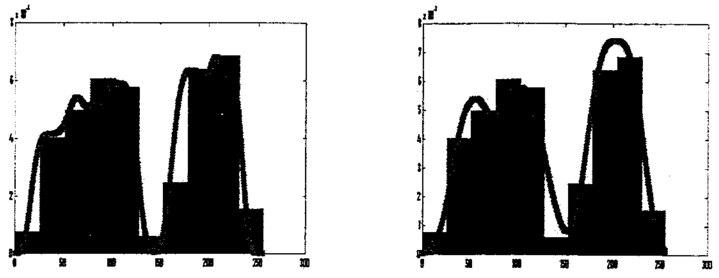


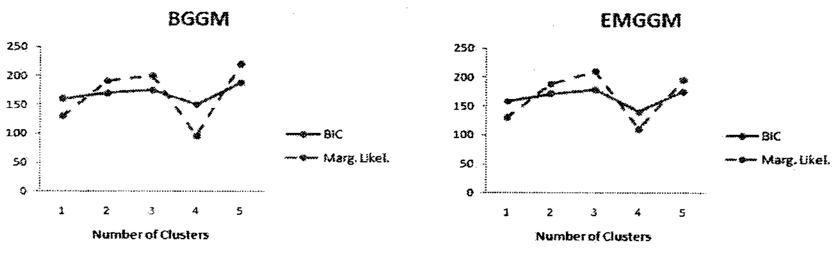
Figure 2.4: Time series plot of Gibbs-within-Metropolis iterations for the first dataset. (a) Iterations for $\hat{\mu}_1$, (b) Iterations for $\hat{\mu}_2$, (c) Iterations for $\hat{\alpha}_1$, (d) Iterations for $\hat{\alpha}_2$, (e) Iterations for $\hat{\beta}_1$, (f) Iterations for $\hat{\beta}_2$, (g) Iterations for $\hat{p}_1$,(h) Iterations for $\hat{p}_2$.

of the two different algorithms are given in Fig. 2.6



(a) (b)

Figure 2.5: Real and estimated histograms for the dataset using both algorithms (a) Bayesian algorithm, (b) EM algorithm.

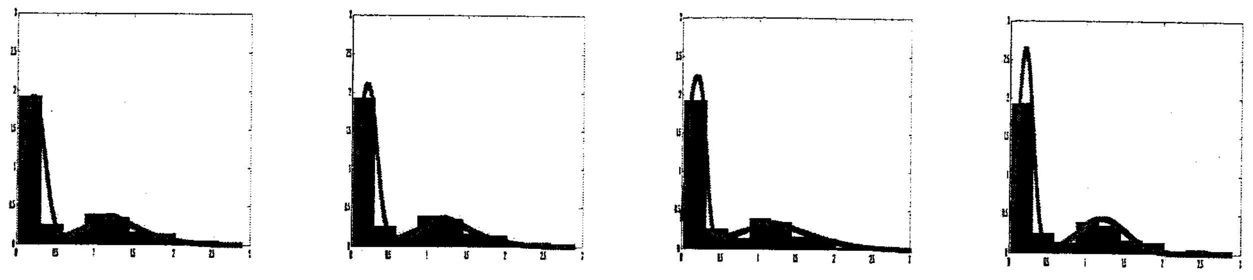


(a) (b)

Figure 2.6: Marginal Likelihood and BIC values for the dataset with different number of clusters using the two algorithms. (a) Bayesian algorithm, (b) EM algorithm.

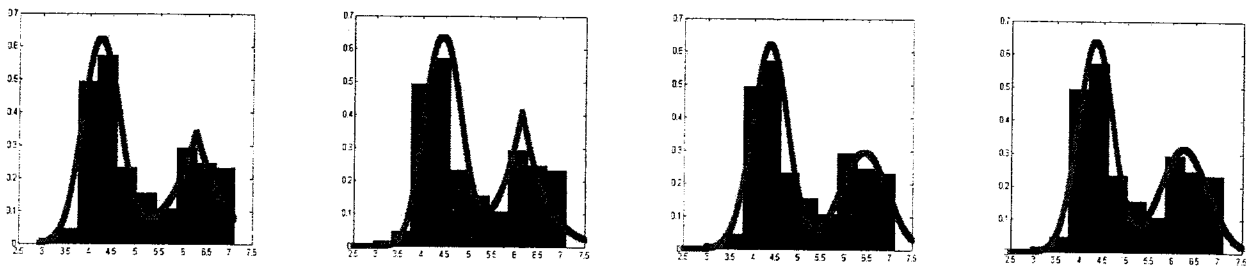
2.3.3 Real Datasets

We devote this section to real datasets. Our method is used to model three standard widely used datasets. The first one describes an enzymatic activity in the blood among a group of 245 unrelated individuals, and the second one is an acidity index measured in a sample of 155 lakes in the Northeastern United States. The third and last one, consists of thickness of 485 postage stamps produced in Mexico. For these three data sets, a mixture of 2 distributions is generally identified [55]. Figures 2.7, 2.8, and 2.9 show the real and the estimated histograms for the three datasets, respectively, when applying the GGM and the GM using EM and Bayesian approaches. In all



(a) (b) (c) (d)

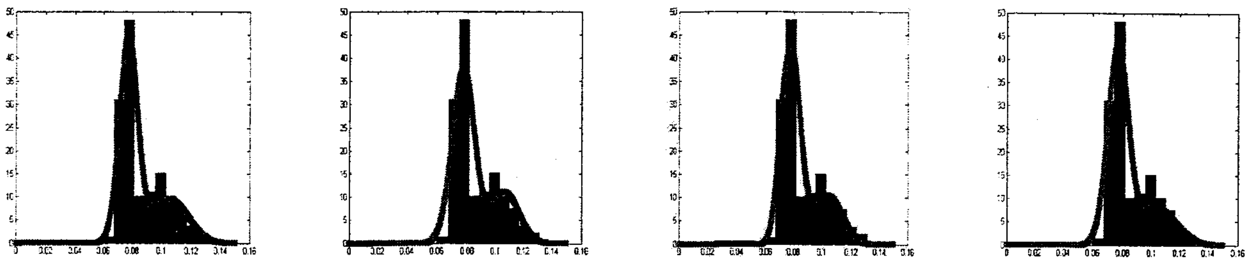
Figure 2.7: Real and estimated histograms for the enzyme data set. (a) Using Bayesian estimation for GGM, (b) Using EM for GGM, (c) Using Bayesian estimation for GM, (d) Using EM for GM.



(a) (b) (c) (d)

Figure 2.8: Real and estimated histograms for the acidity data set. (a) Using Bayesian estimation for GGM, (b) Using EM for GGM, (c) Using Bayesian estimation for GM, (d) Using EM for GM.

cases, it's clear that the GGM and the GM fit the data. The final results of the Bayesian and EM estimations, in the GGM case, are given in table 2.3. The values of the BIC, and marginal likelihood criteria for both GGM and GM using Bayesian and EM methods for different values of M are given in Fig. 2.10. According to these figures the optimal number of components to fit the three datasets in all cases is $M = 2$.



(a) (b) (c) (d)

Figure 2.9: Real and estimated histograms for the stamp data set. (a) Using Bayesian estimation for GGM, (b) Using EM for GGM, (c) Using Bayesian estimation for GM, (d) Using EM for GM.

[ht!]

Table 2.3: Bayesian and EM parameters estimation of the three datasets using two components GGM (^{B} and ^{EM} denote Bayesian and EM estimations, respectively).

Parameters	p_j^B	μ_j^B	α_j^B	β_j^B	p_j^{EM}	μ_j^{EM}	α_j^{EM}	β_j^{EM}
Enzyme Mode 1	0.6325	0.1981	5.0556	2.345	0.6040	0.1899	6.0323	1.8949
Enzyme Mode 2	0.3675	1.2395	2.0059	1.3347	0.3960	1.3119	2.1682	1.4666
Acidity Mode 1	0.5956	4.4279	1.8592	2.2566	0.5998	4.3203	2.2629	1.9693
Acidity Mode 2	0.4044	6.1410	1.9923	1.0495	0.4002	6.2820	1.6287	1.9987
Stamp Mode 1	0.7134	0.0774	101.4945	2.4315	0.6948	0.0773	95.0483	2.0311
Stamp Mode 2	0.2866	0.1025	65.7201	2.2048	0.3056	0.1066	65.0550	2.1729

2.3.4 Classification and Retrieval of Texture Images

Approach

Texture is one of the main characteristics used to describe natural images, which explain its important role in image processing, computer vision and pattern recognition applications. Texture analysis is a fundamental step in a variety of image processing applications such as industrial inspection, medical imaging, remote sensing, and content-based image classification and retrieval [29, 56, 57]. Texture analysis approaches can be divided into four categories: statistical, geometrical, model-based, and signal processing methods [58]. Many classification methods based on images frequency analysis have been proposed in the past. The basic assumption of these methods is that texture can be identified by the energy distribution in the frequency domain via the decomposition of the frequency spectrum into a sufficient number of sub-bands. Then, the statistics of the sub-bands coefficients can be derived and modeled to distinguish different image textures. Indeed, texture information can be modeled using second or higher order statistics [59] and it is well-known that natural image textures generally give rise to non-Gaussian highly-peaked sub-bands densities [60]. In this section we propose an approach for texture images classification and retrieval based on our GGM Bayesian learning algorithm. The used classification methodology, previously adopted in [61] using GM, takes into consideration that signatures of different textures will differ when transformed to frequency domain.

In our classification framework, an image texture is first transformed to gray scale and decomposed into sub-bands using steerable filters [62, 63]. Figure 2.11 shows a texture image and its multiscale version in a pyramid hierarchy. The histograms of the resulted filtered images are show in Fig. 2.12 which shows clearly that the Gaussian assumption would be inappropriate. Then, each sub-band's marginal density is approximated by a GGM model using our Bayesian estimation algorithm (see Fig. 2.13). Finally, the Earth Mover's Distance (EMD) [64] is used to measure the distribution similarity between a set of components representing an input image texture (ie. test image) and sets of components representing texture classes (i.e training images). In our case, EMD can be viewed as the minimum cost of changing one mixture into another, when the cost of moving probability mass from components in the first mixture to components in the second mixture is calculated using Kullback-Leibler (KL) divergence given by:

$$D(f_i||g_j) = \int f_i(x)\log(\frac{f_i(x)}{g_j(x)}) \tag{19}$$

Where f_i is the component i of the input sub-image mixture, g_j is the component j of the class sub-image mixture. The derivation for the KL divergence of the Generalized Gaussian distribution is known to be [29]:

$$D(f_i||g_j) = \log(\frac{\beta_i\alpha_i\Gamma(\frac{1}{\beta_j})}{\beta_j\alpha_j\Gamma(\frac{1}{\beta_i})}) + (\frac{\alpha_j}{\alpha_i})^{\beta_j}\frac{\Gamma(\beta_j + \frac{1}{\beta_i})}{\Gamma(\frac{1}{\beta_i})} - \frac{1}{\beta_i} \tag{20}$$

Where (α_i, β_i) are the parameters of f_i, and (α_j, β_j) are the parameters of g_j. With KL in hand we have to start the minimization problem in which we need to get the $m \times n$ matrix F, where f_{ij} is the amount of weight w_{xi} matched to w_{yj} (w_{xi} and w_{yj} are the weights of the distribution), that will minimize the following equation

$$EMD_{sub} = \sum_{i=1}^{m} \sum_{j=1}^{n} f_{ij} D(f_i || g_j) \tag{21}$$

and subjected to the following constraints:(1) $f_{ij} \geq 0$, where $1 \leq i \leq m$ and $1 \leq j \leq n$, (2) $\sum_{i=1}^{m} f_{ij} = w_{yj}$, where $1 \leq j \leq n$, (3) $\sum_{j=1}^{n} f_{ij} = w_{xi}$, where $1 \leq i \leq m$, (4) $\sum_{i=1}^{m} \sum_{j=1}^{n} f_{ij} = min(w_x, w_y)$, where $w_x = \sum_{i=1}^{m} w_{xi}$, and $w_y = \sum_{j=1}^{n} w_{yj}$. Note that when the image texture is decomposed of L sub-bands, then the total EMD is the sum of that of each sub-band, $EMD = \sum_{l=1}^{L} EMD_{sub_l}$. By computing the EMD between the input texture image and each texture class, each image is affected by the class for which the EMD is the smallest.

Results

The images that we have used in our experiments are from the MIT VisTex database [1]. Six homogeneous texture groups (Bark, Fabric, Food, Metal, Water, and Sand) were considered (Fig. 2.14). For each of the Bark, Fabric, and Metal texture groups, we used four 512×512 images each divided into sixty four 64×64 subimages. And for the Food, Water, and Sand texture groups, we used six 512×512 images each divided into sixty four 64×64 subimages as well. This gives us a total of 256 subimages for each class in the first three groups, and 384 subimages for each class in the second three groups. We then applied our classification approach 10 times, each time using 24 subimages of each original texture image for training and the remaining 40 for testing. This brings us to a total of 720 images from all six groups as training samples for our algorithm, and 1200 as testing samples. We compared the accuracy of our algorithm to classify all 1200 images to those of the three other methods (EM+GGM, Bayesian+GM, EM+GM). We applied our algorithm twice

[1]MIT Vision and Modeling Group (http://vismod.www.media.mit.edu).

Table 2.4: The Average ($\pm$ standard deviation) classification accuracy, over 10 trials, of the four different methods

Method	Using 3 levels pyramid	Using 5 levels pyramid
Bayesian General Gaussian Mixture Models	94.12% $\pm$ 1.62%	95.62% $\pm$ 1.34%
EM General Gaussian Mixture Models	92.58% $\pm$ 1.74%	93.66% $\pm$ 1.69%
Bayesian Gaussian Mixture Models	91.92% $\pm$ 1.36%	92.35% $\pm$ 1.51%
EM Gaussian Mixture Models	90.83% $\pm$ 1.92%	91.97% $\pm$ 2.72%

first using three levels pyramid and second using five levels pyramid to compare the two cases together (see Table 2.4). From these results we can observe two main points: our algorithm has the highest accuracy and as expected the five levels pyramid improves the performance over the three levels pyramids, however this improvement is very small compared to the enormous difference in computational time.

In the retrieval application, we have used each and every subimage as a query and checked if we are able to retrieve all the other 64 subimages coming from the same mother image. Our retrieval approach can be divided into two steps. First task, is the same as the classification approach, we classify the image into one of the six groups. For the second step, we compare the input image with the other images in the same group and retrieve the closest images to our query. We applied our retrieval process twice former using three levels pyramid and latter using five levels pyramid. To measure the retrieval rates (precision and recall), each image was used as a query and the number of relevant images among those that were retrieved was noted. Table 2.5 presents the retrieval rates obtained in terms of precision when 64 images are retrieved each time in response to a query. Note that in this case the precision and recall are the same because for a given image we have at most 64 images which are similar to it. Figure 2.15(a) displays the average (averaged over all the queries) precision rate for different texture classes when we consider only the first 64 images retrieved. Figure 2.15(b) shows two graphs the overall recall of our retrieval method when varying the total number of images retrieved taken into consideration. According to the results in Table 2.5 we can reach the following conclusions. First the highest average retrieval rate is reached using our

[ht!]

Table 2.5: Average Retrieval rate(%) for the four different methods

Method	Using 3 levels pyramid	Using 5 levels pyramid
Bayesian General Gaussian Mixture Models	81.25%	83.81%
EM General Gaussian Mixture Models	76.37%	79.16%
Bayesian Gaussian Mixture Models	72.91%	74.52%
EM Gaussian Mixture Models	71.66%	72.12%

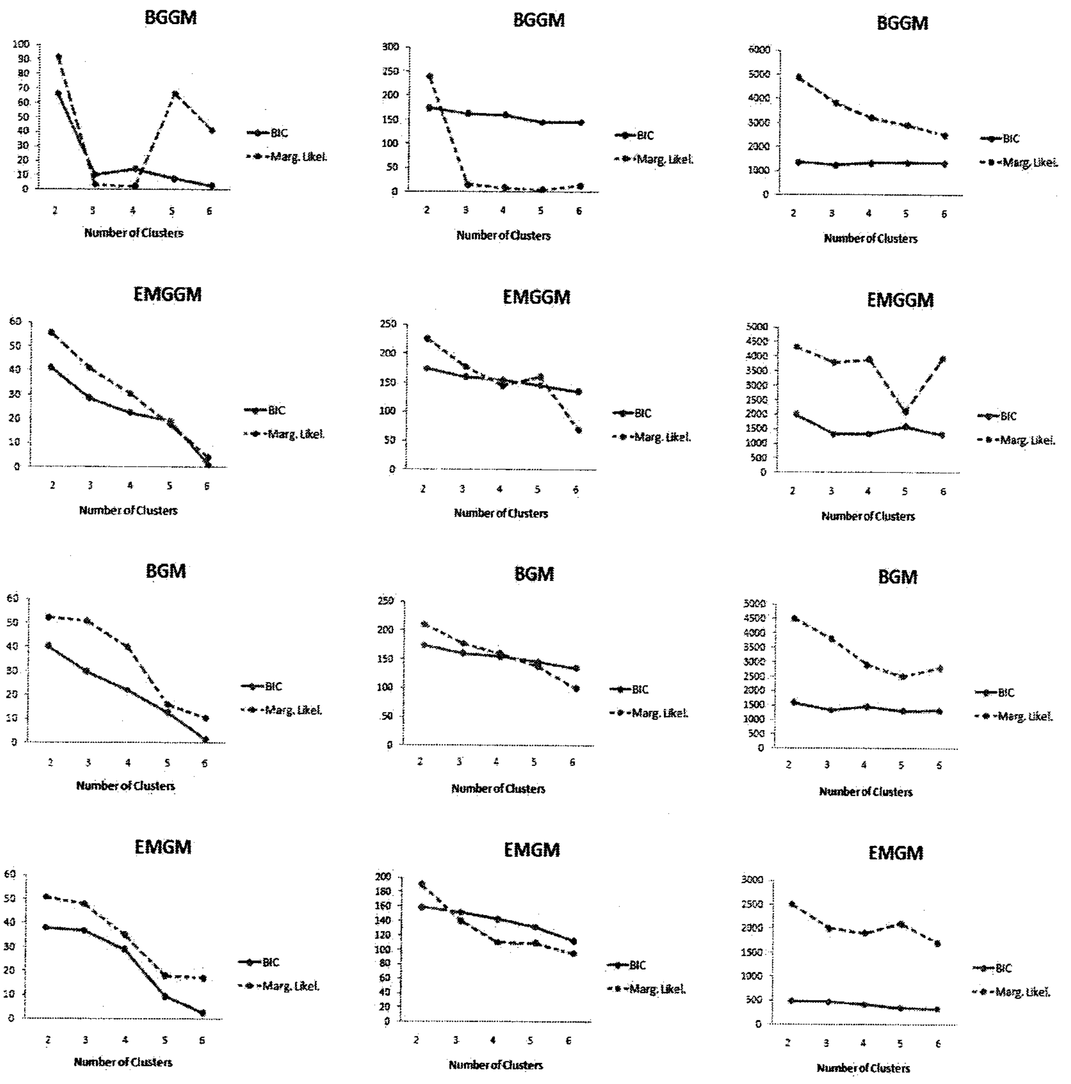
method. Second, it is enough to use three levels pyramid due to the large computational difference between it and the five levels pyramid, and the small difference in their effectiveness.

2.3.5 Image Segmentation

Image segmentation is one of the most significant problems, due to the fact that it is fundamental to many tasks of pattern recognition, image processing, computer vision where the segmentation results generally govern the final quality of interpretation. Several approaches have been proposed in the past. Many techniques, based on finite mixture Gaussians, have been developed, also, where the idea was to partition the image in regions (each associated with one mixture component). However, generally the pixel intensities inside the image regions are heavy-tailed which force the Gaussian mixture model to lose its accuracy. Often, image segmentation must be done in an unsupervised fashion in that training data is not available and the class conditioned feature vectors must be estimated directly from the data. In this section, we apply our estimation algorithm for the segmentation problem by formulating it as a classification problem with mixtures of generalized Gaussian distributions. It is noteworthy to mention that the main purpose of this application is to compare our Bayesian estimation with the EM one and with the results obtained when using Gaussian mixture models. Comparisons with the several other segmentations approaches that have been proposed in the past is out of the scope of our work.

We tested the effectiveness of our method on two different types of images, Printed circuit board (PCB) and Synthetic Aperture Radar (SAR) images, we decided to use these images because of their non-Gaussian characteristics.We began our experiment by applying our algorithm on four complex electronic printed circuit board (PCB) images, to check if the algorithm is able to recognize the background from the printed circuit, Integrated Circuits (IC), wiring, and the pinholes in images. We ran these images with the four different algorithms to compare their effectiveness for separating non-Gaussian data. The first image is an image of a complex PCB (Fig 2.16(a)), applying the four different methods, we obtained three different outputs in regard with the number of clusters. The Bayesian and EM Generalized gaussian algorithms segmented the image into four groups, while the Bayesian Gaussian algorithm divided it into three classes, and the EM Gaussian algorithm separated it into two groups, respectively (Fig. 2.17). We can notice two points: the general Gaussian mixture was able to identify the right number of clusters, and the Bayesian algorithm performs a slightly better refinement in the electronic circuit wiring then the EM. The second image is also an image of a (PCB). We decided to use this image (Fig. 2.16(b)) as it has something written on the board. Applying our segmentation algorithm, it was able to identify the four different classes of the image and to segment it accurately. As for the three other algorithms they were unable to identify the right number of classes which led them to wrong segmentations (Fig 2.18). For the third image (Fig 2.16(c)), we used an image corrupted with Salt and Pepper noise to check if the Bayesian algorithm will be able to segment a corrupted image correctly and to identify its components. We found that our method was able to identify all the small details of the PCB, while all the three other method fail to do so (Fig 2.19). A new area where image processing can be applied and indispensable, is the automated visual inspection of manufactured goods. Most of the electronic components manufactures use image processing to identify missing components in the circuit before shipment. We used the four methods on (Fig 2.16(d)) which contains some missing components. Our segmentation method was able to identify the missing components in the circuit and to separate it to a class that only contains these missing components. In other words, our method identifies that the image contains five classes exactly, where the fifth class contains only the places in the PCB where there are missing components. For the three other methods, they were unable to segment and identify the missing components in the image (Fig 2.20).

We have also tested our method using SAR images. Unlike natural images, SAR images characteristically have a particular kind of noise called speckle, which is introduced due to the coherent imaging process. This causes serious problems for SAR image segmentation process. Hence segmentation techniques that work successfully on natural images may not perform as well on SAR images. We used three different images to investigate the effectiveness of our algorithm. The First and second images are used to illustrate the segmentation effectiveness of our method in segmenting images highly corrupted by atmospheric turbulence (Figs. 2.21(a), 2.21(b)). For both images we can notice that the Bayesian Generalized Gaussian is the most effective approach as it was able to approximate the data with the best estimated histogram compared to the three other methods. The third image (Fig. 2.24) was taken for the Beijing area, China. We can notice that the Bayesian generalized Gaussian mixture estimated histogram is the closest one to the real histogram.



(a) (b) (c)

Figure 2.10: BIC and Marginal Likelihood for several values of *M* when using the four different methods. (a) For the Enzyme dataset, (b) For the Acidity dataset, (c) For the Stamp dataset.

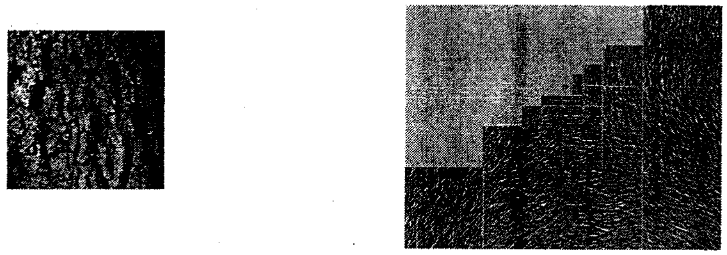

(a) (b)

Figure 2.11: Original image and its steerable pyramid decomposition. (a) Original image from Bark group in Vistex, (b) Sub-images output using five level steerable pyramid.

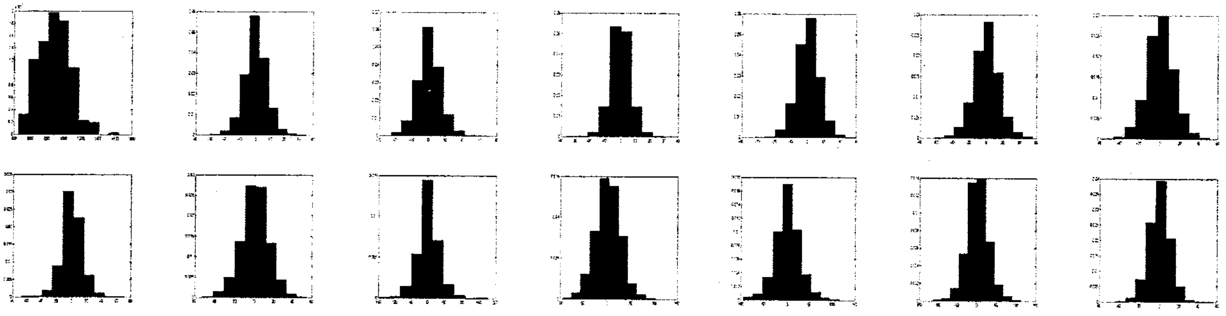


Figure 2.12: Histograms of the 14 Sub-images of the steerable pyramid.

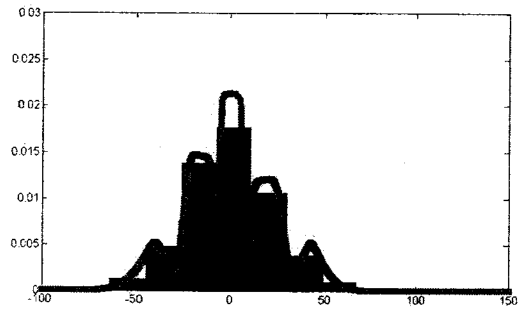


Figure 2.13: An example of the Sub-image real and estimated histogram using 5 components GGM.

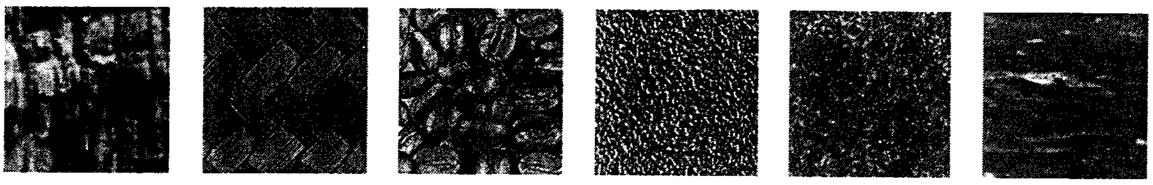
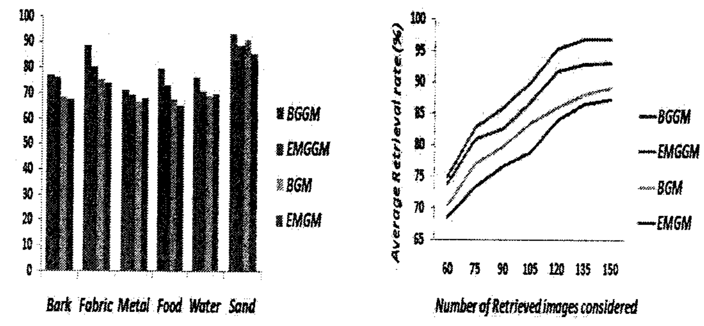


Figure 2.14: Sample images from each group. (a) Bark, (b) Fabric, (c) Food, (d) Metal, (e) Sand, (f) Water.



(a) (b)

Figure 2.15: Average retrieval rate. (a) Precision rate using the 4 different methods for each of the 6 classes used, (b) Overall recall rate using the 4 different methods.

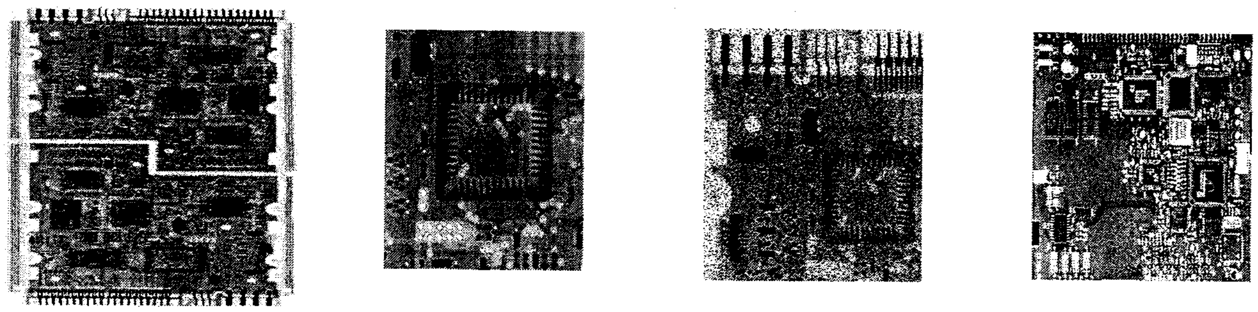


Figure 2.16: Tested images (a) Complex PCB, (b) PCB with text, (c) PCB with noise, (d) PCB with missing components.

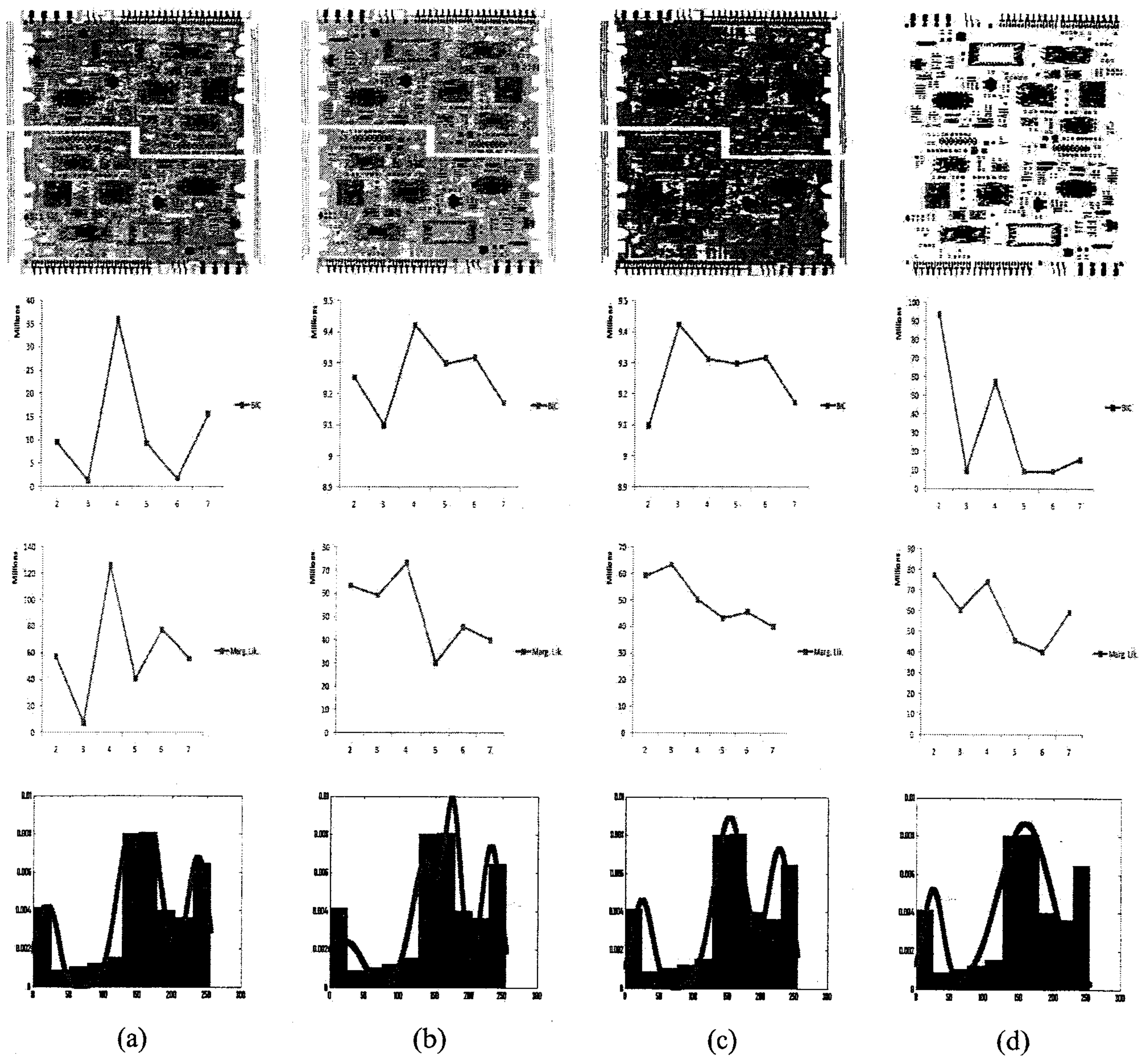


Figure 2.17: Segmentations results for Fig 2.16.a. (a) Bayesian Generalized Gaussian Mixture, (b) E-M Generalized Gaussian Mixture, (c) Bayesian Gaussian Mixture, (d) E-M Gaussian Mixture.

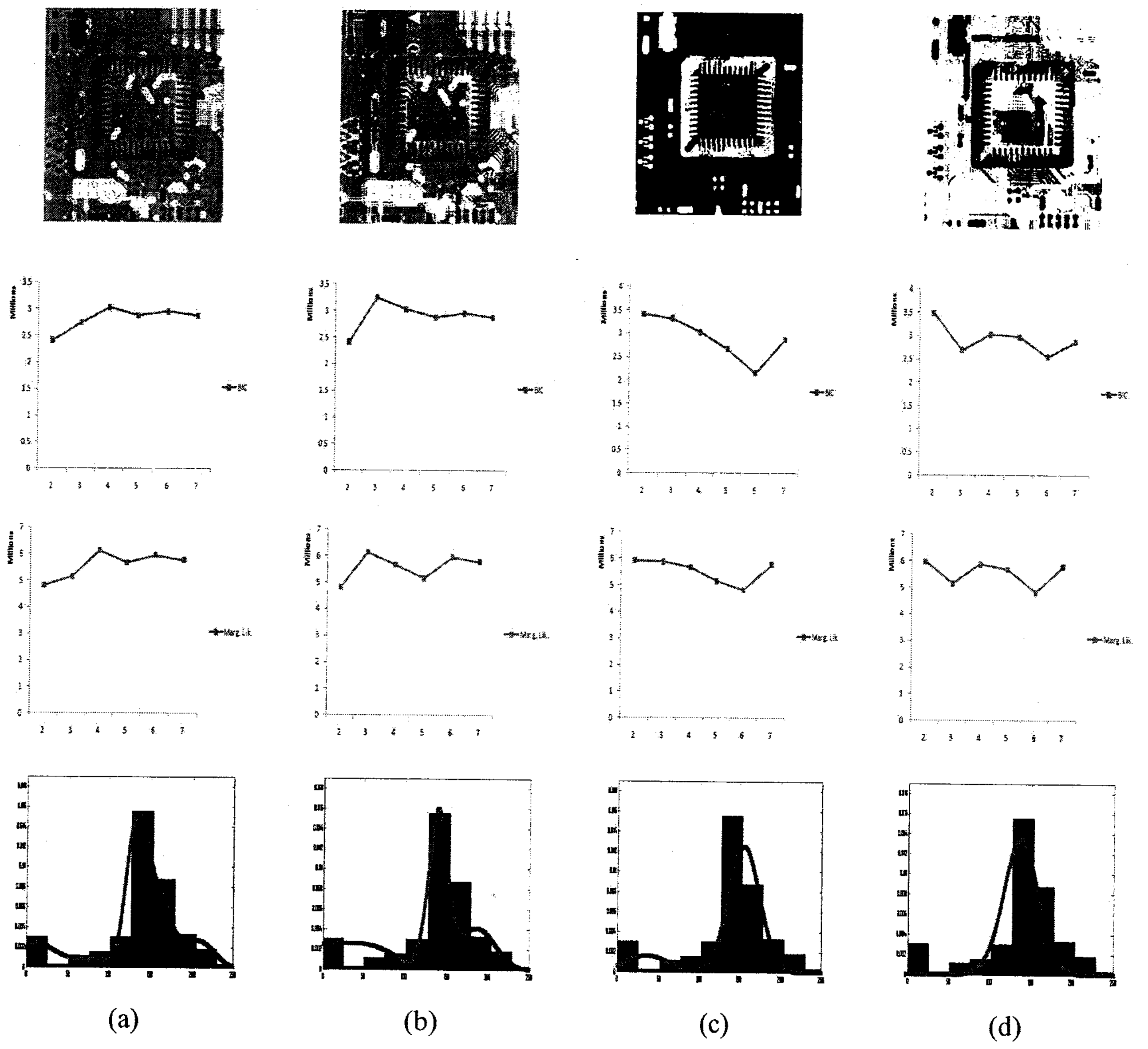


Figure 2.18: Segmentations results for Fig. 2.16.b. (a) Bayesian Generalized Gaussian Mixture, (b) E-M Generalized Gaussian Mixture, (c) Bayesian Gaussian Mixture, (d) E-M Gaussian Mixture.

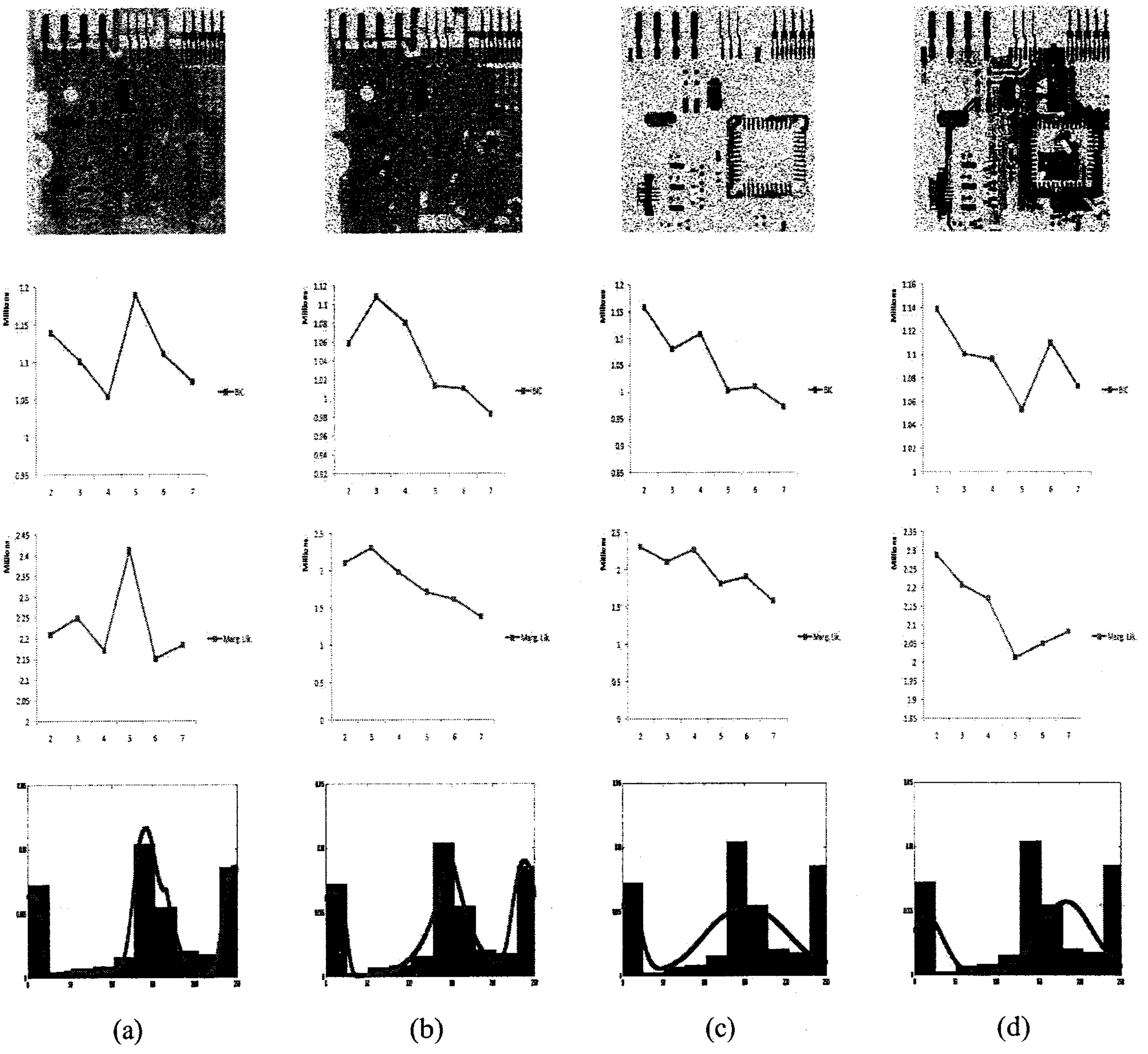


Figure 2.19: PCB Original image corrupted with noise (Fig 2.16.c) and its segmentations using the four methods. (a) Bayesian Generalized Gaussian Mixture, (b) E-M Generalized Gaussian Mixture, (c) Bayesian Gaussian Mixture, (d) E-M Gaussian Mixture.

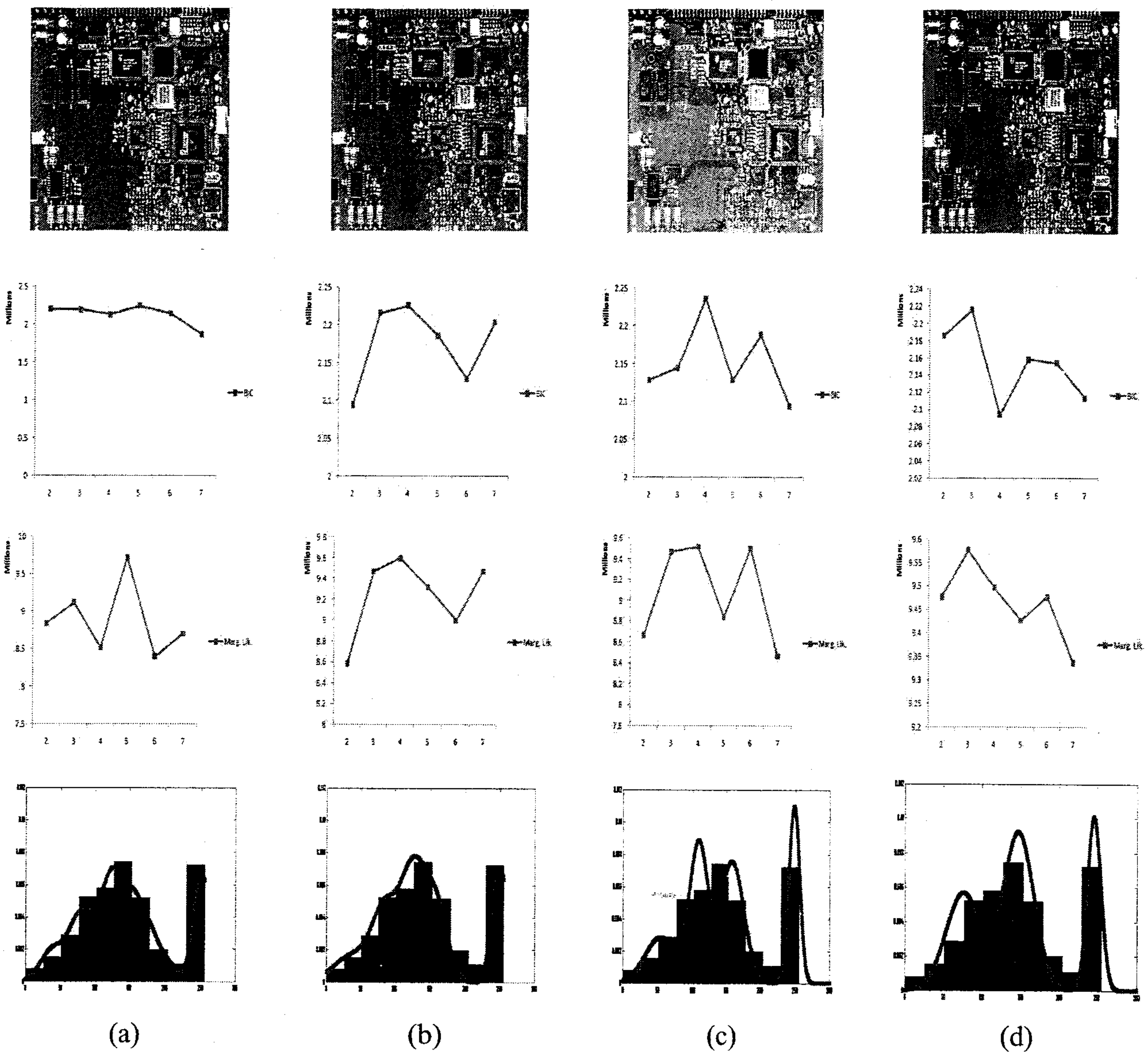
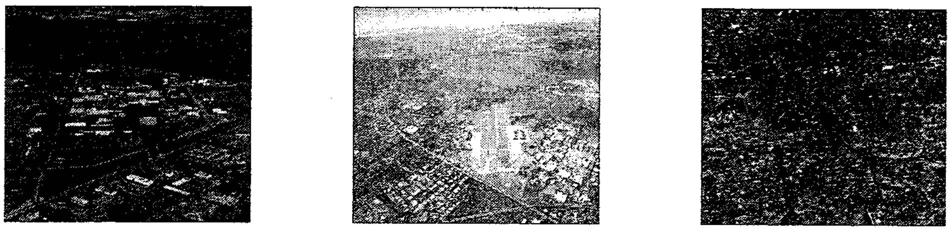
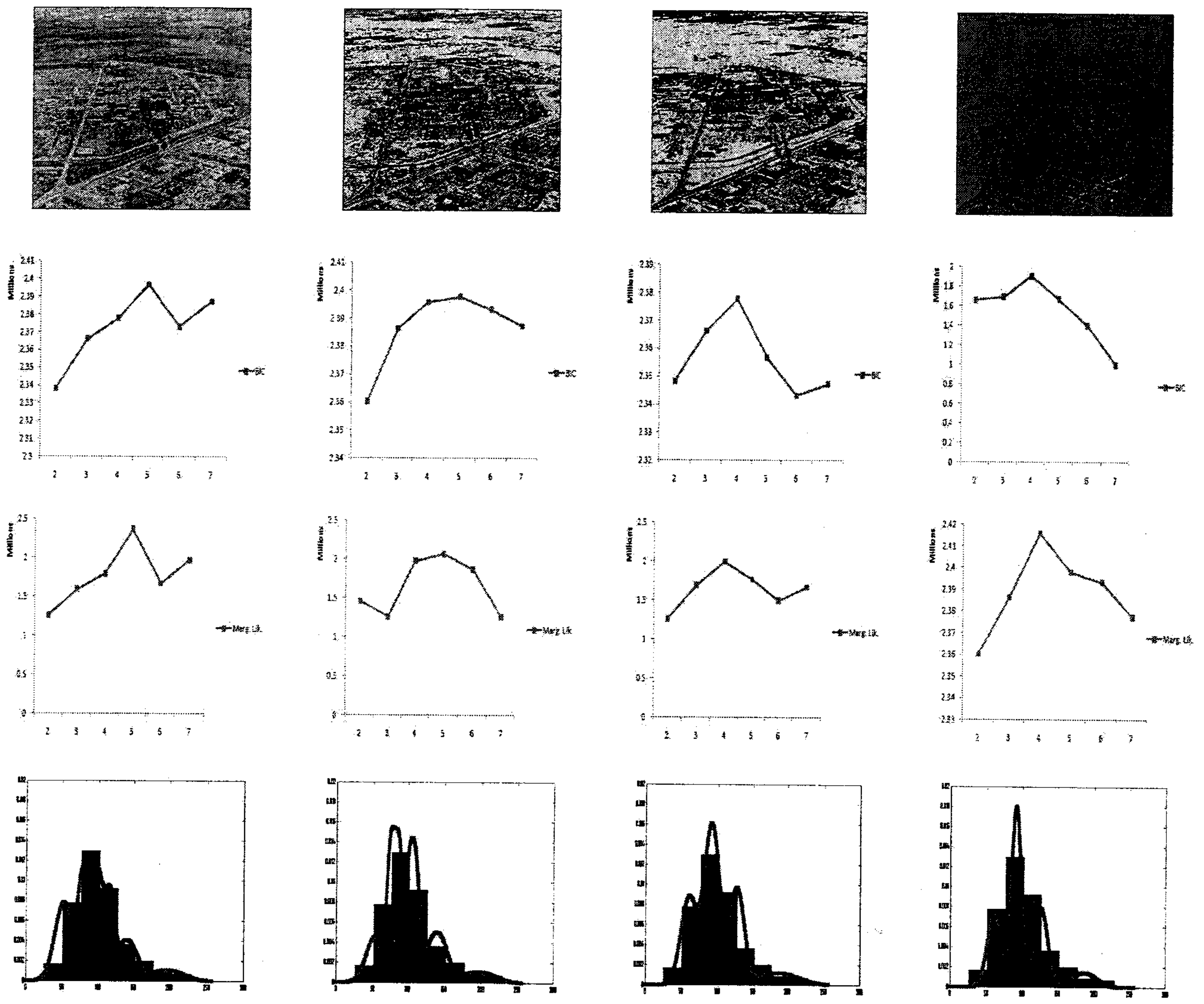


Figure 2.20: Segmentations results for Fig 2.16.d. (a) Bayesian Generalized Gaussian Mixture, (b) E-M Generalized Gaussian Mixture, (c) Bayesian Gaussian Mixture, (d) E-M Gaussian Mixture.



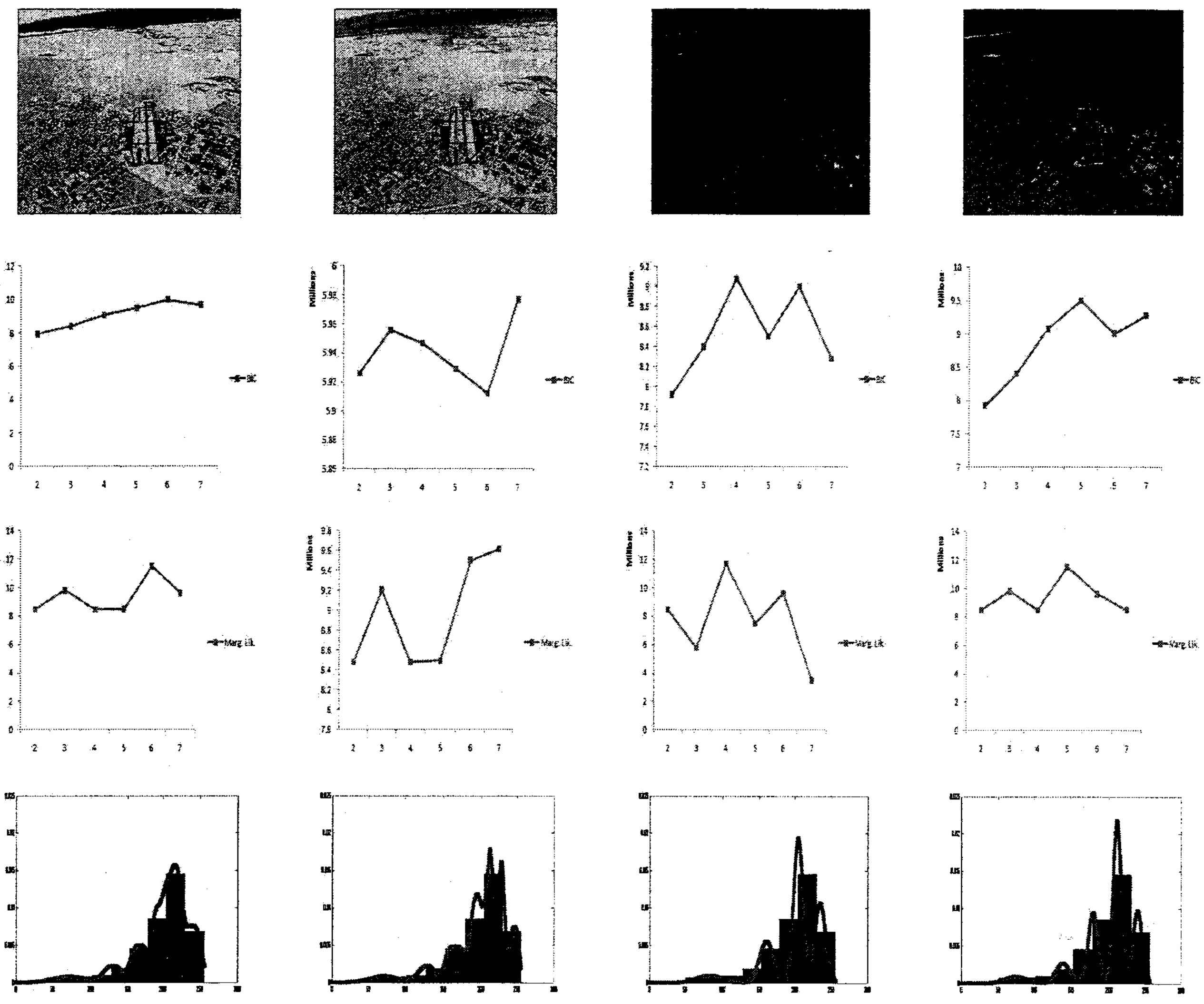
(a) (b) (c)

Figure 2.21: Tested SAR images (a) First image (Courtesy of NASA), (b) Second Image (Courtesy of NASA), (c) SAR image (Courtesy of European Space Agency).



(a) (b) (c) (d)

Figure 2.22: SAR image (Fig. 2.21.a) and its segmentations using the four methods. (a) Bayesian Generalize Gaussian Mixture, (b) E-M Generalize Gaussian Mixture, (c) Bayesian Gaussian Mixture, (d) E-M Gaussian Mixture.



(a) (b) (c) (d)

Figure 2.23: SAR image (Fig. 2.21.b) and its segmentations using the four methods. (a) Bayesian Generalize Gaussian Mixture, (b) E-M Generalize Gaussian Mixture, (c) Bayesian Gaussian Mixture, (d) E-M Gaussian Mixture.

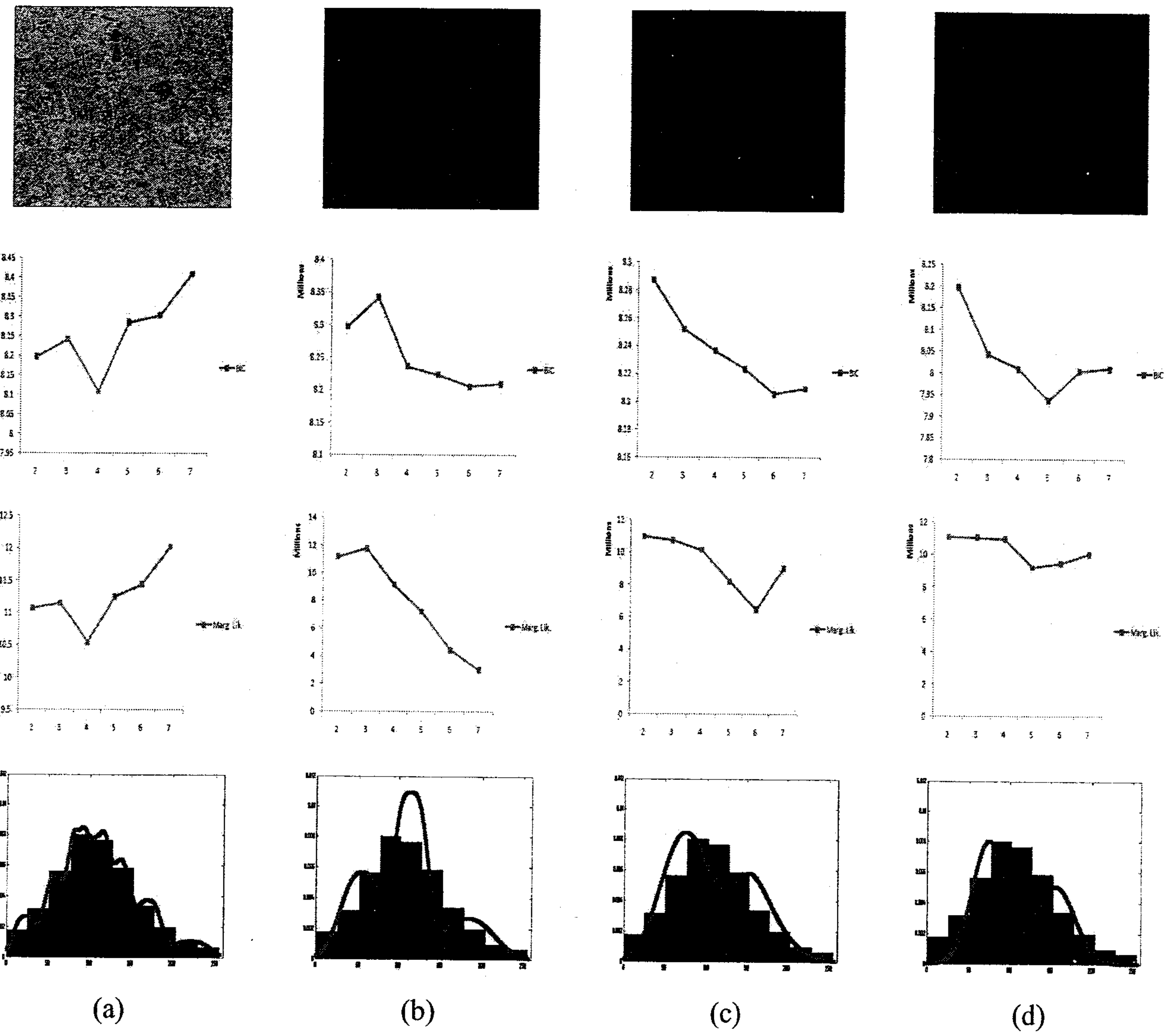


Figure 2.24: SAR image (Fig. 2.21.c) and its segmentations using the four methods(a) Bayesian Generalize Gaussian Mixture, (b) E-M Generalize Gaussian Mixture, (c) Bayesian Gaussian Mixture, (d) E-M Gaussian Mixture.

2.3.6 Biomedical/Bioinformatics Applications

In the context of biomedical image processing and Bioinformatics, an important problem is the development of accurate models for image segmentation and DNA spot detection. In this part we study a highly efficient unsupervised algorithm for biomedical image segmentation and spot detection of cDNA microarray images, based on the Generalized Gaussian mixture models. Our work is motivated by the fact that biomedical and cDNA microarray images both contain non-gaussian characteristics, impossible to model using rigid distributions. Generalized Gaussian mixture models are robust in the presence of noise and outliers, more flexible to adapt the shape of data, and less sensible for over-fitting the number of classes compared to Gaussian mixture.

Biomedical image segmentation

Image segmentation is one of the major challenges in image analysis, since image analysis tasks highly depend on how well previous segmentation is accomplished. Image segmentation is the procedure of dividing an image into different groups with each group enjoying similar properties such as texture, color, boundary, and intensity [65, 66]. Despite, the existence of different segmentation methods, many of them fail to provide satisfactory results when applied on biomedical images. Reasons behind this failure are numerous. First, image segmentation is strongly influenced by the quality of data and biomedical images contain different noises such as speckle, shadows which may cause the boundaries of structures to be indistinct and disconnected. Second, most of image segmentations algorithms are founded on the assumption that the data are Gaussian which is not the case for biomedical images. Further complications arise as the contrast between areas of interest in biomedical images is low, which make the extraction of the desired regions impossible as they are statistically indistinguishable. Last but not least, most of existed segmentation methods do not integrate uncertain prior knowledge.

In this section, we develop a new segmentation methodology, using our GGM Bayesian MCMC algorithm. We can divide our method into two main steps: histogram adjustment, and identification of object of interest using the Bayesian GGM with the integrated likelihood. We validate our algorithm by comparing it to a state of the art segmentation algorithm [67]. This method is divided into two stages: preprocessing, and object segmentation. Preprocessing stage contains histogram adjustment, noise reduction, and layer of interest extraction using K-means algorithm. For the object segmentation a marker-controlled watershed technique is used.

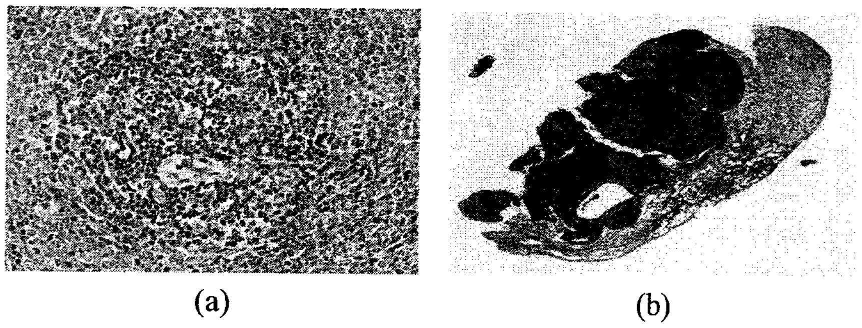


Figure 2.25: Microscopic images used, (a) The rat spleen tissue pulps (Courtesy of Dr. Jinglu Tan), (b) Lung Carcinoid tumor (Courtesy of Dr. Robert Cardiff).

The first image used is the image of a rat spleen tissue pulps (Fig. 2.25(a)). For visual differentiation of cellular components, the tissue section was stained with haematoxylin and eosin $(H\&E)$. Under a microscope, nuclei are usually dark blue, red blood cells orange/red, and muscle fibers deep pink/red. The feature used to differentiate red and white is the density of the lymphocytes. The white pulp has lymphocytes and macrophages surrounding central arterioles. The distribution of the lymphocytes in red pulp is much looser than those in white pulp. Evaluating the severity of infection requires identifying the white pulps. We started by transforming the color image to a gray level image (Fig. 2.26(a)) in order to simplify the processing procedure. For grayscale image nuclei are dark objects within a gray background. Then Histogram adjustment [68] is applied on the image to increase image contrast (Fig 2.26(b)). At this point, we applied our Bayesian GGM to identify the object of interest in the image (Fig 2.26(c)). Comparing the output from our method to the one from the watershed method (Fig 2.26(d)), we can find that we were able to reach a higher identification for the infected regions. Also, the proposed method is less complex due to the fact that we did not need to use neither noise reduction, nor marker-controlled watershed techniques.

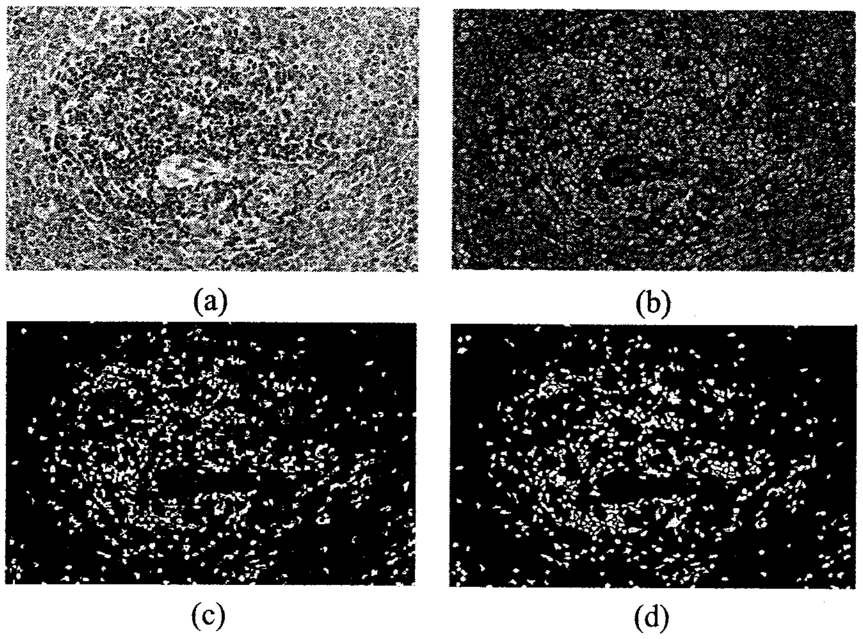


Figure 2.26: The different stage outputs for the two methods on the rat spleen tissue, (a) The gray scale image, (b) the image after histogram adjustment, (c) The identified object of interest using our method, (d) The identified object of interest using the state of the art algorithm.

The second image is an image of a carcinoid tumor seen in the lung of eighty one years old female (Fig. 2.25(b)) . To be able to differentiate visually the cellular components, the tissue section was stained with haematoxylin and eosin ($H\&E$). The size of the tumor is of 2.5 cm long as shown in the image.

First we transformed the image into gray scale image (Fig. 2.27(a)). Next, we applied the histogram adjustment on the image (Fig. 2.27(b)) , and last we applied our algorithm on it to reach the object of interest (Fig. 2.27(c)). Also, it is quite clear here that our algorithm overperformed the watershed algorithm.

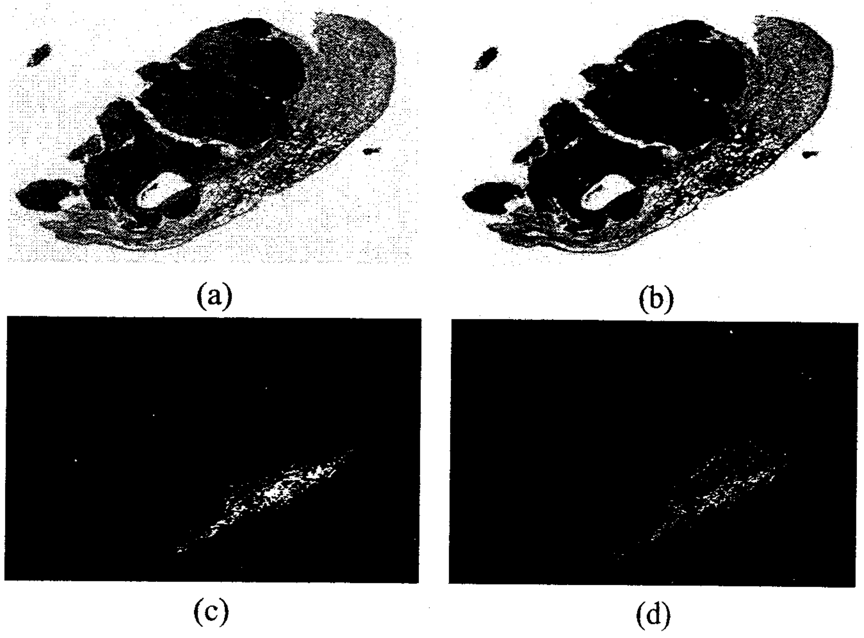


Figure 2.27: The different stage outputs for the two methods on the Lung Carcinoid tumor, (a) The gray scale image, (b) the image after histogram adjustment, (c) The identified object of interest using our method, (d) The identified object of interest using the state of the art algorithm.

Experimental results show that the proposed method is effective and accurate in segmenting microscopic images even without the need of noise reduction stages and marker-controlled watershed techniques to separate the touching objects.

Spot detection and image segmentation in DNA microarray data

In this section, we propose an optimized clustering-based method for microarray image segmentation using GGM. Our algorithm is based on the fact that GGM are flexible to model the shape of data, and have high immunity to noise. To access the performance of our method, we compare it to two well known algorithms: k-means clustering microarray image segmentation (SKMIS) [61], and optimized k-means microarray image (OKMIS) [69]. We evaluate the segmentation performance of the three methods on the spot images from ApoA1 Data [70].

DNA microarray technology is a high throughput technique allowing the comprehensive measurement of the expression level of thousands of genes simultaneously in the studies of genomics for biology and medicine [71, 72]. Complementary single stranded DNA (cDNA) microarrays consist of thousands of individual DNA sequences printed in a high density array. Nowadays, microarray experiments are used to compare gene expression from two samples: target or experimental, and control. The mRNA of both biological tissues (normal and tumor) is extracted, then reversed transcribed into complementary DNA (cDNA) copy, followed by a labeling procedure using two fluorescence dyes, Cyanine Cy3 (green channel) and Cy5 (red channel). After labeling, the two samples are mixed and hybridized with the arrayed DNA sequences. Afterwards, fluorescence measurements are made for each dye separately, and the digital image scanner records the intensity level at each microarray location producing two grayscale images [73].

Image analysis is an highly important aspect of cDNA microarray experiments, as it is responsible for reducing an image of spots into a table with a measure of the intensity for each spot. Efficient, accurate and automatic analysis of cDNA spot images is necessary in order to apply this technology in different biological experiments. cDNA microarray gene expression data analysis involves three main stages: spot localization or gridding, background separation or image segmentation, and intensity estimation. Spot localization or gridding is used to identify blocks and to position rows and columns of spots within each block. Background separation or image segmentation is used to segment the image into background or foreground, and the intensity estimation step gets the red and green intensities and assigns the log ratio after background correction in order to represent the log relative abundance of each spot. These stages are quite important, since the accuracy of the resulting data is essential in posterior analysis.

In cDNA microarray experiments, noise is a challenging problem as it can be produced by laser light reflection, dust on the glass slide, and photon and electronic noise. These noises force microarray images to vary in intensity, in the spot sizes and positions. For this reason, we decided to apply the Bayesian GGM on this problem respectively for its immunity to noise [74].

Over the past few years, many approaches have been proposed for microarray image segmentation. Fixed circle segmentation is the first applied technique on microarray images, its idea is to assign the same circle size to all the spots. Another proposed method in order to avoid the drawback of the fixed circle segmentation is the adaptive circle segmentation technique. This algorithm fits a circle with adaptive size around each spot, in order to characterize the pixels in the circle as signal pixels and the pixels out of the circle as background pixels i.e. foreground or background. Another technique that has been efficiently used in microarray image segmentation is clustering, since it is not restricted to a particular shape and size for the spots. Single k-means clustering microarray image segmentation (SKMIS) attempts to cluster the pixels into two groups, one for foreground, and the other for background. Therefore in SKMIS, feature vector is reduced to a single variable in the Euclidean one-dimensional space. Optimized k-means microarray image segmentation (OKMIS) not only consider the intensity of the pixel but also the shape of the spot based on the fact that the position of the pixel could also influence the result of the clustering. Our algorithm is very simple as we only apply a two component GGM to classify the data to either foreground or background.

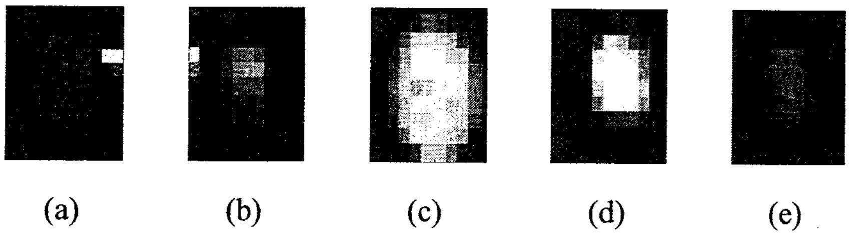


Figure 2.28: Five noisy spots obtained from the 1230c1G/R microarray image.

In order to compare GGM, OKMIS and SKMIS, we applied the three methods on the 1230c1G/R microarray image obtained from the ApoA1 data. Fig. 2.28 shows some examples for the noisy spots in our microarray image. From Fig. 2.29 It is quite clear that our method was able to retrieve the true foreground from the background. We also observe that the GGM overperformed the SKMIS and OKMIS in identifying noisy pixels from foreground. Note that, the GGM was able to take the data form . Hence, the GGM is more suitable when dealing with cDNA microarray image segmentation.

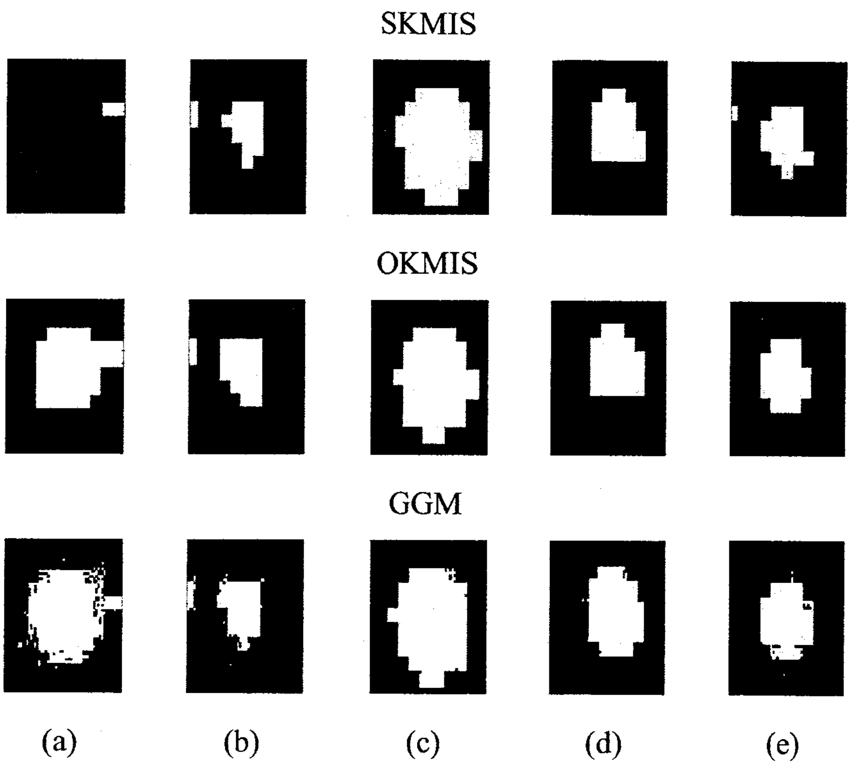


Figure 2.29: The Experiment results of the three methods on the five noisy spots.

2.4 Conclusion

We have presented a Bayesian analysis of finite generalized Gaussian mixtures. Our learning algorithm is based on the Monte Carlo simulation technique of Gibbs sampling mixed with a Metropolis-Hasting step. For the estimation of the number of clusters describing the mixture model, we used the marginal likelihood with Laplace approximation, and the BIC criterion. We have demonstrated clearly by different applications that Bayesian estimation and selection gives reliable estimates. The Bayesian approach provides a natural extension to deal with uncertainty and noise by incorporating prior information.

CHAPTER 3

A Fully Bayesian Model Based on Reversible Jump MCMC and Finite Beta Mixtures for Clustering

The use of mixture models in image and signal processing has proved to be of considerable interest in terms of both theoretical development and in their usefulness in several applications. Researchers have approached the mixture estimation and selection problem, to model complex datasets, with different techniques in the last few years. In theory, it is well-known that full Bayesian approaches, to handle this problem, are fully optimal. The Bayesian learning allows the incorporation of prior knowledge in a formal coherent way that avoids overfitting problems. In this chapter, we propose a fully Bayesian approach for finite Beta mixtures learning using a Reversible Jump Markov Chain Monte Carlo (RJMCMC) technique which simultaneously allows cluster assignments, parameters estimation, and the selection of the optimal number of clusters. The adverb "fully" is justified by the fact that all parameters of interest in our model including number of clusters and missing values are considered as random variables for which priors are specified and posteriors are approximated using RJMCMC. Our work is motivated by the fact that Beta mixtures are able to fit any unknown distributional shape and then can be considered as a useful class of flexible models to address several problems and applications involving measurements and features having well-known marked deviation from the Gaussian shape. The usefulness of the proposed approach is confirmed using synthetic mixture data, real data, and through an interesting application namely texture classification and retrieval.

3.1 Introduction

In recent years Bayesian approaches have found an increased interest in the image and signal processing community [75]. Bayesian inference provides consistent learning frameworks for model uncertainty, through the use of posterior model probabilities, which is fundamental in image processing applications [76]. The majority of the approaches that have been proposed separate the estimation and the selection of the number of components (i.e a certain criterion should be compared for different number of clusters) (see, for instance, [9, 10] for interesting discussions and comparisons between different criteria). Note, however, that both estimation and selection problems are strongly related and depend heavily on the underlying mixture density components choice.

In this chapter, we propose to simultaneously estimate and select finite Beta mixture models using the reversible jump samplers introduced by Green [77] and which have been applied successfully for instance to Gaussian [78–81], Poisson [82, 83], exponential [84] mixtures, to variable selection [85], and to model selection in general [86, 87]. The basic idea of RJMCMC approach is that it is possible to move between parameter subspaces corresponding to statistical models, such as mixture models with different number of components, which offers effective model selection (i.e structure discovery) and produces a good mixing of the Markov chains. Using RJMCMC allows us to explore simultaneously both the parameter and model space by treating the number of clusters as a random variable having a prior distribution [1] and it does not need to be specified in advance, since it can be automatically adjusted during iterations. Our work is motivated by the

[1]It is noteworthy that other works, that we do not investigate in this chapter, have put prior distribution on the number of components (see, for instance, [88–91]).

compactly supported nature of the data generally handled in image and signal processing applications and by the fact that the Beta distribution is able to model any unknown distributional shape generated by this kind of data. Despite these advantages, finite Beta mixtures have been largely ignored and relatively less visited avenue of study compared to finite Gaussian mixtures. Moreover, it is well-know that any continuous density can be well-approximated by a mixture of Beta distributions (See [92], for instance, for formal statement and detailed proof). For this reason the Beta distribution and mixtures of Beta have been widely used to model expert opinion, as a prior, in Bayesian settings [87, 93, 94]. In this work, however, Beta is used as a parent distribution to model directly the data.

This chapter is structured as follows. After presenting our hierarchical finite mixture Beta Bayesian framework in Section 2, the complete RJMCMC algorithm is discussed and developed in Section 3. Section 4 is devoted to the experimental results. Finally, some conclusions are drawn and future research possibilities are highlighted in Section 4.

3.2 Bayesian Analysis of Beta Mixture Model

3.2.1 Finite General Beta Mixture Model

General Beta Distribution

If the random variable x, where $a < x < b$ and $(a, b) \in \mathbb{R}^2$, follows a general Beta distribution with parameters α and β, then the density function is given by [95]:

$$p(x|\alpha,\beta) = \frac{\Gamma(\alpha+\beta)}{(b-a)^{\alpha+\beta-1}\Gamma(\alpha)\Gamma(\beta)}(x-a)^{\alpha-1}(b-x)^{\beta-1} \tag{1}$$

where $\alpha > 0$ and $\beta > 0$. Note that this distribution is reduced to the well-known Beta when $(a, b) = (0, 1)$, which is actually the univariate case of the Dirichlet distribution which has proven high flexibility to model data [96]. The mean and variance of the general Beta distribution are given by:

$$m = E(x) = (b-a)\frac{\alpha}{\alpha+\beta} + a \tag{2}$$

$$v = Var(x) = (b-a)^2\frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)} \quad . \tag{3}$$

Using equation 2, it is easy to obtain the following location-scale parametrization of the general Beta:

$$p(x|m,s) = \frac{\Gamma(s)}{(b-a)^{s-1}\Gamma(\frac{s(m-a)}{b-a})\Gamma(s(1-\frac{m-a}{b-a}))}(x-a)^{\frac{s(m-a)}{b-a}-1}(b-x)^{s(1-\frac{m-a}{b-a})-1} \tag{4}$$

where $s = \alpha + \beta$ and represents the scale of the distribution and m represents the location. Note that this alternative provides interpretable parameters because m and s represent the mean and a measure of the sharpness of the distribution, respectively [97]. A large value of s produces a sharply peaked distribution around the mean m. And when s decreases, the distribution becomes broader. An additional advantage of this parametrization is that m lies within the bounded space $[a, b]$, leading to an increase in computational efficiency. Therefore, this parametrization will be adopted for learning.

Finite General Beta Mixture

A general Beta mixture with M components is defined as:

$$p(x|\Theta) = \sum_{j=1}^{M} p(x|m_j, s_j)p_j \tag{5}$$

where $\{p_j\}$ are the mixing proportions which are constrained to be non-negative and sum to one, and $p(x|m_j, s_j)$ is the general Beta distribution. The symbol $\Theta = (\xi, P)$ refers to the entire set of parameters to be estimated, where $\xi = (m_1, s_1 \ldots, m_M, s_M)$, and $P = (p_1, \ldots, p_M)$.
Given a set of data with N observations $\mathcal{X} = \{x_1, \ldots, x_N\}$, the classical approach to estimate the parameters of a mixture model is to maximize the likelihood through the Expectation Maximization (EM) algorithm which theoretical framework was first introduced in the seminal paper by

Dempster et *al.* [98]. The EM, however, guarantees convergence only to a local maximum which quality depends highly on the initialization step (i.e as a deterministic approach the EM gets stuck at local maxima that are not globally optimal). Moreover, it is well-known that estimation approaches based on maximizing the likelihood can cause overfitting by preferring complex models. Many researchers have developed modifications and extensions of the EM algorithm (the interested reader is refereed to [50, 99, 100]) [2]. A detailed discussion about the drawbacks of deterministic estimation in the case of finite Beta mixtures can be found in [19]. EM is based on the idea of explicitly representing the mixture components generating each observation via latent allocation variables $Z_i, i = 1, \dots, N$. Each Z_i is an integer in $\{1, \dots, M\}$ denoting the unknown component from which x_i is drawn. The unobserved (or missing) vector $Z = (Z_1, \dots, Z_N)$ is generally called the "membership vector" of the mixture model and its different elements Z_i are supposed to be drawn independently from the distributions

$$p(Z_i = j) = p_j \quad j = 1, \dots, M. \tag{6}$$

The same idea has an important role when using Bayesian approaches which are now widely applied as an alternative thanks to modern Bayesian computational tools. Bayesian estimation has become feasible due to the development of simulation-based numerical integration techniques such as Markov chain Monte Carlo (MCMC) methods [11]. MCMC methods have revolutionized Bayesian statistics by allowing inference for highly complex models which can be treated tractably, albeit numerically, through the simulation of required estimates by running appropriate Markov Chains using specific algorithms such as Gibbs sampler. The Gibbs sampler, however, can be difficult to implement when the conditioning distributions have complicated awkward forms. In this case, solutions include the Metropolis-Hastings algorithm [11] which we will use in this work. Among the important problems that arise in using Bayesian techniques is the choice of priors. In the following, we present our Bayesian model, the priors that we have considered and the resulting posteriors.

[2]In particular the authors in [100] classify these extensions into three groups: pure, hybrid and EM-type accelerators.

3.2.2 Hierarchical Model, Priors and Posteriors

Hierarchical Model

Fully Bayesian analysis considers the number of components M as a parameter in the model for which a conditional distribution should be found. Moreover, the unknowns M, ξ and P, in our mixture model, are regarded as random variables drawn from some prior distributions. The joint distribution of all these variables is

$$p(M, P, Z, \xi, \mathcal{X}) = p(M)p(P|M)P(Z|P, M)p(\xi|Z, P, M)p(\mathcal{X}|\xi, Z, P, M)$$

A common approach is to impose conditional independencies [78], $p(\xi|Z, P, M) = p(\xi|M)$ and $p(\mathcal{X}|\xi, Z, P, M) = p(\mathcal{X}|\xi, Z)$, which give us the following joint distribution

$$p(M, P, Z, \xi, \mathcal{X}) = p(M)p(P|M)P(Z|P, M)p(\xi|M)p(\mathcal{X}|\xi, Z)$$

It is worth mentioning that if we condition on Z, the distribution of x_i is simply given by the Z_ith component in the mixture, $p(\mathcal{X}|\xi, Z) = \prod_{i=1}^{N} p(x_i|\xi_{Z_i})$. Moreover, an extra layer can be introduced to the hierarchy to represent the model parameters (M, P, ξ) priors, which gives the following final form of the joint distribution

$$p(\lambda, \delta, \eta, M, P, Z, \xi, \mathcal{X}) = p(\lambda)p(\delta)p(\eta)p(M|\lambda)p(P|M, \delta)p(Z|P, M)p(\xi|M, \eta)\prod_{i=1}^{N} p(x_i|\xi_{Z_i}) \tag{7}$$

where λ, δ and η are the hyperparameters on which M, P and ξ depend, respectively.

Priors and Posteriors

Let us now define the priors, which we suppose that are all drawn independently [3], of the different parameters in our hierarchical model. We know that each location m_j is defined in the compact

[3]The choice of a simple independence prior structure is a common assumption taken generally when defining Bayesian models.

support [a,b], then an appealing flexible choice as a prior is a general Beta distribution, with location ε and scale ζ common to all components, which was found flexible in real applications. Thus, m_j for each component is given the following prior:

$$p(m_j|\varepsilon,\zeta) \sim \frac{\Gamma(\zeta)}{(b-a)^{\zeta-1}\Gamma\big(\frac{\zeta(\varepsilon-a)}{b-a}\big)\Gamma\big(\zeta(1-\frac{\varepsilon-a}{b-a})\big)}(m_j-a)^{\frac{\zeta(\varepsilon-a)}{b-a}-1}(b-m_j)^{\zeta(1-\frac{\varepsilon-a}{b-a})-1} \quad (8)$$

Since s_j control the dispersion of the distributions, a common choice as a prior is an inverse gamma with shape ϑ and scale ϖ common to all components [101], then

$$p(s_j|\vartheta,\varpi) \sim \frac{\varpi^{\vartheta}\exp(-\varpi/s_j)}{\Gamma(\vartheta)s_j^{\vartheta+1}} \quad (9)$$

using the two previous equations, we have

$$p(\xi|M,\eta) = \prod_{j=1}^{M} p(m_j|\varepsilon,\zeta)p(s_j|\vartheta,\varpi) = \frac{\varpi^{M\vartheta}\Gamma(\zeta)^M\prod_{j=1}^{M}\frac{\exp(-\varpi/s_j)(m_j-a)^{\frac{\zeta(\varepsilon-a)}{b-a}-1}(b-m_j)^{\zeta(1-\frac{\varepsilon-a}{b-a})-1}}{s_j^{\vartheta+1}}}{\Gamma(\vartheta)^M(b-a)^{M(\zeta-1)}\Big[\Gamma\big(\frac{\zeta(\varepsilon-a)}{b-a}\big)\Gamma\big(\zeta(1-\frac{\varepsilon-a}{b-a})\big)\Big]^M} \quad (10)$$

Having this priors in hand, the η hyperparameter in equation 7 is actually $(\varepsilon,\zeta,\vartheta,\varpi)$. Thus, according to the previous equation and our joint distribution in equation 7, the full conditional posterior distributions for m_j and s_j are

$$\begin{aligned}
p(m_j|\ldots) &\propto \prod_{j=1}^{M} p(m_j|\varepsilon,\zeta)p(s_j|\vartheta,\varpi)\prod_{i=1}^{N}p(x_i|\xi_{Z_i}) \propto p(m_j|\varepsilon,\zeta)\prod_{i=1}^{N}p(x_i|\xi_{Z_i})\\
&\propto \left[\frac{\Gamma(s_j)}{(b-a)^{s_j-1}\Gamma\big(\frac{s(m_j-a)}{b-a}\big)\Gamma\big(s_j(1-\frac{m_j-a}{b-a})\big)}\right]^{n_j}\prod_{Z_i=j}\left[(x_i-a)^{\frac{s_j(m_j-a)}{b-a}-1}(b-x_i)^{s_j(1-\frac{m_j-a}{b-a})-1}\right]\\
&\times \frac{\Gamma(\zeta)}{(b-a)^{\zeta-1}\Gamma\big(\frac{\zeta(\varepsilon-a)}{b-a}\big)\Gamma\big(\zeta(1-\frac{\varepsilon-a}{b-a})\big)}(m_j-a)^{\frac{\zeta(\varepsilon-a)}{b-a}-1}(b-m_j)^{\zeta(1-\frac{\varepsilon-a}{b-a})-1} \quad (11)
\end{aligned}$$

$$\begin{aligned}
p(s_j|\ldots) &\propto \prod_{j=1}^{M} p(m_j|\varepsilon,\zeta)p(s_j|\vartheta,\varpi)\prod_{i=1}^{N}p(x_i|\xi_{Z_i}) \propto p(s_j|\vartheta,\varpi)\prod_{i=1}^{N}p(x_i|\xi_{Z_i})\\
&\propto \left[\frac{\Gamma(s_j)}{(b-a)^{s_j-1}\Gamma\big(\frac{s(m_j-a)}{b-a}\big)\Gamma\big(s_j(1-\frac{m_j-a}{b-a})\big)}\right]^{n_j}\prod_{Z_i=j}\left[(x_i-a)^{\frac{s_j(m_j-a)}{b-a}-1}(b-x_i)^{s_j(1-\frac{m_j-a}{b-a})-1}\right]\\
&\times \frac{\varpi^{\vartheta}\exp(-\varpi/s_j)}{\Gamma(\vartheta)s_j^{\vartheta+1}} \quad (12)
\end{aligned}$$

where $n_j = \sum_{i=1}^{N} \mathbb{I}_{Z_i=j}$ and represents the number of vectors belonging to cluster j.

Moreover, we know that the vector P is defined on the simplex $\{(p_1, \ldots, p_M) : \sum_{j=1}^{M-1} p_j < 1\}$, then the typical choice, as a prior, for this vector is a Dirichlet distribution with parameters $\delta = (\delta_1, \ldots, \delta_M)$ [11]

$$p(P|M, \delta) = \frac{\Gamma(\sum_{j=1}^{M} \delta_j)}{\prod_{j=1}^{M} \Gamma(\delta_j)} \prod_{j=1}^{M} p_j^{\delta_j - 1} \tag{13}$$

According to equation 6, we have also

$$p(Z|P, M) = \prod_{j=1}^{M} p_j^{n_j} \tag{14}$$

Using the two previous equations and our joint distribution in equation 7, we obtain

$$p(P|\ldots) \propto p(Z|P, M)p(P|M, \delta) \propto \prod_{j=1}^{M} p_j^{n_j} \frac{\Gamma(\sum_{j=1}^{M} \delta_j)}{\prod_{j=1}^{M} \Gamma(\delta_j)} \prod_{j=1}^{M} p_j^{\delta_j - 1} \propto \prod_{j=1}^{M} p_j^{n_j + \delta_j - 1} \tag{15}$$

which is actually proportional to a Dirichlet distribution with parameters $(\delta_1 + n_1, \ldots, \delta_M + n_M)$. It is noteworthy that the prior and the posterior distributions, $p(P|M, \delta)$ and $\pi(P|\ldots)$, are both Dirichlet. In this case we say that the Dirichlet distribution is a conjugate prior for the mixture proportions. In addition, using equations 6 and 7, we have the following posterior for the membership variables

$$p(Z_i = j|\ldots) \propto \frac{p_j \Gamma(s_j)}{(b-a)^{s_j - 1} \Gamma(\frac{s_j(m_j - a)}{b-a}) \Gamma(s_j(1 - \frac{m_j - a}{b-a}))} (x-a)^{\frac{s_j(m_j-a)}{b-a} - 1} (b-x)^{s_j(1 - \frac{m_j - a}{b-a}) - 1} \tag{16}$$

In order to have a more flexible model, we introduce an additional hierarchical level by allowing the hyperparameters to follow some selected distributions. The hyperparameters, ε and ζ, associated with the m_j are given uniform (we have started by testing a Beta prior for ε, and the best experimental results were obtained with location equal to 1 and scale fixed to 2, which corresponds actually to a uniform distribution) and inverse Gamma priors, respectively:

$$p(\varepsilon) \sim \mathcal{U}_{[0,1]} \tag{17}$$

$$p(\zeta|\varphi,\varrho) \sim \frac{\varrho^{\varphi}\exp(-\varrho/\zeta)}{\Gamma(\varphi)\zeta^{\varphi+1}} \tag{18}$$

Thus, according to these two previous equations and equations 10 and 7, we have

$$p(\varepsilon|\ldots) \propto p(\varepsilon)\prod_{j=1}^{M} p(m_j|\varepsilon,\zeta) \propto \prod_{j=1}^{M} \frac{\Gamma(\zeta)}{(b-a)^{\zeta-1}\Gamma\left(\frac{\zeta(\varepsilon-a)}{b-a}\right)\Gamma\left(\zeta(1-\frac{\varepsilon-a}{b-a})\right)}(m_j-a)^{\frac{\zeta(\varepsilon-a)}{b-a}-1}(b-m_j)^{\zeta(1-\frac{\varepsilon-a}{b-a})-1} \tag{19}$$

$$\begin{aligned} p(\zeta|\ldots) &\propto p(\zeta|\varphi,\varrho)\prod_{j=1}^{M} p(m_j|\varepsilon,\zeta) \\ &\propto \frac{\varrho^{\varphi}\exp(-\varrho/\zeta)}{\Gamma(\varphi)\zeta^{\varphi+1}}\prod_{j=1}^{M}\frac{\Gamma(\zeta)}{(b-a)^{\zeta-1}\Gamma\left(\frac{\zeta(\varepsilon-a)}{b-a}\right)\Gamma\left(\zeta(1-\frac{\varepsilon-a}{b-a})\right)}(m_j-a)^{\frac{\zeta(\varepsilon-a)}{b-a}-1}(b-m_j)^{\zeta(1-\frac{\varepsilon-a}{b-a})-1} \end{aligned} \tag{20}$$

The hyperparameters, ϑ and ϖ, associated with the s_j are given inverse Gamma and exponential priors, respectively:

$$p(\vartheta|\lambda,\mu) \sim \frac{\mu^{\lambda}\exp(-\mu/\vartheta)}{\Gamma(\lambda)\vartheta^{\lambda+1}} \tag{21}$$

$$p(\varpi|\phi) \sim \phi\exp(-\phi\varpi) \tag{22}$$

Thus, according to these two previous equations and equations 10 and 7, we have

$$p(\vartheta|\ldots) \propto p(\vartheta|\lambda,\mu)\prod_{j=1}^{M} p(s_j|\vartheta,\varpi) \propto \frac{\mu^{\lambda}\exp(-\mu/\vartheta)}{\Gamma(\lambda)\vartheta^{\lambda+1}}\prod_{j=1}^{M}\frac{\varpi^{\vartheta}\exp(-\varpi/s_j)}{\Gamma(\vartheta)s_j^{\vartheta+1}} \tag{23}$$

$$p(\varpi|\ldots) \propto p(\varpi|\phi)\prod_{j=1}^{M} p(s_j|\vartheta,\varpi) \propto \phi\exp(-\phi\varpi)\prod_{j=1}^{M}\frac{\varpi^{\vartheta}\exp(-\varpi/s_j)}{\Gamma(\vartheta)s_j^{\vartheta+1}} \tag{24}$$

For the number of components M, which has no particular reason to be fixed in advance, we take as a prior a common choice which is a Uniform $\{1,\ldots,\sigma\}$ distribution, where σ is a constant representing the maximum value allowed for M. Our hierarchical model can be displayed as a directed acyclic graph (DAG) as shown in Fig. 3.1.

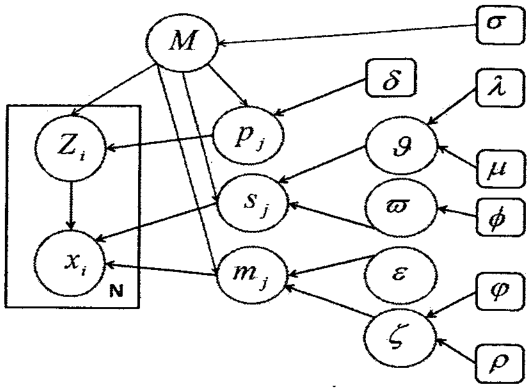


Figure 3.1: Graphical Model representation of the Bayesian Hierarchical finite general Beta mixture model. Nodes in this graph represent random variables, rounded boxes are fixed hyperparameters, boxes indicate repetition (with the number of repetitions in the lower right) and arcs describe conditional dependencies between variables.

3.3 Reversible Jump MCMC Algorithm

3.3.1 RJMCMC Move Types

Let Δ_M denotes the complete set of unknown variables (i.e the sate variable),

$$\Delta_M = (Z, P, M, \xi, \vartheta, \varpi, \varepsilon, \zeta).$$

We consider also a countable family of move types, indexed by $t = 1, 2, \ldots$. In our case, and following [78], the moves consist of: (1) updating the mixing parameters, (2) updating the parameters s and m, (3) updating Z, (4) updating the hyperparameters $\vartheta, \varpi, \varepsilon, \zeta$, (5) splitting one component into two, or merging two into one, (6) the birth or death of an empty component. In [78] *a sweep* is defined as a complete pass over the six moves and is considered as the basis time step of the complete learning algorithm. The first four moves do not change the dimensionality of the parameter vector and are actually classic Gibbs sampling moves. Note, however, that moves (5) and (6) necessitate changing (ξ, P, Z) and changing M by 1. The MCMC step representing move (5) takes the form of a Metropolis-Hastings step by proposing a move from a state Δ_M to Δ'_M with a target probability distribution (posterior distribution) $p(\Delta_M|\mathcal{X})$ and proposal distribution $q_t(\Delta_M, \Delta'_M)$ for the move t. When the current state is Δ_M, a given move t to destination Δ'_M is accepted with probability

$$\pi_t(\Delta_M, \Delta'_M) = \min\left\{1, \frac{p(\Delta'_M|\mathcal{X})q_t(\Delta'_M, \Delta_M)}{p(\Delta_M|\mathcal{X})q_t(\Delta_M, \Delta'_M)}\right\} \tag{25}$$

When we have a move, lying in a higher dimensional space, from a state Δ_M to another state Δ'_M, it is possible to implement this move by drawing a vector of continuous random variables u, independent of Δ_M [78]. And the new Δ'_M state is set through an invertible deterministic function of Δ_M and u: $f(\Delta_M, u)$. Thus, the move acceptance probability is given by

$$\pi_t(\Delta_M, \Delta'_M) = \min\left\{1, \frac{p(\Delta'_M|\mathcal{X})r_t(\Delta'_M)}{p(\Delta_M|\mathcal{X})r_t(\Delta_M)q(u)}\left|\frac{\partial\Delta'_M}{\partial(\Delta_M, u)}\right|\right\} \tag{26}$$

where $r_t(\Delta_M)$ is the probability of choosing move type t when in state Δ_M, $q(u)$ is the density function of u and $\left|\frac{\partial\Delta'_M}{\partial(\Delta_M, u)}\right|$ is the Jacobian function arising from the variable change from (Δ_M, u) to Δ'_M.

3.3.2 Implementation of the Moves

Gibbs Sampling Moves

As we mentioned above the first four moves are classic Gibbs sampling moves. For the first move the mixing parameters are generated from equation 15. The second move is based on the generation of m_j and s_j. It is noteworthy that the sampling of m_j and s_j is more complex, since the conditional posteriors given by equations 11 and 12 do not have known forms. Thus, we have used the Metropolis-Hastings algorithm (M-H) (see, for instance, [102], for a detailed introductory exposition and discussions). At iteration t, the steps of the M-H algorithm, to generate s_j, can be described as follows

1. Generate $\tilde{s}_j \sim q(s_j|s_j^{(t-1)})$ and $u \sim \mathcal{U}_{[0,1]}$
2. Compute $r = \frac{p(\tilde{s}_j|\ldots)q(s_j^{(t-1)}|\tilde{s}_j)}{p(s_j^{(t-1)}|\ldots)q(\tilde{s}_j|s_j^{(t-1)})}$

3. If $r < u$ then $s_j^{(t)} = \tilde{s}_j$ else $s_j^{(t)} = s_j^{(t-1)}$

The major problem in this algorithm is the need to choose the proposal distribution q. The most generic proposal is the random walk Metropolis-Hastings algorithm where each unconstrained parameter is the mean of the proposal distribution for the new value. As $\tilde{s}_j > 0$, we have chosen the following proposal $\tilde{s}_j \sim \mathcal{LN}(\log(s_j^{(t-1)}), e^2)$, where $\mathcal{LN}(\log(s_j^{(t-1)}), e^2)$ refers to the log-normal distribution with mean $\log(s_j^{(t-1)})$ and variance e^2. Note that this is actually equivalent to $\log(\tilde{s}_j) = \log(s_j^{(t-1)}) + \epsilon_j$, where $\epsilon_j \sim \mathcal{N}(0, e^2)$. In the case of m_j we have opted for general Beta proposals, centered at the current values, to assure that $m_j \in [a, b]$. With these proposals the M-H algorithm, to generate m_j, is composed of the following steps:

1. Generate $\tilde{m}_j \sim \mathcal{B}(m_j^{(t-1)}, S)$ and $u \sim \mathcal{U}_{[0,1]}$.
2. Compute $r = \frac{p(\tilde{m}_j|\ldots)\mathcal{B}(m_j^{(t-1)}|\tilde{m}_j, S)}{p(m_j^{(t-1)}|\ldots)\mathcal{B}(\tilde{m}_j|m_j^{(t-1)}, S)}$
3. If $r < u$ then $m_j^{(t)} = \tilde{m}_j$ else $m_j^{(t)} = m_j^{(t-1)}$

where $\mathcal{B}(m_{jl}^{(t-1)}, S)$ is a general Beta distribution with location $m_j^{(t-1)}$ and scale S. The third move is based on the generation of the missing data $Z_i, i = 1, \ldots, N$ from standard uniform random variables r_n, where $Z_i = j$ if $p(Z_i = 1|\ldots) + \ldots, p(Z_i = j-1|\ldots) < r_n \leq p(Z_i = 1|\ldots) + \ldots + p(Z_i = j|\ldots)$ (see equation 16) [80]. The fourth move consists of updating the hyperparameters $\vartheta, \varpi, \varepsilon, \zeta$. The posterior distribution of ε, ζ, ϑ and ϖ, given by equations 19, 20, 23 and 24, respectively, are not of standard forms. However, it is possible to show that they are log-concave [103] (i.e it is straightforward to show that the second derivatives of the logarithms of these functions are negative), then the samples generation is based on the adaptive rejection sampling (ARS) [104].

Split and Merge Moves

In move (5), we have to choose between splitting or merging a given component with probabilities a_M and $b_M = 1 - a_M$, respectively, depending on M. Note that $b_1 = 0$ and $a_\sigma = 0$ (recall that

σ is a constant representing the maximum value allowed for M), otherwise $a_M = b_M = 0.5$. The merging proposal works as follows: choose two components j_1 and j_2, where $m_{j_1} < m_{j_2}$ with no other $m_j \in [m_{j_1}, m_{j_2}]$ (i.e adjacency condition). If these components are merged, we reduce M by 1, which forms a new components $j*$ containing all the observation previously allocated to j_1 and j_2 and then creates values for p_{j*}, s_{j*}, m_{j*}, by preserving the first two moments, as follows (see Appendix B)

$$p_{j*} = p_{j_1} + p_{j_2} \tag{27}$$

$$m_{j*} = \frac{p_{j_1} m_{j_1} + p_{j_2} m_{j_2}}{p_{j_1} + p_{j_2}} \tag{28}$$

$$s_{j*} = \frac{p_{j*}(m_{j*} - a)(b - m_{j*})}{p_{j_1}\left(m_{j_1}^2 + \frac{(m_{j_1}-a)(b-m_{j_1})}{s_{j_1}+1}\right) + p_{j_2}\left(m_{j_2}^2 + \frac{(m_{j_2}-a)(b-m_{j_2})}{s_{j_2}+1}\right) - p_{j*}m_{j*}^2} - 1 \tag{29}$$

When the decision is to split, we choose a component $j*$ at random to define two new components j_1 and j_2 having weights and parameters $(p_{j_1}, m_{j_1}, s_{j_1})$ and $(p_{j_2}, m_{j_2}, s_{j_2})$, respectively, conforming to equations 27, 28 and 29. According to this transformation, there are 3 degrees of freedom, thus we need to generate 3 random numbers $u = (u_1, u_2, u_3)$ drawn from Beta distributions with parameters $(2,2)$, $(2,2)$ and $(1,1)$, respectively [78]. The split transformations are thus defined as following (see Appendix C)

$$p_{j_1} = u_1 p_{j*} \qquad p_{j_2} = (1-u_1)p_{j*} \tag{30}$$

$$m_{j_1} = m_{j*} - u_2\sqrt{\frac{(m_{j*}-a)(b-m_{j*})p_{j_2}}{(s_{j*}+1)p_{j_1}}} \qquad m_{j_2} = m_{j*} + u_2\sqrt{\frac{(m_{j*}-a)(b-m_{j*})p_{j_1}}{(s_{j*}+1)p_{j_2}}} \tag{31}$$

$$s_{j_1} = \frac{(m_{j_1}-a)(b-m_{j_1})}{u_3(1-u_2^2)\frac{(m_{j*}-a)(b-m_{j*})}{s_{j*}+1}\frac{p_{j*}}{p_{j_1}}} - 1 \qquad s_{j_2} = \frac{(m_{j_2}-a)(b-m_{j_2})}{(1-u_3)(1-u_2^2)\frac{(m_{j*}-a)(b-m_{j*})}{s_{j*}+1}\frac{p_{j*}}{p_{j_2}}} - 1 \tag{32}$$

Note that we have also to check the adjacency condition previously defined for the split move. If this condition passed, then we assign the different x_i previously in $j*$ in j_1 or j_2 using equation 16 (i.e Bayes rule). If the condition is not satisfied, we reject the move in order to preserve the reversibility of split/combine moves.

Now, we calculate the acceptance probabilities of split and combine moves: $\min\{1, A\}$ and $\min\{1, A^{-1}\}$, where we have the following according to equation 26:

$$A = \frac{p(Z, P, M+1, \xi, \vartheta, \varpi, \varepsilon, \zeta|\mathcal{X})b_{M+1}}{p(Z, P, M, \xi, \vartheta, \varpi, \varepsilon, \zeta|\mathcal{X})a_M P_{alloc} q(u)} \left|\frac{\partial \Delta'_M}{\partial(\Delta_M, u)}\right| \tag{33}$$

where P_{alloc} is the probability of making this particular current allocation of data to components j_1 and j_2:

$$\begin{aligned} P_{alloc} &= \prod_{Z_i=j_1} \frac{p_{j_1} p(x_i|m_{j_1}, s_{j_1})}{p_{j_1} p(x_i|m_{j_1}, s_{j_1}) + p_{j_2} p(x_i|m_{j_2}, s_{j_2})} \prod_{Z_i=j_2} \frac{p_{j_2} p(x_i|m_{j_2}, s_{j_2})}{p_{j_1} p(x_i|m_{j_1}, s_{j_1}) + p_{j_2} p(x_i|m_{j_2}, s_{j_2})} \\ &= \frac{\prod_{Z_i=j_1} p_{j_1} p(x_i|m_{j_1}, s_{j_1}) \prod_{Z_i=j_2} p_{j_2} p(x_i|m_{j_2}, s_{j_2})}{\prod_{Z_i=j*} p_{j_1} p(x_i|m_{j_1}, s_{j_1}) + p_{j_2} p(x_i|m_{j_2}, s_{j_2})} \end{aligned} \tag{34}$$

$$q(u) = p(u_1)p(u_2)p(u_3) \tag{35}$$

$\frac{p(Z,P,M+1,\xi,\vartheta,\varpi,\varepsilon,\zeta|\mathcal{X})}{p(Z,P,M,\xi,\vartheta,\varpi,\varepsilon,\zeta|\mathcal{X})}$ is developed in Appendix D and $\left|\frac{\partial \Delta'_M}{\partial(\Delta_M, u)}\right|$ is given by

$$\left|\frac{\partial \Delta'_M}{\partial(\Delta_M, u)}\right| = \left|\frac{\partial(p_{j_1}, p_{j_2}, m_{j_1}, m_{j_2}, s_{j_1}, s_{j_2})}{\partial(u_1, p_{j*}, m_{j*}, u_2, s_{j*}, u_3)}\right| = p_{j*} \frac{(m_{j_2} - m_{j_1})(s_{j_1}+1)(s_{j_2}+1)}{u_2(1-u_2^2)u_3(1-u_3)(s_{j*}+1)} \tag{36}$$

which is the Jacobian that arises from transforming $(p_{j*}, u_1, m_{j*}, u_2, s_{j*}, u_3)$ to $(p_{j_1}, p_{j_2}, m_{j_1}, m_{j_2}, s_{j_1}, s_{j_2})$ and is developed in Appendix E.

Birth and Death Moves

In move (6), birth-and-death, the first step is to choose randomly between birth and death with probabilities a_M and b_M as above. The birth step consists in adding a new Beta component in the mixture by generating its parameters, m_{j*} and s_{j*}, from the associated prior distributions given by equations 8 and 9, respectively. The weight of the new component, p_{j*}, is generated from the marginal distribution of p_{j*} derived from the distribution of $P = (p_1, \ldots, p_M, p_{j*})$. The vector P follows a Dirichlet with parameters $(\delta_1, \ldots, \delta_M, \delta_{j*})$ (see equation 13), thus the marginal of p_{j*} is a Beta distribution with parameters $(\delta_{j*}, \sum_{j=1}^{M} \delta_j)$ [95]. Note that in order to keep the mixture constraint $\sum_{j=1}^{M} p_j + p_{j*} = 1$, the previous weights $p_j, j = 1, \ldots, M$ have to be rescaled and then all multiplied by $(1 - p_{j*})$. The Jacobian corresponding to the birth move is then $(1 - p_{j*})^M$. For the opposite move, we choose randomly an existing empty component to delete, then of course the remaining weights have to be rescaled to keep the unit-sum constraint. The acceptance probabilities of birth and death moves: $\min\{1, A\}$ and $\min\{1, A^{-1}\}$, are calculated according to equation 26 (see Appendix F):

$$A = \frac{p(M+1)}{p(M)} \frac{\Gamma(\delta_{j*} + \sum_{j=1}^{M} \delta_j)}{\Gamma(\delta_{j*})\Gamma(\sum_{j=1}^{M} \delta_j)} p_{j*}^{\delta_{j*}-1} (1-p_{j*})^{N+\sum_{j=1}^{M} \delta_j - M} (M+1) \frac{b_{M+1}}{a_M(M_0+1)} \frac{1}{p(p_{j*})} (1-p_{j*})^M \tag{37}$$

where M_0 is the number of empty components before the birth.

3.4 Experimental results

In this section we report results on different interesting applications. In the first application, we briefly discuss the results obtained with some artificially generated data sets. The discussion will not be very detailed since, the experiments with generated data are not as significant as those with real data. We investigate the effectiveness of our algorithm by applying it on four well-known real data sets, while comparing it to the RJMCMC in the case of Gaussian mixture model [78] in the second application. Last but not least, we demonstrate the usefulness of our algorithm for texture image classification and retrieval. In these applications our specific choices for the hyperparameters were $\eta_1 = \ldots, \eta_M = 1$, $(\varphi, \varrho, \lambda, \mu, \phi)$=(2,5,0.2,2,1), S and e^2 (in the M-H algorithms) were set to 2 and 0.01, respectively, and σ (the maximum value allowed for M) was set to 30.

3.4.1 Synthetic Data Sets

We dedicate this section for the analysis of generated data. The goal of this section is to investigate if our algorithm is able to: estimate the mixture parameters and select the number of clusters effectively. We generated 100 data sets using different parameters and number of clusters, in order

Table 3.1: Summary of the results for the 100 generated data sets. $\hat{M}$ denotes the obtained number of clusters.

Number of clusters M	Number of datasets	$\hat{M}=2$	$\hat{M}=3$	$\hat{M}=4$	$\hat{M}=5$	$\hat{M}=6$	$\hat{M}=7$
M=2	25	100%	0%	0%	0%	0%	0%
M=3	20	0%	100%	0%	0%	0%	0%
M=4	15	0%	6.67%	92.23%	0%	0%	0%
M=5	15	0%	0%	0%	100%	0%	0%
M=6	15	0%	0%	6.67%	6.67%	86.66%	0%
M=7	10	0%	0%	0%	0%	10%	90%

Table 3.2: Parameters of four different generated data sets. N represents the number of elements in each data set. m_j, s_j, and p_j are the real parameters. $\hat{m}_j$, $\hat{s}_j$, and $\hat{p}_j$ are the estimated parameters.

	j	m_j	s_j	p_j	$\hat{m}_j$	$\hat{s}_j$	$\hat{p}_j$
Data 1 (N=3365)	1	2.00	10.00	0.60	1.94	12.31	0.57
	2	5.00	19.00	0.40	4.90	17.71	0.43
Data 2 (N=3647)	1	1.00	12.00	0.35	0.90	15.11	0.34
	2	3.50	22.00	0.35	3.35	26.00	0.35
	3	6.00	13.00	0.30	5.90	16.57	0.31
Data 3 (N=3703)	1	1.00	15.00	0.10	1.17	18.90	0.11
	2	3.00	14.00	0.30	3.01	12.70	0.28
	3	5.00	19.00	0.20	4.85	18.79	0.20
	4	6.50	17.00	0.40	6.51	23.40	0.41
Data 4 (N=3706)	1	1.00	15.00	0.10	1.11	19.22	0.12
	2	2.00	25.00	0.20	2.03	23.91	0.19
	3	4.00	14.00	0.30	3.82	15.83	0.28
	4	5.00	19.00	0.20	5.14	19.05	0.21
	5	6.50	17.00	0.20	6.53	16.69	0.20

to investigate the method on a wide range of data. Applying our methods on these data sets, we found that it was able to identify the right number of clusters in 96% of the cases as shown in table 3.1. For the 4% wrongly modeled data sets, we can notice that they were fitted with a smaller number of components. It is noteworthy that in each of these four cases the probabilities for the right number of components were quite close to the ones chosen. We choose four data sets for more in details inspection. The real and estimated parameters for these data sets are given in table 3.2, and the real and estimated histograms are drawn in Fig. 3.2. According to these results it is clear that our algorithm has very good learning capabilities.

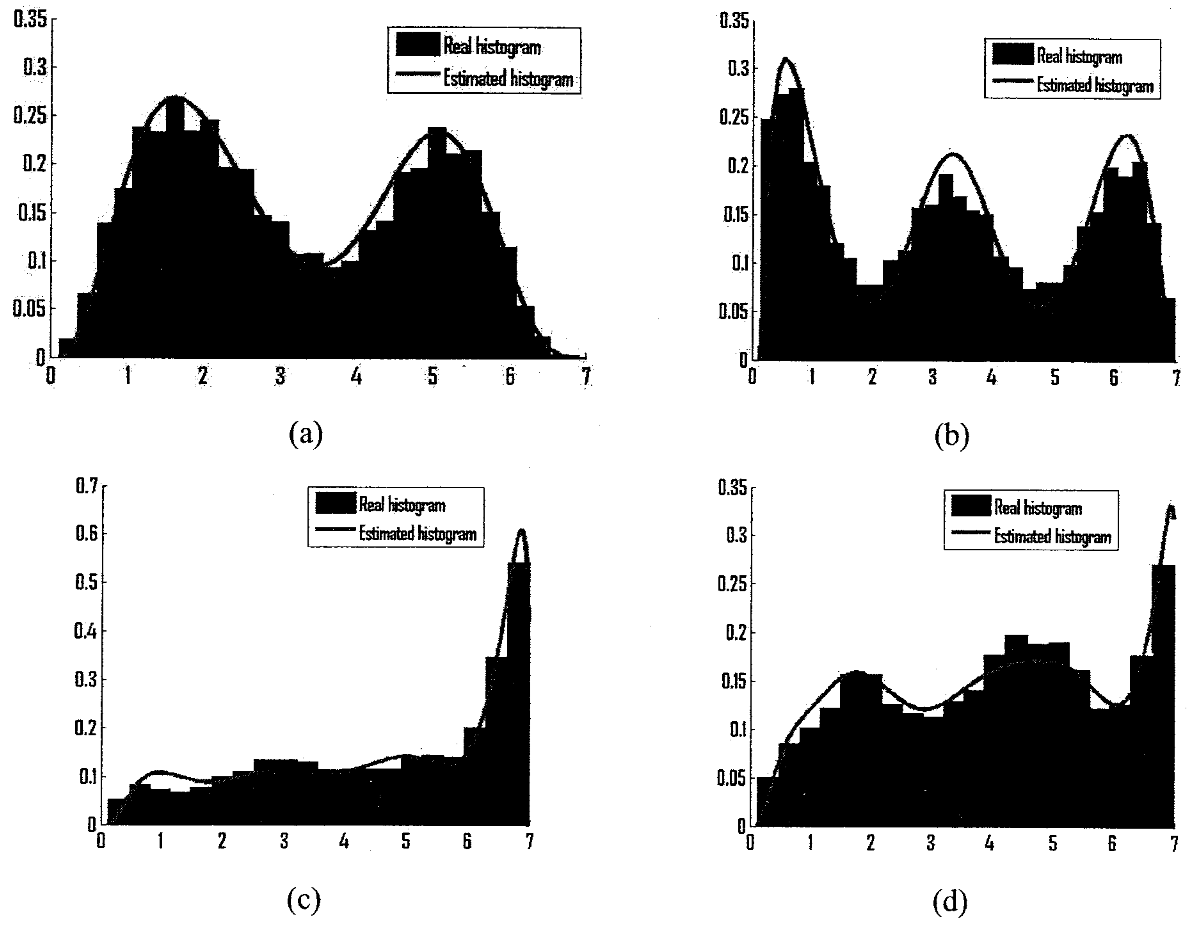


Figure 3.2: Real and estimated histograms for four generated data sets. (a) a 2 components mixture, (b) a 3 components mixture, (c) a 4 components mixture, (d) a 5 components mixture.

3.4.2 Real Data Sets

We devote this section to real data modeling and analysis. We apply our algorithm on four standard widely used data sets: enzyme, acidity, galaxy, and stamp [4]. The first data describes an enzymatic activity in the blood among a group of 245 unrelated individuals, and the second one is an acidity index measured in a sample of 155 lakes in the northeastern United States. The third one consists

[4]http://www.maths.uq.edu.au/~gjm/DATA/mmdata.html

Table 3.3: Estimated posterior probabilities of the number of components given the data for the four data sets, with percentage of accepted Split-Combine, and Birth-Death moves.

Data set	N	$p(k\|\mathcal{X})$	Proportion (%) of Split -Combine moves accepted	Proportion (%) of Birth -Death moves accepted
Enzyme	245	$p(1\|\mathcal{X}) = 0.0000\, p(2\|\mathcal{X}) = 0.1712\, p(3\|\mathcal{X}) = 0.4152\, p(4\|\mathcal{X}) = 0.3782$ $p(5\|\mathcal{X}) = 0.0341 \sum_{k>6}^{\sigma} p(k\|\mathcal{X}) = 0.0013$	6.68%	3.02%
Acidity	155	$p(1\|\mathcal{X}) = 0.0000\, p(2\|\mathcal{X}) = 0.4307\, p(3\|\mathcal{X}) = 0.3354\, p(4\|\mathcal{X}) = 0.1942$ $p(5\|\mathcal{X}) = 0.0231 p(6\|\mathcal{X}) = 0.0154 \sum_{k>7}^{\sigma} p(k\|\mathcal{X}) = 0.0012$	10.17%	7.52%
Galaxy	82	$p(1\|\mathcal{X}) = 0.0000\, p(2\|\mathcal{X}) = 0.0010\, p(3\|\mathcal{X}) = 0.0210\, p(4\|\mathcal{X}) = 0.2031$ $p(5\|\mathcal{X}) = 0.3671 p(6\|\mathcal{X}) = 0.0798\, p(7\|\mathcal{X}) = 0.3167\, p(8\|\mathcal{X}) = 0.0094$ $\sum_{k>9}^{\sigma} p(k) = 0.0019$	9.32%	16.71%
Stamp	485	$p(1\|\mathcal{X}) = 0.0000\, p(2\|\mathcal{X}) = 0.0000\, p(3\|\mathcal{X}) = 0.0001\, p(4\|\mathcal{X}) = 0.5612$ $p(5\|\mathcal{X}) = 0.3574 p(6\|\mathcal{X}) = 0.0231\, p(7\|\mathcal{X}) = 0.0556\, p(8\|\mathcal{X}) = 0.0012$ $\sum_{k>9}^{\sigma} p(k\|\mathcal{X}) = 0.0014$	4.87%	2.16%

of the velocities of 82 distant galaxies, diverging from our own galaxy, as for the last data set it consists of thickness of 485 postage stamps produced in Mexico. The enzyme data set was analyzed in several research papers such as in [105] where it was modeled by two skewed distributions, and in [78] where the use of three to five Gaussian components was favored. For the acidity data and Galaxy data sets, three to five components were generally identified [78]. The 1872 Hidalgo postage stamps of Mexico data set was introduced in [106] and has been used in several research papers (see, for instance, [107, 108]) which identified seven and three components Gaussian model with equal and unequal variances, respectively. Using these four datasets we compared our model to the one in [78]. In all the runs the number of components has never exceeded fifteen. Estimated posterior probabilities of the number of components given the data for the four data are given in table 3.3. For the enzyme data our algorithm favors 3-5 components with maximum posterior probability for three components, same as the GMM, this is due to the fact that the enzyme data are not skewed (see Fig. 3.3) . For the acidity data set a mixture of two components was chosen as shown in Fig. 3.4. For the galaxy and stamp data sets the data are highly skewed and spread which force the algorithm to use a higher number of components, for this reason our algorithm supports the use of five components for both data sets (See Figs. 3.5 and 3.6). In each case, we can relate the number of components to the skewness to the data. Also note that the general Beta can model skewed data which is not the case for the Gaussian, and this advantage is demonstrated for the acidity and galaxy datasets where our algorithm favors the use of two and five components, respectively,

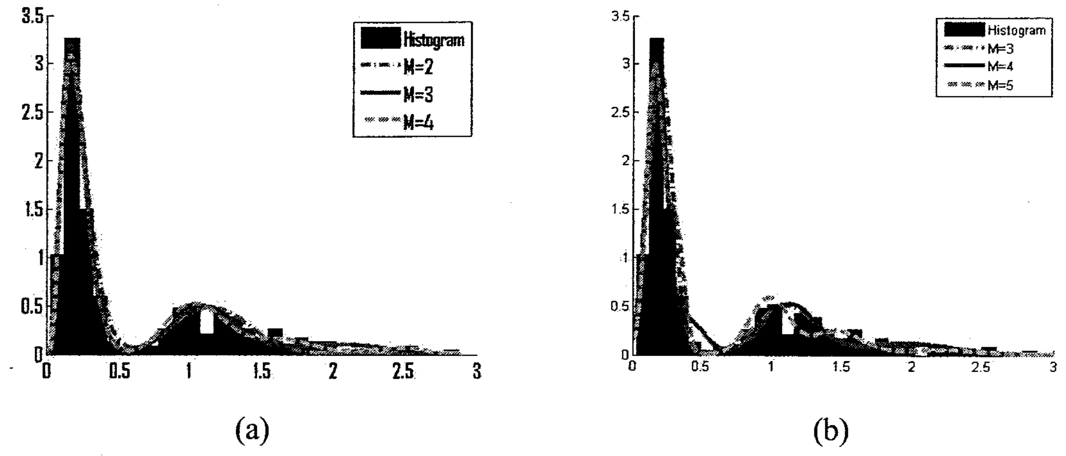


Figure 3.3: Enzyme data modeling when considering the mixtures with the highest probabilities. (a) Beta mixture models, (b) Gaussian mixture models.

compared to three and six for the GMM (See table 3.5). According to the experiments presented here it is clear that the general Beta mixture model outperforms the Gaussian one, by representing the data effectively with less number of components. This result was already expected due to the fact that the general Beta mixture model is more flexible which helps it to represent highly spread and hard to model data.

3.4.3 Texture Images Classification and Retrieval

Approach

An interesting difficult problem in image processing is texture analysis. Indeed, texture provides important characteristics for surface and object identification (depth and orientation, for instance) in many types of images (satellite, medical, etc.) and plays an important role in several applications such as content-based image categorization, browsing and retrieval [109]. Methods for texture analysis can be grouped into four major categories: statistical, geometrical, model based and signal processing approaches [58]. An efficient technique for analyzing image textures is the

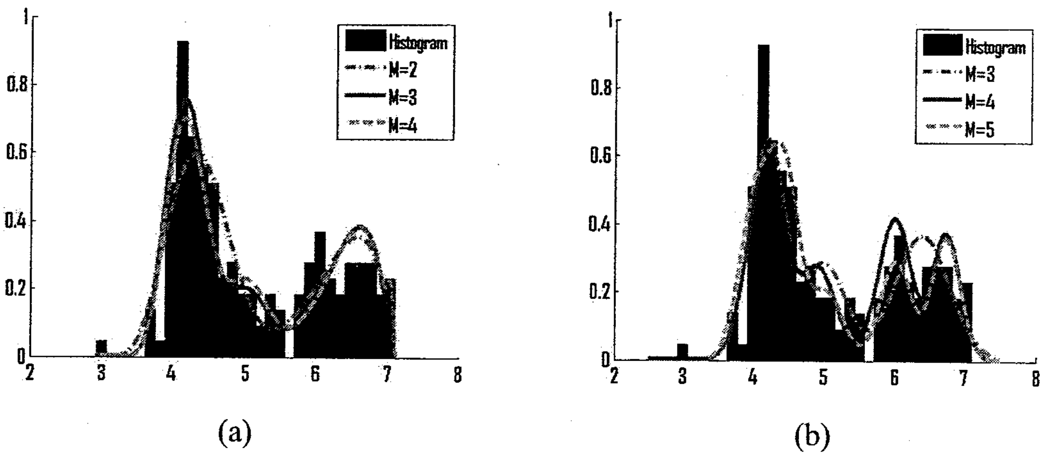


Figure 3.4: Acidity data modeling when considering the mixtures with the highest probabilities. (a) Beta mixture models, (b) Gaussian mixture models.

Table 3.4: Parameters of the mixture models representing the different tested real data sets. j component number. m_j, s_j, and p_j are the real parameters. $\hat{m}_j$, $\hat{s}_j$, and $\hat{p}_j^\beta$ are the Beta mixture estimated parameters. $\hat{\mu}_j$, $\hat{\sigma}_j^2$, and $\hat{p}_j^g$ are the Gaussian mixture estimated parameters

	j	$\hat{m}_j$	$\hat{s}_j$	$\hat{p}_j^\beta$	$\hat{\mu}_j$	$\hat{\sigma}_j^2$	$\hat{p}_j^g$
Enzyme	1	0.1960	64.1054	0.6450	0.1962	0.0078	0.6431
	2	1.1008	42.9680	0.2630	1.1006	0.0448	0.2637
	3	1.9492	13.4360	0.0920	1.9492	0.1271	0.0932
Acidity	1	4.3615	23.7136	0.3500	4.1865	0.0691	0.4240
	2	6.3149	11.8544	0.3854	5.0168	0.0868	0.2086
	3				6.3926	0.1593	0.3674
Galaxy	1	9.7101	69.0788	0.0854	9.7101	0.1931	0.0988
	2	16.1737	106.9899	0.0320	17.5649	1.5392	0.1006
	3	20.0898	109.0967	0.5366	19.9782	0.3540	0.3416
	4	24.1064	54.3933	0.3095	22.7039	0.4731	0.2692
	5	32.9518	28.1897	0.0366	25.1517	1.2112	0.1655
	6				33.0427	1.0256	0.0243
Stamp	1	0.0705	103.0930	0.2202	0.0710	0.2320×10^{-4}	0.2133
	2	0.0803	129.4502	0.4270	0.0789	0.2392×10^{-4}	0.4152
	3	0.0971	57.5257	0.2094	0.0907	0.2709×10^{-4}	0.1015
	4	0.1140	28.4719	0.1168	0.1016	0.3080×10^{-4}	0.1666
	5	0.1284	580.7500	0.0265	0.1157	0.7166×10^{-4}	0.1035

multichannel decomposition approach, using the assumption that the energy distribution in the frequency domain identifies texture, based on Gabor filters [59, 110–112], wavelet transforms [113–116] and steerable pyramids [61–63]. A detailed survey and intersting discussions can be found in [117], also. The texture can then be modeled by the marginal densities of the coefficients of

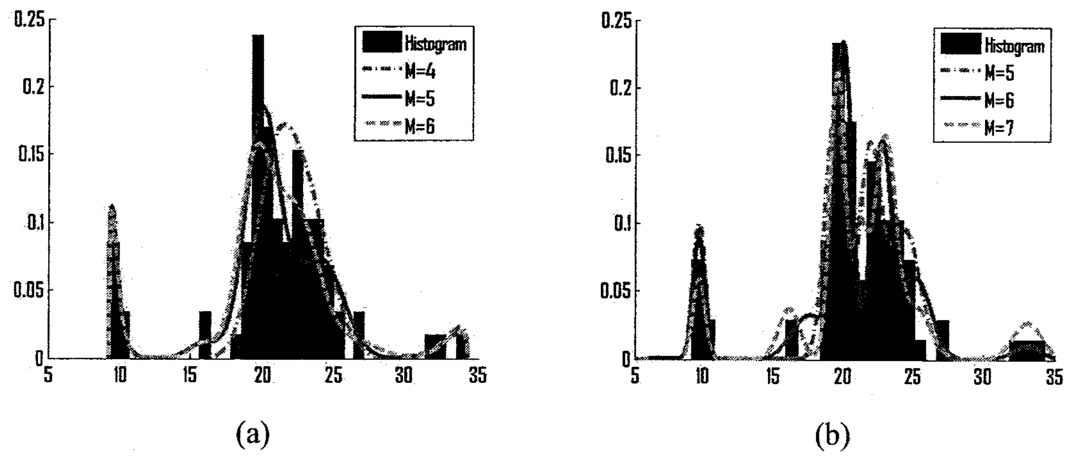
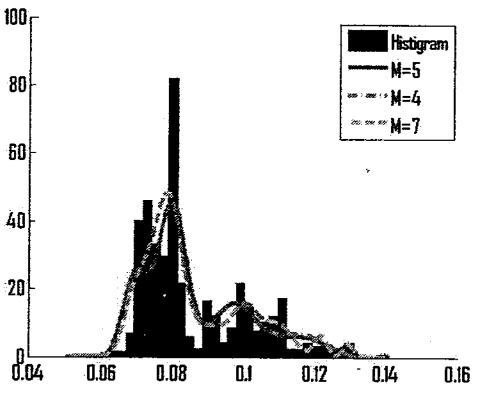
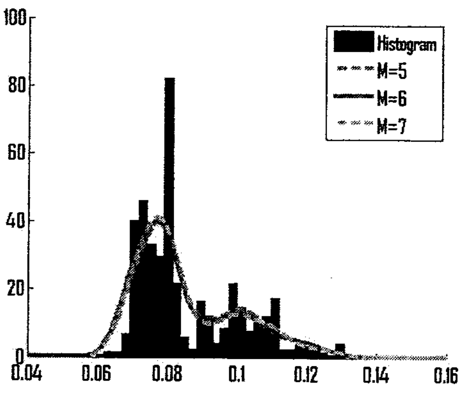


Figure 3.5: Galaxy data modeling when considering the mixtures with the highest probabilities. (a) Beta mixture models, (b) Gaussian mixture models.

the resulted filtered subband images which allows a more compact representation than histograms which necessitate many parameters (hundreds). This approach is also justified by some psychological researches on human texture perception which have shown that textures producing similar marginal densities are very difficult to discriminate [29]. It is noteworthy that the sub-band marginal densities are generally non-Gaussian especially for natural images texture [118]. In this section we propose an approach for texture images classification and retrieval based on our finite general Beta mixture model. In our classification framework, an image texture is first transformed to gray scale and decomposed into sub-bands using steerable filters [62, 63]. Figure 3.7 shows a texture image and its multiscale version in a pyramid hierarchy. The histograms of the resulted filtered images are show in Fig. 3.8 which shows clearly that the Gaussian assumption would be inappropriate. Then, each sub-band's marginal density is approximated by a finite general Beta mixture model using our Bayesian learning algorithm (See Fig. 3.9). As a result each texture image will be represented by a set of finite general Beta mixture models which can be viewed as the signatures of the image. Finally the Earth Mover's Distance (EMD) [64] is used to measure the distribution similarity between a set of components representing an input image texture (ie. test

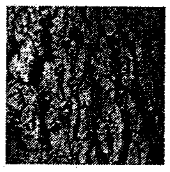


(a)

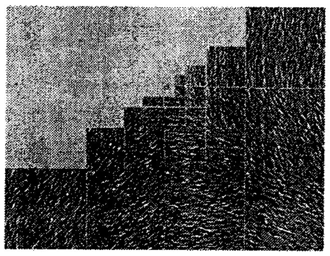


(b)

Figure 3.6: Stamp data modeling when considering the mixtures with the highest probabilities. (a) Beta mixture models, (b) Gaussian mixture models.



(a)



(b)

Figure 3.7: Original image and its steerable pyramid decomposition. (a) Original image from Bark group in Vistex, (b) Sub-images output using five level steerable pyramid.

image) and sets of components representing texture classes (i.e training images). In our case, EMD can be viewed as the minimum cost of changing one mixture into another, when the cost of moving probability mass from components in the first mixture to components in the second mixture is calculated using Kullback-Leibler (KL) divergence given by:

$$D(f_i||g_j) = \int f_i(x)\log(\frac{f_i(x)}{g_j(x)})dx \quad (38)$$

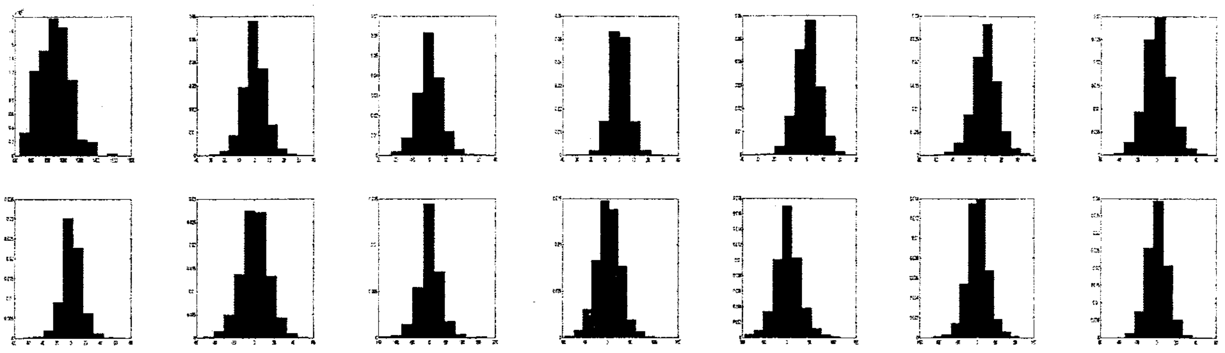


Figure 3.8: Histograms of the 14 sub-images of the steerable pyramid.

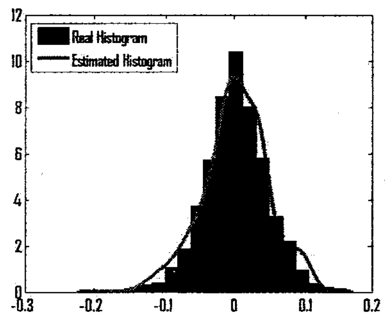


Figure 3.9: A sub-image histogram fitted by a Beta mixture model.

Where f_i is the component i of the input sub-image mixture which we suppose that it has m components with weights p_{fi}, and g_j is the component j of the class sub-image mixture which we suppose that it has n components with weights p_{gj}. For two general Beta distributions f_i and g_j the KL divergence has a closed form expression and we can show that is given by (see Appendix G)

$$\begin{aligned} D(f_i \| g_j) &= \log\left[\frac{\Gamma(\alpha_i+\beta_i)\Gamma(\alpha_j)\Gamma(\beta_j)}{(b-a)^{\alpha_i+\beta_i-\alpha_j-\beta_j}\Gamma(\alpha_j+\beta_j)\Gamma(\alpha_i)\Gamma(\beta_i)}\right] - (\beta_j-\beta_i+\alpha_j-\alpha_i)\log(b-a) \\ &+ (\alpha_j-\alpha_i+\beta_j-\beta_i)\Psi(\alpha_i+\beta_i) - (\alpha_j-\alpha_i)\Psi(\alpha_i) - (\beta_j-\beta_i)\Psi(\beta_i) \qquad (39) \end{aligned}$$

Where (α_i, β_i) are the parameters of f_i, (α_j, β_j) are the parameters of g_j, and Ψ is the digamma function. With the KL divergence in hand we have to start the minimization problem in which we need to get the $m \times n$ matrix F, where f_{ij} is the amount of weight p_{fi} matched to p_{gj}, that will minimize the following equation

$$EMD_{sub} = \sum_{i=1}^{m} \sum_{j=1}^{n} f_{ij} D(f_i || g_j) \tag{40}$$

and subjected to the following constraints:(1) $f_{ij} \geq 0$, where $1 \leq i \leq m$ and $1 \leq j \leq n$, (2) $\sum_{i=1}^{m} f_{ij} = p_{gj}$, where $1 \leq j \leq n$, (3) $\sum_{j=1}^{n} f_{ij} = p_{fi}$, where $1 \leq i \leq m$, (4) $\sum_{i=1}^{m} \sum_{j=1}^{n} f_{ij} = min(\sum_{i=1}^{m} p_{fi}, \sum_{j=1}^{n} p_{gj}) = 1$. Note that when the image texture is decomposed of L sub-bands, then the total EMD is the sum of that of each sub-band, $EMD = \sum_{l=1}^{L} EMD_{sub_l}$. By computing the EMD between the input texture image and each texture class, each image is affected to the class for which the EMD is the smallest.

Results

We performed our classifications experiments using the Vistex data set [5]. Six homogeneous texture groups (Bark, Fabric, Food, Metal, Water, and Sand) were considered (See Fig 3.10). We used four 512×512 images from each of the Bark, Fabric, and Metal texture groups, and six 512×512 from each of the Food, Water, and Sand texture groups, then we divided each image into sixty four 64×64 subimages. Thus, we obtained a total of 256 subimages for each class in the first three groups, and 384 subimages for each class in the second three groups. We then applied our classification approach 10 times, each time using 24 subimages of each original texture image for training and the remaining 40 for testing. This brought us to a total of 720 images from all six groups as training samples for our algorithm, and 1200 as testing samples. Moreover, we applied our algorithm by using first three levels pyramid and second by using five levels pyramid. The classification results, when using both finite general Beta and Gaussian mixture models, are given in table 3.5. From these results we can observe that our algorithm has a higher accuracy then the Gaussian which is a further endorsement of our model. In addition, and as expected, the five levels pyramid improves the performance over the three levels pyramids, yet this improvement is very small compared to the enormous difference in computational time.

[5]MIT Vision and Modeling Group (http://vismod.www.media.mit.edu).

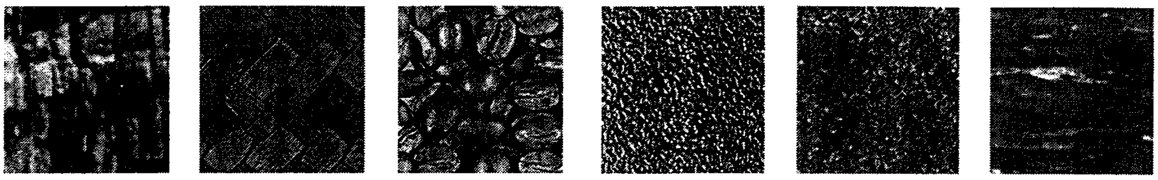


Figure 3.10: Sample images from each group. (a) Bark, (b) Fabric, (c) Food, (d) Metal, (e) Sand, (f) Water.

Table 3.5: The Average classification accuracy (%) of the two different methods.

Method	Using 3 levels pyramid	Using 5 levels pyramid
General Beta Mixture Models	92.50% ± 1.41%	93.58% ± 1.33%
Gaussian Mixture Models	91.67% ± 1.66%	92.58% ± 1.08%

We conducted another experiment designed to retrieve images similar to a given query. Our retrieval approach can be divided into two steps. First task, is the same as in the classification approach, since we have to choose the nearest texture group to the query. For the second step, we compared the input image (i.e, query) with the other images in the same group and retrieve the closest images to our query using the EMD. We applied our retrieval process twice former using three levels pyramid and latter using five levels pyramid. To measure the retrieval rates (precision and recall), each image was used as a query and the number of relevant images among those that were retrieved was noted. Table 3.6 presents the retrieval rates obtained in terms of precision when 64 images are retrieved each time in response to a query. Note that in this case the precision and recall are the same because for a given image we have at most 64 images which are similar to it. Figure 3.11(a) represents the average (averaged over all the queries) precision rate for different texture classes when we consider only the first 64 images retrieved. Figures 3.11(b) and 3.11(c) show two graphs illustrating the overall precision and recall, respectively, of our retrieval method when varying the total number of images retrieved taken into consideration.

Table 3.6: Average precsion rate (%) of the two different methods.

Method	Using 3 levels pyramid	Using 5 levels pyramid
General Beta Mixture Models	75.32%	78.12%
Gaussian Mixture Models	71.56%	73.32%

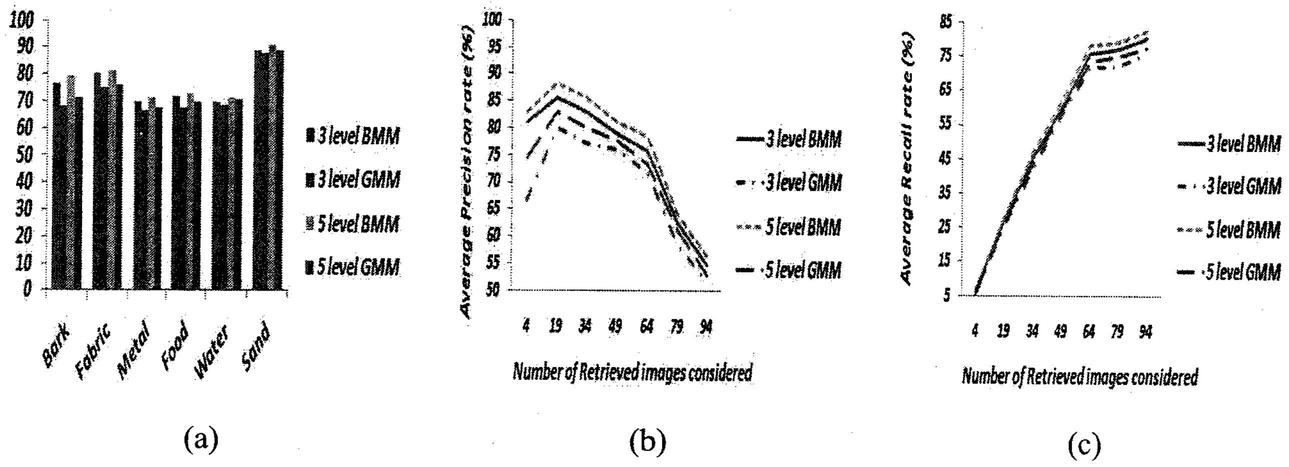


Figure 3.11: Precision and recall. (a) Average precision when 64 retrieved images are considered for each class, (b) Average precision when varying the number of retrieved images, (c) Average recall when varying the number of retrieved images.

3.5 Conclusion

We presented a fully Bayesian analysis, coupled with MCMC techniques, of finite Beta mixtures with unknown number of components. The proposed algorithm automatically handles the problem of the specification of the number of clusters on the basis of the RJMCMC approach, which allows varying the dimension of the mixture, by constructing split and merge moves that rely on moment matching. The results from applying the proposed model to different applications have been presented and justify further the recent interest on the use of Bayesian machinery in image processing. The finite Beta mixture has many appealing advantages that make it useful for a variety of image processing applications which require the modeling of non Gaussian data.

CHAPTER 4

Conclusions

The problem of clustering data into similar groups is one of the most widely studied problems and has applications in several domains and areas. Finite mixture models offer a formal approach to this problem. In order to use mixture models, three main points have to be identified: the choice of the probability density function, the approaches used for parameters estimation and the selection of the number of clusters. The majority of approaches previously proposed the use of the Gaussian density in the mixture modeling of data. However, it is certainly not always the best approximation especially in image and signal processing applications.

In this thesis, first we have proposed a new Bayesian approach for finite generalized Gaussian mixtures. The Monte Carlo simulation technique of Gibbs sampling mixed with a Metropolis-Hasting step was used to learn the model parameters. The integrated likelihood was used to estimate the number of clusters describing the data. In order to study the effectiveness of our algorithm, we have applied it in different image processing applications. The results not only demonstrate the reliability of the method, but also its capability to reach better estimates when compared to different Bayesian and deterministic methods.

Second, we have extended the application of RJMCMC sampler to the general Beta mixture. The proposed algorithm handles the problem of the specification of the number of clusters on the basis of the RJMCMC approach, which allows varying the dimension of the mixture, by constructing split and merge moves that rely on moment matching. We demonstrated the effectiveness of this approach when applied to signal and image processing applications.

In conclusion, compared to existing techniques, generally based on the Gaussian assumption, our approaches not only can model non-Gaussian data, but also can reach better approximation for the model parameters, and even a better selection for the number of clusters.

Over the last decade, the use of technological advances have brought an explosion of enormous multidimensional data. In order to analyze and understand these data, the use of multidimensional mixtures is needed. In future, we will try to apply both algorithms for the case of multidimensional data. As noted before, the estimation of the number of clusters that best describes the data without over- or under-fitting is one of the most challenging aspects when using mixture models. For this purpose, we will investigate the GGM approach in the infinite case.

APPENDIX A

Proof of Equations 11, 12, and 13

The derivation of (Eq. 11) is as follows:

$$\pi(\mu_j|Z,\mathcal{X}) \propto \pi(\mu_j) \prod_{Z_{ij=1}} p(\mathcal{X}|\mu_j,\alpha_j,\beta_j) \tag{1}$$

Thus,

$$\pi(\mu_j|Z,\mathcal{X}) \propto \frac{1}{\sqrt{2\pi}\sigma_0}\mathrm{e}^{\frac{(\mu_j-\mu_0)^2}{2\sigma_0^2}} \prod_{Z_{ij=1}} (\frac{\alpha_j\beta_j}{2\Gamma(1/\beta_j)}\mathrm{e}^{-(\alpha_j|\mathrm{x}_i-\mu_j|)^{\beta_j}}) \tag{2}$$

In this case we have σ_0 as a constant hyperparameter, also α_j, β_j are considered as constant parameters, which gives us

$$\pi(\mu_j|Z,\mathcal{X}) \propto \mathrm{e}^{\frac{(\mu_j-\mu_0)^2}{2\sigma_0^2}} \mathrm{e}^{\sum_{Z_{ij=1}}(-\alpha_j|x_i-\mu_j|)^{\beta_j}} \tag{3}$$

The derivation for (Eq. 12) is as follows:

$$\pi(\alpha_j|Z,\mathcal{X}) \propto \pi(\alpha_j) \prod_{Z_{ij=1}} p(\mathrm{x}_i|\mu_j,\alpha_j,\beta_j) \tag{4}$$

Thus,

$$\pi(\alpha_j|Z,\mathcal{X}) \propto \frac{\alpha_j^{\alpha_\alpha-1}\beta_\alpha^{\alpha_\alpha}\mathrm{e}^{-\beta_\alpha\alpha_j}}{\Gamma(\alpha_\alpha)} \prod_{Z_{ij=1}} (\frac{\alpha_j\beta_j}{2\Gamma(1/\beta_j)}\mathrm{e}^{-(\alpha_j|\mathrm{x}_i-\mu_j|)^{\beta_j}}) \tag{5}$$

In this case we have (α_α, β_α) are constant hyperparameters, also β_j is considered as a constant parameter, which gives us

$$\pi(\alpha_j|Z,\mathcal{X}) \propto \alpha_j^{\alpha_\alpha-1}\mathrm{e}^{-\beta_\alpha\alpha_j}(\alpha_j)^{n_j}\mathrm{e}^{\sum_{Z_{ij=1}}(-\alpha_j|x_i-\mu_j|)^{\beta_j}} \tag{6}$$

The derivation for (Eq. 13) is the same as for (Eq. 12). $(\alpha_\beta, \beta_\beta)$ are constant hyperparameters, and α_j is considered as a constant parameter.

$$\pi(\beta_j|Z,\mathcal{X}) \propto \beta_j^{\alpha_\beta-1}\mathrm{e}^{-\beta_\beta\beta_j}(\frac{\beta_j}{\Gamma(1/\beta_j)})^{n_j}\mathrm{e}^{\sum_{Z_{ij}=1}(-\alpha_j|x_i-\mu_j|)^{\beta_j}} \tag{7}$$

APPENDIX B

Proof of Equation 29

We can show that the new variance v_{j*} for component $j*$ satisfies [78]

$$p_{j*}(m_{j*}^2 + v_{j*}) = p_{j_1}(m_{j_1}^2 + v_{j_1}) + p_{j_2}(m_{j_2}^2 + v_{j_2}) \tag{1}$$

Besides, according to equation 2 we have

$$\alpha_j = \frac{m_j - a}{b - a} s_j \quad \beta_j = s_j - \frac{m_j - a}{b - a} s_j = s_j(1 - \frac{m_j - a}{b - a}) = \frac{b - m_j}{b - a} s_j$$

Using the two previous equations, equation 3 becomes

$$v_j = (b-a)^2 \frac{\alpha_j \beta_j}{(\alpha_j + \beta_j)^2 (\alpha_j + \beta_j + 1)} = (b-a)^2 \frac{\alpha_j \beta_j}{s_j^2 (s_j + 1)} = \frac{(m_j - a)(b - m_j)}{s_j + 1} \tag{2}$$

substituting the previous equation into equation 1, we obtain

$$p_{j*}\left(m_{j*}^2 + \frac{(m_{j*} - a)(b - m_{j*})}{s_{j*} + 1}\right) = p_{j_1}\left(m_{j_1}^2 + \frac{(m_{j_1} - a)(b - m_{j_1})}{s_{j_1} + 1}\right) + p_{j_2}\left(m_{j_2}^2 + \frac{(m_{j_2} - a)(b - m_{j_2})}{s_{j_2} + 1}\right)$$

Thus, $\frac{(m_{j*}-a)(b-m_{j*})}{s_{j*}+1} = \frac{p_{j_1}\left(m_{j_1}^2 + \frac{(m_{j_1}-a)(b-m_{j_1})}{s_{j_1}+1}\right) + p_{j_2}\left(m_{j_2}^2 + \frac{(m_{j_2}-a)(b-m_{j_2})}{s_{j_2}+1}\right)}{p_{j*}} - m_{j*}^2$, and

$$s_{j*} = \frac{p_{j*}(m_{j*} - a)(b - m_{j*})}{p_{j_1}\left(m_{j_1}^2 + \frac{(m_{j_1}-a)(b-m_{j_1})}{s_{j_1}+1}\right) + p_{j_2}\left(m_{j_2}^2 + \frac{(m_{j_2}-a)(b-m_{j_2})}{s_{j_2}+1}\right) - p_{j*}m_{j*}^2} - 1$$

APPENDIX C

Proof of Equation 32

When we split a component $j*$ to define two new components j_1 and j_2 having weights and parameters $(p_{j_1}, m_{j_1}, s_{j_1})$ and $(p_{j_2}, m_{j_2}, s_{j_2})$, respectively, we can set the following [78]

$$v_{j_1} = u_3(1 - u_2^2)v_{j*}\frac{p_{j*}}{p_{j_1}} \tag{1}$$

$$v_{j_2} = (1 - u_3)(1 - u_2^2)v_{j*}\frac{p_{j*}}{p_{j_2}} \tag{2}$$

By substituting equation 2 into equation 1, we obtain $\frac{(m_{j_1}-a)(b-m_{j_1})}{s_{j_1}+1} = u_3(1-u_2^2)\frac{(m_{j*}-a)(b-m_{j*})}{s_{j*}+1}\frac{p_{j*}}{p_{j_1}}$, thus

$$s_{j_1} = \frac{(m_{j_1} - a)(b - m_{j_1})}{u_3(1 - u_2^2)\frac{(m_{j*}-a)(b-m_{j*})}{s_{j*}+1}\frac{p_{j*}}{p_{j_1}}} - 1$$

By substituting equation 2 into equation 2, we obtain $\frac{(m_{j_2}-a)(b-m_{j_2})}{s_{j_2}+1} = (1-u_3)(1-u_2^2)\frac{(m_{j*}-a)(b-m_{j*})}{s_{j*}+1}\frac{p_{j*}}{p_{j_1}}$, thus

$$s_{j_2} = \frac{(m_{j_2} - a)(b - m_{j_2})}{(1 - u_3)(1 - u_2^2)\frac{(m_{j*}-a)(b-m_{j*})}{s_{j*}+1}\frac{p_{j*}}{p_{j_1}}} - 1$$

APPENDIX D

Proof of Equation 33

$$\frac{p(Z,P,M+1,\xi,\vartheta,\varpi,\varepsilon,\zeta|\mathcal{X})}{p(Z,P,M,\xi,\vartheta,\varpi,\varepsilon,\zeta|\mathcal{X})} = \tag{1}$$
$$(\text{likelihood ratio})\frac{(M+1)!p(M+1)p(P|M+1,\delta)p(Z|P,M+1)p(\xi|M+1,\eta)}{M!p(M)p(P|M,\delta)p(Z|P,M)p(\xi|M,\eta)}p(\varepsilon)p(\zeta|\varphi,\varrho)p(\vartheta|\lambda,\mu)p(\varpi|\phi)$$

where “likelihood” ratio is the ratio of the likelihood using the new parameter set, corresponding to $M+1$ components, to that for the old one corresponding to M components:

$$\text{likelihood ratio} = \frac{\prod_{i=1,Z_i=j_1}^{N} p(x_i|\Theta)\prod_{i=1,Z_i=j_2}^{N} p(x_i|\Theta)}{\prod_{i=1,Z_i=j*}^{N} p(x_i|\Theta)} \tag{2}$$

$$\begin{aligned}\frac{p(P|M+1,\delta)}{p(P|M,\delta)} &= \frac{\frac{\Gamma(\sum_{j=1}^{M+1}\delta_j)}{\prod_{j=1}^{M+1}\Gamma(\delta_j)}\prod_{j=1}^{M+1}p_j^{\delta_j-1}}{\frac{\Gamma(\sum_{j=1}^{M}\delta_j)}{\prod_{j=1}^{M}\Gamma(\delta_j)}\prod_{j=1}^{M}p_j^{\delta_j-1}} = \frac{\frac{\Gamma(\sum_{j=1}^{M-1}\delta_j+\delta_{j_1}+\delta_{j_2})}{\prod_{j=1}^{M-1}\Gamma(\delta_j)\Gamma(\delta_{j_1})\Gamma(\delta_{j_2})}\prod_{j=1}^{M-1}p_j^{\delta_j-1}p_{j_1}^{\delta_{j_1}-1}p_{j_2}^{\delta_{j_1}-1}}{\frac{\Gamma(\sum_{j=1}^{M-1}\delta_j+\delta_{j*})}{\prod_{j=1}^{M-1}\Gamma(\delta_j)\Gamma(\delta_{j*})}\prod_{j=1}^{M-1}p_j^{\delta_j-1}p_{j*}^{\delta_{j*}-1}} \\ &= \frac{\frac{\Gamma(\sum_{j=1}^{M-1}\delta_j+\delta_{j_1}+\delta_{j_2})}{\Gamma(\delta_{j_1})\Gamma(\delta_{j_2})}p_{j_1}^{\delta_{j_1}-1}p_{j_2}^{\delta_{j_1}-1}}{\frac{\Gamma(\sum_{j=1}^{M-1}\delta_j+\delta_{j*})}{\Gamma(\delta_{j*})}p_{j*}^{\delta_{j*}-1}} = \frac{\Gamma(\sum_{j=1}^{M-1}\delta_j+\delta_{j_1}+\delta_{j_2})\Gamma(\delta_{j*})p_{j_1}^{\delta_{j_1}-1}p_{j_2}^{\delta_{j_1}-1}}{\Gamma(\delta_{j_1})\Gamma(\delta_{j_2})\Gamma(\sum_{j=1}^{M-1}\delta_j+\delta_{j*})p_{j*}^{\delta_{j*}-1}}\end{aligned} \tag{3}$$

$$\frac{p(Z|P,M+1)}{p(Z|P,M)} = \frac{p_{j_1}^{n_{j_1}}p_{j_2}^{n_{j_2}}\prod_{j=1}^{M-1}p_j^{n_j}}{p_{j*}^{n_{j*}}\prod_{j=1}^{M-1}p_j^{n_j}} = \frac{p_{j_1}^{n_{j_1}}p_{j_2}^{n_{j_2}}}{p_{j*}^{n_{j*}}} \tag{4}$$

where n_{j_1} and n_{j_2} are the numbers of observations to be assigned to components j_1 and j_2.

$$\frac{p(\xi|M+1,\eta)}{p(\xi|M,\eta)} = \frac{\varpi^{\vartheta}\Gamma(\zeta)\exp\Big(-\varpi\big(\frac{s_{j_1}+s_{j_2}}{s_{j_1}s_{j_2}}-\frac{1}{s_{j*}}\big)\Big)\Big(\frac{(m_{j_1}-a)(m_{j_2}-a)}{m_{j*}-a}\Big)^{\frac{\zeta(\varepsilon-a)}{b-a}-1}\Big(\frac{(b-m_{j_1})(b-m_{j_2})}{b-m_{j*}}\Big)^{\zeta(1-\frac{\varepsilon-a}{b-a})-1}}{\big(\frac{s_{j_1}s_{j_2}}{s_{j*}}\big)^{\vartheta+1}\Gamma(\vartheta)(b-a)^{(\zeta-1)}\Big[\Gamma\big(\frac{\zeta(\varepsilon-a)}{b-a}-1\big)\Gamma\big(\zeta(1-\frac{\varepsilon-a}{b-a})\big)\Big]} \tag{5}$$

and where the term $M!$ arises due to the exchangeability of the priors of the ξ parameters. Indeed, it is known that Label-switching is of important concern, and numerous papers have discussed this subject (see, for instance, [119]). In our case we have adopted a simple approach that has been found effective in practice and according to our experimental results. Indeed, we impose an identifiability constraint on the parameter space which is $m_1 \leq m_2 \leq \ldots \leq m_M$. It is noteworthy that using this constraint results in $M!$ ways of labeling the mixture components.

APPENDIX E

Proof of Equation 36

$$\left|\frac{\partial \Delta'_M}{\partial(\Delta_M, u)}\right| = \left|\frac{\partial(p_{j_1}, p_{j_2}, m_{j_1}, m_{j_2}, s_{j_1}, s_{j_2})}{\partial(u_1, p_{j*}, m_{j*}, u_2, s_{j*}, u_3)}\right|$$

By computing the derivatives, we obtain:

$$\frac{\partial p_{j_1}}{\partial u_1} = p_{j*} \qquad \frac{\partial p_{j_2}}{\partial u_1} = -p_{j*} \qquad \frac{\partial m_{j_1}}{\partial u_1} = \frac{\partial m_{j_2}}{\partial u_1} = \frac{\partial s_{j_1}}{\partial u_1} = \frac{\partial s_{j_2}}{\partial u_1} = 0$$

$$\frac{\partial p_{j_1}}{\partial p_{j*}} = u_1 \qquad \frac{\partial p_{j_2}}{\partial p_{j*}} = 1 - u_1 \qquad \frac{\partial m_{j_1}}{\partial p_{j*}} = \frac{\partial m_{j_2}}{\partial p_{j*}} = 0$$

$$\frac{\partial s_{j_1}}{\partial p_{j*}} = -\frac{p_{j_1}}{p_{j*}^2}\frac{(m_{j_1}-a)(b-m_{j_1})}{(m_{j*}-a)(b-m_{j*})}\frac{(s_{j*}+1)}{u_3(1-u_2^2)} = \frac{-(s_{j_1}+1)}{p_{j*}}$$

$$\frac{\partial s_{j_2}}{\partial p_{j*}} = -\frac{p_{j_2}}{p_{j*}^2}\frac{(m_{j_2}-a)(b-m_{j_2})}{(m_{j*}-a)(b-m_{j*})}\frac{(s_{j*}+1)}{(1-u_3)(1-u_2^2)} = \frac{-(s_{j_2}+1)}{p_{j*}}$$

$$\frac{\partial p_{j_1}}{\partial m_{j*}} = \frac{\partial p_{j_2}}{\partial m_{j*}} = 0$$

$$\frac{\partial m_{j_1}}{\partial m_{j*}} = 1 - u_2\frac{a+b-2m_{j*}}{2}\sqrt{\frac{p_{j_2}}{(m_{j*}-a)(b-m_{j*})(s_{j*}+1)p_{j_1}}} = 1 + \frac{(m_{j_1}-m_{j*})(a+b-2m_{j*})}{2(m_{j*}-a)(b-m_{j*})}$$

$$\frac{\partial m_{j_2}}{\partial m_{j*}} = 1 + u_2\frac{a+b-2m_{j*}}{2}\sqrt{\frac{p_{j_1}}{(m_{j*}-a)(b-m_{j*})(s_{j*}+1)p_{j_2}}} = 1 + \frac{(m_{j_2}-m_{j*})(a+b-2m_{j*})}{2(m_{j*}-a)(b-m_{j*})}$$

$$\frac{\partial s_{j_1}}{\partial m_{j*}} = -\frac{p_{j_1}(a+b-2m_{j*})(m_{j_1}-a)(b-m_{j_1})(s_{j*}+1)}{p_{j*}(m_{j*}-a)^2(b-m_{j*})^2u_3(1-u_2^2)} = -\frac{(a+b-2m_{j*})}{(m_{j*}-a)(b-m_{j*})}(s_{j_1}+1)$$

$$\frac{\partial s_{j_2}}{\partial m_{j*}} = -\frac{p_{j_2}(a+b-2m_{j*})(m_{j_2}-a)(b-m_{j_2})(s_{j*}+1)}{p_{j*}(m_{j*}-a)^2(b-m_{j*})^2(1-u_3)(1-u_2^2)} = -\frac{(a+b-2m_{j*})}{(m_{j*}-a)(b-m_{j*})}(s_{j_2}+1)$$

$$\frac{\partial p_{j_1}}{\partial u_2} = \frac{\partial p_{j_2}}{\partial u_2} = 0$$

$$\frac{\partial m_{j_1}}{\partial u_2} = -\sqrt{\frac{p_{j_2}}{p_{j_1}}\frac{(m_{j*}-a)(b-m_{j*})}{(s_{j*}+1)}} = \frac{(m_{j_1}-m_{j*})}{u_2}$$

$$\frac{\partial m_{j_2}}{\partial u_2} = \sqrt{\frac{p_{j_1}}{p_{j_2}}\frac{(m_{j*}-a)(b-m_{j*})}{(s_{j*}+1)}} = \frac{(m_{j_2}-m_{j*})}{u_2}$$

$$\frac{\partial s_{j_1}}{\partial u_2} = 2u_2\frac{(m_{j_1}-a)(b-m_{j_1})}{(m_{j*}-a)(b-m_{j*})}\frac{(s_{j*}+1)}{u_3(1-u_2^2)^2}\frac{p_{j_1}}{p_{j*}} = \frac{2u_2}{(1-u_2^2)}(s_{j_1}+1)$$

$$\frac{\partial s_{j_2}}{\partial u_2} = 2u_2\frac{(m_{j_2}-a)(b-m_{j_2})}{(m_{j*}-a)(b-m_{j*})}\frac{(s_{j*}+1)}{(1-u_3)(1-u_2^2)^2}\frac{p_{j_2}}{p_{j*}} = \frac{2u_2}{(1-u_2^2)}(s_{j_2}+1)$$

$$\frac{\partial p_{j_1}}{\partial s_{j*}} = \frac{\partial p_{j_2}}{\partial s_{j*}} = 0$$

$$\frac{\partial m_{j_1}}{\partial s_{j*}} = \frac{u2}{2(s_{j*}+1)}\sqrt{\frac{(m_{j*}-a)(b-m_{j*})}{(s_{j*}+1)}\frac{p_{j_2}}{p_{j_1}}} = \frac{(m_{j*}-m_{j_1})}{2(s_{j*}+1)}$$

$$\frac{\partial m_{j_2}}{\partial s_{j*}} = -\frac{u2}{2(s_{j*}+1)}\sqrt{\frac{(m_{j*}-a)(b-m_{j*})}{(s_{j*}+1)}\frac{p_{j_1}}{p_{j_2}}} = \frac{(m_{j*}-m_{j_2})}{2(s_{j*}+1)}$$

$$\frac{\partial s_{j_1}}{\partial s_{j*}} = \frac{(m_{j_1}-a)(b-m_{j_1})}{(m_{j*}-a)(b-m_{j*})}\frac{p_{j_1}}{p_{j*}}\frac{1}{u_3(1-u_2^2)} = \frac{(s_{j_1}+1)}{(s_{j*}+1)}$$

$$\frac{\partial s_{j_2}}{\partial s_{j*}} = \frac{(m_{j_2}-a)(b-m_{j_2})}{(m_{j*}-a)(b-m_{j*})}\frac{p_{j_2}}{p_{j*}}\frac{1}{(1-u_3)(1-u_2^2)} = \frac{(s_{j_2}+1)}{(s_{j*}+1)}$$

$$\frac{\partial p_{j_1}}{\partial u_3} = \frac{\partial p_{j_2}}{\partial u_3} = \frac{\partial m_{j_1}}{\partial u_3} = \frac{\partial m_{j_2}}{\partial u_3} = 0$$

$$\frac{\partial s_{j_1}}{\partial u_3} = -\frac{(m_{j_1}-a)(b-m_{j_1})}{(m_{j*}-a)(b-m_{j*})}\frac{p_{j_1}}{p_{j*}}\frac{(s_{j*}+1)}{u_3^2(1-u_2^2)} = -\frac{(s_{j_1}+1)}{u_3}$$

$$\frac{\partial s_{j_2}}{\partial u_3} = \frac{(m_{j_2}-a)(b-m_{j_2})}{(m_{j*}-a)(b-m_{j*})}\frac{p_{j_2}}{p_{j*}}\frac{(s_{j*}+1)}{(1-u_3)^2(1-u_2^2)} = \frac{(s_{j_2}+1)}{(1-u_3)}$$

$$\left(\begin{array}{cccccc}
p_{j*} & -p_{j*} & 0 & 0 & 0 & 0 \\
u_1 & 1-u_1 & 0 & 0 & \frac{-(s_{j_1}+1)}{p_{j*}} & \frac{-(s_{j_2}+1)}{p_{j*}} \\
0 & 0 & 1+\frac{(m_{j_1}-m_{j*})(a+b-2m_{j*})}{2(m_{j*}-a)(b-m_{j*})} & 1+\frac{(m_{j_2}-m_{j*})(a+b-2m_{j*})}{2(m_{j*}-a)(b-m_{j*})} & -\frac{(a+b-2m_{j*})(s_{j_1}+1)}{(m_{j*}-a)(b-m_{j*})} & -\frac{(a+b-2m_{j*})(s_{j_2}+1)}{(m_{j*}-a)(b-m_{j*})} \\
0 & 0 & \frac{(m_{j_1}-m_{j*})}{u_2} & \frac{(m_{j_2}-m_{j*})}{u_2} & \frac{2u_2(s_{j_1}+1)}{(1-u_2^2)} & \frac{2u_2(s_{j_2}+1)}{(1-u_2^2)} \\
0 & 0 & \frac{(m_{j*}-m_{j_1})}{2(s_{j*}+1)} & \frac{(m_{j*}-m_{j_2})}{2(s_{j*}+1)} & \frac{(s_{j_1}+1)}{(s_{j*}+1)} & \frac{(s_{j_2}+1)}{(s_{j*}+1)} \\
0 & 0 & 0 & 0 & -\frac{(s_{j_1}+1)}{u_3} & \frac{(s_{j_2}+1)}{(1-u_3)}
\end{array}\right. \quad (1)$$

$$= p_{j*}\frac{(m_{j*}-m_{j_1})(m_{j_2}-m_{j*})(s_{j_1}+1)(s_{j_2}+1)}{u_3(1-u_3)(s_{j*}+1)}\left[\left(\frac{(m_{j_2}-m_{j_1})}{(m_{j*}-m_{j_1})(m_{j_2}-m_{j*})}\right)\left(\frac{1}{u_2(1-u_2^2)}\right)\right] \quad (2)$$

$$= p_{j*}\frac{(m_{j_2}-m_{j_1})(s_{j_1}+1)(s_{j_2}+1)}{u_2(1-u_2^2)u_3(1-u_3)(s_{j*}+1)} \quad (3)$$

APPENDIX F

Proof of Equation 37

According to equation 26, we have the following in the case of the birth of an empty component, where now (p_{j*}, m_{j*}, s_{j*}) play the role of u:

$$A = \frac{p(Z, P, M+1, \xi, \vartheta, \varpi, \varepsilon, \zeta|\mathcal{X})b_{M+1}}{p(Z, P, M, \xi, \vartheta, \varpi, \varepsilon, \zeta|\mathcal{X})a_M p(p_{j*})p(m_{j*})p(s_{j*})}\left|\frac{\partial \Delta'_M}{\partial(\Delta_M, u)}\right| \tag{1}$$

$\frac{p(Z,P,M+1,\xi,\vartheta,\varpi,\varepsilon,\zeta|\mathcal{X})}{p(Z,P,M,\xi,\vartheta,\varpi,\varepsilon,\zeta|\mathcal{X})}$ is developed in Appendix C. In the birth case, however, the likelihood ratio is 1 and we have also

$$\begin{aligned}\frac{p(P|M+1,\delta)}{p(P|M,\delta)} &= \frac{\frac{\Gamma(\sum_{j=1}^M \delta_j)\Gamma(\delta_{j*}+\sum_{j=1}^M \delta_j)}{\Gamma(\delta_{j*})\Gamma(\sum_{j=1}^M \delta_j)\prod_{j=1}^M \Gamma(\delta_j)}\prod_{j=1}^M (p_j(1-p_{j*}))^{\delta_j-1}p_{j*}^{\delta_{j*}-1}}{\frac{\Gamma(\sum_{j=1}^M \delta_j)}{\prod_{j=1}^M \Gamma(\delta_j)}\prod_{j=1}^M p_j^{\delta_j-1}} \\ &= \frac{\Gamma(\delta_{j*}+\sum_{j=1}^M \delta_j)}{\Gamma(\delta_{j*})\Gamma(\sum_{j=1}^M \delta_j)}p_{j*}^{\delta_{j*}-1}(1-p_{j*})^{\sum_{j=1}^M \delta_j - M}\end{aligned} \tag{2}$$

and

$$\frac{p(Z|P,M+1)}{p(Z|P,M)} = \frac{p_{j*}^0\prod_{j=1}^M (p_j(1-p_{j*}))^{n_j}}{\prod_{j=1}^M p_j^{n_j}} = (1-p_{j*})^{\sum_{j=1}^M n_j} = (1-p_{j*})^N \tag{3}$$

Note also that our specific choice of generating m_{j*} and s_{j*}, from the associated prior distributions given by equations 8 and 9, respectively, simplify the calculations, since

$$\frac{p(\xi|M+1,\eta)}{p(\xi|M,\eta)q(u)} = \frac{1}{p(p_{j*})} \tag{4}$$

Recall that the Jacobian is $(1-p_{j*})^M$, thus

$$A = \frac{p(M+1)}{p(M)}\frac{\Gamma(\delta_{j*}+\sum_{j=1}^M \delta_j)}{\Gamma(\delta_{j*})\Gamma(\sum_{j=1}^M \delta_j)}p_{j*}^{\delta_{j*}-1}(1-p_{j*})^{N+\sum_{j=1}^M \delta_j - M}(M+1)\frac{b_{M+1}}{a_M(M_0+1)}\frac{1}{p(p_{j*})}(1-p_{j*})^M \tag{5}$$

where M_0 is the number of empty components before the birth and $p(p_{j*})$ is a Beta distribution with parameters $(\delta_{j*}, \sum_{j=1}^{M} \delta_j)$.

APPENDIX G

Proof of Equation 39

If a 2-parameter density p belongs to the exponential family, then we can write it as the following [120]

$$p(x|\theta) = H(x)\exp\left(G(\theta)^{tr}T(x) + \Phi(\theta)\right) \tag{1}$$

where $G(\theta) = (G_1(\theta), G_2(\theta))$, $T(x) = (T_1(x), T_2(x))$ and tr denotes the transpose. The K-L divergence between two exponential distributions is given by [120]

$$D\big(p(x|\theta)\|p'(x|\theta')\big) = \Phi(\theta) - \Phi(\theta') + [G(\theta) - G(\theta')]^{tr}E_\theta[T(x)] \tag{2}$$

where E_θ is the expectation with respect to $p(x|\theta)$. Moreover, we have the following [120]

$$E_\theta[T(x)] = -\Phi'(\theta) \tag{3}$$

The general Beta distribution can be written as an exponential density. In fact, we can easily show that

$$\begin{aligned} p(x|\alpha,\beta) &= \frac{\Gamma(\alpha+\beta)}{(b-a)^{\alpha+\beta-1}\Gamma(\alpha)\Gamma(\beta)}(x-a)^{\alpha-1}(b-x)^{\beta-1} \\ &= \exp\Big[\log\big(\Gamma(\alpha+\beta)\big) - (\alpha+\beta-1)\log(b-a) - \log\big(\Gamma(\alpha)\big) \\ &\quad - \log\big(\Gamma(\beta)\big) + (\alpha-1)\log(x-a) + (\beta-1)\log(b-x)\Big] \end{aligned}$$

Then by letting

$$\Phi(\alpha,\beta) = \log\big(\Gamma(\alpha+\beta)\big) - \log\big(\Gamma(\alpha)\big) - \log\big(\Gamma(\beta)\big) - (\alpha+\beta)\log\big(b-a\big)$$

$$G_1(\alpha,\beta) = \alpha \qquad G_2(\alpha,\beta) = \beta \qquad T_1(x) = \log(x-a) \qquad T_2(x) = \log(b-x)$$

$$H(x) = \exp\Big(- \log(x-a) - \log(b-x) + \log(b-a)\Big)$$

we obtain $E_\theta[\log(x-a)] = -\Psi(\alpha+\beta) + \Psi(\alpha) + \log\big(b-a\big)$, $E_\theta[\log(b-x)] = -\Psi(\alpha+\beta) + \Psi(\beta) + \log\big(b-a\big)$, thus

$$\begin{aligned}
& D\big(p(x|\theta)\|p'(x|\theta')\big) = \log\big(\Gamma(\alpha+\beta)\big) - \log\big(\Gamma(\alpha'+\beta')\big) + \log\big(\Gamma(\alpha')\big) - \log\big(\Gamma(\alpha)\big) \\
+ \quad & \log\big(\Gamma(\beta')\big) - \log\big(\Gamma(\beta)\big) - (\alpha+\beta-\alpha'-\beta')\log(b-a) \\
+ \quad & (\alpha'-\alpha)[\Psi(\alpha+\beta) - \Psi(\alpha) - \log\big(b-a\big)] + (\beta'-\beta)[\Psi(\alpha+\beta) - \Psi(\beta) - \log\big(b-a\big)] \\
= \quad & \log\left[\frac{\Gamma(\alpha+\beta)\Gamma(\alpha')\Gamma(\beta')}{(b-a)^{\alpha+\beta-\alpha'-\beta'}\Gamma(\alpha'+\beta')\Gamma(\alpha)\Gamma(\beta)}\right] \\
+ \quad & (\alpha'-\alpha)[\Psi(\alpha+\beta) - \Psi(\alpha) - \log\big(b-a\big)] + (\beta'-\beta)[\Psi(\alpha+\beta) - \Psi(\beta) - \log\big(b-a\big)]
\end{aligned}$$

List of References

[1] S.Y. Kung, M.W. Mak, and S.H. Lin. *Biometric Authentication: A Machine Learning Approach*. Prentice Hall, information and system sciences series edition, 2004.

[2] E. Osuna , R. Freund , F. Girosi . Training Support Vector Machines: an Application to Face Detection. In *Proc. of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 130–136, 1997.

[3] A.K. Jain, R.P.W. Duin, and J. Mao. Statistical Pattern Recognition: A Review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):4–37, 2000.

[4] Y. Linde, A. Buzo, and R. Gray. An Algorithm for Vector Quantizer Design. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(1):84–95, 1980.

[5] L. Ding, Z. Lin, A. Radwan, M. S. El-Hennawey, and R. A. Goubran. Non-intrusive single-ended speech quality assessment in VoIP. *Speech Communication*, 49(6):477–489, 2007.

[6] N. Bouguila, J. H. Wang, A. Ben Hamza. A Bayesian approach for software quality prediction. In *4th International IEEE Conference on Intelligent Systems*, pages 11–49–11–54, 2008.

[7] P. Guo and M. R. Lyu. Software Quality Prediction Using Mixture Models with EM Algorithm. In *The First Asia-Pacific Conference on Quality Software*, page 69, 2000.

[8] Xu Wenlong, Zhang Xianghua, and Feng Huanqing. Using Simple Gaussian Mixture Model for Multi-Class Classification Based on Tumor Gene Expression Data. In *International Conference on Bioinformatics and Biomedical Engineering*, pages 470–473, 2008.

[9] G.J. McLachlan and D. Peel. *Finite Mixture Models.* New York: Wiley, 2000.

[10] N. Bouguila and D. Ziou. High-Dimensional Unsupervised Selection and Estimation of a Finite Generalized Dirichlet Mixture Model Based on Minimum Message Length. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(10):1716–1731, 2007.

[11] C.P. Robert. *The Bayesian Choice From Decision-Theoretic Foundations to Computational Implementation.* Springer, second edition, 2007.

[12] S. Meignen and H. Meignen. On the Modeling of Small Sample Distributions with Generalized Gaussian Density in a Maximum Likelihood Framework. *IEEE Transactions on Image Processing*, 15(6):1647–1652, 2006.

[13] F. Chen, Z. Gao and J. Villasenor. Lattice Vector Quantization of Generalized Gaussian Sources. *IEEE Transactions on Information Theory*, 43(1):92–103, 1997.

[14] R. L. Joshi, V. J. Crump and T. R. Fischer. Image Subband Coding Using Arithmetic Coded Trellis Coded Quantization. *IEEE Transactions on Circuits and Systems for Video Technology*, 5(6):515–523, 1995.

[15] R. Laroia and N. Farvardin. A Structured Fixed-Rate Vector Quantizer Derived from a Variable-Length Scalar Quantizer: Part I-Memoryless Sources. *IEEE Transactions on Information Theory*, 39(3):851–867, 1993.

[16] J. H. Miller and J. B. Thomas. Detectors for Discrete-Time Signals in Non-Gaussian Noise. *IEEE Transactions on Information Theory*, 18(2):241–250, 1972.

[17] N. Farvardin and J. W. Modestino. Optimum Quantizer Performance for a Class of Non-Gaussian Memoryless Sources. *IEEE Transactions on Information Theory*, 30(3):485–497, 1984.

[18] Z. Gao, B. Belzer and J. Villasenor. A Comparison of the Z, E_8, and Leech Lattices for Quantization of Low-Shape-Parameter Generalized Gaussian Sources. *IEEE Signal Processing Letters*, 2(10):197–199, 1995.

[19] N. Bouguila, D. Ziou and E. Monga. Practical Bayesian Estimation of a Finite Beta Mixture Through Gibbs Sampling and its Applications. *Statistics and Computing*, 16(2):215–225, 2006.

[20] W. J. Fitzgerald. Markov Chain Monte Carlo Methods With Applications to Signal Processing. *Signal Processing*, 81(1):3–18, 2001.

[21] C. Andrieu, P. M. Djurić and A. Doucet. Model Selection by MCMC Computation. *Signal Processing*, 81(1):19–37, 2001.

[22] J.K. Ghosh, M. Delampady and T. Samanta. *An Introduction to Bayesian Analysis Theory and Methods*. Springer, 2006.

[23] J. R. Ohm. *Multimedia Communication Technology, Representation, Transmission and Identification of Multimedia Signals*. Springer, 2004.

[24] W. Mauersberger. Experimental Results on the Performance of Mismatched Quantizers. *IEEE Transactions on Information Theory*, 25(4):381–386, 1979.

[25] S. G. Mallat. A Theory for Multiresolution Signal Decomposition: The Wavelet Representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7):674–693, 1989.

[26] T. Naveen and J. W. Woods. Motion Compensated Multiresolution Transmission of High Definition Video. *IEEE Transactions on Circuits and Systems for Video Technology*, 4(1):29–41, 1994.

[27] K. Sharifi and A. Leon-Garcia. Estimation of Shape Parameter for Generalized Gaussian Distributions in Subband Decomposition of Video. *IEEE Transactions on Circuits and Systems for Video Technology*, 5(1):52–56, 1995.

[28] G. Calvagno, C. Ghirardi, G. A. Mian and R. Rinaldo. Modeling of Subband image Data for Buffer Control. *IEEE Transactions on Circuits and Systems for Video Technology*, 7(2):402–408, 1997.

[29] M. N. Do and M. Vetterli. Wavelet-Based Texture Retrieval Using Generalized Gaussian Density and Kullback-Leibler Distance. *IEEE Transactions on Image Processing*, 11(2):146–158, 2002.

[30] J-F. Aujol, G. Aubert and L. Blanc-Féraud. Wavelet-Based Level Set Evolution for Classification of Textured Images. *IEEE Transactions on Image Processing*, 12(12):1634–1641, 2003.

[31] S-K. Choi and C-S. Tong. Supervised Texture Classification Using Characteristic Generalized Gaussian Density. *Journal of Mathematical Imaging and Vision*, 29(1):35–47, 2007.

[32] P. Moulin and J. Liu. Analysis of Multiresolution Image Denoising Schemes Using Generalized Gaussian and Complexity Priors. *IEEE Transactions on Information Theory*, 45(3):909–919, 1999.

[33] T. R. Fischer. A Pyramid Vector Quantizer. *IEEE Transactions on Information Theory*, 32(4):568–583, 1986.

[34] K. A. Birney and T. R. Fischer. On the Modeling of DCT and Subband Image Data for Compression. *IEEE Transactions on Image Processing*, 4(2):186–193, 1995.

[35] C. Bouman and K. Sauer. A Generalized Gaussian Image Model for Edge-Preserving MAP Estimation. *IEEE Transactions on Image Processing*, 2(3):296–310, 1993.

[36] Y. Bazi, L. Bruzzone and F. Melgani. Image Thresholding Based on the EM Algorithm and the Generalized Gaussian Distribution. *Pattern Recognition*, 40(2):619–634, 2007.

[37] S-K. S. Fan, Y. Lin and C-C. Wu. Image Thresholding using a Novel Estimation Method in Generalized Gaussian Distribution Mixture Modeling. *Neurocomputing*, 72(1-3):500–512, 2008.

[38] S. Gazor and W. Zhang. Speech Probability Distributions. *IEEE Signal Processing Letters*, 10(7):204–207, 2003.

[39] K. Kokkinakis and A. K. Nandi. Exponent Parameter Estimation for Generalized Gaussian Probability Density Functions with Application to Speech Modeling. *Signal Processing*, 85(9):1852–1858, 2005.

[40] M.S. Allili, N. Bouguila, and D. Ziou. Finite General Gaussian Mixture Modeling and Application to Image and Video Foreground Segmentation. *Journal of Electronic Imaging*, 17(1):1–13, 2008.

[41] S-K. S. Fan and Y. Lin. A Fast Estimation Method for the Generalized Gaussian Mixture Distribution on Complex Images. *Computer Vision and Image Understanding*, 113(7):839–853, 2009.

[42] G. Moser and S. B. Serpico. Generalized minimum-error thresholding for unsupervised change detection from SAR amplitude imagery. *IEEE Transactions on Geoscience and Remote Sensing*, 44(10):2972–2982, 2006.

[43] D. Cantzos, A. Mouchtari, and C. Kyriakakis. Multichannel Audio Resynthesis Based on a Generalized Gaussian Mixture Model and Cepstral Smoothing. In *Proc. of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustic*, pages 215–218, 2005.

[44] M. K. Varanasi and B. Aazhang. Parametric Generalized Gaussian Density Estimation. *The Journal of the Acoustical Society of America*, 86(4):1404–1415, 1989.

[45] R. Krupinski and J. Purczynski. Approximated fast estimator for the shape parameter of generalized Gaussian distribution. *Signal Processing*, 86(2):205–211, 2006.

[46] B. Aiazzi, L. Alpaone and S. Baronti. Estimation Based on Entropy Matching for Generalized Gaussian PDF Modeling. *IEEE Signal Processing Letters*, 6(6):138–140, 1999.

[47] M. Pi. Improve Maximum Likelihood Estimation for Subband GGD Parameters. *Pattern Recognition Letters*, 27(14):1710–1713, 2006.

[48] F. Müller. Distribution Shape of Two-Dimensional DCT Coefficients of Natural Images. *Electronic Letters*, 29(22):1935–1936, 1993.

[49] S. Geman and D. Geman. Stochastic Relaxation, Gibbs Distributions and the Bayesian Restoration of Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):721–741, 1984.

[50] G. J. McLachlan and T. Krishnan. *The EM Algorithm and Extensions*. New York: Wiley-Interscience, 1997.

[51] C.P. Robert and G. Casella. *Monte Carlo Statistical Methods.* Springer, second edition, 2004.

[52] J.E. Gentle and W. Härdle. *Handbook of Computational Statistics.Concepts and Fundamentals*, volume one. Springer, 2004.

[53] S.M. Lewis , A.E. Raftery. Estimating Bayes Factors via Posterior Simulation with the Laplace-Metropolis Estimator. *Journal of the American Statistical Association*, 90:648–655, 1997.

[54] G. Schwarz. Estimating the Dimension of a Model. *Annals of Statistics*, 16:461–464, 1978.

[55] S. L. Crawford. An Application of the Laplace Method to Finite Mixture Distributions. *Journal of the American Statistical Association*, 89:259–267, 1994.

[56] R. M. Haralick. Automatic remote sensor image processing. *Digital Picture Analysis*, 2:5–63, 1976.

[57] J. R. Smith and S. Chang. Transform Features for Texture Classification and Discrimination in Large Image Databases. In *Proc. of the IEEE International Conference on Image Processing*, pages 407–411, 1994.

[58] M. Tuceryan and A. K. Jain. Texture Analysis. In *The Handbook of Pattern Recognition and Computer Vision*, pages 207–248, 1998.

[59] D. Dunn, W. E. Higgins and J. Wakeley. Texture Segmentation Using 2-D Gabor Elementary Functions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(2):130–149, 1994.

[60] E. P. Simoncelli and E. H. Adelson. Noise Removal Via Bayesian Wavelet Coring. In *Proc. of the IEEE International Conference on Image Processing*, pages 379–382, 1996.

[61] Y. Wu, K. L. Chan and Y. Huang. Image Texture Classification Based on Finite Gaussian Mixture Models. In *Proc. of the 3rd International workshop on texture analysis and synthesis, 9th International Conference on Computer Vision*, pages 107–112, 2003.

[62] W. T. Freeman and E. H. Adelson. The Design and Use of Steerable filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(9):891–906, 1991.

[63] E. P. Simoncelli and W. T. Freeman. The Steerable Pyramid: A Flexible Architecture for Multi-Scale Derivative Computation. In *Proc. of the IEEE International Conference on Image Processing*, pages 444–447, 1995.

[64] Y. Rubner, C. Tomasi and L. J. Guibas. A Metric for Distributions with Applications to Image Databasese. In *Proc. of the IEEE International Conference on Computer Vision*, pages 59–66, 1998.

[65] J.C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms.* New York: Plenum, 1981.

[66] T.N. Pappas. An adaptive clustering algorithm for image segmentation. *IEEE Transactions on Signal Processing*, 40(4):901–914, 1992.

[67] J. Yu and J. Tan. Object density-based image segmentation and its applications in biomedical image analysis. *Computer Methods and Programs in Biomedicine*, 96(3):193–204, 2009.

[68] G.W. Larson, H. Rushmeier, C. Piatko. A visibility matching tone reproduction operator for high dynamic range scenes. *IEEE Transactions on Visualization and Computer Graphics*, 3(4):291–306, 1997.

[69] L. Rueda and L. Qin. An Improved Clustering-based Approach for DNA Microarray Image Segmentation. In *Proc. of the International Conference on Image Analysis and Recognition*, pages 644–652, 2004.

[70] M.J. Callow, S. Dudoit, E.L. Gong, T.P. Speed, and E.M. Rubin. Microarray expression profiling identifies genes with altered expression in HDL deficient mice. *Genome Research*, 10(12):2022–2029, 2000.

[71] M. Schena. *Microarray analysis.* John Wiley & Sons, 2002.

[72] P. Brown and D. Botstein. Exploring the new world of the genome with DNA microarrays. In *Nat Genet*, pages 33–37, 1999.

[73] L. Qin, L. Rueda, A. Ali and A. Ngom. Spot Detection and Image Segmentation in DNA Microarray Data. *Applied Bioinformatics*, 4(1):1–11, 2005.

[74] H. Yijun and W. Guirong. Segmentation of cDNA Microarray Spots Using K-means Clustering Algorithm and Mathematical Morphology. In *Proc. of the 2009 WASE International Conference on Information Engineering*, pages 110–113, 2009.

[75] W. J. Fitzgerald, S. J. Godsill, A. C. Kokaram and J. A. Stark. Bayesian Methods in Signal and Image Processing. In A. P. Dawid J. M. Bernardo, J. O. Berger and A. F. M. Smith, editors, *Bayesian Statistics 6*, pages 239–254. Oxford University Press, 1999.

[76] R. Wilson and G. H. Granlund. The Uncertainty Principle in Image Processing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):758–767, 1984.

[77] P. J. Green. Reversible Jump Markov Chain Monte Carlo Computation and Bayesian Model Determination. *Biometrika*, 82(4):711–732, 1995.

[78] S. Richardson and P. J. Green. On Bayesian Analysis of Mixtures with an Unknown Number of Components (with discussion). *Journal of the Royal Statistical Society: Series B*, 59(4):731–792, 1997.

[79] A. D. Marrs. An Application of Reversible-Jump MCMC to Multivariate Spherical Gaussian Mixtures. In *Advances in Neural Information Processing Systems (NIPS)*, pages 577–583, 1997.

[80] Z. Zhang, K. L. Chan, Y. Wu and C. Chen. Learning a Multivariate Gaussian Mixture Model with the Reversible Jump MCMC Algorithm. *Statistics and Computing*, 14(4):343–355, 2004.

[81] P. Dellaportas and I. Papageorgiou. Multivariate Mixtures of Normals with Unknown Number of Components. *Statistics and Computing*, 16(1):57–68, 2006.

[82] V. Viallefont, S. Richardson and P. J. Green. Bayesian Analysis of Poisson Mixtures. *Journal of Nonparametric Statistics*, 14(1-2):181–202, 2002.

[83] L. Meligkotsidou. Bayesian multivariate Poisson mixtures with an unknown number of components. *Statistics and Computing*, 17(2):93–107, 2007.

[84] M-A. Gruet, A. Philippe and C. P. Robert. MCMC Control Spreadsheets for Exponential Mixture Estimation. *Journal of Computational and Graphical Statistics*, 8(2):298–317, 1999.

[85] P. Dellaportas, J. J. Forster and I. Ntzoufras. On Bayesian Model and Variable Selection using MCMC. *Statistics and Computing*, 12(1):27–36, 2002.

[86] C. Andrieu and A. Doucet. Joint Bayesian Model Selection and Estimation of Noisy Sinusoids via Reversible Jump MCMC. *IEEE Transactions on Signal Processing*, 47(10):2667–2676, 1999.

[87] S. P. Brooks. On Bayesian Analyses and Finite Mixtures for Proportions. *Statistics and Computing*, 11(2):179–190, 2001.

[88] K. Roeder and L. Wasserman. Practical Bayesian Density Estimation Using Mixture of Normals. *Journal of the American Statistical Association*, 92:894–902, 1997.

[89] M. Stephens. Bayesian Analysis of mixture Models with an Unknown Number of Components: An Alternative to reversible Jump Methods. *Annals of Statistics*, 28:40–74, 2000.

[90] A. Nobile. On the Posterior Distribution of the Number of Components in a Finite Mixture. *Annals of Statistics*, 32(5):2044–2073, 2004.

[91] A. Nobile and A. T. Fearnside. Bayesian finite mixtures with an unknown number of components: The allocation sampler. *Statistics and Computing*, 17(2):147–162, 2007.

[92] P. Diaconis and D. Ylvisaker. Quantifying Prior Opinion. In D. V. Lindley J. M. Bernardo, M. H. DeGroot and A. F. M. Smith, editors, *Bayesian Statistics 2*, pages 133–156, 1985.

[93] A. E. Gelfand, B. K. Mallick and D. K. Dey. Modeling Expert Opinion Arising as a Partial Probabilistic Specification. *Journal of the American Statistical Association*, 90(430):598–604, 1995.

[94] J. Berkhof, I. Van Mechelen and A. Gelman. A Bayesian Approach to the Selection and Testing of Mixture Models. *Statistica Sinica*, 13:423–442, 2003.

[95] N. L. Johnson, S. Kotz and N. Balakrishman. *Continous Univariate Distributions*, volume 2. John Wiley and Sons: New York, 1995.

[96] N. Bouguila, D. Ziou and J. Vaillancourt. Unsupervised Learning of a Finite Mixture Model Based on the Dirichlet Distribution and its Application. *IEEE Transactions on Image Processing*, 13(11):1533–1543, 2004.

[97] D. J. C. Mackay and L. Peto. A Hierarchical Dirichlet Language Model. *Natural Language Engineering*, 1(3):1–19, 1994.

[98] A. P. Dempster, N. M. Laird and D. B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm (with discussion). *Journal of the Royal Statistical Society, B*, 39:1–38, 1977.

[99] X. Meng and D. van Dyk. The EM Algorithm - An Old Folk Song Sung to a Fast New Tune (with discussion). *Journal of the Royal Statistical Society, B*, 59(3):511–567, 1997.

[100] M. Jamshidian and R. I. Jennrich. Acceleration of the EM Algorithm by using Quasi-Newton Methods. *Journal of the Royal Statistical Society: Series B*, 59(3):569–587, 1997.

[101] B. P. Carlin and T. A. Louis. *Bayes and Empirical Bayes Methods for Data Analysis*. Chapman & Hall/CRC, second edition, 2000.

[102] S. Chib and E. Greenberg. Understanding the Metropolis-Hastings Algorithm. *The American Statistician*, 49(4):327–335, 1995.

[103] D. Applegate and R. Kannan. Sampling and Integration of Near Log-concave Functions. In *Proc. of the Twenty-third Annual ACM Symposium on Theory of Computing*, pages 156–163, 1991.

[104] W. R. Gilks and P. Wild. Algorithm AS 287: Adaptive Rejection Sampling from Log-Concave Density Functions. *Applied Statistics*, 42(4):701–709, 1993.

[105] Y. C. Bechtel, C. Bonaiti-Pellie, N. Poisson, J. Magnette and P. R. Bechtel. A Population and Family Study of N-acetyltransferase using Caffeine Urinary Metabolites. *Clinical Pharmacology and Therapeutics*, 54(2):134–141, 1993.

[106] A. J. Izenman and C. J. Sommer. Philatelic Mixtures and Multimodal Densities. *Journal of the American Statistical Association*, 83(404):941–953, 1988.

[107] K. E. Basford, G. J. McLachlan and M. G. York. Modelling the Distribution of Stamp Paper Thickness via Finite Normal Mixtures: The 1872 Hidalgo Stamp Issue of Mexico Revisited. *Journal of Applied Statistics*, 24(2):169–179, 1997.

[108] X. Yang and J. Liu. Mixture Density Estimation with Group Membership Functions. *Pattern Recognition Letters*, 23(5):501–512, 2002.

[109] B. S. Manjunath and W. Y. Ma. Texture Features for Browsing and Retrieval of Image Data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(8):837–842, 1996.

[110] A. C. Bovik, M. Clark and W. S. Geisler. Multichannel Texture Analysis Using Localized Spatial Filters. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 12(1):55–73, 1990.

[111] D. Dunn and W. E. Higgins. Optimal Gabor Filters for Texture Segmentation. *IEEE Transactions on Image Processing*, 4(7):947–964, 1995.

[112] S. E. Grigorescu, N. Petkov and P. Kruizinga. Comparison of Texture Features Based on Gabor Filters. *IEEE Transactions on Image Processing*, 11(10):1160–1167, 2002.

[113] T. Chang and C.-C. J. Kuo. Texture Analysis and Classification with Tree-Structured Wavelet Transform. *IEEE Transactions on Image Processing*, 2(4):429–441, 1993.

[114] A. Laine and J. Fan. Texture Classification by Wavelet Packet Signatures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(11):1186–1191, 1993.

[115] M. Unser. Texture Classification and Segmentation Using Wavelet frames. *IEEE Transactions on Image Processing*, 4(11):1549–1560, 1995.

[116] G. V. Wouwer, P. Scheunders and D. V. Dyck. Statitical Texture Characterization from Discrete Wavelet Representations. *IEEE Transactions on Image Processing*, 8(4):592–598, 1999.

[117] T. Randen and J. H. Husøy. Filtering for Texture Classification: A Comparative Study. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 21(4):291–310, 1999.

[118] M. J. Wainwright and E. P. Simoncelli. Scale Mixtures of Gaussians and the Statistics of Natural Images. In *Advances in Neural Information Processing Systems (NIPS)*, pages 855–861, 2000.

[119] M. Stephens. Dealing with Label Switching in Mixture Models. *Journal of the Royal Statistical Society, B*, 62(4):795–809, 2000.

[120] L. D. Brown. *Fundamentals of Statistical Exponential Families with Applications in Statistical Decision Theory*. Hayward, CA: Institute of Mathematical Statistics, 1986.