6

Mixture Models

I must have missed something.
—**Ian Rankin**, ***The Hanging Garden***.—

Roadmap

This chapter covers a class of models where a rather simple distribution is made more complex and less informative by a mechanism that mixes together several known or unknown distributions. This representation is naturally called a mixture of distributions, as illustrated above. Inference about the parameters of the elements of the mixtures and the weights is called mixture estimation, while recovery of the original distribution of each observation is called classification (or, more exactly, unsupervised classification to distinguish it from the supervised classification to be discussed in Chap. 8).

Both aspects almost always require advanced computational tools since even the representation of the posterior distribution may be complicated. Typically, Bayesian inference for these models was not correctly treated until the introduction of MCMC algorithms in the early 1990s. This chapter also covers the case of a mixture with an unknown number of components, for which a specific approximation of the Bayes factor was designed by Chib (1995).

J.-M. Marin and C.P. Robert, *Bayesian Essentials with R*, Springer Texts in Statistics, DOI 10.1007/978-1-4614-8687-9_6,

6.1 Missing Variable Models

In some cases, the complexity of a model originates from the fact that some piece of information about an original and more standard (simpler) model is *missing*. For instance, we have encountered a missing variable model in Chap. 5 with the Arnason–Schwarz model (Sect. 5.5), where the fact of ignoring the characteristics of the individuals outside their capture periods makes inference much harder. Similarly, we have seen in Chap. 4 that the probit model can be reinterpreted as a missing-variable model in that we only observe the sign of a normal variable.

Formally, all models that are defined via a marginalization mechanism, that is, such that the density of the observables $\mathbf{x}$, $f(\mathbf{x}|\theta)$, is given by an integral

$$f(\mathbf{x}|\theta) = \int_{\mathscr{Z}} g(\mathbf{x}, \mathbf{z}|\theta)\, \mathrm{d}\mathbf{z}\,, \tag{6.1}$$

can be considered as belonging to a *missing variable* (or *missing data*) model.[1]

This chapter focus on the case of the mixture model, which is *the* archetypical missing-variable model in that its simple representation (and interpretation) is mirrored by a need for complex processing. Later, in Chap. 7, we will also discuss *hidden Markov models* that add to the missing structure a temporal dependence dimension.

Although image analysis is the topic of Chap. 8, the dataset used in this chapter is derived from an image of a license plate, called **License** and not available in bayess, as

```
> image(license,col=grey(0:255/255),axes=FALSE,xlab="",
        ylab="")
```

represented in Fig. 6.1 (top). The actual histogram of the grey levels is concentrated on 256 values because of the poor resolution of the image, but we transformed the original data as

```
> license=scan("license.txt")
> license=jitter(license,10)
> datha=log((license-min(license)+.01)/
+ (max(license)+.01-license))
```

where `jitter` is used to randomize the dataset and avoid repetitions (as already described on page 156). The second line of code is a logit transform.

[1] This is not a definition in the mathematical sense since all densities can formally be represented that way. We thus stress that the model itself must be introduced that way. This point is not to be mistaken for a requirement that the variable $\mathbf{z}$ be meaningful for the data at hand. In many cases, for instance the probit model, the missing variable representation remains formal.

The transformed data used in this chapter has been stored in the file `datha.txt`.

```
> data(datha)
> datha=as.matrix(datha)
> hist(datha,nclas=200,xlab="",xlim=c(min(datha),max(datha)),
      ylab="",prob=TRUE,main="")
```

As seen from Fig. 6.1 (bottom), the resulting structure of the data is compatible with a sample from a mixture of several normal distributions (with at least two components). We point out at this early stage that mixture modeling is often used in image smoothing. Unsurprisingly, the current plate image would instead require feature recognition, for which this modeling does not help, because it requires spatial coherence and thus more complicated models that will be presented in Chap. 8.

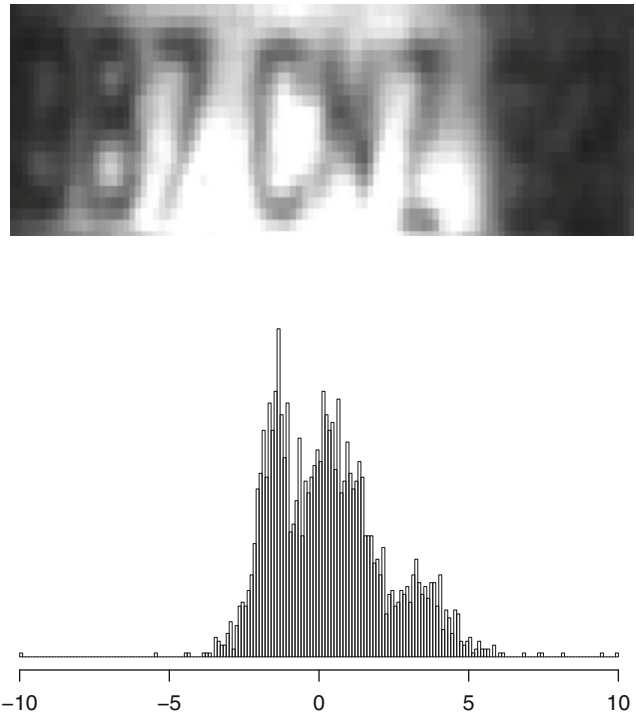

Fig. 6.1. Dataset **License**: (*top*) Image of a car license plate and (*bottom*) histogram of the transformed grey levels of the dataset

6.2 Finite Mixture Models

We now introduce the specific case of mixtures as it exemplifies the complexity of missing-variable models, both by its nature (in the sense that it is inherently linked with a missing variable) and by its processing, which also requires the incorporation of the missing structure.[2]

A *mixture distribution* is a convex combination

$$\sum_{j=1}^{k} p_j f_j(x)\,, \qquad p_j \geq 0\,, \qquad \sum_{j=1}^{k} p_j = 1\,,$$

of k distributions f_j ($k > 1$). In the simplest situations, the f_j's are known and inference focuses either on the unknown proportions p_j or on the allocations of the points of the sample $(x_1, \ldots, x_n)$ to the components f_j, i.e. on the probability that x_i is generated from f_j by opposition to being generated from f_ℓ, say. In most cases, however, the f_j's are from a parametric family like the normal or Beta distributions, with unknown parameters θ_j, leading to the mixture model

$$\sum_{j=1}^{k} p_j f(x|\theta_j)\,, \tag{6.2}$$

with parameters including both the weights p_j and the component parameters θ_j ($j = 1, \ldots, k$). It is actually relevant to distinguish the weights p_j from the other parameters in that they are solely associated with the missing-data structure of the model, while the others are related to the observations. This distinction is obviously irrelevant in the computation of the likelihood function or in the construction of the prior distribution, but it matters in the interpretation of the posterior output, for instance.

There are several motivations for considering mixtures of distributions as a useful extension to "standard" distributions. The most natural approach is to envisage a dataset as made of several latent (that is, missing, unobserved) strata or subpopulations. For instance, one of the earliest occurrences of mixture modeling can be found in Bertillon (1887),[3] where the bimodal structure of the heights of (military) conscripts in central France (Doubs) can be explained a posteriori by the aggregation of two populations of young men, one from the plains and one from the mountains. The mixture structure appears because the origin of each observation (that is, the allocation to a specific subpopulation or stratum) is lost. In the example of the military conscripts, this means that the geographical origin of each young man was not recorded.

[2] We will see later that the missing structure of a mixture actually *need* not be simulated but, for more complex missing-variable structures like hidden Markov models (introduced in Chap. 7), this completion cannot be avoided.

[3] The Frenchman Alphonse Bertillon is also the father of scientific police investigation. For instance, he originated the use of fingerprints in criminal investigations.

Depending on the setting, the inferential goal associated with a sample from a mixture of distributions may be either to reconstitute the groups by estimating the missing component z, an operation usually called classification (or *clustering*), to provide estimators for the parameters of the different groups, or even to estimate the number k of groups.

A completely different (if more involved) approach to the interpretation and estimation of mixtures is the *semiparametric* perspective. This approach considers that since very few phenomena obey probability laws corresponding to the most standard distributions, mixtures such as (6.2) can be seen as a good trade-off between fair representation of the phenomenon and efficient estimation of the underlying distribution. If k is large enough, there is theoretical support for the argument that (6.2) provides a good approximation (in some functional sense) to most distributions. Hence, a mixture distribution can be perceived as a type of (functional) basis approximation of unknown distributions, in a spirit similar to wavelets and splines, but with a more intuitive flavor (for a statistician at least). However, this chapter mostly focuses on the "parametric" case, that is, on situations when the partition of the sample into subsamples with different distributions f_j does make sense from the dataset or modelling point of view (even though the computational processing is the same in both cases). In other words, we consider settings where clustering the sample into strata or subpopulations is of interest.

6.3 Mixture Likelihoods and Posteriors

Let us consider an iid sample $\mathbf{x} = (x_1, \ldots, x_n)$ from model (6.2). The likelihood is such that

$$\ell(\boldsymbol{\theta}, \mathbf{p}|\mathbf{x}) = \prod_{i=1}^{n} \sum_{j=1}^{k} p_j \, f(x_i|\theta_j) \,.$$

This likelihood contains k^n terms when the inner sums are expanded. While this expansion is not necessary for computing the likelihood at a given value $(\boldsymbol{\theta}, \mathbf{p})$, a computation that is feasible in $\mathrm{O}(nk)$ operations as demonstrated by the representation in Fig. 6.2, it remains a necessary step in the understanding of the mixture structure. Alas, the computational difficulty in using the expanded version precludes analytic solutions for either maximum likelihood or Bayes estimators.

Example 6.1. Consider the simple case of a two-component normal mixture

$$p\,\mathcal{N}(\mu_1, 1) + (1-p)\,\mathcal{N}(\mu_2, 1)\,, \tag{6.3}$$

where the weight $p \neq 0.5$ is known. The likelihood surface can be computed by an R code as in the following `plotmix` function, which relies on the `image` function and a discretization of the (μ_1, μ_2) space into pixels. Given a sample

`sampl` that is generated in the first lines of the function, the log-likelihood surface is computed by

```
pbar=1-p
mu1=mu2=seq(min(sampl),max(sampl),.1)
mo1=mu1%*%t(rep(1,length(mu2)))
mo2=rep(1,length(mu2))%*%t(mu2)
ca1=-0.5*mo1*mo1
ca2=-0.5*mo2*mo2
like=0*mo1
for (i in 1:n)
   like=like+log(p*exp(ca1+sampl[i]*mo1)+
   pbar*exp(ca2+sampl[i]*mo2))
like=like+.1*(ca1+ca2)
```

and plotted by

```
image(mu1,mu2,like,xlab=expression(mu[1]),
  ylab=expression(mu[2]),col=heat.colors(250))
contour(mu1,mu2,like,levels=seq(min(like),max(like),lengthl),
  add=TRUE,drawlabels=FALSE)
```

We note that the outcome of the `plotmix` function is the list `list(sample=sampl,like=like)`, used in subsequent analyses of the data. For instance, this outcome, including the level sets obtained by `contour`, is provided in Fig. 6.2. In this case, the parameters are identifiable: μ_1 cannot be confused with μ_2 when p is different from 0.5. Nonetheless, the log-likelihood surface in this figure exhibits two modes, one being close to the true value of the parameters used to simulate the dataset and one corresponding to an inverse separation of the dataset into two groups.[4] ◀

For any prior $\pi(\boldsymbol{\theta}, \mathbf{p})$, the posterior distribution of $(\boldsymbol{\theta}, \mathbf{p})$ is available up to a multiplicative constant:

$$\pi(\boldsymbol{\theta}, \mathbf{p}|\mathbf{x}) \propto \left[\prod_{i=1}^{n} \sum_{j=1}^{k} p_j \, f(x_i|\theta_j)\right] \pi(\boldsymbol{\theta}, \mathbf{p}) \; . \tag{6.4}$$

While $\pi(\boldsymbol{\theta}, \mathbf{p}|\mathbf{x})$ can thus be computed for a given value of $(\boldsymbol{\theta}, \mathbf{p})$ at a cost of order $\mathrm{O}(kn)$, we now explain why the derivation of the posterior characteristics, and in particular of posterior expectations of quantities of interest, is only possible in an exponential time of order $\mathrm{O}(k^n)$.

To explain this difficulty in more detail, we consider the rather intuitive missing-variable representation of mixture models: With each x_i is associated

[4] To get a better understanding of this second mode, consider the limiting setting when $p = 0.5$. In that case, there are two equivalent modes of the likelihood, (μ_1, μ_2) and (μ_2, μ_1). As p moves away from 0.5, this second mode gets lower and lower compared with the other mode, but it still remains.

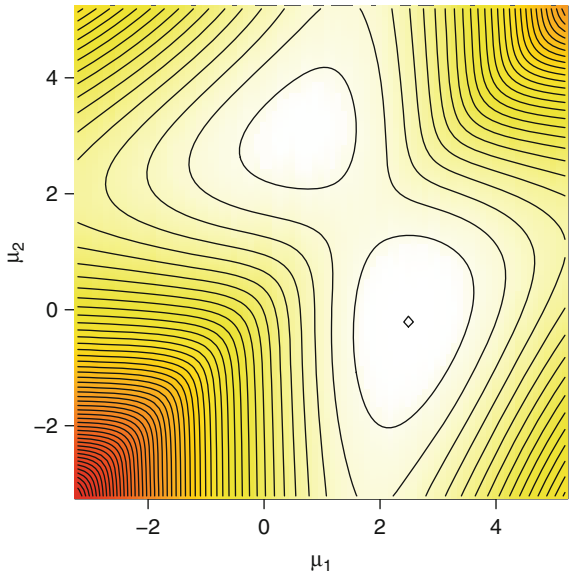

Fig. 6.2. R image representation of the log-likelihood of the mixture (6.3) for a simulated dataset of 500 observations and a true value $(\mu_1, \mu_2, p) = (2.5, 0, 0.7)$. Besides a mode (represented by a *diamond*) in the neighborhood of the true value, the R contour function exhibits an additional mode on the likelihood surface

a missing variable z_i that indicates "its" component, i.e. the index z_i of the distribution from which it was generated. Formally, this means that we have a hierarchical structure associated with the model:

$$z_i|\mathbf{p} \sim \mathscr{M}_k(p_1, \ldots, p_k)$$

and

$$x_i|z_i, \boldsymbol{\theta} \sim f(\cdot|\theta_{z_i})\,.$$

The completed likelihood corresponding to the missing structure is such that

$$\ell(\boldsymbol{\theta}, \mathbf{p}|\mathbf{x}, \mathbf{z}) = \prod_{i=1}^{n} p_{z_i}\, f(x_i|\theta_{z_i})$$

and

$$\pi(\boldsymbol{\theta}, \mathbf{p}|\mathbf{x}, \mathbf{z}) \propto \left[\prod_{i=1}^{n} p_{z_i}\, f(x_i|\theta_{z_i})\right] \pi\,(\boldsymbol{\theta}, \mathbf{p})\,,$$

where $\mathbf{z} = (z_1, \ldots, z_n)$. If we denote by $\mathcal{Z} = \{1, \ldots, k\}^n$ the set of the k^n possible values of the vector $\mathbf{z}$, we can decompose $\mathcal{Z}$ into a partition of subsets

$$\mathcal{Z} = \cup_{j=1}^{\mathfrak{r}} \mathcal{Z}_j$$

as follows (see Exercise 6.2 for the value of $\mathfrak{r}$: For a given allocation size vector $(n_1, \ldots, n_k)$, where $n_1 + \ldots + n_k$, i.e. a given number of observations allocated to each component, we define the *partition sets*

$$\mathcal{Z}_j = \left\{ \mathbf{z} : \sum_{i=1}^{n} \mathbb{I}_{z_i=1}, \ldots, \sum_{i=1}^{n} \mathbb{I}_{z_i=k} \right\},$$

which consist of all allocations with the given allocation size $(n_1, \ldots, n_k)$. We label those partition sets with $j = j(n_1, \ldots, n_k)$ by using, for instance, the lexicographical ordering on the $(n_1, \ldots, n_k)$'s. (This means that $j = 1$ corresponds to $(n_1, \ldots, n_k) = (n, 0, \ldots, 0)$, $j = 2$ to $(n_1, \ldots, n_k) = (n - 1, 1, \ldots, 0)$, $j = 3$ to $(n_1, \ldots, n_k) = (n - 1, 0, 1, \ldots, 0)$, and so on). Using this partition, the posterior distribution of $(\boldsymbol{\theta}, \mathbf{p})$ can be written in closed form as

$$\pi(\boldsymbol{\theta}, \mathbf{p}|\mathbf{x}) = \sum_{\mathbf{z} \in \mathcal{Z}} \pi(\boldsymbol{\theta}, \mathbf{p}|\mathbf{x}, \mathbf{z}) = \sum_{i=1}^{\mathfrak{r}} \sum_{\mathbf{z} \in \mathcal{Z}_i} \omega(\mathbf{z}) \, \pi(\boldsymbol{\theta}, \mathbf{p}|\mathbf{x}, \mathbf{z}), \tag{6.5}$$

where $\omega(\mathbf{z})$ represents the marginal posterior probability of the allocation $\mathbf{z}$ conditional on the observations $\mathbf{x}$ (derived by integrating out the parameters $\boldsymbol{\theta}$ and $\mathbf{p}$). With this representation, a Bayes estimator of $(\boldsymbol{\theta}, \mathbf{p})$ can also be written in closed form as

$$\mathbb{E}^{\pi}[\boldsymbol{\theta}, p|\mathbf{x}] = \sum_{i=1}^{\mathfrak{r}} \sum_{\mathbf{z} \in \mathcal{Z}_i} \omega(\mathbf{z}) \, \mathbb{E}^{\pi}[\boldsymbol{\theta}, \mathbf{p}|\mathbf{x}, \mathbf{z}] .$$

Continuation of Example 6.1. In the special case of model (6.3), if we take two different independent normal priors on both means,

$$\mu_1 \sim \mathcal{N}(0, 4), \quad \mu_2 \sim \mathcal{N}(2, 4),$$

the posterior weight of a given allocation vector $\mathbf{z}$ is

$$\begin{aligned} \omega(\mathbf{z}) \propto & \sqrt{(n_1 + 1/4)(n - n_1 + 1/4)} \, p^{n_1} (n_1 - p)^{n-l} \\ & \times \exp\left\{ -[(n_1 + 1/4)\hat{s}_1(\mathbf{z}) + n_1 \{\bar{x}_1(\mathbf{z})\}^2/4]/2 \right\} \\ & \times \exp\left\{ -[(n - n_1 + 1/4)\hat{s}_2(\mathbf{z}) + (n - n_1)\{\bar{x}_2(\mathbf{z}) - 2\}^2/4]/2 \right\}, \end{aligned}$$

$$\bar{x}_1(\mathbf{z}) = \frac{1}{n_1} \sum_{i=1}^{n} \mathbb{I}_{z_i=1} x_i, \quad \bar{x}_2(\mathbf{z}) = \frac{1}{n - n_1} \sum_{i=1}^{n} \mathbb{I}_{z_i=2} x_i,$$

$$\hat{s}_1(\mathbf{z}) = \sum_{i=1}^{n} \mathbb{I}_{z_i=1} (x_i - \bar{x}_1(\mathbf{z}))^2, \quad \hat{s}_2(\mathbf{z}) = \sum_{i=1}^{n} \mathbb{I}_{z_i=2} (x_i - \bar{x}_2(\mathbf{z}))^2$$

(if we set $\bar{x}_1(\mathbf{z}) = 0$ when $n_1 = 0$ and $\bar{x}_2(\mathbf{z}) = 0$ when $n - n_1 = 0$). Implementing this derivation in R is quite straightforward:

```
omega=function(z,x,p){
  n=length(x)
  n1=sum(z==1);n2=n-n1
  if (n1==0) xbar1=0 else xbar1=sum((z==1)*x)/n1
  if (n2==0) xbar2=0 else xbar2=sum((z==2)*x)/n2
  ss1=sum((z==1)*(x-xbar1)^2)
  ss2=sum((z==2)*(x-xbar2)^2)
  return(sqrt((n1+.25)*(n2+.25))*p^n1*(1-p)^n2*
     exp(-((n1+.25)*ss1+(n2+.25)*ss2)/2)*
     exp(-(n1*xbar1^2+n2*xbar2)/8))
  }
```

leading for instance to

```
> omega(z=sample(1:2,4,rep=TRUE),
+ x=plotmix(n=4,plot=FALSE)$samp,p=.8)
[1] 0.0001781843
> omega(z=sample(1:2,4,rep=TRUE),
+ x=plotmix(n=4,plot=FALSE)$sample,p=.8)
[1] 5.152284e-09
```

Note that the `omega` function is not and cannot be normalized, so the values must be interpreted on a relative scale. ◀

The decomposition (6.5) makes a lot of sense from an inferential point of view. The posterior distribution simply considers each possible partition $\mathbf{z}$ of the dataset, then allocates a posterior probability $\omega(\mathbf{z})$ to this partition, and at last constructs a posterior distribution for the parameters conditional on this allocation. Unfortunately, the computational burden resulting from this decomposition is simply too intensive because there are k^n terms in the sum.

However, there exists a solution that overcomes this computational problem. It uses an MCMC approach that takes advantage of the missing-variable structure and removes the requirement to explore the k^n possible values of $\mathbf{z}$ by only looking at the most likely ones.

Although this is beyond the scope of the book, let us point out here that there also exists in the statistical literature a technique that predates MCMC simulation algorithms but still relates to the same missing-data structure and completion mechanism. It is called the *EM Algorithm*[5] and consists of an iterative but deterministic sequence of "E" (for *expectation*) and "M" (for *maximization*) steps that converge to a local maximum of the likelihood. At iteration t, the "E" step corresponds to the computation of the function

$$Q\{(\boldsymbol{\theta}^{(t)}, \mathbf{p}^{(t)}), (\boldsymbol{\theta}, \mathbf{p})\} = \mathbb{E}_{(\boldsymbol{\theta}^{(t)}, \mathbf{p}^{(t)})} \left[\log \ell(\boldsymbol{\theta}, \mathbf{p}|\mathbf{x}, \mathbf{z})|\mathbf{x}\right],$$

[5] In non-Bayesian statistics, the EM algorithm is certainly the most ubiquitous numerical method, even though it only applies to (real or artificial) missing variable models.

where the likelihood $\ell(\boldsymbol{\theta}, \mathbf{p}|\mathbf{x}, \mathbf{z})$ is the joint distribution of $\mathbf{x}$ and $\mathbf{z}$, while the expectation is computed under the conditional distribution of $\mathbf{z}$ given $\mathbf{x}$ and the value $(\boldsymbol{\theta}^{(t)}, \mathbf{p}^{(t)})$ for the parameter. The "M" step corresponds to the maximization of $Q((\boldsymbol{\theta}^{(t)}, \mathbf{p}^{(t)}), (\boldsymbol{\theta}, \mathbf{p}))$ in $(\boldsymbol{\theta}, \mathbf{p})$, with solution $(\boldsymbol{\theta}^{(t+1)}, \mathbf{p}^{(t+1)})$. As we will see in Sect. 6.4, the Gibbs sampler takes advantage of exactly the same conditional distribution. Further details on EM and its Monte Carlo versions (namely, when the "E" step is not analytically feasible) are given in Robert and Casella (2004, Chap. 5; 2009, Chap. 5).

6.4 MCMC Solutions

For the joint distribution (6.4), the full conditional distribution of $\mathbf{z}$ given $\mathbf{x}$ and the parameters is always available as

$$\pi(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta}, \mathbf{p}) \propto \prod_{i=1}^{n} p_{z_i} f(x_i|\theta_{z_i})$$

and can thus be computed at a cost of $\mathrm{O}(n)$. Since, for standard distributions $f(\cdot|\theta)$, the full posterior conditionals are also easily simulated when using conjugate priors, this implies that the Gibbs sampler can be derived in this setting.[6]

If $\mathbf{p}$ and $\boldsymbol{\theta}$ are independent a priori, then, given $\mathbf{z}$, the vectors $\mathbf{p}$ and $\mathbf{x}$ are independent; that is,

$$\pi(\mathbf{p}|\mathbf{z}, \mathbf{x}) \propto \pi(\mathbf{p}) f(\mathbf{z}|\mathbf{p}) f(\mathbf{x}|\mathbf{z}) \propto \pi(\mathbf{p}) f(\mathbf{z}|\mathbf{p}) \propto \pi(\mathbf{p}|\mathbf{z})\,.$$

Moreover, in that case, $\boldsymbol{\theta}$ is also independent a posteriori from $\mathbf{p}$ given $\mathbf{z}$ and $\mathbf{x}$, with density $\pi(\boldsymbol{\theta}|\mathbf{z}, \mathbf{x})$. If we apply the Gibbs sampler in this problem, it involves the successive simulation of $\mathbf{z}$ and $(\mathbf{p}, \boldsymbol{\theta})$ conditional on one another and on the data:

[6] Historically, missing-variable models constituted one of the first instances where the Gibbs sampler was used by completing the missing variables by simulation under the name of *data augmentation* (see Tanner, 1996, and Robert and Casella, 2004, Chaps. 9 and 10).

Algorithm 6.11 MIXTURE GIBBS SAMPLER

Initialization: Choose $\mathbf{p}^{(0)}$ and $\boldsymbol{\theta}^{(0)}$ arbitrarily.
Iteration t $(t \geq 1)$:

1. For $i = 1, \ldots, n$, generate $z_i^{(t)}$ such that

$$\mathbb{P}\left(z_i = j|\boldsymbol{\theta}, \mathbf{p}\right) \propto p_j^{(t-1)} f\left(x_i|\theta_j^{(t-1)}\right) .$$

2. Generate $\mathbf{p}^{(t)}$ according to $\pi(\mathbf{p}|\mathbf{z}^{(t)})$.
3. Generate $\boldsymbol{\theta}^{(t)}$ according to $\pi(\boldsymbol{\theta}|\mathbf{z}^{(t)}, \mathbf{x})$.

The simulation of the p_j's is also generally obvious since there exists a conjugate prior (as detailed below). In contrast, the complexity in the simulation of the θ_j's will depend on the type of sampling density $f(\cdot|\theta)$ as well as the prior π.

The marginal (sampling) distribution of the z_i's is a multinomial distribution $\mathscr{M}_k(p_1, \ldots, p_k)$, which allows for a conjugate prior on $\mathbf{p}$, namely the Dirichlet distribution $\mathbf{p} \sim \mathscr{D}(\gamma_1, \ldots, \gamma_k)$, with density

$$\frac{\Gamma(\gamma_1 + \ldots + \gamma_k)}{\Gamma(\gamma_1) \cdots \Gamma(\gamma_k)} p_1^{\gamma_1} \cdots p_k^{\gamma_k}$$

on the simplex of $\mathbb{R}^k$,

$$\mathscr{S}_k = \left\{ (p_1, \ldots, p_k) \in [0,1]^k ; \sum_{j=1}^{k} p_j = 1 \right\} .$$

In this case, denoting $n_j = \sum_{l=1}^{n} \mathbb{I}_{z_l = j}$ $(1 \leq j \leq k)$ the allocation sizes, the posterior distribution of $\mathbf{p}$ given $\mathbf{z}$ is

$$\mathbf{p}|\mathbf{z} \sim \mathscr{D}(n_1 + \gamma_1, \ldots, n_k + \gamma_k) .$$

It is rather peculiar that, despite its importance for Bayesian statistics, the Dirichlet distribution is not available in R (at least in the standard stat package). It is however fairly straightforward to code, using a representation based on gamma variates, as shown below.

```
rdirichlet=function(n=1,par=rep(1,2)){

  k=length(par)
  mat=matrix(0,n,k)
  for (i in 1:n){
```

```
    sim=rgamma(k,shape=par,scale=1)
    mat[i,]=sim/sum(sim)
    }
  mat
  }
```

When the density $f(\cdot|\theta)$ also allows for conjugate priors, the simulation of $\boldsymbol{\theta}$ can be specified further since an independent conjugate prior on each θ_j leads to independent and conjugate posterior distributions on the θ_j's, given $\mathbf{z}$ and $\mathbf{x}$.

Continuation of Example 6.1. For the mixture (6.3), under independent normal priors $\mathcal{N}(\delta, 1/\lambda)$ (both $\delta \in \mathbb{R}$ and $\lambda > 0$ are fixed hyperparameters) on both μ_1 and μ_2, the parameters μ_1 and μ_2 are independent given $(\mathbf{z}, \mathbf{x})$, with conditional distributions

$$\mathcal{N}\left(\frac{\lambda\delta + n_1\bar{x}_1(\mathbf{z})}{\lambda + n_1}, \frac{1}{\lambda + n_1}\right) \quad \text{and} \quad \mathcal{N}\left(\frac{\lambda\delta + (n-n_1)\bar{x}_2(\mathbf{z})}{\lambda + n - n_1}, \frac{1}{\lambda + n - n_1}\right),$$

respectively. Similarly, the conditional posterior distribution of the z_i's given (μ_1, μ_2) is $(i = 1, \ldots, n)$

$$\mathbb{P}(z_i = 1|\mu_1, x_i) \propto p \exp\left(-0.5\,(x_i - \mu_1)^2\right).$$

We can thus construct an R function like the following one to generate a sample from the posterior distribution: assuming $\delta = 0$ and $\lambda = 1$,

```
gibbsmean=function(p,datha,niter=10^4){

  n=length(datha)
  z=rep(0,n); ssiz=rep(0,2)
  nxj=rep(0,2)
  mug=matrix(mean(datha),nrow=niter+1,ncol=2)

  for (i in 2:(niter+1)){
    for (t in 1:n){
      prob=c(p,1-p)*dnorm(datha[t],mean=mug[i-1,])
      z[t]=sample(c(1,2),size=1,prob=prob)
      }
    for (j in 1:2){
      ssiz[j]=1+sum(z==j)
      nxj[j]=sum(as.numeric(z==j)*datha)
      }
    mug[i,]=rnorm(2,mean=nxj/ssiz,sd=sqrt(1/ssiz))
    }
  mug
  }
```

which can be used as

```
> dat=plotmix()$sample
> simu=gibbsmean(0.7,dat)
> points(simu,pch=".")
```

to produce Figs. 6.3 and 6.4. This R code illustrates two possible behaviors of this algorithm if we use a simulated dataset of 500 points from the mixture $0.7\mathcal{N}(0,1) + 0.3\mathcal{N}(2.5,1)$, which corresponds to the level sets on both pictures. The starting point in both cases is located at the saddle point between the two modes, i.e. at an instable equilibrium. Depending on the very first (random) iterations of the algorithm, the final sample may end up located on the upper or on the lower mode. For instance, in Fig. 6.3, the Gibbs sample based on 10,000 iterations is in agreement with the likelihood surface, since the second mode discussed in Example 6.1 is much lower than the mode where the simulation output concentrates. In Fig. 6.4, the Gibbs sample ends up being trapped by this lower mode. ◀

Example 6.2. If we consider the more general case of a mixture of two normal distributions with all parameters unknown,

$$p\mathcal{N}(\mu_1, \sigma_1^2) + (1-p)\mathcal{N}(\mu_2, \sigma_2^2)\,,$$

and for the conjugate prior distribution ($j = 1, 2$)

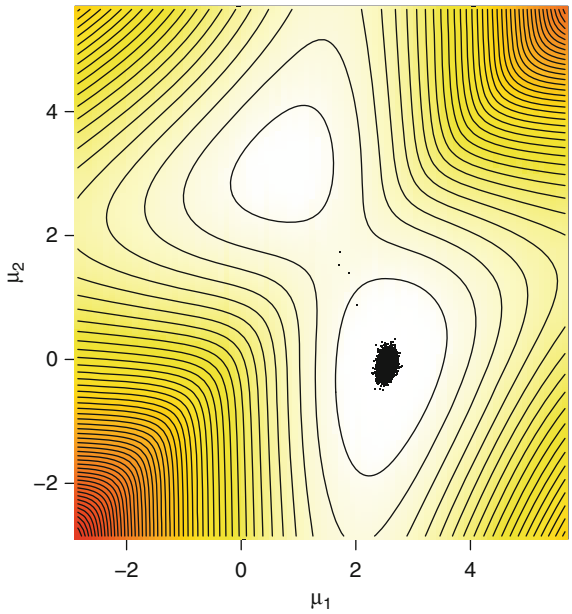

Fig. 6.3. Log-likelihood surface and the corresponding Gibbs sample for the model (6.3), based on 10,000 iterations

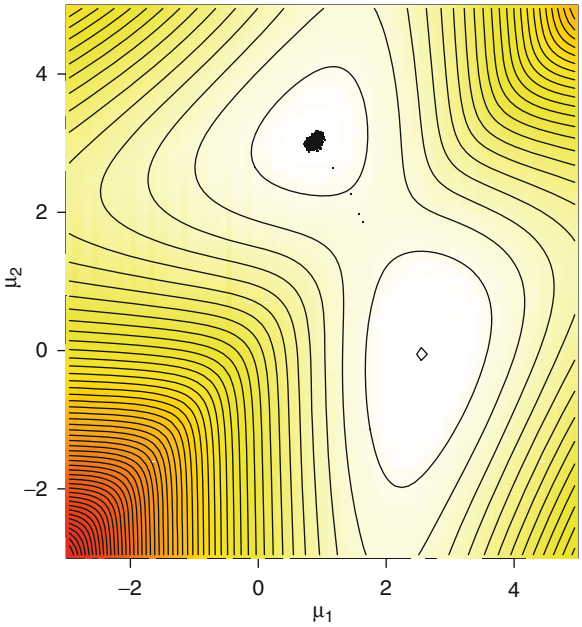

Fig. 6.4. Same legend as Fig. 6.3, with the same starting point located at the saddle point. In this instance, the Gibbs sample ends up around a lower mode

$$\mu_j|\sigma_j \sim \mathcal{N}(\xi_j, \sigma_j^2/l_j)\,, \quad \sigma_j^2 \sim \mathcal{IG}(\nu_j/2, s_j^2/2)\,, \quad p \sim \mathcal{B}e(\alpha, \beta)\,,$$

the same decomposition conditional on $\mathbf{z}$ and straightforward (if dreary) algebra imply that

$$p|\mathbf{x}, \mathbf{z} \sim \mathcal{B}e(\alpha + n_1, \beta + n_2),$$

$$\mu_j|\sigma_j, \mathbf{x}, \mathbf{z} \sim \mathcal{N}\left(\xi_1(\mathbf{z}), \frac{\sigma_j^2}{n_j + l_j}\right)\,, \ \sigma_j^2|\mathbf{x}, \mathbf{z} \sim \mathcal{IG}((\nu_j + n_j)/2, s_j(\mathbf{z})/2)\,,$$

where n_j is the number of z_i equal to j, $\bar{x}_j(\mathbf{z})$ and $\hat{s}_j^2(\mathbf{z})$ are the empirical mean and variance (biased) for the subsample with z_i equal to j, and

$$\xi_j(\mathbf{z}) = \frac{l_j\xi_j + n_j\bar{x}_j(\mathbf{z})}{l_j + n_j}\,, \quad s_j(\mathbf{z}) = s_j^2 + n_j\hat{s}_j^2(\mathbf{z}) + \frac{l_j n_j}{l_j + n_j}(\xi_j - \bar{x}_j(\mathbf{z}))^2\,.$$

The modification of the above R code is also straightforward and we do not reproduce it here to save space. The extension to more than two components is equally straightforward, as described below for **License**. ◀

If we model **License** by a $k = 3$ component normal mixture model, we start by deriving the prior distribution from the scale of the problem. Namely, we choose a $\mathcal{D}_3(1/2, 1/2, 1/2)$ prior for the weights (although picking parameters less than 1 in the Dirichlet prior has the potential drawback that it may allow very small weights for some components), a $\mathcal{N}(\overline{x}, \sigma_i^2)$ distribution on the means μ_i, and a $\mathcal{G}a(10, \hat{\sigma}^2)$ distribution on the precisions σ_i^{-2}, where $\overline{x}$ and $\hat{\sigma}^2$ are the empirical mean and variance of **License**, respectively. (This empirical choice of a prior is debatable on principle, as it depends on the dataset, but this is relatively harmless since it is equivalent to standardizing the dataset so that the empirical mean and variance are equal to 0 and 1, respectively.) If we define the parameter vector `mix` as a list,

```
> mix=list(k=k,p=p,mu=mu,sig=sig)
```

our R function

```
gibbsnorm=function(niter,mix)
```

is made of an initialization step:

```
n=length(datha);k=mix$k
z=rep(0,n) #missing data
nxj=rep(0,k)
ssiz=ssum=rep(0,k)
mug=sigg=prog=matrix(0,nrow=niter,ncol=k)
lopost=rep(0,niter) #log-posterior
lik=matrix(0,n,k)
prog[1,]=rep(1,k)/k;mug[1,]=rep(mix$mu,k)
sigg[1,]=rep(mix$sig,k)
#current log-likelihood
for (j in 1:k)
  lik[,j]=prog[1,j]*dnorm(x=datha,mean=mug[1,j],
       sd=sqrt(sigg[1,j]))
lopost[1]=sum(log(apply(lik,1,sum)))+
  sum(dnorm(mug[1,],mean(datha),sqrt(sigg[1,]),log=TRUE))-
  (10+1)*sum(log(sigg[1,]))-sum(var(datha)/sigg[1,])+
  .5*sum(log(prog[1,]))
```

and of the main loop for data completion and conditional parameter simulation:

```
for (i in 1:(niter-1)){
 for (t in 1:n){   #missing data completion
   prob=prog[i,]*dnorm(datha[t],mug[i,],sqrt(sigg[i,]))
   if (sum(prob)==0) prob=rep(1,k)/k
   z[t]=sample(1:k,1,prob=prob)
   }
```

```
#conditional parameter simulation
for (j in 1:k){
  ssiz[j]=sum(z==j)
  nxj[j]=sum(as.numeric(z==j)*datha)
  }
mug[i+1,]=rnorm(k,(mean(datha)+nxj)/(ssiz+1),
          sqrt(sigg[i,]/(ssiz+1)))
for (j in 1:k)
  ssum[j]=sum(as.numeric(z==j)*(datha-nxj[j]/ssiz[j])^2)
   sigg[i+1,]=1/rgamma(k,shape=.5*(20+ssiz),rate=var(datha)+
    .5*ssum+.5*ssiz/(ssiz+1)*(mean(datha)-nxj/ssiz)^2)
prog[i+1,]=rdirichlet(1,par=ssiz+0.5)
#current log-likelihood
for (j in 1:k)
  lik[,j]=prog[i+1,j]*dnorm(x=datha,mean=mug[i+1,j],
   sd=sqrt(sigg[i+1,j]))
 lopost[i+1]=sum(log(apply(lik,1,sum)))+
sum(dnorm(mug[i+1,],mean(datha),sqrt(sigg[i+1,]),log=TRUE))-
   (10+1)*sum(log(sigg[i+1,]))-sum(var(datha)/sigg[i+1,])+
   .5*sum(log(prog[i+1,]))
}
```

returning all simulated values as a list

```
list(k=k,mu=mug,sig=sigg,p=prog,lopost=lopost)
```

The output of this R function, represented in Fig. 6.5 as an overlay of the **License** histogram is then produced by the R code

```
mix=list(k=3,mu=mean(datha),sig=var(datha))
simu=gibbsnorm(1000,mix)
hist(datha,prob=TRUE,main="",xlab="",ylab="",nclass=100)
x=y=seq(min(datha),max(datha),length=150)
yy=matrix(0,ncol=150,nrow=1000)
for (i in 1:150){
  yy[,i]=apply(simu$p*dnorm(x[i],mean=simu$mu,
  sd=sqrt(simu$sig)),1,sum)
  y[i]=mean(yy[,i])
  }
for (t in 501:1000)
  lines(x,yy[t,],col="gold")
lines(x,y,lwd=2.3,col="sienna2")
```

This output demonstrates that this crude prior modeling is sufficient to capture the modal features of the histogram as well as the tail behavior in a surprisingly small number of Gibbs iterations, despite the large sample size of

2,625 points. The range of the simulated densities represented in Fig. 6.5 reflects the variability of the posterior distribution, while the estimate of the density is obtained by averaging the simulated densities over the 500 iterations.[7]

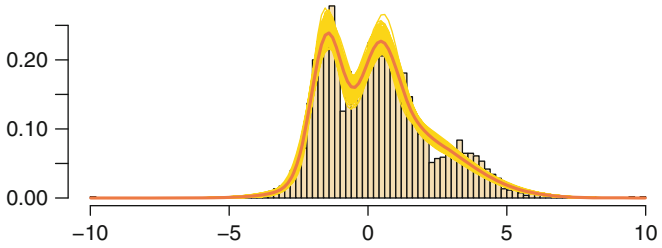

Fig. 6.5. Dataset **License**: Representation of 500 Gibbs iterations for the mixture estimation. (The accumulated *lines* correspond to the estimated mixtures at each iteration and the overlaid *curve* to the density estimate obtained by summation.)

The experiment produced in Example 6.1, page 184, gives a false sense of security about the performance of the Gibbs sampler because it hides the structural dependence of the sampler on its initial conditions. The fundamental feature of Gibbs sampling—its derivation from conditional distributions—implies that it is often restricted in the width of its moves and that, in some situations, this restriction may even jeopardize convergence. In the case of mixtures of distributions, conditioning on $\mathbf{z}$ implies that the proposals for $(\boldsymbol{\theta}, \mathbf{p})$ are quite concentrated and do not allow drastic changes in the allocations at the next step. To obtain a significant modification of $\mathbf{z}$ requires a considerable number of iterations once a stable position has been reached.[8] Figure 6.4 illustrates this phenomenon for the very same sample as in Fig. 6.3: A Gibbs sampler initialized at the saddlepoint may get close to the second mode in the very first iterations and is then unable to escape its (fatal) attraction, even after a large number of iterations, for the reason given above. It is quite interesting to see that this Gibbs sampler suffers from the same pathology as the EM algorithm. However, this is not immensely surprising given that it is based on a similar principle.

In general, there is very little one can do about improving the Gibbs sampler since its components are given by the joint distribution. The solutions are (a) to change the parameterization and thus the conditioning (see

[7] That this is a natural estimate of the model, compared with the "plug-in" density using the estimates of the parameters, will be explained more clearly in Sect. 6.5.

[8] In practice, the Gibbs sampler never leaves the vicinity of a given mode if the attraction of this mode is strong enough, for instance in the case of many observations.

Exercise 6.6), (b) to use tempering to facilitate exploration (see Sect. 6.7), or (c) to mix the Gibbs sampler with another MCMC algorithm.

To look for alternative MCMC algorithms is not a difficulty in this setting, given that the likelihood of mixture models is available in closed form, being computable in $\mathrm{O}(kn)$ time, and the posterior distribution is thus known up to a multiplicative constant. We can therefore use any Metropolis–Hastings algorithm, as long as the proposal distribution q provides a correct exploration of the posterior surface, since the acceptance ratio

$$\frac{\pi(\boldsymbol{\theta}', \mathbf{p}'|\mathbf{x})}{\pi(\boldsymbol{\theta}, \mathbf{p}|\mathbf{x})} \frac{q(\boldsymbol{\theta}, \mathbf{p}|\boldsymbol{\theta}', \mathbf{p}')}{q(\boldsymbol{\theta}', \mathbf{p}'|\boldsymbol{\theta}, \mathbf{p})} \wedge 1$$

can be computed in $\mathrm{O}(kn)$ time. For instance, we can use a random walk Metropolis–Hastings algorithm where each parameter is the mean of the proposal distribution for the new value, that is,

$$\widetilde{\xi}_j = \xi_j^{(t-1)} + u_j,$$

where $u_j \sim \mathcal{N}(0, \zeta^2)$ and ζ is chosen to achieve a reasonable acceptance rate.

Continuation of Example 6.1. For the posterior associated with (6.3), the Gaussian random walk proposal is

$$\widetilde{\mu_1} \sim \mathcal{N}\left(\mu_1^{(t-1)}, \zeta^2\right) \quad \text{and} \quad \widetilde{\mu_2} \sim \mathcal{N}\left(\mu_2^{(t-1)}, \zeta^2\right)$$

which leads to an acceptance probability of

$$r = \min\left\{1, \pi\left(\widetilde{\mu_1}, \widetilde{\mu_2}\middle| x\right) \middle/ \pi\left(\mu_1^{(t-1)}, \mu_2^{(t-1)}\middle| x\right)\right\}.$$

The corresponding R function is then of the form

```
hmmean=function(dat,niter,var=1){
  mu=matrix(0,niter,2)
  mu[1,]=rnorm(2)
  for (i in 2:niter){
    muprop=rnorm(2,mu[i-1,],sqrt(var))
    bound=lpost(dat,muprop)-lpost(dat,mu[i-1,])
    if (runif(1)<=exp(bound)) mu[i,]=muprop else
      mu[i,]=mu[i-1,]
      }
    mu
    }
```

used as in

```
> dat=plotmix()$sample
> simu=hmmeantemp(dat,niter=10^4)
> points(simu,pch=".")
```

when `lpost` is the log-posterior density R function:

```
lpost=function(x,mu,p=0.7,delta=0,lambda=1){
  sum(log(p*dnorm(x,mu[1])+(1-p)*dnorm(x,mu[2])))+
  sum(log(dnorm(mu,delta,1/sqrt(lambda))))
  }
```

For the same simulated dataset as in Fig. 6.4, Fig. 6.6 shows how quickly this algorithm escapes the attraction of the spurious mode. After a few iterations of the algorithm, the chain drifts away from the poor mode and converges almost deterministically to the proper region of the posterior surface. The Gaussian random walk is scaled as $\zeta = 1$, although slightly smaller scales do work as well but would require more iterations to reach the proper modal regions. Too small a scale sees the same trapping phenomenon appear, as the chain does not have sufficient energy to escape the attraction of the current mode (see Example 6.1, page 199, and Fig. 6.8 below). Nonetheless, for a large enough scale, the Metropolis–Hastings algorithm overcomes the drawbacks of the Gibbs sampler. ◀

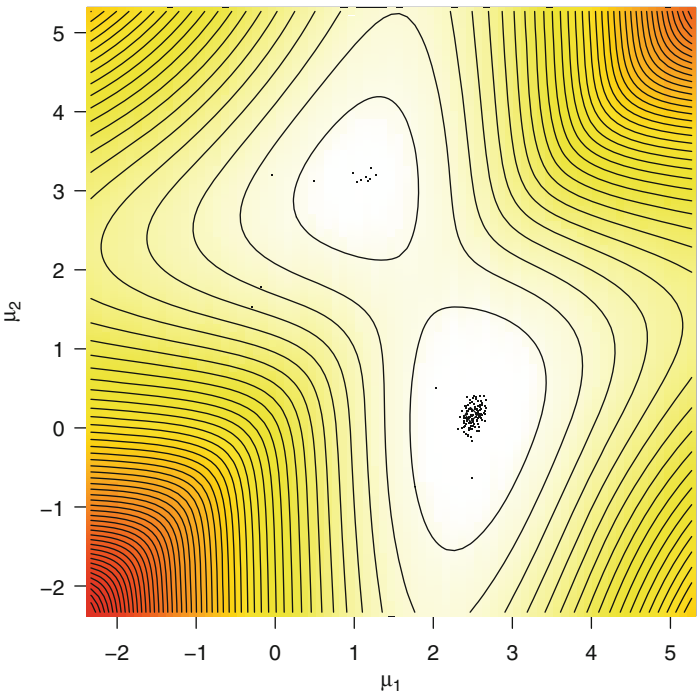

Fig. 6.6. Outcome of a 10,000 iteration random walk Metropolis–Hastings sample on the log-likelihood surface; the starting point is equal to $(1, 3)$. The scale ζ of the random walk is equal to 1

We must point out that, for constrained parameters, the unconstrained random walk Metropolis–Hastings proposal remains valid but is not efficient because when the chain $(\xi_j^{(t)})$ gets close to the boundary of the parameter space, it moves very slowly, given that the proposed values are often incompatible with the constraint and thus rejected at the Metropolis–Hastings acceptance step.

For instance, this lack of efficiency has an impact on the simulation of the weight vector $\mathbf{p}$ since $\sum_{i=1}^k p_k = 1$ in addition to positivity constraints. A practical resolution of this difficulty is to overparameterize the weights of (6.2) into

$$p_j = w_j \Big/ \sum_{l=1}^k w_l\,, \quad w_j > 0 \qquad (1 \le j \le k)\,.$$

Obviously, the w_j's are not identifiable, but this is not a difficulty from a simulation point of view and the p_j's remain identifiable (up to a permutation of indices). Perhaps paradoxically, using overparameterized representations often helps with the mixing of the corresponding MCMC algorithms since those algorithms are less constrained by the dataset or by the likelihood. The reader may have noticed that the w_j's are also constrained by a positivity requirement (just like the variances in a normal mixture or the scale parameters for a Gamma mixture), but this weaker constraint can be bypassed using the reparameterization $\eta_j = \log w_j$. The proposed random walk move on the w_j's is thus

$$\log(\widetilde{w_j}) = \log\left\{w_j^{(t-1)}\right\} + u_j\,,$$

where $u_j \sim \mathcal{N}(0, \zeta^2)$. An important difference from the original random walk Metropolis–Hastings algorithm is that the acceptance ratio also involves a Jacobian term. For instance, the acceptance ratio for a move from $w^{(t-1)}$ to $\tilde{w}$ is then

$$1 \wedge \frac{\pi(\tilde{w})}{\pi(w^{(t-1)})} \prod_{j=1}^k \frac{\widetilde{w_j}}{w_j^{(t-1)}}\,. \tag{6.6}$$

Note that, while being a fairly natural algorithm, the random walk Metropolis–Hastings algorithm usually falls victim to the curse of dimensionality since, obviously the same scale cannot perform well for every component of the parameter vector. In large or even moderate dimensions, a reparameterization of the parameter and preliminary estimation of the information matrix of the distribution are thus often necessary and must sometimes be completed by Gibbs steps operating in lower dimensions.

6.5 Label Switching Difficulty

A basic but extremely important feature of a mixture model is that it is invariant under permutations of the indices of the components. For instance, the normal mixtures $0.3\mathcal{N}(0,1)+0.7\mathcal{N}(2.5,1)$ and $0.7\mathcal{N}(2.5,1)+0.3\mathcal{N}(0,1)$

are exactly the same. Therefore, the $\mathcal{N}(2.5,1)$ distribution cannot be called the "first" component of the mixture! In other words, the component parameters θ_i are not identifiable *marginally* in the sense that θ_1 may be 2.5 as well as 0 in the example above. In this specific case, the pairs (θ_1, p) and $(\theta_2, 1-p)$ are exchangeable.

First, in a k-component mixture, the number of modes of the likelihood is of order $O(k!)$ since if $((\theta_1,\ldots,\theta_k),(p_1,\ldots,p_k))$ is a local maximum of the likelihood function, so is $\tau(\boldsymbol{\theta},\mathbf{p}) = (\theta_{\tau(1)},\ldots,\theta_{\tau(k)},p_{\tau(1)},\ldots,p_{\tau(k)})$ for every permutation $\tau \in \mathfrak{S}_k$, the set of all permutations of $\{1,\ldots,k\}$. This makes maximization and even exploration of the posterior surface obviously harder because modes are separated by valleys that most samplers find difficult to cross.

Second, if an exchangeable prior is used on $(\boldsymbol{\theta},\mathbf{p})$ (that is, a prior invariant under permutation of the indices), all the posterior marginals on the θ_i's are identical, a fact which means for instance that the posterior expectation of θ_1 is identical to the posterior expectation of θ_2. Therefore, alternatives to posterior expectations must be considered to provide pertinent estimators.

Continuation of Example 6.1. In the special case of model (6.3), if we take *the same* normal prior on both μ_1 and μ_2, $\mu_1,\mu_2 \sim \mathcal{N}(0,10)$, say, the posterior weight conditional on $\mathbf{p}$ associated with an allocation $\mathbf{z}$ for which n_1 values are attached to the first component will simply be

$$\begin{aligned}\omega(\mathbf{z}) &\propto p^{n_1}(1-p)^{n-n_1}\int e^{-n_1(\mu_1-\bar{x}_1(\mathbf{z}))^2/2-(n-1)(\mu_2-\bar{x}_2(\mathbf{z}))^2/2}\,\mathrm{d}\pi(\mu_1)\,\mathrm{d}\pi(\mu_2)\\ &\qquad\times\exp\left(-\left\{s_1^2(\mathbf{z})+s_2^2(\mathbf{z})\right\}/2\right)\\ &\propto \sqrt{(n_1+1/10)(n-n_1+1/10)}\,p^{n_1}(1-p)^{n-n_1}\,\exp\left(-\left\{s_1^2(\mathbf{z})+s_2^2(\mathbf{z})\right.\right.\\ &\qquad\left.\left.+n_1\{\bar{x}_1(\mathbf{z})\}^2/(10n_1+1)+(n-n_1)\{\bar{x}_2(\mathbf{z})\}^2/(10(n-n_1)+1)\right\}/2\right),\end{aligned}$$

where $s_1^2(\mathbf{z})$ and $s_2^2(\mathbf{z})$ denote the sums of squares for both groups. ◀

For the Gibbs output of **License** discussed above, the exchangeability predicted by the theory is not observed at all, as shown in Fig. 6.7. This figure is derived from an R code repeating dual plots like

```
> simu=gibbsnorm(1000,mix)
> plot(simu$mu[,1],ylim=range(simu$mu),
+ ylab=expression(mu[i]),xlab="n",type="l",col="sienna3")
> lines(simu$mu[,2],col="gold4")
> lines(simu$mu[,3],col="steelblue")
> plot(simu$mu[,2],simu$p[,2],col="sienna3",
+ xlim=range(simu$mu),ylim=range(simu$p),
+ xlab=expression(mu[i]),ylab=expression(p[i]))
> points(simu$mu[,3],simu$p[,3],col="steelblue")
```

In Fig. 6.7, we see that each component is thus identified by its mean, and the posterior distributions of the means are very clearly distinct. Although this result has the appeal of providing distinct estimates for the three components, it suffers from the severe drawback that the Gibbs sampler has not explored the whole parameter space after 1,000 iterations. Running the algorithm for a much longer period does not solve this problem since the Gibbs sampler cannot simultaneously switch enough component allocations in this highly peaked setup. In other words, the algorithm is unable to explore more than one of the $3! = 6$ equivalent modes of the posterior distribution. Therefore, it is difficult to trust the estimates derived from such an output.

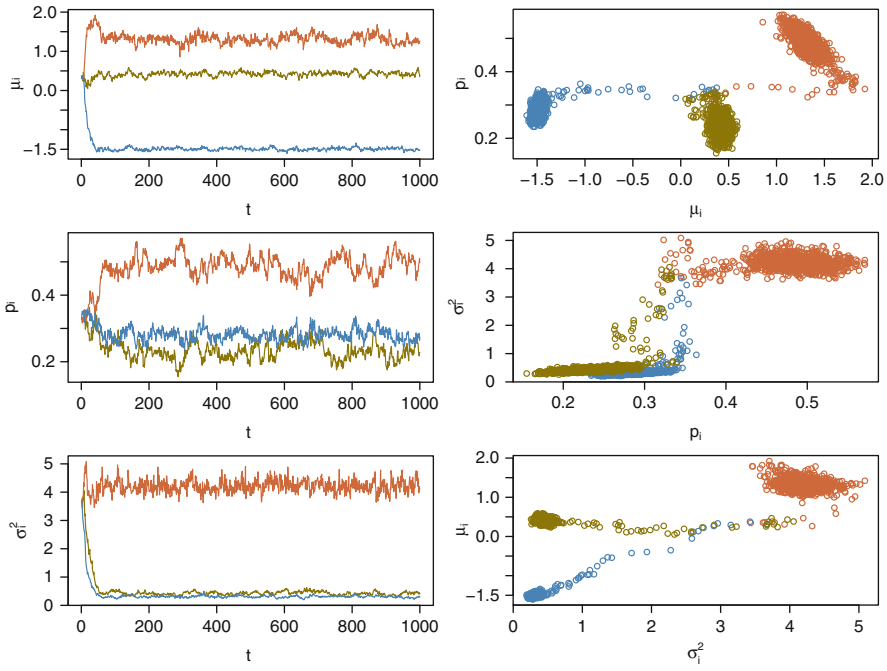

Fig. 6.7. Dataset **License**: (*left*) Convergence of the three types of parameters of the normal mixture, each component being identified by a different *grey level*/*color*; (*right*) 2×2 plot of the Gibbs sample for the three types of parameters of a normal mixture

This identifiability problem related to the exchangeability of the posterior distribution, often called "label switching," thus requires either an alternative prior modeling or a more tailored inferential approach. A naïve answer to the problem is to impose an *identifiability constraint* on the parameters, for instance defining the components by ordering the means (or the variances or

the weights) in a normal mixture (see Exercise 6.3). From a Bayesian point of view, this amounts to truncating the original prior distribution, going from $\pi(\boldsymbol{\theta}, \mathbf{p})$ to

$$\pi(\boldsymbol{\theta}, \mathbf{p})\, \mathbb{I}_{\mu_1 \leq \ldots \leq \mu_k}$$

for instance. While this seems innocuous (given that the sampling distribution is the same with or without this indicator function), the introduction of an identifiability constraint has severe consequences on the resulting inference, both from a prior and from a computational point of view. When reducing the parameter space to its constrained part, the imposed truncation has no reason to respect the topology of either the prior or the likelihood. Instead of singling out one mode of the posterior, the constrained parameter space may then well include parts of several modes and the resulting posterior mean could, for instance, lie in a very low probability region between the modes, while the high posterior probability zones are located at the boundaries of this space.

In addition, the constraint may radically modify the prior modeling and come close to contradicting the prior information. For large values of k, the introduction of a constraint also has a consequence on posterior inference: With many components, the ordering of components in terms of one of the parameters of the mixture is unrealistic. Some components will be close in mean while others will be close in variance or in weight. This may even lead to very poor estimates of the parameters if the inappropriate ordering is chosen.

Note that while imposing a constraint that is not directly related to the modal regions of the target distribution may considerably reduce the efficiency of an MCMC algorithm, it must be stressed that the constraint does not need to be imposed *during* the simulation but can instead be imposed *after* simulation by reordering the MCMC output according to the constraint. For instance, if the constraint imposes an ordering of the means, once the simulation is over, the components can be relabeled for each MCMC iteration according to this constraint; that is, defining the first component as the one associated with the smallest simulated mean and so on. From this perspective, identifiability constraints have nothing to do with (or against) simulation.

An empirical resolution of the label switching problem that avoids imposing the constraints altogether consists of arbitrarily selecting one of the $k!$ modal regions of the posterior distribution once the simulation step is over and only then operate the relabeling in terms of proximity to this region.

Given an MCMC sample of size M, we can find a Monte Carlo approximation of the *maximum a posteriori* (MAP) estimator by taking $\boldsymbol{\theta}^{(i^*)}, \mathbf{p}^{(i^*)}$ such that

$$i^* = \arg \max_{i=1,\ldots,M} \pi\left\{(\boldsymbol{\theta}, \mathbf{p})^{(i)} | \mathbf{x}\right\};$$

that is, the simulated value that gives the maximal posterior density. (Note that π does not need its normalizing constant for this computation.) This value is quite likely to be in the vicinity of one of the $k!$ modes, especially if we run many simulations. The approximate MAP estimate will thus act as a *pivot* in the sense that it gives a good approximation to a mode and we can reorder the other iterations with respect to this mode.

Rather than selecting the reordering based on a Euclidean distance in the parameter space, we use a distance in the space of allocation probabilities. Indeed, the components of the parameter vary in different spaces, from the real line for the means to the simplex for the weights. Let $\mathfrak{S}_k$ be the k-permutation set and $\tau \in \mathfrak{S}_k$. We suggest to minimize in τ an entropy distance summing the relative entropies between the $\mathbb{P}(z_t = j|\boldsymbol{\theta}^{(i^\star)}, \mathbf{p}^{(i^\star)})$'s and the $\mathbb{P}(z_t = j|\tau\left\{(\boldsymbol{\theta}^{(i)}, \mathbf{p}^{(i)})\right\})$'s, namely

$$h(i,\tau) = \sum_{t=1}^{n}\sum_{j=1}^{k} \mathbb{P}(z_t = j|\boldsymbol{\theta}^{(i^\star)}, \mathbf{p}^{(i^\star)}) \times \log\left\{\mathbb{P}(z_t = j|\boldsymbol{\theta}^{(i^\star)}, \mathbf{p}^{(i^\star)})/\mathbb{P}(z_t = j|\tau\left[(\boldsymbol{\theta}^{(i)}, \mathbf{p}^{(i)})\right])\right\} .$$

The selection of the permutations reordering the MCMC output thus reads as follows:

Algorithm 6.12 Pivotal Reordering
At iteration $i \in \{1, \ldots, M\}$:

1. Compute
$$\tau_i = \arg\min_{\tau\in\mathfrak{S}_k} h(i,\tau)\,,$$
2. Set $(\boldsymbol{\theta}^{(i)}, \mathbf{p}^{(i)}) = \tau_i\{(\boldsymbol{\theta}^{(i)}, \mathbf{p}^{(i)})\}$.

Thanks to this reordering, most iteration labels get switched to the same mode (when n gets large, this is almost a certainty), and the identifiability problem is thus solved. Therefore, after this reordering step, the Monte Carlo estimate of the posterior expectation $\mathbb{E}^\pi[\theta_i|\mathbf{x}]$,

$$\sum_{j=1}^{M} (\theta_i)^{(j)}/M\,,$$

can be used as in a standard setting because the reordering automatically gives different meanings to different components. Obviously, $\mathbb{E}^{\pi}[\theta_i|\mathbf{x}]$ (or its approximation) should also be compared with $\theta^{(i^*)}$ to check convergence.[9]

Using the Gibbs output `simu` of **License** (which is the `datha` of the following code) as in the previous illustration, the corresponding R code involves the determination of the MAP approximation

```
indimap=order(simu$lopost,decreasing=TRUE)[1]
map=list(mu=simu$mu[indimap,],sig=simu$sig[indimap,],
    p=simu$p[indimap,])
```

that is easily derived by storing the values of the log-likelihood densities in the Gibbs sampling function `gibbsnorm`. The corresponding (MAP) allocation probabilities for the data are then

```
lili=alloc=matrix(0,length(datha),3)
for (t in 1:length(datha)){
  lili[t,]=map$p*dnorm(datha[t],mean=map$mu,
  sd=sqrt(map$sig))
  lili[t,]=lili[t,]/sum(lili[t,])
  }
```

They are used as reference for the reordering:

```
ormu=orsig=orp=matrix(0,ncol=3,nrow=1000)
library(combinat)
perma=permn(3)
for (t in 1:1000){
  entropies=rep(0,factorial(3))
  for (j in 1:n){
    alloc[j,]=simu$p[t,]*dnorm(datha[j],mean=simu$mu[t,],
    sd=sqrt(simu$sig[t,]))
    alloc[j,]=alloc[j,]/sum(alloc[j,])
    for (i in 1:factorial(3))
     entropies[i]=entropies[i]+
     sum(lili[j,]*log(alloc[j,perma[[i]]]))
    }
  best=order(entropies,decreasing=TRUE)[1]
  ormu[t,]=simu$mu[t,perma[[best]]]
  orsig[t,]=simu$sig[t,perma[[best]]]
  orp[t,]=simu$p[t,perma[[best]]]
  }
```

[9] While this resolution seems intuitive enough, there is still a lot of debate in academic circles on whether or not label switching should be observed on an MCMC output and, in case it should, on which substitute to the posterior mean should be used.

An output comparing the original MCMC sample and the one corresponding to this reordering for the **License** dataset is then constructed. However, since the Gibbs sampler does not switch between the $k!$ modes in this case, the above reordering does not modify the labelling and we thus abstain from producing the corresponding graph as it is identical to Fig. 6.7.

6.6 Prior Selection

After[10] insisting in Chap. 2 that conjugate priors are not the only possibility for prior modeling, we seem to be using them quite extensively in this chapter! The fundamental reason for this is that, as explained below, it is not possible to use the standard alternative of noninformative priors on the components. Nonconjugate priors can be used as well (with Metropolis–Hastings steps) but are difficult to fathom when the components have no specific "real" meaning (as, for instance, when the mixture is used as a nonparametric proxy).

The representation (6.2) of a mixture model precludes the use of independent improper priors,

$$\pi(\boldsymbol{\theta}) = \prod_{j=1}^{k} \pi_j(\theta_j)\,,$$

since if, for any $1 \le j \le k$,

$$\int \pi_j(\theta_i)\mathrm{d}\theta_j = \infty\,,$$

then, for every n,

$$\int \pi(\boldsymbol{\theta}, \mathbf{p}|\mathbf{x})\mathrm{d}\boldsymbol{\theta}\mathrm{d}\mathbf{p} = \infty\,.$$

The reason for this inconvenient behavior is that among the k^n terms in the expansion (6.5) of $\pi(\boldsymbol{\theta}, \mathbf{p}|\mathbf{x})$, there are $(k-1)^n$ terms without *any* observation allocated to the ith component and thus there are $(k-1)^n$ terms with a conditional posterior $\pi(\theta_i|\mathbf{x}, \mathbf{z})$ that is equal to the prior $\pi_i(\theta_i)$.

The inability to use improper priors may be seen by some as a *marginalia*, a fact of little importance, since they argue that proper priors with large variances can be used instead. However, since mixtures are ill-posed problems,[11] this difficulty with improper priors is more of an issue, given that the

[10] This section may be skipped by most readers, as it only addresses the very specific issue of handling improper priors in mixture estimation.

[11] By nature, *ill-posed* problems are not precisely defined. They cover classes of models such as *inverse problems*, where the complexity of getting back from the data to the parameters is huge. They are not to be confused with nonidentifiable problems, though.

influence of a particular proper prior, no matter how large its variance, cannot be truly assessed. In other words, the prior gives a specific meaning to what distinguishes one component from another.

ϟ Prior distributions must always be chosen with the utmost care when dealing with mixtures and their bearing on the resulting inference assessed by a sensitivity study. The fact that some noninformative priors are associated with undefined posteriors, no matter what the sample size, is a clear indicator of the complex nature of Bayesian inference for those models.

6.7 Tempering

The notion of *tempering* can be found in different areas under many different denominations, but it always comes down to the same intuition that governs simulated annealing (Chap. 8), namely that when you flatten a posterior surface, it is easier to move around, while if you sharpen it, it gets harder to do so except around peaks.

More formally, given a density $\pi(x)$, we can define an associated density $\pi_\alpha(x) \propto \pi(x)^\alpha$ for $\alpha > 0$ large enough (if α is too small, $\pi(x)^\alpha$ does not integrate). An important property of this family of distributions is that they all share the same modes. When $\alpha > 1$, the surface of π_α is more contrasted than the surface of π: Peaks are higher and valleys are lower. Increasing α to infinity results in a Dirac mass at the modes of π, and this is the principle behind simulated annealing. Conversely, lowering α to values less than 1 makes the surface smoother by lowering peaks and raising valleys. In a compact space, lowering α to 0 ends up with the uniform distribution.

This rather straightforward intuition can be exploited in several directions for simulation. For instance, a tempered version of π, π_α, can be simulated in a preliminary step to determine where the modal regions of π are. (Different values of α can be used in parallel to compare the results.) This preliminary exploration can then be used to build a more appropriate proposal. Alternatively, these simulations may be pursued and associated with appropriate importance weights. Note also that a regular Metropolis–Hastings algorithm may be used with π_α just as well as with π since the acceptance ratio is transformed into

$$\left(\frac{\pi(\boldsymbol{\theta}', \mathbf{p}'|\mathbf{x})}{\pi(\boldsymbol{\theta}, \mathbf{p}|\mathbf{x})}\right)^\alpha \frac{q(\boldsymbol{\theta}, \mathbf{p}|\boldsymbol{\theta}', \mathbf{p}')}{q(\boldsymbol{\theta}', \mathbf{p}'|\boldsymbol{\theta}, \mathbf{p})} \wedge 1 \tag{6.7}$$

in the case of the mixture parameters, with the same irrelevance of the normalizing constants.

Continuation of Example 6.1. If we consider once more the posterior associated with (6.3), we can check in Fig. 6.8 the cumulative effect of a small

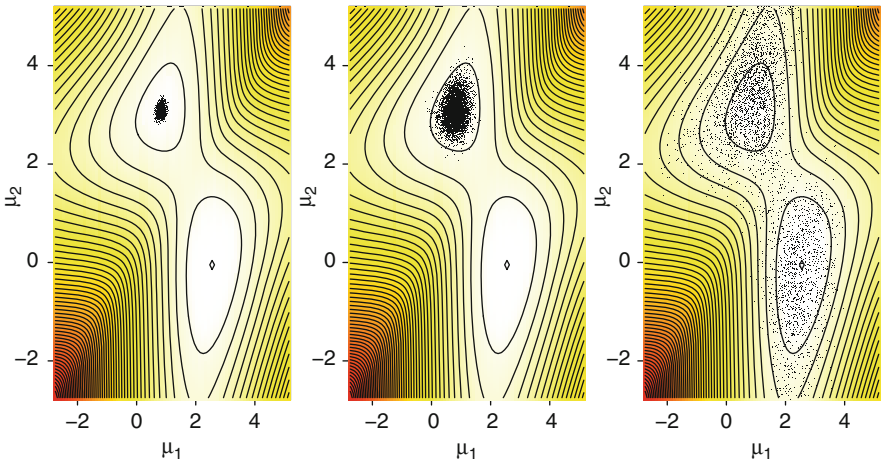

Fig. 6.8. Comparison of Metropolis–Hastings samples of 10^4 points started in the vicinity of the spurious mode for the target distributions π_α when $\alpha = 1, 0.1, 0.01$ (*from left to right*), π is the same as in Fig. 6.6, and the proposal is a random walk with variance 0.1 (the shape of log-likelihood does not changed)

variance for the random walk proposal (chosen here as 0.1) and a decrease in the power α. The R function used to produce this figure is

```
hmmeantemp=function(dat,niter=100,var=.1,alpha=1){
  mu=matrix(0,niter,2)
  mu[1,]=c(1,3)
  for (i in 2:niter){
    muprop=rnorm(2,mu[i-1,],sqrt(var))
    bound=lpost(dat,muprop)-lpost(dat,mu[i-1,])
    if (runif(1)<=exp(alpha*bound)) mu[i,]=muprop else
      mu[i,]=mu[i-1,]
    }
  mu
  }
```

It thus constitutes a very straightforward modification of the original Metropolis–Hastings algorithm. For the genuine target distribution π *(left)*, 10,000 iterations of the Metropolis–Hastings algorithm are not nearly sufficient to remove the attraction of the lower mode. When $\alpha = 0.1$, we can reasonably hope that a few thousand more iterations could bring the Markov chain toward the other mode. For $\alpha = 0.01$, only a few iterations suffice to switch modes, given that the saddle between both modes is not much lower than the modes themselves. (The best way to check this fact and to select α in practice is to run the R code!) ◀

6.8 Mixtures with an Unknown Number of Components

While the standard interpretation of mixtures gives each component a meaning, the semiparametric approach to mixtures only perceives components as base elements in a representation of an unknown density. In that perspective, the number k of components represents the degree of approximation, and it has no particular reason to be fixed in advance. Even from the traditional perspective, it may also happen that the number of homogeneous groups within the population of interest is unknown and that inference first seeks to determine this number. For instance, in a marketing study of Web-browsing behaviors, it may well be that the number of different behaviors is unknown. Also, for instance, in the analysis of financial stocks, the number of different patterns in the evolution of these stocks may be unknown to the analyst. For these different situations, it is thus necessary to extend the previous setting to include inference on the number k of components itself.

Inference on such a structure is somehow more complicated than on single models, especially when there are an infinite number of submodels, i.e. when k is not bounded, and it can be tackled from two different (or even opposite) perspectives. The first approach is to consider the variable dimension model as a whole and to estimate quantities that are meaningful for the whole model (such as moments or predictives) as well as quantities that only make sense for submodels (such as posterior probabilities of submodels and posterior moments of θ_k). From a Bayesian perspective, once a prior is defined on $\boldsymbol{\theta}$, the only difficulty is in finding an efficient way to explore the complex parameter space in order to produce these estimators. The second perspective on variable dimension models is to resort to *testing*, rather than estimation, by adopting a *model choice* stance. This requires choosing among all possible submodels the "best one" in terms of an appropriate criterion, usually through the Bayes factor (Sect. 2.3.2). The computational resolution of the comparison when the number of models is infinite requires MCMC exploration, while the variability of the resulting inference may be underestimated if the selection of the model is not accounted for in the assessment of the variability. Nonetheless, this is an approach often used in linear and generalized linear models (Chaps. 3 and 4) where subgroups of covariates are compared against a given dataset.

Mixtures with an unknown number of components are one particular instance of *variable dimension models*. Other cases include the selection of covariates among k possible covariates in a generalized linear model (Chap. 4) which can be seen as a collection of 2^k submodels (depending on the presence or absence of each covariate). Similarly, in a time series model such as the AR and MA models (Chap. 7), the value of the lag dependence can be left open, depending on the data at hand. Other instances are the determination of the order in a hidden Markov model (Chap. 7), as in DNA sequences where the dependence of the past bases may go back for one, two, or more steps, or even in a capture–recapture experiment (Chap. 5) when one estimates the number of species from the observed species.

While we opt here for a testing perspective, a more generic simulation technique called *reversible jump* has been developed by Green (1995). While it was exposed in the earlier edition (Marin and Robert, 2007), it requires both a high degree of formalization and a very sensitive calibration. In the specific case of mixtures, the number of models under comparison (i.e., the range of k) is usually small enough to prefer an enumeration of all models and hence an approximation of all marginal likelihoods.

Similarly, *Dirichlet processes* are often advanced as alternative to the estimation of the number of components for mixtures because they naturally embed a clustering mechanism. A Dirichlet process is a nonparametric object that formally involves a countably infinite number of components. Nonetheless, inference on Dirichlet processes for a finite sample size produces a random number of clusters, which can be used as an estimate of the number of components. Since the technical complexity of those objects is too high for this book, we refer to Hjort et al. (2010) for detail.

Once testing is adopted as the setting of reference, the implementation of the principle boils down to study some proposals regarding approximations of the Bayes factor oriented towards the direct exploitation of outputs from single model MCMC runs.

In fact, the major difference between approximations of Bayes factors based on those outputs and approximations based on the output from the reversible jump chains is that the latter requires a sufficiently efficient choice of proposals to move around models, which can be difficult. If we can instead concentrate the simulation effort on single models, the complexity of the algorithm decreases (a lot) and there exist ways to evaluate the performance of the corresponding MCMC samples. In addition, it is often the case that few models are in competition when estimating k and it is therefore possible to visit the whole range of potentials models in an exhaustive manner.

We have

$$f_J(\mathbf{x}|\boldsymbol{\lambda}_J) = \prod_{i=1}^{n} \sum_{j=1}^{J} p_j f(x_i|\theta_j)$$

where $\boldsymbol{\lambda}_J = (\boldsymbol{\theta}, \boldsymbol{p}) = (\theta_1, \ldots, \theta_J, p_1, \ldots, p_J)$. Most solutions (see, e.g. Frühwirth-Schnatter, 2006, Sect. 5.4) revolve around an importance sampling approximation to the marginal likelihood integral

$$m_J(x) = \int f_J(\mathbf{x}|\boldsymbol{\lambda}_J)\, \pi_J(\boldsymbol{\lambda}_J)\, \mathrm{d}\boldsymbol{\lambda}_J$$

where J denotes the model index (that is the number of components in the present case). A different possibility is to use Gelfand and Dey (1994) representation: starting from an arbitrary density g_J, the equality

$$1 = \int g_J(\boldsymbol{\lambda}_J)\,\mathrm{d}\boldsymbol{\lambda}_J = \int \frac{g_J(\boldsymbol{\lambda}_J)}{f_J(\mathbf{x}|\boldsymbol{\lambda}_J)\,\pi_J(\boldsymbol{\lambda}_J)} f_J(\mathbf{x}|\boldsymbol{\lambda}_J)\,\pi_J(\boldsymbol{\lambda}_J)\,\mathrm{d}\boldsymbol{\lambda}_J$$
$$= m_J(\mathbf{x}) \int \frac{g_J(\boldsymbol{\lambda}_J)}{f_J(\mathbf{x}|\boldsymbol{\lambda}_J)\,\pi_J(\boldsymbol{\lambda}_J)} \pi_J(\boldsymbol{\lambda}_J|\mathbf{x})\,\mathrm{d}\boldsymbol{\lambda}_J$$

implies that a potential estimate of $m_J(\mathbf{x})$ is

$$\hat{m}_J(\mathbf{x}) = 1 \Big/ \frac{1}{T} \sum_{t=1}^{T} \frac{g_J(\boldsymbol{\lambda}_J^{(t)})}{f_J(\mathbf{x}|\boldsymbol{\lambda}_J^{(t)})\,\pi_J(\boldsymbol{\lambda}_J^{(t)})}$$

when the $\boldsymbol{\lambda}_J^{(t)}$'s are produced by a Monte Carlo or an MCMC sampler targeted at $\pi_J(\boldsymbol{\lambda}_J|\mathbf{x})$.

While this solution can be easily implemented in low dimensional settings, calibrating the auxiliary density g_k is always an issue. The auxiliary density could be selected as a non-parametric estimate of $\pi_k(\boldsymbol{\lambda}_J|x)$ based on the sample itself but this is very costly. Another difficulty is that the estimate may have an infinite variance and thus be too variable to be trustworthy.

Yet another approximation to the integral $m_J(\mathbf{x})$ is to consider it as the expectation of $f_J(\mathbf{x}|\boldsymbol{\lambda}_J)$, when $\boldsymbol{\lambda}_J$ is distributed from the prior. While a brute force approach simulating $\boldsymbol{\lambda}_J$ from the prior distribution requires a huge number of simulations!

We consider here a further solution, first proposed by Chib (1995), that is straightforward to implement in the setting of mixtures. Although this method may fail because of the lack of label switching, we show below how the difficulty can easily be removed. Chib's method is directly based on the expression of the marginal distribution (loosely called *marginal likelihood* in this section) in Bayes' theorem:

$$m_J(\mathbf{x}) = \frac{f_J(\mathbf{x}|\boldsymbol{\lambda}_J)\,\pi_J(\boldsymbol{\lambda}_J)}{\pi_J(\boldsymbol{\lambda}_J|\mathbf{x})}$$

and on the property that the rhs of this equation is constant in $\boldsymbol{\lambda}_J$. Therefore, if an arbitrary value of $\boldsymbol{\lambda}_J$, $\boldsymbol{\lambda}_J^*$ say, is selected and if a good approximation to $\pi_J(\boldsymbol{\lambda}_J|\mathbf{x})$ can be constructed, $\hat{\pi}_J(\boldsymbol{\lambda}_J|\mathbf{x})$, Chib's approximation to the marginal likelihood is

$$\hat{m}_J(\mathbf{x}) = \frac{f_J(\mathbf{x}|\boldsymbol{\lambda}_J^*)\,\pi_J(\boldsymbol{\lambda}_J^*)}{\hat{\pi}_J(\boldsymbol{\lambda}_J^*|\mathbf{x})}\,. \tag{6.8}$$

In the case of mixtures, a natural approximation to $\pi_J(\boldsymbol{\lambda}_J|\mathbf{x})$ is the Rao–Blackwell estimate

$$\hat{\pi}_J(\boldsymbol{\lambda}_J^*|\mathbf{x}) = \frac{1}{T} \sum_{t=1}^{T} \pi_J(\boldsymbol{\lambda}_J^*|\mathbf{x}, \mathbf{z}^{(t)})\,, \tag{6.9}$$

where the $\mathbf{z}^{(t)}$'s are the latent variables simulated by the MCMC sampler. To be efficient, this method requires

(a) a good choice of $\boldsymbol{\lambda}_J^*$ but, since in the case of mixtures, the likelihood is computable, $\boldsymbol{\lambda}_J^*$ can be chosen as the MCMC approximation to the MAP estimator (see Algorithm 6.12) and,
(b) a good approximation to $\pi_J(\boldsymbol{\lambda}_J|\mathbf{x})$.

This latter requirement is paramount: while, at a formal level, $\hat{\pi}_J(\boldsymbol{\lambda}_J^*|\mathbf{x})$ is a converging (parametric) approximation to $\pi_J(\boldsymbol{\lambda}_J|\mathbf{x})$ by virtue of the ergodic theorem, this obviously requires the chain $(\mathbf{z}^{(t)})$ to converge to its stationarity distribution. Unfortunately, as discussed previously, in the case of mixtures, the Gibbs sampler rarely converges because of the label switching phenomenon, so the approximation $\hat{\pi}_J(\boldsymbol{\lambda}_J^*|\mathbf{x})$ is untrustworthy. It is easily seen via a numerical experiment that (6.8) is *significantly different from the true value* $m_J(\mathbf{x})$ when label switching *does not occur*. There is, however, a fix to this problem which is to recover the label switching symmetry a posteriori, replacing $\hat{\pi}_J(\boldsymbol{\lambda}_J^*|\mathbf{x})$ in (6.9) above with

$$\hat{\pi}_J(\boldsymbol{\lambda}_J^*|\mathbf{x}) = \frac{1}{T\,J!} \sum_{\sigma \in \mathfrak{S}_J} \sum_{t=1}^{T} \pi_J(\sigma(\boldsymbol{\lambda}_J^*)|\mathbf{x}, \mathbf{z}^{(t)})\,,$$

where $\mathfrak{S}_J$ denotes the set of all permutations of $\{1, \ldots, J\}$ and $\sigma(\boldsymbol{\lambda}_J^*)$ denotes the transform of $\boldsymbol{\lambda}_J^*$ where components are switched according to the permutation σ. Note that the permutation can equally be applied to $\boldsymbol{\lambda}_J^*$ or to the $\mathbf{z}^{(t)}$'s but that the former is usually more efficient from a computational point of view given that the sufficient statistics only have to be computed once. The justification for this modification stems from a Rao–Blackwellization argument, namely that the permutations are ancillary for the problem and should be integrated out.

Example 6.3. In the case of the normal mixture case and a benchmark called the "galaxy dataset" (Robert and Casella, 2004, Chap. 11, Table 11.1) Gibbs sampling does not produce any label switching. If we compute $\log \hat{m}_J(\mathbf{x})$ using Chib's original estimate (6.8), the [logarithm of the] estimated marginal likelihood is

$$\hat{\rho}_J(\mathbf{x}) = -105.1396$$

for $J = 3$ (based on 10^3 simulations), while introducing the permutations leads to

$$\hat{\rho}_J(\mathbf{x}) = -103.3479\,.$$

As noted by Frühwirth-Schnatter (2006), the difference between the original Chib's approximation and the true marginal likelihood is close to $\log(J!)$ (only) when the Gibbs sampler remains concentrated around a single mode of the posterior distribution. In the current case, we have that

$$-116.3747 + \log(2!) = -115.6816$$

exactly! (We also checked this numerical value of the marginal likelihood against a brute-force estimate obtained by simulating from the prior and

averaging the likelihood, up to a fourth digit agreement.) A similar result holds for $J = 3$, with

$$-105.1396 + \log(3!) = -103.3479\,.$$

For $J = 4$, we get for instance that the original Chib's approximation is -104.1936, while the average over permutations gives -102.6642. Similarly, for $J = 5$, the difference between -103.91 and -101.93 is less than $\log(5!)$. The $\log(J!)$ difference cannot therefore be used as a direct correction for Chib's approximation because of this difficulty in controlling the amount of overlap. However, it is unnecessary since using the permutation average resolves the difficulty. Table 6.1 shows that the preferred value of J for the **Galaxy** dataset and the current choice of prior distribution is $J = 5$.

J	2	3	4	5	6	7	8
$\hat{\rho}_J(\mathbf{x})$	-115.68	-103.35	-102.66	-101.93	-102.88	-105.48	-108.44

Table 6.1. Dataset **Galaxy**: estimations of the marginal log-likelihoods by the symmetrized Chib's approximation

When the number of components J grows too large for all permutations in $\mathfrak{S}_J$ to be considered in the average, a (random) subsample of permutations can be simulated to keep the computing time to a reasonable level (obviously keeping the identity as one of the selected permutations!), as in Table 6.1 for $J = 6, 7$. Note also that the discrepancy between the original Chib's (1995) approximation and the average over permutations is a good indicator of the mixing properties of the Markov chain, if a further convergence indicator is requested.

We implemented Chib's method for the **License** dataset in the function `gibbsnorm(niter,mix)`. The code relies on the combinatorial package `combinat` in order to store all possible permutations:

```
lolik=rep(0,niter)
library(combinat)
perms=matrix(unlist(permn(k)),ncol=k,byrow=T)
nperms=dim(perms)[1]
```

The marginal likelihood is then averaged over iterations and permutations

```
chibdeno=0
for (j in 1:nperms)
chibdeno=chibdeno+exp(sum(dnorm(mug[i+1,perms[j,]],
    mean=(mean(datha)+nxj)/(1+ssiz),
```

```
      sd=sqrt(sigg[i+1,perms[j,]])/sqrt((1+ssiz)),log=TRUE))+
    sum(dgamma(1/sigg[i+1,perms[j,]],shape=.5*(20+ssiz),
      rate=var(datha)+.5*ssum+.5*ssiz/
  (ssiz+1)*(mean(datha)-nxj/ssiz)^2,log=TRUE)
    -2*log(sigg[i+1,perms[j,]]))+
    sum((ssiz-0.5)*log(prog[i+1,perms[j,]]))+
    lgamma(sum(ssiz+0.5))-sum(lgamma(ssiz+0.5)))
```

the function returning a list `list(...,lolik=lolik,deno=chibdeno)`. Using the code,

```
> simu=gibbsnorm(1000,mix)
> lopos=order(simu$lopost)[1000]
> lnum1=simu$lolik[lopos]
> lnum2=sum(dnorm(simu$mu[lopos,],
+ mean=mean(datha),sd=simu$sig[lopos,],log=TRUE)+
+ dgamma(1/simu$sig[lopos,],10,var(datha),log=TRUE)-
+ 2*log(simu$sig[lopos,]))+
+ sum((rep(0.5,k)-1)*log(simu$p[lopos,]))+
+ lgamma(sum(rep(0.5,k)))-sum(lgamma(rep(0.5,k)))
> lchibapprox2=lnum1+lnum2-log(simu$deno)
```

we obtain Table 6.2 which gives the approximations of the marginal likelihoods from $k = 2$ to $k = 8$. For the **License** dataset, the favored number of components is thus $k = 4$.

k	2	3	4	5	6
$\hat{\rho}_k(\mathbf{x})$	−5373.445	−5315.351	−5308.79	−5336.23	−5341.524

Table 6.2. Dataset **License**: estimations of the marginal log-likelihoods by the symmetrized Chib's approximation

6.9 Exercises

6.1 Show that a mixture of Bernoulli distributions is again a Bernoulli distribution. Extend this to the case of multinomial distributions.

6.2 Show that the number of nonnegative integer solutions of the decomposition of n into k parts such that $n_1 + \ldots + n_k$ is equal to

$$\mathfrak{r} = \binom{n+k-1}{n}.$$

Deduce that the number of partition sets is of order $O(n^{k-1})$. (*Hint:* This is a classical combinatoric problem.)

6.3 For a mixture of two normal distributions with all parameters unknown,

$$p\mathcal{N}(\mu_1,\sigma_1^2)+(1-p)\mathcal{N}(\mu_2,\sigma_2^2)\,,$$

and for the prior distribution $(j=1,2)$

$$\mu_j|\sigma_j \sim \mathcal{N}(\xi_j,\sigma_j^2/n_j)\,, \quad \sigma_j^2 \sim \mathcal{IG}(\nu_j/2,s_j^2/2)\,, \quad p\sim\mathcal{B}e(\alpha,\beta)\,,$$

show that

$$p|\mathbf{x},\mathbf{z}\sim\mathcal{B}e(\alpha+\ell_1,\beta+\ell_2),$$

$$\mu_j|\sigma_j,\mathbf{x},\mathbf{z}\sim\mathcal{N}\left(\xi_1(\mathbf{z}),\frac{\sigma_j^2}{n_j+\ell_j}\right)\,,\ \sigma_j^2|\mathbf{x},\mathbf{z}\sim\mathcal{IG}((\nu_j+\ell_j)/2,s_j(\mathbf{z})/2)\,,$$

where ℓ_j is the number of z_i equal to j, $\bar{x}_j(\mathbf{z})$ and $\hat{s}_j^2(\mathbf{z})$ are the empirical mean and variance for the subsample with z_i equal to j, and

$$\xi_j(\mathbf{z})=\frac{n_j\xi_j+\ell_j\bar{x}_j(\mathbf{z})}{n_j+\ell_j}\,,\quad s_j(\mathbf{z})=s_j^2+\ell_j\hat{s}_j^2(\mathbf{z})+\frac{n_j\ell_j}{n_j+\ell_j}(\xi_j-\bar{x}_j(\mathbf{z}))^2\,.$$

Compute the corresponding weight $\omega(\mathbf{z})$.

6.4 For the normal mixture model of Exercise 6.3, compute the function $Q(\theta_0,\theta)$ and derive both steps of the EM algorithm. Apply this algorithm to a simulated dataset and test the influence of the starting point θ_0.

6.5 In the mixture model with independent priors on the θ_j's, show that the θ_j's are dependent on each other given (only) $\mathbf{x}$ by summing out the $\mathbf{z}$'s.

6.6 Construct and test the Gibbs sampler associated with the (ξ,μ_0) parameterization of (6.3), when $\mu_1=\mu_0-\xi$ and $\mu_2=\mu_0+\xi$.

6.7 Show that, if an exchangeable prior π is used on the vector of weights $(p_1,\ldots,p_k)$, then, necessarily, $\mathbb{E}^\pi[p_j]=1/k$ and, if the prior on the other parameters $(\theta_1,\ldots,\theta_k)$ is also exchangeable, then $\mathbb{E}^\pi[p_j|x_1,\ldots,x_n]=1/k$ for all j's.

6.8 Show that running an MCMC algorithm with target $\pi(\theta|\mathbf{x})^\gamma$ will increase the proximity to the MAP estimate when $\gamma>1$ is large. (*Note*: This is a crude version of the *simulated annealing* algorithm. See also Chap. 8.) Discuss the modifications required in Algorithm 6.11 to achieve simulation from $\pi(\theta|\mathbf{x})^\gamma$ when $\gamma\in\mathbb{N}^*$ is an integer.

6.9 Show that the ratio (6.7) goes to 1 when α goes to 0 when the proposal q is a random walk. Describe the average behavior of this ratio in the case of an independent proposal.

6.10 If one needs to use importance sampling weights, show that the simultaneous choice of several powers α requires the computation of the normalizing constant of π_α.

6.11 In the setting of the mean mixture (6.3), run an MCMC simulation experiment to compare the influence of a $\mathcal{N}(0,100)$ and of a $\mathcal{N}(0,10000)$ prior on (μ_1,μ_2) on a sample of 500 observations.

6.12 Show that, for a normal mixture $0.5\,\mathcal{N}(0,1)+0.5\,\mathcal{N}(\mu,\sigma^2)$, the likelihood is unbounded. Exhibit this feature by plotting the likelihood of a simulated sample using the R image procedure.