AN INTRODUCTION TO FINITE MIXTURE MODELS

ABEL RODRÍGUEZ

January, 2018

PREFACE

This monograph provides an introduction to finite mixture models, which are an extremely popular tools in statistics and machine learning. It is intended to be a companion of the homonymous course offered by the author and UCSC on the Coursera platform (https://www.coursera.org/).

Mixture models provide a flexible approach to modeling data and are useful in density estimation, clustering and classification problems:

1. Standard families of probability distributions such as the Gaussian, exponential or Poisson are often too restrictive for modeling features of real data such as multimodality or zero inflation. Mixture models, which can be related to kernel density estimation procedures, address this issue in a way that allows for natural generalizations of well-known procedures.

2. In addition to providing flexible probability distributions, finite mixture models have a strong relationship with classical clustering and classification procedures such as K-mean clustering, as well as linear and quadratic discriminant analysis. More generally they provide a tool to understand and generalize these approaches, as well as to quantify the uncertainty associated with the estimates and predictions generated by them.

In addition to the basics of calculus-based probability, this course assumes that you are familiar with the principles of maximum likelihood and Bayesian estimation, including familiarity with common computational tools such as the Expectation-Maximization and Markov chain Monte Carlo algorithms (particularly with Gibbs sampling and Random Walk Metropolis-Hastings methods). It also assumes basic knowledge of the `R` language. If needed, there are a number of excellent online courses (in Coursera, as well as in other platforms) that you can use to strengthen your background.

Notations

We reserve the capital letters F, G, H and P to denote generic probability distribution functions, while their lowercase counterparts f, g, h and p represent the associated densities/probability mass functions. Similarly, the capital letters such as X, Y and Z will be used to denote random variables, while their lowercase counterparts x, y and z to represent realizations of the random variable. Both vectors and scalars are denoted with either Greek or Latin letters not reserved for probability densities or realizations of random variables, but we do not distinguish between. On the other hand, matrices are denoted with uppercase letters not reserved for probability distributions or random variables.

CONTENTS

1 BASIC CONCEPTS

1.1 DEFINITION OF A FINITE MIXTURE MODEL

Let $\omega_1, \ldots, \omega_K$ be a collection of real numbers such that $0 \leqslant \omega_k \leqslant 1$ and $\sum_{k=1}^{K} \omega_k = 1$, and $G_1, \ldots, G_K$ be a collection of cumulative distribution functions. A random variable X with cumulative distribution function $F(x) = \Pr(X \leqslant x)$ of the form

$$F(x) = \sum_{k}^{K} \underbrace{\omega_k}_{\text{Weight}} \underbrace{G_k(x)}_{\text{Component}}$$

is said to follow finite mixture distribution with K components. The associated density/probability mass function takes the form

$$f(x) = \sum_{k}^{K} \omega_k g_k(x),$$

where $g_k(x)$ is the density associated with $G_k(x)$.

The values $\omega_1, \ldots, \omega_K$ are usually called the "weights" of the mixture, and the distributions $G_1, \ldots, G_K$ are called the "components" of the mixture. Each component will typically belong to a parametric family that is indexed by its own parameter θ_k. We will write $G_k(x) = G_k(x \mid \theta_k)$ whenever it is necessary to highlight the dependence on these parameters.

It is often the case that $G_1, \ldots G_K$ all belong to the same family and differ only in the value parameters associated with each of the distributions, so that $G_k(x \mid \theta_k) = G(x \mid \theta_k)$. In that case, the function G (and sometimes its density/probability mass function g) are called the "kernel" of the mixture. For example, we could define a mixture with $K = 3$ components, with $G(x \mid \theta_1)$, $G(x \mid \theta_2)$ and $G(x \mid \theta_3)$ all corresponding to exponential distributions with means θ_1, θ_2 and θ_3 respectively. In that case, the cumulative distribution function of the mixture is given by

$$F(x) = \omega_1 \left(1 - \exp\left\{\frac{x}{\theta_1}\right\}\right) + \omega_2 \left(1 - \exp\left\{\frac{x}{\theta_2}\right\}\right) + \omega_3 \left(1 - \exp\left\{\frac{x}{\theta_3}\right\}\right),$$

for $x \geqslant 0$ and 0 otherwise, while the associated density function is

$$f(x) = \frac{\omega_1}{\theta_1} \exp\left\{\frac{x}{\theta_1}\right\} + \frac{\omega_2}{\theta_2} \exp\left\{\frac{x}{\theta_2}\right\} + \frac{\omega_3}{\theta_3} \exp\left\{\frac{x}{\theta_3}\right\}, \tag{1}$$

again for $x \geqslant 0$ and 0 otherwise. Similarly,

$$f(x) = \begin{cases} \omega \frac{x^{\nu_1 - 1}}{\Gamma(\nu_1)\lambda_1^{\nu_1}} \exp\left\{\frac{x}{\lambda_1}\right\} + (1-\omega) \frac{x^{\nu_2 - 1}}{\Gamma(\nu_2)\lambda_2^{\nu_2}} \exp\left\{\frac{x}{\lambda_2}\right\} & x \geqslant 0 \\ 0 & \text{otherwise} \end{cases}$$

is the density of a mixture of two Gammas with mixture weights ω and $1 - \omega$, and component-specific parameters $\theta_1 = (\nu_1, \lambda_1)$ and $\theta_2 = (\nu_2, \lambda_2)$ (in the previous expression, $\Gamma(a) = \int_0^\infty z^{a-1} \exp\{-z\} dz$ denotes the well-known Gamma function). Plots of the cumulative distribution and density

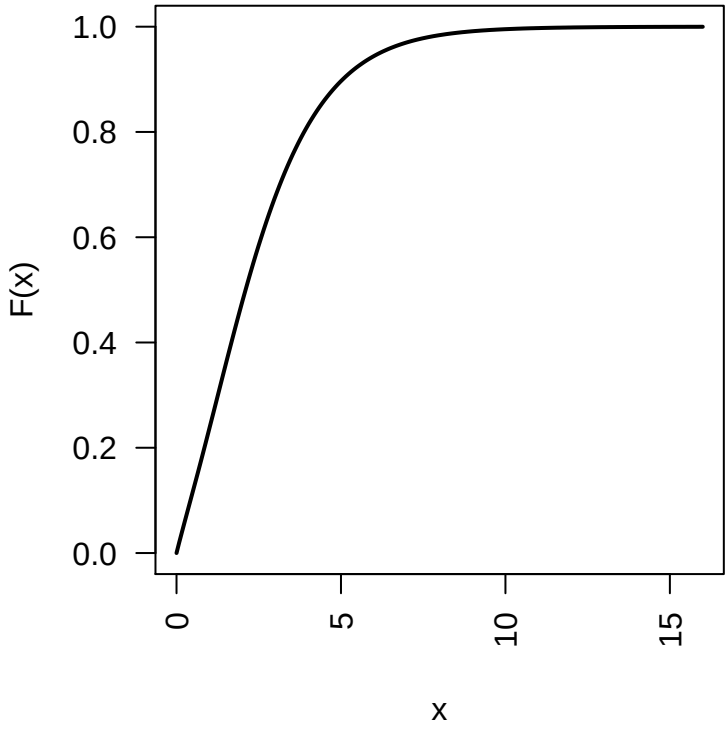

Figure 1: Probability distribution function for a mixture of two Gamma distribution with parameters $\omega = 1/2$, $\nu_1 = 3$, $\nu_2 = 1$, $\lambda_1 = 1$ and $\lambda_2 = 2$.

functions for this mixture in the case $\omega = 1/2$, $\nu_1 = 3$, $\nu_2 = 1$, $\lambda_1 = 1$ and $\lambda_2 = 2$ can be seen Figures 1 and 2, respectively. Note that the density function in particular has a very particular shape around the origin that is not commonly observed when working with standard distributions

The mean and variance of a random variable that follows a mixture distribution can be computed in terms of the weights and the mean and variances of each component:

$$\begin{aligned}
E_F(X) &= \sum_{k=1}^{K} \omega_k E_{G_k}(X) \\
Var_F(X) &= E_F(X^2) - \{E_F(X)\}^2 \\
&= \sum_{k=1}^{K} \omega_k \left\{ E_{G_k}(X^2) \right\} - \left\{ \sum_{k=1}^{K} \omega_k E_{G_k}(X) \right\}^2 \\
&= \sum_{k=1}^{K} \omega_k \left\{ Var_{G_k}(X) + \left[E_{G_k}(X) \right]^2 \right\} - \left\{ \sum_{k=1}^{K} \omega_k E_{G_k}(X) \right\}^2,
\end{aligned}$$

where $E_F(X)$ and $Var_F(X)$ denotes, respectively, the expected value and the variance of X if X is distributed according to F. Note that, while the mean of the mixture is a linear combination of the means of the components, that is

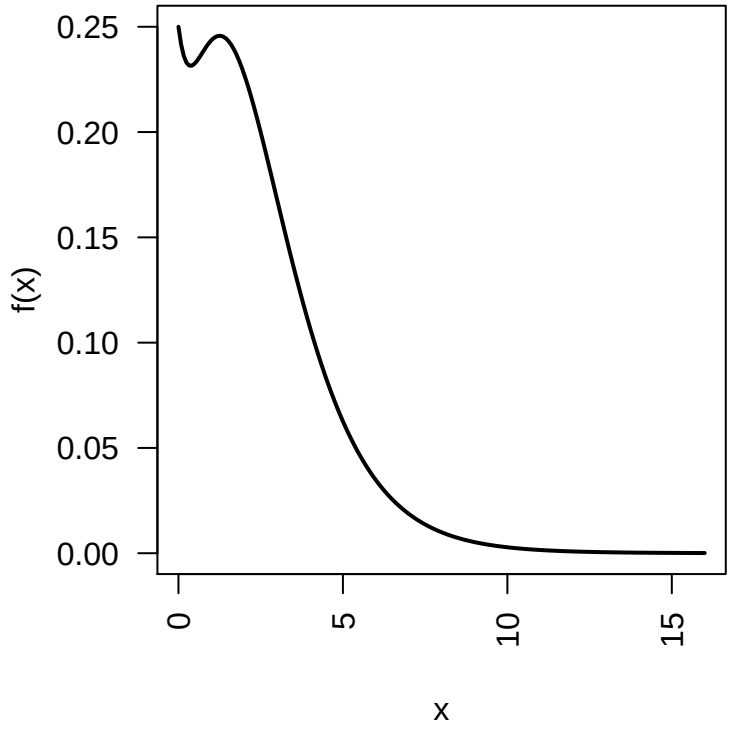

Figure 2: Probability distribution function for a mixture of two Gamma distribution with parameters $\omega = 1/2$, $\nu_1 = 3$, $\nu_2 = 1$, $\lambda_1 = 1$ and $\lambda_2 = 2$.

not typically the case for the variance. Once of the few exceptions is when the mean of each component is zero, in which case

$$Var_F(X) = \sum_{k=1}^{K} \omega_k \left\{ Var_{G_k}(X) + \left[\cancelto{0}{E_{G_k}(X)} \right]^2 \right\} - \left\{ \sum_{k=1}^{K} \omega_k \cancelto{0}{E_{G_k}(X)} \right\}^2 \tag{2}$$

$$= \sum_{k=1}^{K} \omega_k Var_{G_k}(X). \tag{3}$$

As an example, consider again the three-component mixture of exponential distributions with density 1. The mean is simply

$$E_F(X) = \omega_1\theta_1 + \omega_2\theta_2 + \omega_3\theta_3$$

while the variance is

$$Var_F(X) = \omega_1 2\theta_1^2 + \omega_2 2\theta_2^2 + \omega_3 2\theta_3^2 - \{\omega_1\theta_1 + \omega_2\theta_2 + \omega_3\theta_3\}^2$$

1.2 WHY FINITE MIXTURE MODELS?

Finite mixtures of distributions within a single family provide a lot of flexibility. For example, a mixture of Gaussian distributions can have a bimodal density. An example is given by

$$f(x) = 0.6\frac{1}{\sqrt{2\pi}} \exp\left\{-\frac{1}{2}x^2\right\} + 0.4\frac{1}{\sqrt{2\pi}}\frac{1}{2} \exp\left\{-\frac{1}{2}\frac{(x-5)^2}{4}\right\}.$$

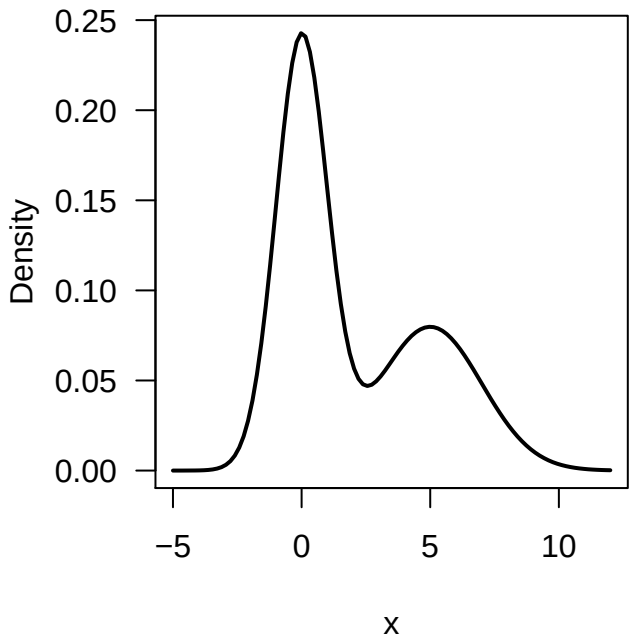

Figure 3: An example of a bimodal density generated by a mixture of two Gaussian distributions

The fact that this distribution is bimodal can be easily verified by plotting the density function in R (see Figure 3):

```
# Mixture of univariate Gaussians leading to bimodality
x = seq(-5, 12, length=100)
y = 0.6*dnorm(x, 0, 1) + 0.4*dnorm(x, 5, 2)
par(mar=c(4,4,1,1)+0.1)
plot(x, y, type="l", ylab="Density", las=1, lwd=2)
```

Alternatively, the mixture can look like a unimodal but skewed distribution. For example, the mixture

$$f(x) = 0.55\frac{1}{\sqrt{2\pi}\sqrt{2}}\exp\left\{-\frac{1}{2}\frac{x^2}{2}\right\} + 0.45\frac{1}{\sqrt{2\pi}}\frac{1}{4}\exp\left\{-\frac{1}{2}\left(\frac{x-3}{4}\right)^2\right\},$$

has a heavier right tail, which you can again verify by plotting it (see Figure 4).

```
# Mixture of univariate Gaussians, unimodal skewed
x = seq(-5, 12, length=100)
y = 0.55*dnorm(x, 0, sqrt(2)) + 0.45*dnorm(x, 3, 4)
par(mar=c(4,4,1,1)+0.1)
plot(x, y, type="l", ylab="Density", las=1, lwd=2)
```

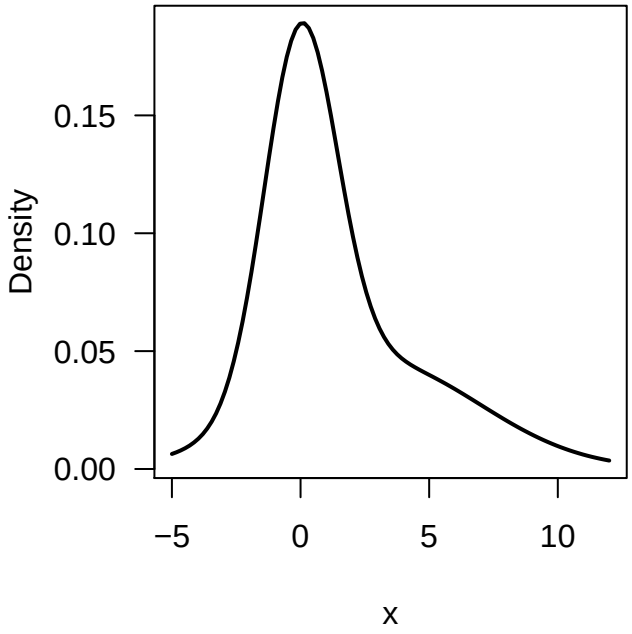

Figure 4: An example of a unimodal but skewed density generated by a mixture of two Gaussian distributions

Finally, a mixture can have symmetric but heavy-tailed distribution. For example, the following code can be used to compare the density of the three-component mixture

$$f(x) = 0.4\frac{1}{\sqrt{2\pi}\sqrt{2}}\exp\left\{-\frac{1}{2}\frac{x^2}{2}\right\} + 0.4\frac{1}{\sqrt{2\pi}}\frac{1}{4}\exp\left\{-\frac{1}{2}\frac{x^2}{16}\right\} + 0.2\frac{1}{\sqrt{2\pi}}\frac{1}{20}\exp\left\{-\frac{1}{2}\frac{x^2}{20}\right\}.$$

with that of a single normal distribution with the same variance $0.4 \times 2 + 0.4 \times 16 + 0.2 \times 20 = 11.2$ (note that in this case the expected value of each component, and therefore the mixture, is equal to zero, and equation (2) applies). See Figure 5.

```
# Mixture of univariate Gaussians, unimodal heavy tail
x = seq(-12, 12, length=100)
y = 0.40*dnorm(x, 0, sqrt(2)) +
    0.40*dnorm(x, 0, sqrt(16)) +
    0.20*dnorm(x, 0, sqrt(20))
z = dnorm(x, 0, sqrt(0.4*2+0.4*16+0.2*20))
par(mar=c(4,4,1,1)+0.1)
plot(x, y, type="l", ylab="Density", las=1, lwd=2)
lines(x, z, lty=2, lwd=2)
legend(2, 0.16, c("Mixture","Gaussian"), lty=c(1,2), bty="n",
       cex=0.77, lwd=c(2,2))
```

This flexibility means that mixtures that involve a single kernel (and, in particular, mixtures of Gaussian distributions) can be a very useful tool for density estimation. Nonetheless, not all useful mixture models involve components that belong to the same family of distributions. In some circum-

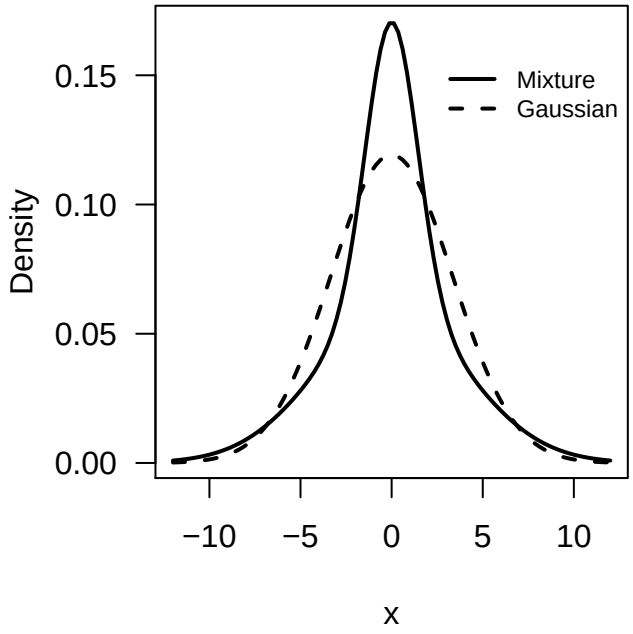

Figure 5: An example of a unimodal, heavy tailed density generated by a mixture of three Gaussian distributions. To make the presence of heavy tails clear, the figure also depicts the density of a Gaussian distribution with the same mean and variance as the mixture.

stances it is useful to allow for different components to belong to different families, and even to blend discrete a continuous distributions. For example for zero-inflated data (i.e., data that contains a large number zeros) mixing a member of a standard family of distributions with a degenerate distribution at zero can be particularly helpful. For example, a zero-inflated negative binomial distribution with probability mass function:

$$p(x) = \begin{cases} \omega_1 + (1-\omega_1)\theta^r & x = 0 \\ (1-\omega_1)\binom{x+r-1}{x}\theta^r(1-\theta)^x & x \geqslant 1 \end{cases}$$

is an example of a mixture distribution in which component 1 corresponds to a point mass at zero and component 2 to a negative binomial distribution with parameters r and θ (note that we have explicitly used the fact that $\omega_2 = 1-\omega_1$). In this case the value of ω_1 controls the number of additional zeros that you observed over those expected under a negative binomial distribution with parameters r and θ. This can most easily be seen by plotting the probability mass function of the regular negative binomial distribution

$$p^*(x) = \binom{x+r-1}{x}\theta^r(1-\theta)^x$$

against that of the zero inflated version (see Figures 6).

```
x = seq(0, 15)
y = dnbinom(x, 8, 0.6)
z = 0.2*c(1,rep(0,length(x)-1)) + (1-0.2)*y
par(mfrow=c(2,1))
par(mar=c(4,4,1,1)+0.1)
```

```
barplot(y, names.arg=x, las=1, xlab = "x", ylab="Probability",
        border=NA)
par(mar=c(4,4,1,1)+0.1)
barplot(z, names.arg=x, las=1, xlab = "x", ylab="Probability",
        border=NA)
```

Similarly, we can write a zero-inflated log-Gaussian distribution as a mixture of log-Gaussian distribution and a degenerate distribution at zero. The corresponding density takes the form

$$f(x) = \begin{cases} \omega_1 \delta_0(x) + (1-\omega_1) \frac{1}{\sqrt{2\pi}\sigma x} \exp\left\{-\frac{\ln x - \mu}{2\sigma^2}\right\} & x \geqslant 0 \\ 0 & \text{otherwise,} \end{cases}$$

where $\delta_a(x)$ denotes the Dirac delta function. The cumulative distribution function is easy to graph (see Figure 7) but note that graphing the density f is more difficult because it involves plotting the Dirac delta function.

```
x = seq(-2, 15, length=1000)
y = plnorm(x, 1.5, 0.5)
z = 0.3*as.numeric(x>=0) + (1-0.3)*y
par(mar=c(4,4,1,1)+0.1)
plot(x, y, type="l", las=1, lty=2, xlab="x",
     ylab="Cumulative distribution Function", lwd=2)
lines(x, z, lty=1, lwd=2)
legend(4, 0.45, c("Zero infla. log Gaussian","log Gaussian"),
     lty=c(1,2), bty="n", lwd=c(2,2))
```

1.3 HIERARCHICAL REPRESENTATION OF FINITE MIXTURES

Recall that the cumulative distribution function of a mixture takes the form

$$F(x) = \sum_{k}^{K} \omega_k G_k(x),$$

where $G_1(x), \ldots, G_K(x)$ are themselves well-defined cumulative distribution functions. Hence, the statement "X is distributed according to F", $X \sim F$, can be rewritten by introducing a (discrete) random variables $c \in \{1, \ldots, K\}$ such that

$$X \mid c \sim G_c, \qquad P(c = k) = \omega_k.$$

Indeed, note that the marginal distribution for X obtained by integrating the implied joint distribution of X and c over c is $\int P(x \mid c) P(dc) = \sum_{k=1}^{K} G_k \omega_k = F(x)$ as expected.

This hierarchical representation is key both for computation and interpretation. Consider for example the problem of generating a sample $x_1, \ldots, x_n$ of independent and identically realizations from a mixture model with cumulative distribution function F. If you know how to simulate from each of the components $G_1, \ldots, G_K$, then you can easily simulate from the mixture by first randomly sampling the component that generated each observation, and then sampling the actual observation from the chosen component. In particular, to generate an independent sample $x_1, \ldots, x_n$ from a mixture distribution F you can repeat the following two steps for each $i = 1, 2, \ldots, n$:

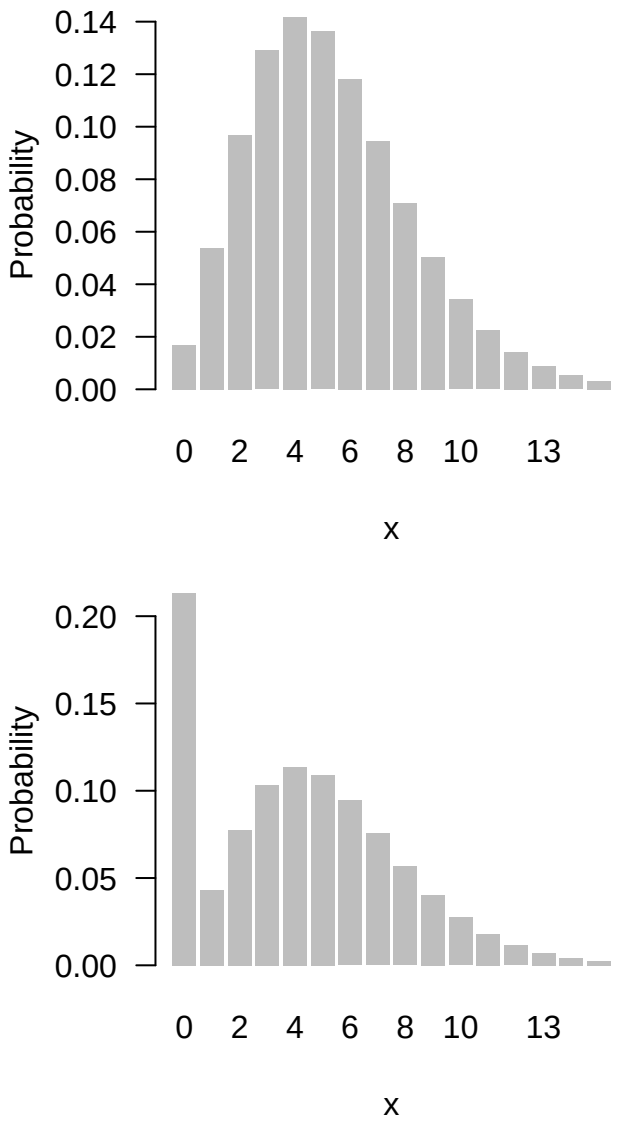

Figure 6: Comparison of the probability mass function of a negative binomial distribution (top panel) against a zero inflated mixture of that same distribution and a point mass at zero (bottom panel). Mixtures of this type are often useful to model zero inflated data.

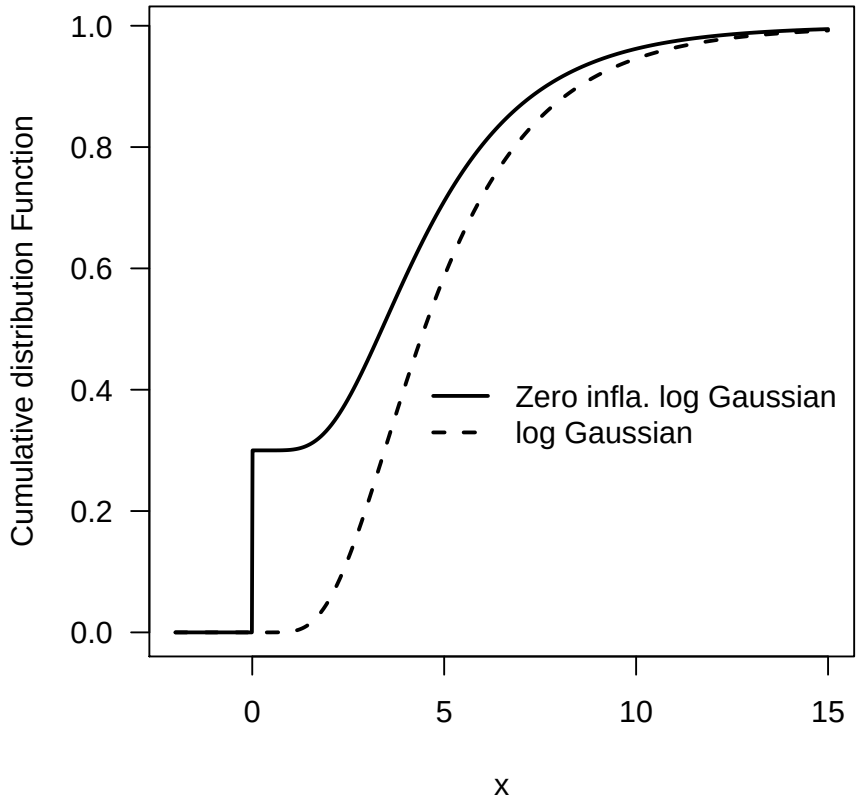

Figure 7: Comparison of the probability distribution function of a log-Gaussian distribution against a zero inflated mixture of that same distribution and a point mass at zero. Mixtures of this type are often useful to model zero inflated data.

1. Select c_i independently from a discrete distribution on $\{1,\ldots,K\}$ with associated probabilities $\omega_1,\ldots,\omega_K$.
2. Simulate x_i from G_{c_i}.

For example, we can use the following R code to simulate 50 observations from a mixture of two Gaussian distributions

$$f(x) = 0.6\frac{1}{\sqrt{2\pi}}\exp\left\{-\frac{1}{2}x^2\right\} + 0.4\frac{1}{\sqrt{2\pi}2}\exp\left\{-\frac{1}{2}\left(\frac{x-5}{2}\right)^2\right\}.$$

(See also Figure 8).

```
# Generate n observations from a mixture of two Gaussian
# distributions
n     = 50
w     = c(0.6, 0.4)  # Weights
mu    = c(0, 5)      # Means
sigma = c(1, 2)      # Standard deviations
cc    = sample(1:2, n, replace=T, prob=w)
x     = rnorm(n, mu[cc], sigma[cc])

# Plot f(x) along with the observations
# just sampled
xx = seq(-5, 12, length=200)
yy = w[1]*dnorm(xx, mu[1], sigma[1]) +
     w[2]*dnorm(xx, mu[2], sigma[2])
par(mar=c(4,4,1,1)+0.1)
plot(xx, yy, type="l", ylab="Density", xlab="x", las=1, lwd=2)
points(x, y=rep(0,n), pch=1)
```

The simulation exercise should make it clear that the auxiliary variable c_i indicates which component of the mixture generates the i-th observation, and that the weight ω_k represents the average number of observations from component k that we expect to see in a random sample. The variables $c_1,\ldots,c_n$ will be particularly important when we use mixture models for clustering and classification (unsupervised and supervised classification).

1.4 THE LIKELIHOOD FUNCTION FOR MIXTURE MODELS

We have discussed some of the properties of mixture models as a data-generation mechanism, i.e., as a tool that can be used to generate data once the parameters of the mixture $\theta = (\theta_1,\ldots,\theta_K)$ and $\boldsymbol{\omega} = (\omega_1,\ldots,\omega_K)$ have been specified. However, in most practical applications we are more interested on first solving the inverse problem: learning the values of these parameters from an independent and identically distributed random sample $\mathbf{x} = (x_1,\ldots,x_n)$. The likelihood function, which as you might recall is defined as a function of the parameters that is proportional to the joint probability density/mass function of the data given the parameters, will be the key tool we use to solve this inverse problem.

The likelihood function associated with an independent and identically distributed sample from a mixture model takes the form

$$L_O(\theta,\boldsymbol{\omega};\mathbf{x}) \propto p(\mathbf{x}\mid\theta,\boldsymbol{\omega}) = \prod_{i=1}^{n}\sum_{k=1}^{K}\omega_k g_k(x_i\mid\theta_k),$$

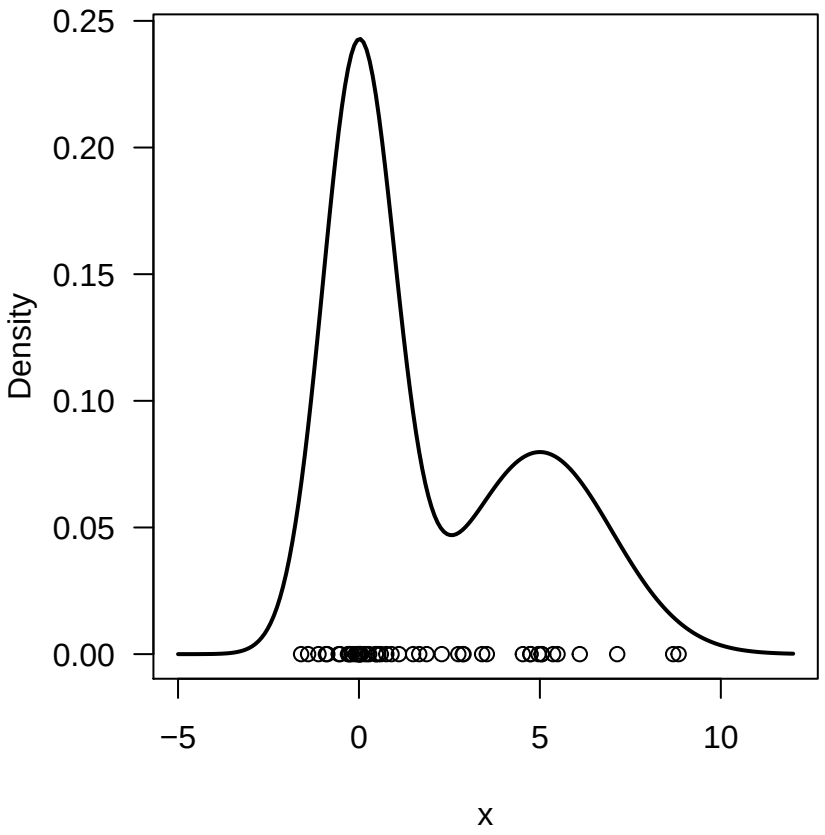

Figure 8: Density of a mixture of two Gaussian distributions along with 50 randomly generated samples from it.

where g_k is either the density of the probability mass function associated with G_k. We call this the *observed-data* likelihood because if is based only of the vector of observations $\mathbf{x}$ and does not involve the indicators $\mathbf{c} = (c_1, \ldots, c_n)$.

We can also define a *complete-data likelihood* that involves the observations $\mathbf{x}$ as well as the indicators $\mathbf{c}$ using the hierarchical representation discussed in Section 1.3.

$$L(\theta, \omega; \mathbf{x}, \mathbf{c}) p(\mathbf{x}, \mathbf{c} \mid \theta, \omega) = \prod_{i=1}^{n} \prod_{k=1}^{K} [\omega_k g_k(x_i \mid \theta_k)]^{\mathbf{1}(c_i = k)},$$

where $\mathbf{1}(\cdot)$ represents the indicator function, so that

$$\mathbf{1}(c_i = k) = \begin{cases} 1 & c_i = k, \\ 0 & \text{otherwise}. \end{cases}$$

By introducing the indicators $c_1, \ldots, c_n$ we have transformed the sum that appears in the definition of the density of a mixture into a product over the components. Note, however, that for a given value of c_i, only one of the K terms involved in the innermost product is different from zero (the one corresponding to $k = c_i$). This version of the full-data likelihood will play a key role in the development of Expectation-Maximization algorithms for maximum likelihood estimation.

An alternative way to write the full data likelihood is as the product of two distributions

$$p(\mathbf{x}, \mathbf{c} \mid \theta, \omega) = p(\mathbf{x}, \mid \mathbf{c}, \theta) p(\mathbf{c} \mid \omega),$$

where

$$p(\mathbf{x}, \mid \mathbf{c}, \theta) = \prod_{i=1}^{n} g_{c_i}(x_i \mid \theta_{c_i})$$

$$p(\mathbf{c} \mid \omega) = \prod_{k=1}^{K} \omega_k^{\sum_{i=1}^{n} \mathbf{1}(c_i = k)}.$$

This representation will be key when we develop Markov chain Monte Carlo algorithm for Bayesian inference in finite mixture models.

1.5 PARAMETER IDENTIFIABILITY

You might recall that a probability model is identifiable if and only if different values of the parameters generate different probability distributions of the observable variables.

One challenge involved in working with mixture models is that they are not fully identifiable. To see why, consider a mixture of two univariate Gaussian distributions where $\omega_1 = 0.3$, $\omega_2 = 0.7$, $\mu_1 = 0$, $\mu_2 = 1$, $\sigma_1 = 1$ and $\sigma_2 = 2$. The associated density

$$f(x) = 0.3 \frac{1}{\sqrt{2\pi}} \exp\left\{-\frac{1}{2}x^2\right\} + 0.7 \frac{1}{\sqrt{2\pi 2}} \exp\left\{-\frac{1}{2}\left(\frac{x-1}{2}\right)^2\right\}$$

is identical to the density of another mixture with parameters $\omega_1 = 0.7$, $\omega_2 = 0.3$, $\mu_1 = 1$, $\mu_2 = 0$, $\sigma_1 = 2$ and $\sigma_2 = 1$.

$$f(x) = 0.7 \frac{1}{\sqrt{2\pi 2}} \exp\left\{-\frac{1}{2}\left(\frac{x-1}{2}\right)^2\right\} + 0.3 \frac{1}{\sqrt{2\pi}} \exp\left\{-\frac{1}{2}x^2\right\}.$$

In other words, the labels used to distinguish the components in the mixture are not identifiable. The literature sometimes refers to this type of lack of identifiability as the *label switching* "problem". Whether label switching is an actual problem or not depends on the computational algorithm being used to fit the model, and the task we are attempting to complete in any particular case. For example, label switching tends to not be an issue for the purpose of density estimation or classification problems, but it can lead to serious difficulties in clustering problems. We delay a more thorough discussion of identification issues until Chapters 4 and 5.

Identifiability problems in mixture models are not restricted to label switching and can arise also when attempting to determine the number of components in the mixture. For example, note that we can write any probability distribution as a mixture of K identical components, e.g.,

$$\begin{aligned} f(x) &= \frac{1}{\sqrt{2\pi}\sigma} \exp\left\{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right\} \\ &= \sum_{k=1}^{K} \omega_k \frac{1}{\sqrt{2\pi}\sigma} \exp\left\{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right\} \end{aligned}$$

for any set of weights $\omega_1, \ldots, \omega_K$. Similarly, any density can be written as a mixture with K components in which $K-1$ have zero weights.

In practical terms, this means that a K-component mixture that involves rare (i.e., low weight) components or where some components have parameters that are very close to each other will be hard to differentiate from mixtures with a $K' < K$ components. This makes the problem of selecting the number of components in the mixture a difficult one. Again, we delay further discussion of these issues until Chapter 5.

2 MAXIMUM LIKELIHOOD ESTIMATION FOR MIXTURE MODELS

Maximum likelihood estimation is the most common approach to estimate the parameters of statistical models. However, attempting to obtain maximum likelihood estimates (MLEs) $\hat{\omega}$ and $\hat{\theta}$ by directly maximizing the observed-data likelihood

$$(\hat{\omega}, \hat{\theta}) = \arg\max_{\omega,\theta} \prod_{i=1}^{n} \sum_{k=1}^{K} \omega_k g_k(x_i \mid \theta_k),$$

is often computationally unfeasible. Indeed, there is rarely a closed-form solution for this maximization problem, and standard numerical algorithms such as the Newton-Raphson algorithm often present convergence issues, particularly for mixtures with a large number of components.

2.1 EXPECTATION MAXIMIZATION ALGORITHMS FOR MIXTURE MODELS

A common computational approach used for maximum likelihood estimation of mixture models is the expectation-maximization (EM) algorithm. The EM algorithm is an iterative algorithm for computing maximum likelihood estimators in the presence of missing data/latent variables. In the case of mixture models, the latent indicators $c_1, \ldots, c_n$ act as our missing variables. It is worthwhile noting that the version of the EM algorithm for mixture models was developed in the 1970s well before the general EM algorithm was published.

The EM algorithm is iterative, and therefore requires that we select initial values $\omega^{(0)}$ and $\theta^{(0)}$ for the parameters of the model. Given these initial values, the algorithm proceeds by repeatedly updating these values using two steps:

$$\begin{aligned} &\textbf{E step: Set} \quad Q\left(\omega, \theta \mid \omega^{(t)}, \theta^{(t)}, x\right) = E_{c \mid \omega^{(t)}, \theta^{(t)}, x}\{\log p(x, c \mid \omega, \theta)\}, \\ &\textbf{M step: Set} \quad \left(\hat{\omega}^{(t+1)}, \hat{\theta}^{(t+1)}\right) = \arg\max_{\omega,\theta} Q\left(\omega, \theta \mid \hat{\omega}^{(t)}, \hat{\theta}^{(t)}, y\right). \end{aligned}$$

These two steps are repeated until a stopping criteria is satisfied (usually, the relative change in the expected value of the full-data log-likelihood Q is below a given threshold ϵ, e.g. $\epsilon = 10^{-5}$).

Let's investigate in more detail the algorithm. Note that conditionally on ω, θ and x the different c_is are independent from each other and

$$p(c_i = k \mid \omega, \theta, x) = \frac{\omega_k g_k(x_i \mid \theta_k)}{\sum_{l=1}^{K} \omega_l g_l(x_i \mid \theta_l)} = \nu_{i,k}(\omega, \theta).$$

The value of $\nu_{i,k}(\omega, \theta)$ can be interpreted as the probability that observation i was generated from component k if we assume that the true parameters of the mixture model are ω and θ. Also, remember that

$$p(x, c \mid \theta, \omega) = \prod_{i=1}^{n} \prod_{k=1}^{K} [\omega_k g_k(x_i \mid \theta_k)]^{1(c_i = k)},$$

which implies that

$$\log p(x, c \mid \theta, \omega) = \sum_{i=1}^{n} \sum_{k=1}^{K} 1(c_i = k) \left[\log \omega_k + \log g_k(x_i \mid \theta_k)\right].$$

Hence,

$$Q\left(\omega, \theta \mid \hat{\omega}^{(t)}, \hat{\theta}^{(t)}, x\right) = \sum_{i=1}^{n} \sum_{k=1}^{K} E_{c \mid \hat{\omega}^{(t)}, \hat{\theta}^{(t)}, x} \left\{ 1(c_i = k) \underbrace{\left[\log \omega_k + \log g_k(x_i \mid \theta_k)\right]}_{\text{constant with respect to } c} \right\}$$

and therefore:

$$Q\left(\omega, \theta \mid \hat{\omega}^{(t)}, \hat{\theta}^{(t)}, x\right) = \sum_{i=1}^{n} \sum_{k=1}^{K} \nu_{i,k}^{(t+1)} \left(\hat{\omega}^{(t)}, \hat{\theta}^{(t)}\right) \left[\log \omega_k + \log g_k(x_i \mid \theta_k)\right].$$

(Remember that the term that is constant with respect to c can come out of the expectation, and that the expected value of an indicator function is just the probability of the event inside the indicator). Hence, roughly speaking, we can see that the Q function is in this case equivalent to a weighted average of the log likelihoods associated with each of the components in the mixture.

2.2 THE EM ALGORITHM FOR A LOCATION MIXTURE OF TWO GAUSSIAN DISTRIBUTIONS

A specific example is useful to fully understanding the algorithm. In particular, consider estimating the parameters of a location mixture of two univariate normal distributions with means μ_1 and μ_2, and common variance σ^2. The density of this mixture takes the form:

$$f(x \mid \omega, \mu_1, \mu_2, \sigma) = \omega \frac{1}{\sqrt{2\pi}\sigma} \exp\left\{-\frac{(x-\mu_1)^2}{2\sigma^2}\right\} + (1-\omega) \frac{1}{\sqrt{2\pi}\sigma} \exp\left\{-\frac{(x-\mu_2)^2}{2\sigma^2}\right\}$$

The expected weights are

$$\nu_{i,1}^{(t+1)} = \nu_{i,1}^{(t+1)} \left(\hat{\omega}^{(t)}, \hat{\mu}_1^{(t)}, \hat{\mu}_2^{(t)}, \hat{\sigma}^{(t)}\right) = \frac{\hat{\omega}^{(t)} \exp\left\{-\frac{1}{2}\left(\frac{x-\hat{\mu}_1^{(t)}}{\hat{\sigma}^{(t)}}\right)^2\right\}}{\hat{\omega}^{(t)} \exp\left\{-\frac{1}{2}\left(\frac{x-\hat{\mu}_1^{(t)}}{\hat{\sigma}^{(t)}}\right)^2\right\} + \left(1-\hat{\omega}^{(t)}\right) \exp\left\{-\frac{1}{2}\left(\frac{x-\hat{\mu}_2^{(t)}}{\hat{\sigma}^{(t)}}\right)^2\right\}}$$

and $\nu_{i,2}^{(t+1)} = 1 - \nu_{i,1}^{(t+1)}$. Therefore, the Q function is:

$$
\begin{aligned}
&Q\left(\omega, \mu_1, \mu_2, \sigma \mid \hat{w}^{(t)}, \hat{\mu}_1^{(t)}, \hat{\mu}_2^{(t)}, \hat{\sigma}^{(t)}, x\right) = \\
&\sum_{i=1}^{n} \left\{ \nu_{i,1}^{(t+1)} \left[\log \omega - \frac{1}{2}\log 2\pi - \log \sigma - \frac{(x-\mu_1)^2}{2\sigma^2} \right] \right. \\
&\qquad \left. + \nu_{i,2}^{(t+1)} \left[\log(1-\omega) - \frac{1}{2}\log 2\pi - \log \sigma - \frac{(x-\mu_2)^2}{2\sigma^2} \right] \right\}
\end{aligned}
$$

To maximize Q we compute its partial derivatives:

$$\frac{\partial Q}{\partial \omega} = \left\{ \sum_{i=1}^{n} \nu_{i,1}^{(t+1)} \right\} \frac{1}{\omega} - \left\{ \sum_{i=1}^{n} \nu_{i,2}^{(t+1)} \right\} \frac{1}{1-\omega} \tag{4}$$

$$\frac{\partial Q}{\partial \mu_k} = -\sum_{i=1}^{n} \nu_{i,k}^{(t+1)} \frac{1}{\sigma^2}(x_i - \mu_k) \tag{5}$$

$$\frac{\partial Q}{\partial \sigma} = \sum_{i=1}^{n}\sum_{k=1}^{2} \nu_{i,k}^{(t+1)} \left\{ -\frac{1}{\sigma} + \frac{1}{\sigma^3}(x_i - \mu_k)^2 \right\} \tag{6}$$

By setting (4) equal to zero we get

$$
\begin{aligned}
& \left\{ \sum_{i=1}^{n} \nu_{i,2}^{(t+1)} \right\} \omega^{(t+1)} = \left\{ \sum_{i=1}^{n} \nu_{i,1}^{(t+1)} \right\} \left(1 - \omega^{(t+1)}\right) \\
\Rightarrow \quad & \left\{ \sum_{i=1}^{n} \nu_{i,1}^{(t+1)} \right\} = \left\{ \sum_{i=1}^{n} \nu_{i,1}^{(t+1)} + \sum_{i=1}^{n} \nu_{i,2}^{(t+1)} \right\} \omega^{(t+1)} \\
\Rightarrow \quad & \omega^{(t+1)} = \frac{\sum_{i=1}^{n} \nu_{i,1}^{(t+1)}}{\sum_{i=1}^{n} \nu_{i,1}^{(t+1)} + \nu_{i,2}^{(t+1)}} = \frac{1}{n}\sum_{i=1}^{n} \nu_{i,1}^{(t+1)}.
\end{aligned}
$$

(Note that, by definition, $\nu_{i,1}^{(t)} + \nu_{i,2}^{(t)} = 1$, and therefore the $\sum_{i=1}^{n} \nu_{i,1}^{(t)} + \nu_{i,2}^{(t)} = n$.) Hence, the partial estimator of ω is just the average of the probabilities that observations belong to component 1. When the components are well separated, the values of $\nu_{i,k}$ will all be close to either 0 or 1, and therefore the partial estimate of ω will be approximately equal to the fraction of observations that come from component 1.

By setting (5) to zero we get

$$
\begin{aligned}
& 0 = \sum_{i=1}^{n} \nu_{i,k}^{(t+1)} \left(x_i - \mu_k^{(t+1)}\right) \\
\Rightarrow \quad & \sum_{i=1}^{n} \nu_{i,k}^{(t+1)} x_i = \sum_{i=1}^{n} \nu_{i,k}^{(t+1)} \mu_k^{(t+1)} \\
\Rightarrow \quad & \mu_k^{(t+1)} = \frac{\sum_{i=1}^{n} \nu_{i,k}^{(t+1)} x_i}{\sum_{i=1}^{n} \nu_{i,k}^{(t+1)}}.
\end{aligned}
$$

Note that the partial estimator for μ_k is a weighted average of the x_is, with the weight associated with observation i being proportional to the probability that such observation was generated by component k. Again, if the components are well separated and values of $\nu_{i,k}$ are all close to either 0 or

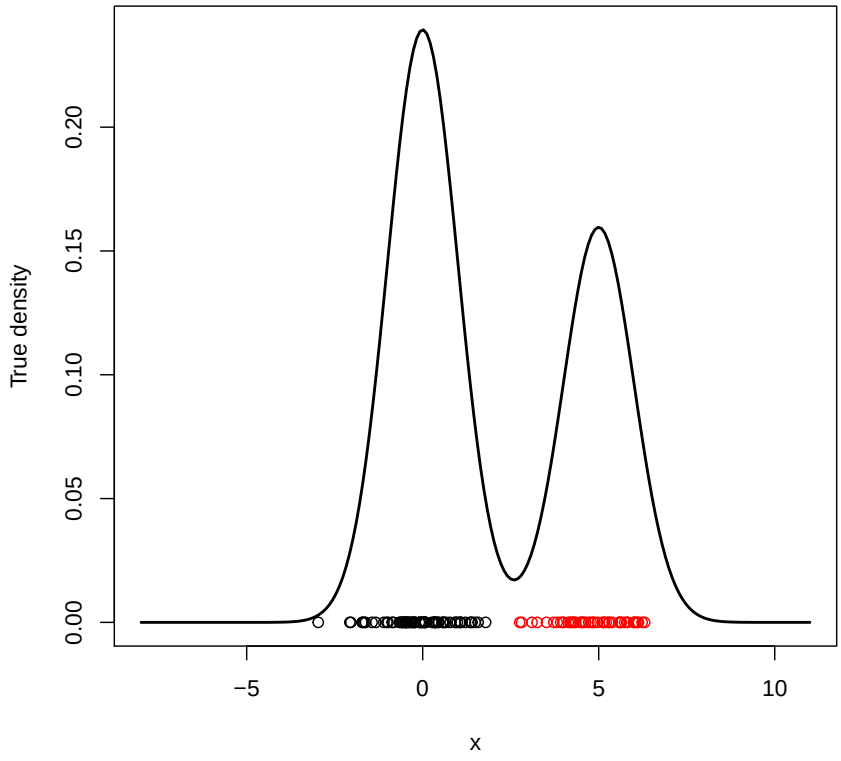

Figure 9: True density and observations used to illustrate the EM algorithm for fitting a location mixture of two univariate Gaussian distributions.

1, this weighted average is roughly the average of the observations coming from component k.

Finally, making (6) equal to zero,

$$\sum_{i=1}^{n}\sum_{k=1}^{2} v_{i,k}^{(t+1)} = \left(\frac{1}{\sigma^{(t+1)}}\right)^2 \sum_{i=1}^{n}\sum_{k=1}^{2} v_{i,k}^{(t+1)} \left(x_i - \mu_k^{(t+1)}\right)^2$$

$$\Rightarrow \qquad \sigma^{(t+1)} = \sqrt{\frac{\sum_{i=1}^{n}\sum_{k=1}^{2} v_{i,k}^{(t+1)} \left(x_i - \mu_k^{(t+1)}\right)^2}{\sum_{i=1}^{n}\sum_{k=1}^{2} v_{i,k}^{(t+1)}}}.$$

The code contained in the file `EM_univariate_normal.R` implements this algorithm and uses it to fit the model to a simulated data set generated from a true model with $\omega_{\text{True}} = 0.6$, $\mu_{1,\text{True}} = 0$, $\mu_{2,\text{True}} = 4$ and $\sigma_{\text{True}} = 1$, i.e.,

$$f_{\text{True}}(x) = 0.6\frac{1}{\sqrt{2\pi}}\exp\left\{-\frac{x^2}{2}\right\} + 0.4\frac{1}{\sqrt{2\pi}}\exp\left\{-\frac{(x-4)^2}{2}\right\}.$$

(See Figure 9.) Note that the code fixes the seed of the random number generator in order to make the results reproducible. We adopt this practice for all of our examples in this course, but you will want to remove this line of code when writing the algorithms for other data sets.

For $\epsilon = 10^{-5}$ the algorithm converges for this data set in 11 iterations, yielding the following maximum likelihood estimates: $\hat{\omega} = 0.6106$, $\hat{\mu}_1 =$

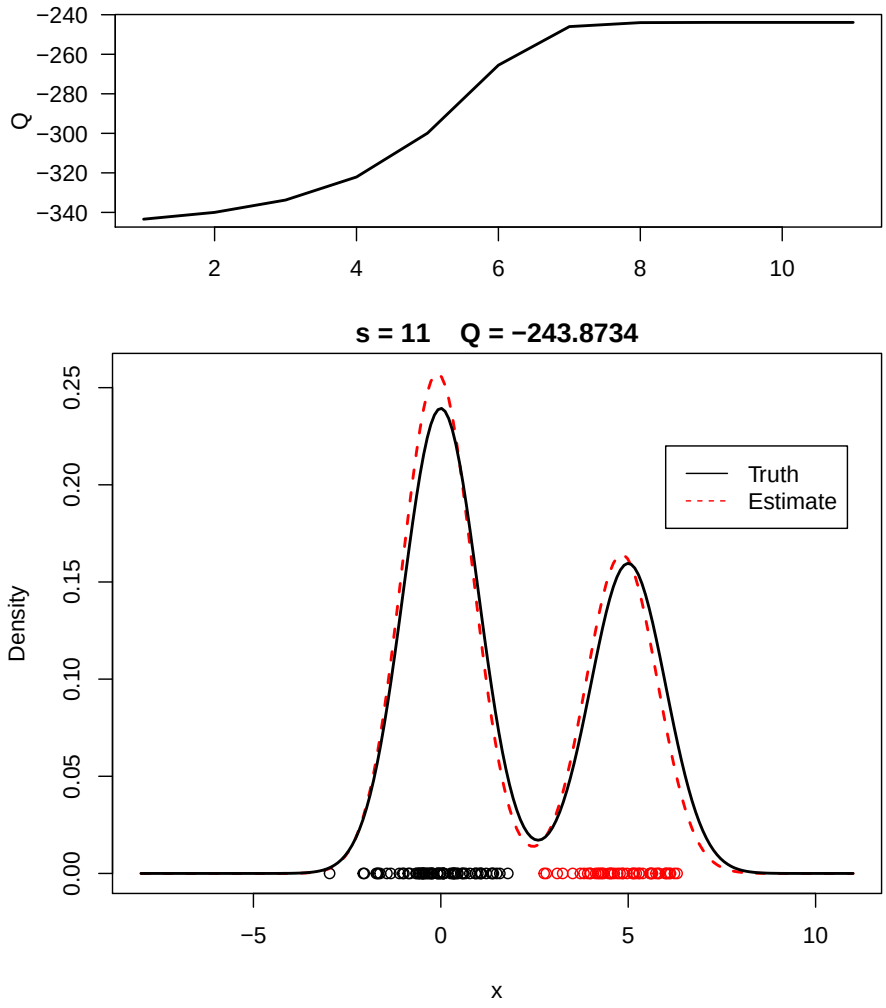

Figure 10: Value of the expected loglikelihood Q as a function of the iteration number (top panel) and a comparison of the maximum likelihood estimate $\hat{f}$ at convergence against the true density f_{True} (bottom panel) for a simulated dataset of 120 observations. The dots at the bottom of the plot correspond to the data, with the colors of the dots representing the true components that generated each of the observations.

-0.0949, $\hat{\mu}_2 = 4.8287$ and $\hat{\sigma} = 0.9463$. The evolution of the objective function Q used to monitor convergence together with a graph of the estimated density

$$\hat{f}(x) = \hat{\omega}\frac{1}{\sqrt{2\pi}\hat{\sigma}}\exp\left\{-\frac{(x-\hat{\mu}_1)^2}{2\hat{\sigma}^2}\right\} + (1-\hat{\omega})\frac{1}{\sqrt{2\pi}\hat{\sigma}}\exp\left\{-\frac{(x-\hat{\mu}_2)^2}{2\hat{\sigma}^2}\right\},$$

the true distribution f_{True} and the data can be seen in Figure 10. Note that the objective Q is strictly increasing. This should not be a surprise, as the EM algorithm is guarranteed to monotonically converge to a *local maximum*. Also, note that in this case the estimate of the density is very close to the true density of the data in spite of the relatively small sample size.

2.3 GENERAL LOCATION AND SCALE MIXTURES OF p-VARIATE GAUSSIAN DISTRIBUTIONS

The expressions used for the updates of the EM algorithm for a location and scale mixture of K q-variate Gaussian distributions

$$f(x) = \sum_{k=1}^{K} \omega_k \left(\frac{1}{2\pi}\right)^{q/2} |\Sigma_k|^{-1/2} \exp\left\{(x-\theta_k)^T \Sigma_k^{-1} (x-\theta_k)\right\},$$

can be obtained in a very similar way:

$$\begin{aligned}
\nu_{i,k}^{(t+1)} &= \frac{\omega_k^{(t)}|\Sigma_k^{(t)}|^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}(x-\mu_k^{(t)})^T \left[\Sigma_k^{(t)}\right]^{-1} (x-\mu_k^{(t)})\right\}}{\sum_{l=1}^{K} \omega_l^{(t)}|\Sigma_l^{(t)}|^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}(x-\mu_l^{(t)})^t \left[\Sigma_l^{(t)}\right]^{-1} (x-\mu_l^{(t)})\right\}} \\
\omega_k^{(t+1)} &= \frac{\sum_{i=1}^{n} \nu_{i,k}^{(t+1)}}{\sum_{l=1}^{K}\sum_{i=1}^{n} \nu_{i,l}^{(t+1)}} \\
\mu_k^{(t+1)} &= \frac{1}{\sum_{i=1}^{n} \nu_{i,k}^{(t+1)}} \sum_{i=1}^{n} \nu_{i,k}^{(t+1)} x_i \\
\Sigma_k^{(t+1)} &= \frac{1}{\sum_{i=1}^{n} \nu_{i,k}^{(t+1)}} \sum_{i=1}^{n} \nu_{i,k}^{(t+1)} \left(x_i - \mu_k^{(t+1)}\right)\left(x_i - \mu_k^{(t+1)}\right)^T
\end{aligned}$$

The R code in the file `EM_multivariate_normal.R` provides an implementation of this algorithm. Again, the implementation is illustrated using a simulated data set, but in this case $q = 2$ and $K = 3$. Figure 11 shows the contour plots associated with the true components of the mixture, as well as the random sample used in the illustration. Figure 12 shows the evolution of the objective function Q, as well as contour plots of the components estimated by the EM algorithm. For $\epsilon = 10^{-5}$ the EM algorithm converges after 12 iterations. Note that the algorithm seems to misclassify observation 54, which leads to underestimating the variance of the top left component of the mixture.

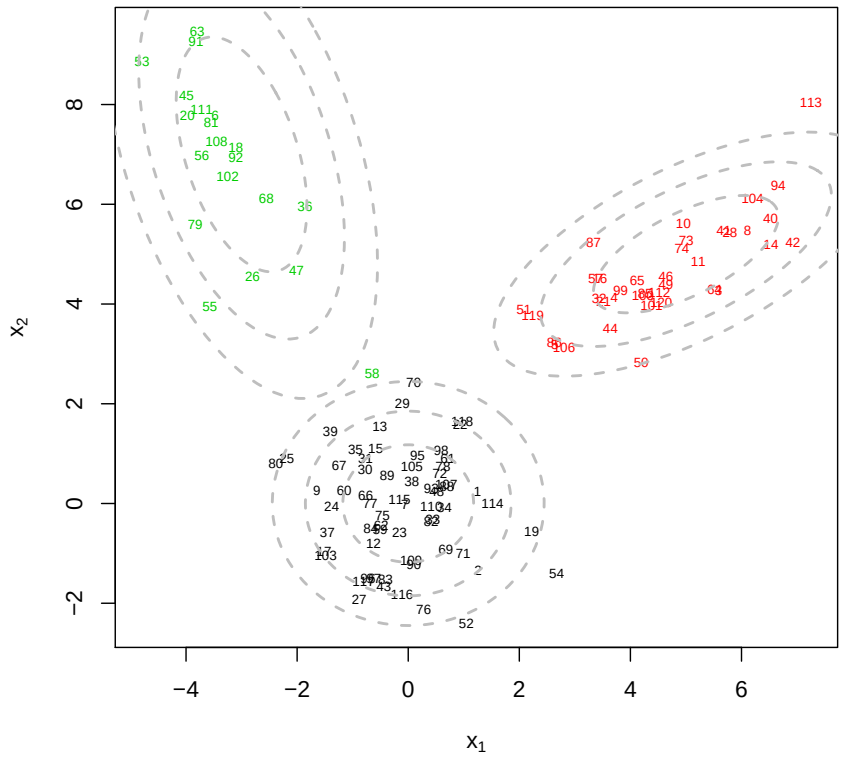

Figure 11: True density and observations used to illustrate the EM algorithm for fitting a location and scale mixture of three bivariate Gaussian distributions.

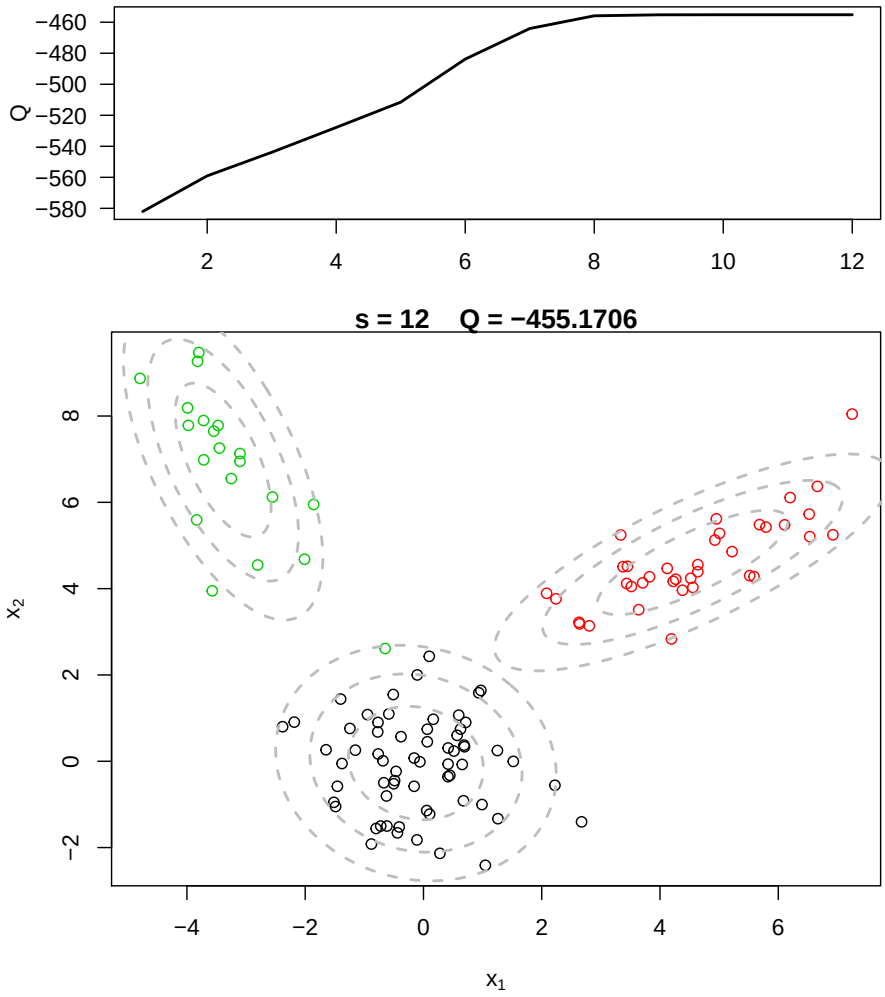

Figure 12: Value of the expected loglikelihood Q as a function of the iteration number (top panel) and a contour plots of the maximum likelihood estimate $\hat{f}$ (bottom panel) at convergence for a simulated dataset of 120 observations. The dots correspond to the data, with the colors of the dots representing the true components that generated each of the observations.

3 BAYESIAN INFERENCE FOR FINITE MIXTURE MODELS

We discuss now Bayesian inference for the mixture model with density/probability mass function

$$f(x) = \sum_{k}^{K} \omega_k g_k(x \mid \theta_k).$$

Bayesian inference requires that we elicit prior distributions for the vector of weights $(\omega_1, \ldots, \omega_K)$ and each of the component-specific parameters $\theta_1, \ldots, \theta_k$. We will adhere to standard practice and use a Dirichlet prior for $\omega = (\omega_1, \ldots, \omega_K)$,

$$p(\omega) = \frac{\Gamma\left(\sum_{k=1}^{K} a_k\right)}{\prod_{k=1}^{K} \Gamma(a_k)} \prod_{k=1}^{K} \omega_k^{a_k - 1}, \qquad \sum_{k=1}^{K} \omega_k = 1.$$

The symmetric version $a_1 = \cdots = a_K = a$ is popular, particularly the case $a = 1$ (which corresponds to a uniform prior on the simplex). In the case of the θ_ks, the choice of priors depends on the form of $g_1, \ldots, g_K$, but often implies that assumption of independence across k. Furthermore, when a conjugate or conditionally conjugate prior exists for g_k, the prior for θ_k is usually chosen to be in that conjugate family of priors.

It is very important not to use improper priors in the context of mixture models, as they might easily lead to improper posteriors. This is specially true when we are interested in selecting the number of components K based on the data (see Chapter 5). The warning extends to proper but flat priors that, in the limit, become improper. In general priors should be chosen so that the a priori support of f roughly matches the support of the observed data.

3.1 MARKOV CHAIN MONTE CARLO ALGORITHMS FOR MIXTURE MODELS

Markov chain Monte Carlo (MCMC) algorithm are typically used to perform Bayesian inference in complex models. MCMC algorithms repeatedly sample from the full conditional distributions of each (block of) parameters given fixed values for the rest. After an appropriate burn-in period, they generate samples that are dependent but identically distributed according to the posterior distribution of interest.

To develop a MCMC algorithm for mixture models we will use the hierarchical representation of the likelihood,

$$p(x, c \mid \theta, \omega) = p(x, \mid c, \theta) p(c \mid \omega),$$

where

$$p(x, \mid c, \theta) = \prod_{i=1}^{n} g_{c_i}(x_i \mid \theta_{c_i}),$$

$$p(c \mid \omega) = \prod_{i=1}^{n} \prod_{k=1}^{K} \omega_k^{1(c_i = k)} = \prod_{k=1}^{K} \omega_k^{\sum_{i=1}^{n} 1(c_i = k)}.$$

When combined with the priors this leads to a joint posterior distribution of the form:

$$p(c, \theta, \omega \mid x) \propto \left\{ \prod_{i=1}^{n} g_{c_i}(x_i \mid \theta_{c_i}) \right\} \left\{ \prod_{i=1}^{n} \prod_{k=1}^{K} \omega_k^{1(c_i=k)} \right\} p(\omega) p(\theta).$$

Each of the full conditional distributions can be derived from this joint posterior by retaining the terms that involve the parameter of interest, and recognizing the product of the selected terms as the kernel of a known family of distributions. For example, in the case of the ω, the only two relevant terms are $p(c \mid \omega)$ and $p(\omega)$. Furthermore, since $p(\omega) \propto \prod_{k=1}^{K} \omega_k^{a_k}$, the full conditional distribution of the weights is given by

$$p(\omega \mid c, \theta, x) \propto p(c \mid \omega) p(\omega) = \prod_{k=1}^{K} \omega_k^{a_k + \sum_{i=1}^{n} 1(c_i=k) - 1}.$$

This clearly corresponds to the kernel of another Dirichlet distribution with updated parameters

$$a_k^* = a_k + m_k, \qquad k = 1, \ldots, K,$$

where $m_k = \sum_{i=1}^{n} 1(c_i = k)$ is the number of observation coming from component k. For small values of a_k the expected value of the weights under this full conditional distribution roughly correspond to the proportion of observations in the sample that have been assigned to each component.

Similarly, for the full conditional of c_i we have

$$p(c_i \mid c_1, \ldots, c_{i-1}, c_{i+1}, c_n, \theta, \omega, x) \propto p(x_i \mid c_i, \theta) p(c_i \mid \omega).$$

Hence,

$$p(c_i = k \mid c_1, \ldots, c_{i-1}, c_{i+1}, c_n, \theta, \omega, x) = \frac{\omega_k g_k(x_i \mid \theta_k)}{\sum_{l=1}^{K} \omega_l g_l(x_i \mid \theta_l)}.$$

(Note the similarity with the formula for the expected weights $v_{i,k}$ in the EM algorithm.) Finally, the posterior for each θ_k reduces to

$$p(\theta_k \mid c, \theta_1, \ldots, \theta_{k-1}, \theta_{k+1}, \ldots, \theta_K, \omega, x) \propto p(\theta_k \mid \theta_1, \ldots, \theta_{k-1}, \theta_{k+1}, \ldots, \theta_K) \prod_{i:c_i=k} p(x_i \mid \theta_k)$$

In the most common case in which the priors for θ_k are independent this is simply:

$$p(\theta_k \mid c, \theta_1, \ldots, \theta_{k-1}, \theta_{k+1}, \ldots, \theta_K, \omega, x) \propto p(\theta_k) \prod_{i:c_i=k} p(x_i \mid \theta_k)$$

3.2 THE MCMC ALGORITHM FOR A LOCATION MIXTURE OF TWO GAUSSIAN DISTRIBUTIONS

As in Chapter 2, we proceed to derive the full conditionals for a mixture of two univariate normal distributions:

$$f(x \mid \omega, \mu_1, \mu_2, \sigma) = \omega \frac{1}{\sqrt{2\pi}\sigma} \exp\left\{ -\frac{(x-\mu_1)^2}{2\sigma^2} \right\} + (1-\omega) \frac{1}{\sqrt{2\pi}\sigma} \exp\left\{ -\frac{(x-\mu_2)^2}{2\sigma^2} \right\}.$$

We use a beta distribution with parameters $a_1 = 1$ and $a_2 = 1$ for ω (which corresponds to a uniform distribution, and is just a special case of the Dirichlet distribution when $K = 2$), independent Gaussian priors for each μ_k so that $\mu_k \sim N(\eta, \tau^2)$, i.e.,

$$p(\mu_k) = \frac{1}{\sqrt{2\pi}\tau} \exp\left\{-\frac{(\mu_k - \eta)^2}{2\tau^2}\right\},$$

and an inverse Gamma prior with shape parameter a and scale parameter b for σ^2, i.e.,

$$p(\sigma^2) = \frac{1}{b^a \Gamma(a)} \left(\frac{1}{\sigma^2}\right)^{d+1} \exp\left\{-\frac{q}{\sigma^2}\right\}.$$

In practice, in the absence of real prior information we typically employ the observed data to guide the selection of the hyperparameter η, τ^2, d and q, in an approach that is reminiscent of empirical Bayes. In particular, we attempt to make the means of the different component lie in the same support of the observed data, so we take η to be approximately equal the mean (or median) of the observations, and τ^2 to be roughly equal to their variance. Similarly, for the prior on the variance σ^2 we set $d = 2$ (which implies that $E(\sigma^2) = q$ and an infinite prior variance) and q to be roughly equal to the variance of the observations. Posteriors are often not very sensitive to changes on the prior means that remain within an order of magnitude of the values suggested above.

Under the priors just discussed we have:

$$\begin{aligned}
&p(\mu_k \mid c, \mu_1, \ldots, \mu_{k-1}, \mu_{k+1}, \ldots, \mu_K, \omega, x) \\
&\quad \propto \exp\left\{-\frac{(\mu_k - \eta)^2}{2\tau^2}\right\} \prod_{i:c_i=k} \exp\left\{-\frac{(x_i - \mu_k)^2}{2\sigma^2}\right\} \\
&\quad \propto \exp\left\{-\frac{1}{2}\left[m_k \frac{\mu_k^2}{\sigma^2} - 2\frac{\mu_k \sum_{i:c_i=k} x_i}{\sigma^2} + \frac{\mu_k}{\tau^2} - 2\frac{\mu_k \eta}{\tau^2}\right]\right\} \\
&\quad \propto \exp\left\{-\frac{1}{2}\left[\frac{m_k}{\sigma^2} + \frac{1}{\tau^2}\right]\left[\mu_k - \frac{\frac{1}{\sigma^2}\sum_{i:c_i=k} x_i + \frac{\eta}{\tau^2}}{\frac{m_k}{\sigma^2} + \frac{1}{\tau^2}}\right]\right\},
\end{aligned}$$

which is just the kernel of a normal distribution with updated mean

$$\eta_k^* = \frac{\frac{1}{\sigma^2}\sum_{i:c=k} x_i + \frac{\eta}{\tau^2}}{\frac{m_k}{\sigma^2} + \frac{1}{\tau^2}}$$

and updated standard deviation

$$\tau_k^* = \left[\frac{m_k}{\sigma^2} + \frac{1}{\tau^2}\right]^{-1/2}.$$

Finally,

$$\begin{aligned}
p(\sigma^2 \mid c, \mu, \omega, x) &\propto \left(\frac{1}{\sigma^2}\right)^{d+1} \exp\left\{-\frac{q}{\sigma^2}\right\} \\
&\qquad \left(\frac{1}{\sigma^2}\right)^{n/2} \exp\left\{-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(x_i - \mu_{c_i})^2\right\} \\
&= \left(\frac{1}{\sigma^2}\right)^{n/2+d+1} \exp\left\{-\frac{1}{\sigma^2}\left[\frac{\sum_{i=1}^{n}(x_i - \mu_{c_i})^2}{2} + q\right]\right\},
\end{aligned}$$

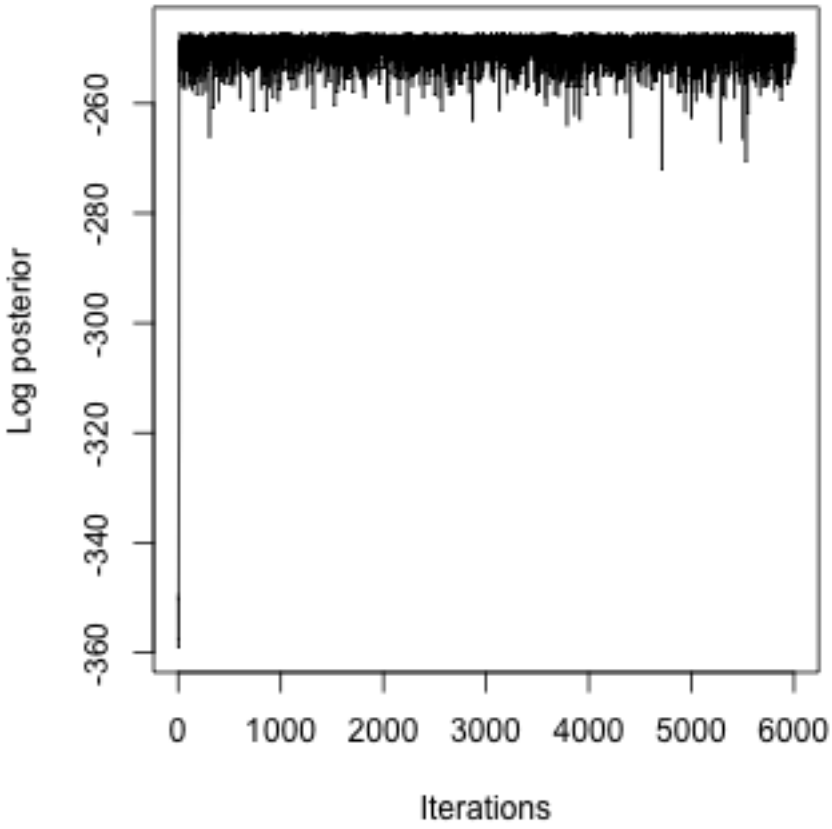

Figure 13: Trace plot of the unnormalized posterior distribution for a mixture of two univariate Gaussian distributions fitted to our simulated dataset.

which is the kernel of another inverse Gamma distribution with shape $d^* = n/2 + d$ and rate parameter

$$q^* = \frac{1}{2}\sum_{i=1}^{n}(x_i - \mu_{c_i})^2 + q.$$

The code contained in the file `MCMC_univariate_normal.R` implements this algorithm and tests it in the same data set we used in Section 2.2. Figure 13 presents the trace of the (unnormalized) posterior distribution for the 6,000 samples generated by the algorithm. The graph suggests that the MCMC algorithm converges quickly and that the mixing of is reasonable. Hence, we discard the first 1,000 iterations as burn-in and use the remaining 5,000 to carry out our posterior inference.

The posterior mean and 95% symmetric credible intervals for the parameters in the model are presented in Table 1. Note that the posterior means are very close to the maximum likelihood estimators we reported in Section 2.2.

Figure 14 shows the posterior mean density along with pointwise 95% credible intervals for that density. Note that the estimate looks very similar to the true density that generated the data, and that there is little uncertainty about the multimodality of the distribution.

Table 1: Posterior estimates for the parameters of the two-component mixture model for our simulated data set.

Parameter	Posterior mean	95% credible interval
ω	0.6104	(0.5217, 0.6971)
μ_1	-0.0942	(-0.3164, 0.1325)
μ_2	4.8275	(4.5483, 5.1081)
σ	0.9563	(0.8417, 1.0941)

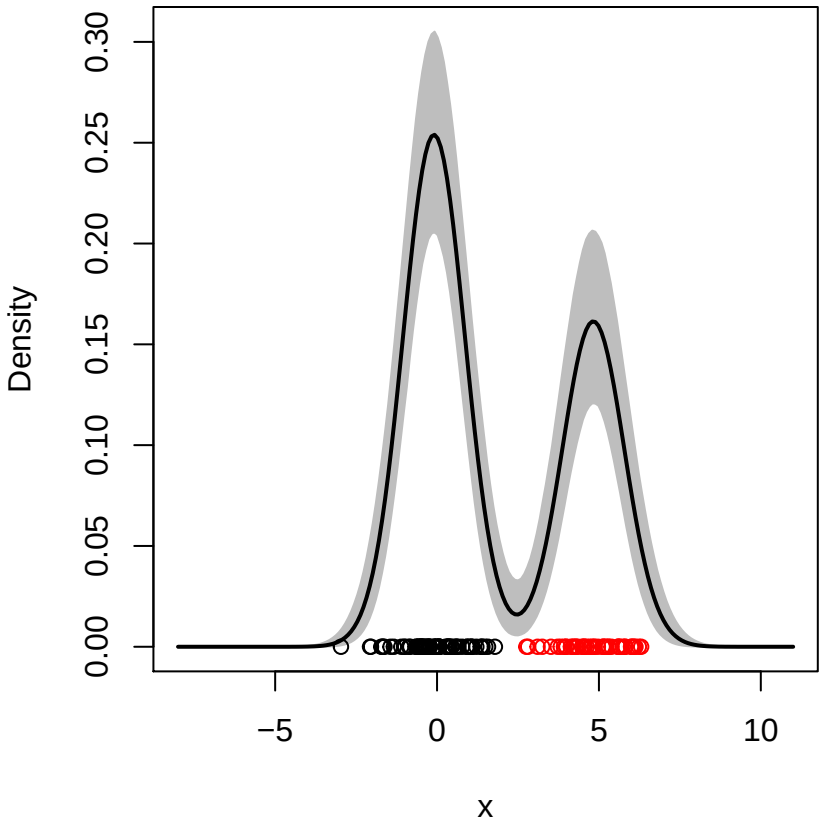

Figure 14: Posterior mean and pointwise 95% credible intervals for our Bayesian density estimate, along with 50 randomly generated samples from it.

3.3 GENERAL LOCATION AND SCALE MIXTURES OF p-VARIATE GAUSSIAN DISTRIBUTIONS

The previous MCMC algorithm can be extended to a location and scale mixture of K multivariate normal distributions, each with its own mean and variance-covariance matrix:

$$\sum_{k=1}^{K} \omega_k \left(\frac{1}{2\pi}\right)^{p/2} |\Sigma_k|^{-1/2} \exp\left\{(x-\mu_k)^T \Sigma_k^{-1} (x-\mu_k)\right\}$$

as long as we assume a multivariate Gaussian prior on the mean with mean b and variance B and an inverse Wishart prior for the covariance matrices with parameters ν and S. Since these priors are conditionally conjugate, the full-conditional posterior distribution for μ_k and Σ_k turn out to also be Gaussian and inverse Wishart, respectively. In the case of μ_k, the updated parameters are

$$b_k^* = \left(B^{-1} + m_k \Sigma_k^{-1}\right)^{-1} \left(B^{-1} b + \Sigma_k^{-1} \sum_{i:c_i=k} x_i\right)$$

and

$$B_k^* = \left(B^{-1} + m_k \Sigma_k^{-1}\right)^{-1},$$

while the updated parameters for Σ_k are

$$\nu_k^* = \nu + m_k$$

and

$$S_k^* = S + \sum_{i:c_i=k} (x_i - \mu_k)(x_i - \mu_k)^T.$$

The file `MCMC_multivariate_normal.R` contains an implementation of this code, which we illustrate with the same simulated data set used in Section 2.3 (see Figure 11). We run the algorithm for 1,000 iterations. Figure 15 shows a trace plot of the unnormalized posterior distribution. As in the univariate case, the algorithm seems to converge quite quickly and mix well. Figure 16 shows an estimate of the mixture density based on only the last iteration of the MCMC algorithm, which confirms that the posterior distribution seems to have converged.

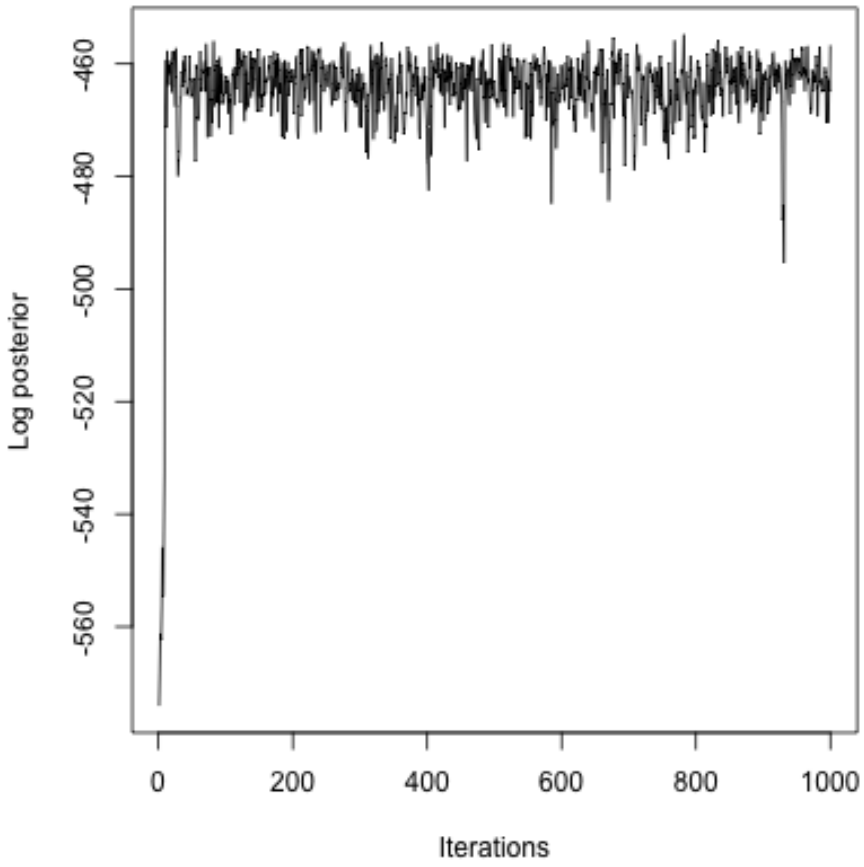

Figure 15: Trace plot of the unnormalized posterior distribution for a mixture of three bivariate Gaussian distributions fitted to our simulated dataset.

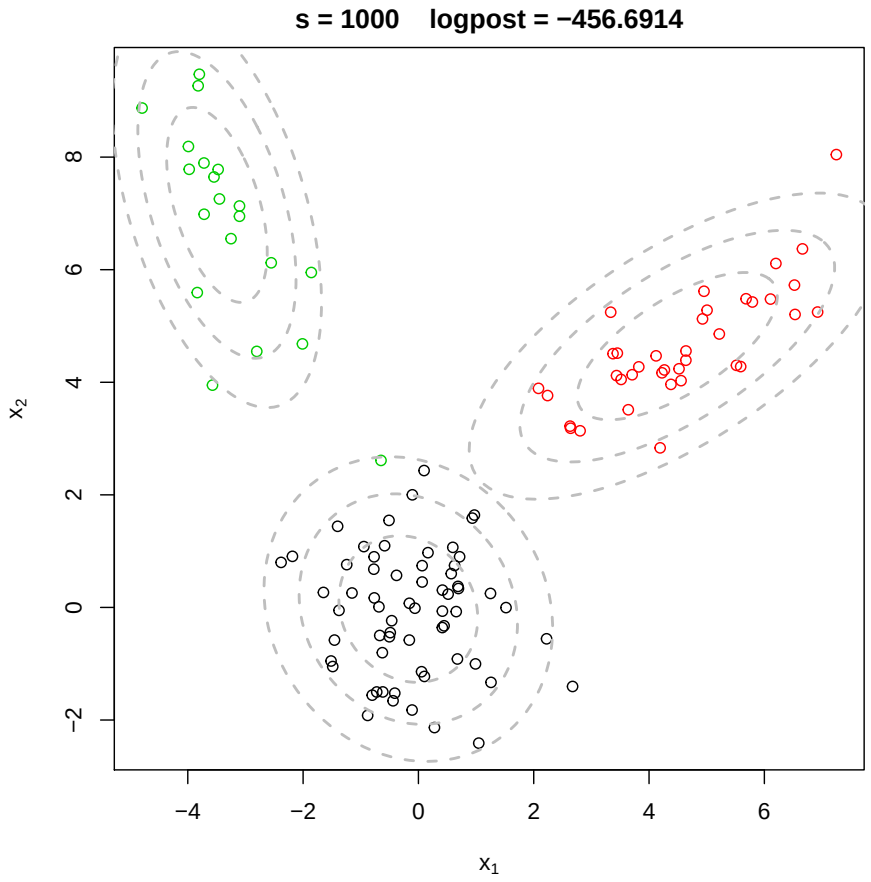

Figure 16: Estimate of the mixture density for the three-component mixture of bivaraite densities based on the last iteration of the MCMC algorithm.

4 APPLICATIONS OF MIXTURE MODELS

4.1 DENSITY ESTIMATION

The goal in density estimation tasks is to generate flexible (non-parametric) estimates of the probability density function that generated an independent and identically distributed sample $x_1, \ldots, x_n$. We are already familiar with this application, as we have implicitly used it to motivate mixture models.

The most widely used approach to density estimation is *kernel density estimation*, which relies on estimators of the form

$$\tilde{f}(x) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{h} g\left(\frac{1}{h} \|x - x_i\|\right), \tag{7}$$

where g is a density (often refereed to as the *kernel*). The scale parameter h is often called the *bandwidth*, and is usually estimated using generalized cross-validation. Kernel density estimators can be seen as smoothed versions of the empirical distribution function $\sum_{i=1}^{n} \frac{1}{n} \delta_{x_i}(x)$ (recall that $\delta_a(\cdot)$ represents Dirac's delta function at a). The smoothing arises from a convolution with the scaled kernel $\frac{1}{h} g\left(\frac{x}{h}\right)$.

Note that (7) looks like a mixture model with n components (as many as observations) and equal weights for all its components. This becomes even clearer if you consider the widely used Gaussian kernel density estimator:

$$\tilde{f}(x) = \sum_{i=1}^{n} \frac{1}{n} \frac{1}{\sqrt{2\pi}h} \exp\left\{\frac{1}{2}\left(\frac{x - x_i}{h}\right)^2\right\}. \tag{8}$$

Note that this is a location mixture of n normals with uniform weights $\omega_1 = \omega_2 = \cdots = \omega_n = 1/n$, component-specific means that are equal to the observations in the sample, and common standard deviation h for all components.

In order to understand the relationship between kernel density estimation and mixture models it is useful to contrast (8) with the density estimate

$$\hat{f}(x) = \sum_{K=1}^{K} \hat{\omega}_k \frac{1}{\sqrt{2\pi}\hat{\sigma}} \exp\left\{\frac{1}{2\hat{\sigma}^2}(x - \hat{\mu}_k)^2\right\} \tag{9}$$

obtained by plugging-in the maximum likelihood estimates of the parameters, $\hat{\omega}_1, \ldots, \hat{\omega}_K$, $\hat{\mu}_1, \ldots, \hat{\mu}_K$ and $\hat{\sigma}$ of a location mixture of K univariate Gaussian distributions.

Recall that the maximum likelihood estimator for μ_k and ω_k under the mixture model correspond, roughly speaking, to the mean and the fraction of observations in the group, respectively. In the limiting case $K = n$, these become the value of the single observation in the group, and $1/n$. These are exactly the values used by the kernel density estimator. Note, however, that when $K = n$ the maximum likelihood estimator for σ does not exist. Indeed, σ must be strictly positive and a direct application of the formula we derived in Chapter 2 yields 0 as the optimal value for $\hat{\sigma}$. Another way to interpret this observation is to realize that, when $K = n$, the estimate $\hat{f}$ reduces to the empirical cumulative distribution function. In fact, this highlights that in

estimating location mixtures there is a clear trade-off between the number of components and the variance of those components: the more components are included, the lower the variance will tend to be. We will explore this trade-off in Chapter 5.

The connection we just highlighted allows us to use mixture models to generalize kernel density estimators in a number of ways. For example by considering location and scale mixtures of two-parameter kernels such as the Gaussian (i.e., by allowing both the mean and the variance of the components to be different) we obtain adaptive-bandwidth kernel density estimates that allow the degree of smoothness induced by the convolution to be different in various regions of the support. Similarly, if a Bayesian procedure is used instead to estimate the mixture, the predictive distribution $p(x \mid x_1, \ldots, x_n)$ obtained from the fitting procedure can be interpreted as a Bayesian version of kernel density estimation.

To illustrate the use of mixture models for density estimation we use the well-known `galaxies` data set, which contains the velocities (in km/sec) of 82 galaxies from 6 well-separated conic sections of an unfilled survey of the Corona Borealis region. Multimodality in such surveys is evidence of voids and superclusters in the far universe. We fit location mixtures of $K = 6$ Gaussian distributions corresponding to (9) using both an EM and a MCMC algorithm, and compare the results against the Gaussian kernel density estimate corresponding to (8), which is obtained using the `R` function `density`. The code is contained in `galaxies_density_estimation.R`.

Figure 17 contains all three density estimates as well pointwise credible bands associated with the Bayesian posterior distribution. Note that all three estimates agree quite closely in terms of the shape of the tails of the distribution. However, they somewhat disagree in terms of the middle section of the density. In particular, while the estimators based on mixture models tend to clearly identify two modes in this region, the kernel density estimate tends to smooth out the density, suggesting a more unimodal (albeit skewed) shape. Note, however, that the Bayesian estimate obtained using the MCMC algorithm seems to be overall closer to the kernel density estimate than the frequentist estimator obtained using the EM algorithm.

It should be clear from this example that label switching is not an issue when using mixture models for density estimation since, in this case, we are only interested in evaluating the density f, which is invariant to permutations in the labels.

4.2 CLUSTERING (UNSUPERVISED CLASSIFICATION)

The goal in clustering tasks (sometimes called unsupervised classification tasks in the machine learning literature) is to divide a heterogeneous sample into a set of homogeneous groups. This needs to be accomplished in the absence of any labeled data that provides prior information about the shape of the groups (hence, the term "unsupervised" classification). For example, measurements of the physical characteristics of a number of plants might be available and we are interested in dividing the sample into groups that correspond to various species.

There is a rich literature covering a variety of approaches to clustering. K-means clustering is particularly relevant for our discussion as it directly connects with mixture models. K-means clustering assumes that there exists labels $s_1, \ldots, s_n \in \{1, \ldots, K\}$ that divide the sample into K "spherical" clusters in Euclidean space, which are centered at locations $\gamma_1, \ldots, \gamma_K$. The

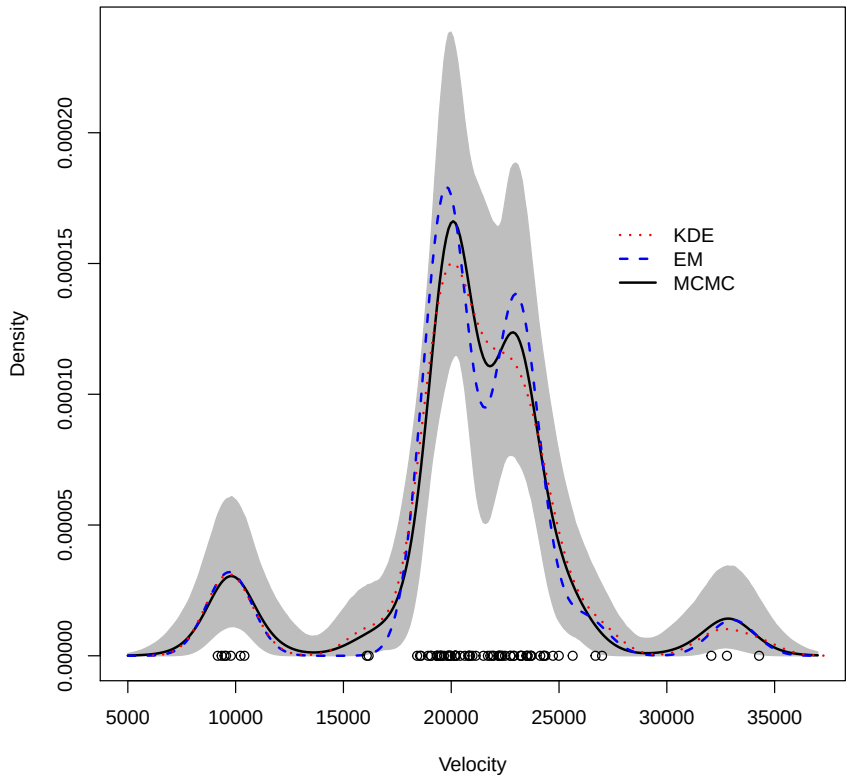

Figure 17: Comparison of a kernel density estimate with estimates from a mixture model with $K = 6$ components obtained using an EM and a MCMC algorithm for the `galaxies` data set. The grey area corresponds to 95% pointwise credible bands for the posterior distribution of the predictive distribution $p(x \mid x_1, \ldots, x_n)$, which provides a measure of uncertainty associated with the Bayesian point estimator.

clusters are estimated iteratively: given an initial guess for the centers, we proceed by first assigning observations to the cluster corresponding to the closest center γ_k (measured in Euclidean distance), and then recomputing the location of the centers as the average of the observations assigned to the cluster. These two steps are repeated until the location of the centers does not change from one iteration to the next.

Inferences for the auxiliary variables $c_1, \ldots, c_n$ that we introduced to derive the EM and MCMC algorithms for mixture models can be used to solve clustering problems, as they essentially play the same role as the indicators $s_1, \ldots, s_n$ that appear in the k-means clustering algorithm. Indeed, remember that c_i can be interpreted as the label of the component that generated observation x_i. For example, if the EM algorithm is used for inference, it is natural to set

$$\hat{c}_i = \arg\max_k \hat{v}_{i,k},$$

where the $\hat{v}_{i,k}$s correspond the value of weights at the last iteration of the algorithm. (Recall that $v_{i,k}$ is the probability that x_i comes from component k, so we are assigning x_i to the component with the largest probability.)

There is a close relationship between k-means clustering and Gaussian mixture models. To highlight this relationship, consider a location mixture of p-variate Gaussian distribution in which all components have the same variance-covariance matrix, $\sigma^2 I$, and where the weights are assumed uniform and known, i.e.,

$$f(x) = \sum_{k=1}^{K} \frac{1}{K} \left(\frac{1}{\sqrt{2\pi}\sigma} \right)^p \exp\left\{ -\frac{1}{2\sigma^2} (x - \mu_k)^T (x - \mu_k) \right\} \tag{10}$$

and compare the steps associated with the EM algorithm used to fit this model with the steps used to conduct k-means clustering (recall that $(x - \mu_k)^T (x - \mu_k) = \|x - \mu_k\|^2$):

K-means clustering	**EM algorithm**
Assignment step: Set	*E step:* Set
$c_i^{(t+1)} = \arg\min_k \|x_i - \gamma_k^{(t)}\|$	$v_{i,k}^{(t+1)} \propto \exp\left\{ -\frac{1}{2\sigma^2} \|x_i - \mu_k^{(t)}\|^2 \right\}$
Center update step: Set	*M step:* Set
$\gamma_k^{(t+1)} = \frac{\sum_{i=1}^n \mathbf{1}\left(c_i^{(t+1)} = k\right) x_i}{\sum_{i=1}^n \mathbf{1}\left(c_i^{(t+1)} = k\right)}$	$\mu_k^{(t+1)} = \frac{\sum_{i=1}^n v_{i,k}^{(t+1)} x_i}{\sum_{i=1}^n v_{i,k}^{(t+1)}}$

To contrast the *Assignment step* against the *E step*, note that for any σ

$$\arg\min_k \|x_i - \mu_k\| = \arg\max_k \exp\left\{ -\frac{1}{2\sigma^2} \|x_i - \mu_k\|^2 \right\}.$$

Furthermore, remember that, for problems with well separated clusters, there is one component k for which $v_{i,k} \approx 1$, while $v_{i,k} \approx 0$ for all other components. Hence, for well separated components we have that $\hat{v}_{i,k} \approx \mathbf{1}(\hat{c}_i = k)$ for all i and k. And because of this, we will have $\hat{\gamma}_k \approx \hat{\mu}_k$ in the

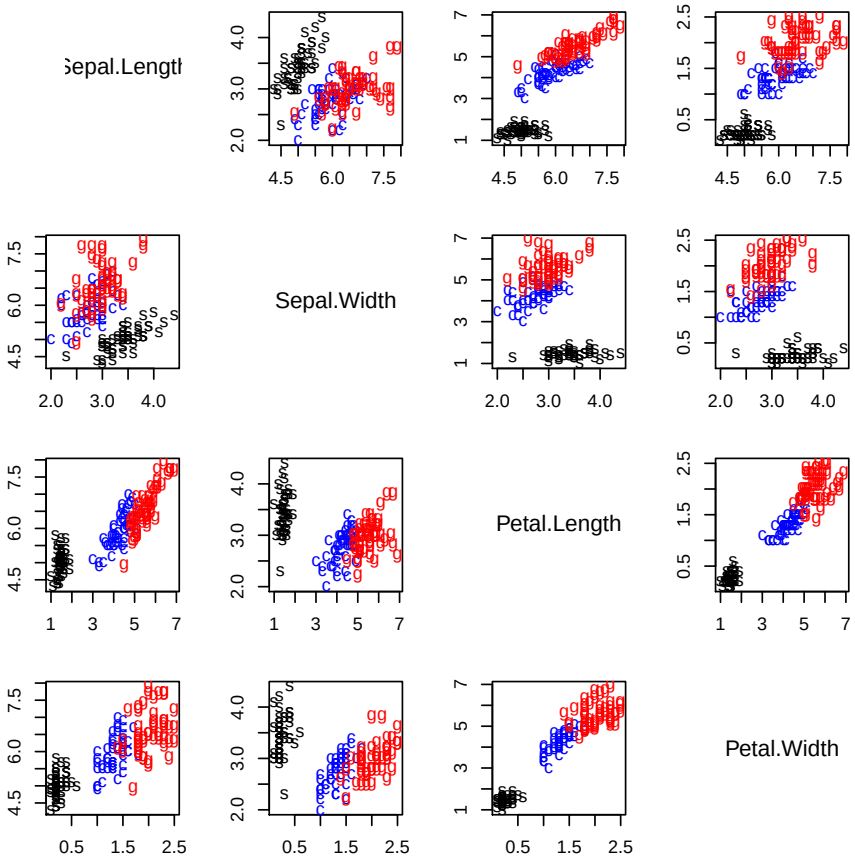

Figure 18: Pair plots for the `iris` dataset. Letters correspond to the true species of each sample (the true cluster labels, `s` for *setosa*, `g` for *virginica*, and `c` for *versicolor*).

optimum. Hence, we can think of k-means clustering as a very close approximation to the results that would be obtained by using the EM algorithm to fit the K-component mixture in (10).

By casting k-means clustering as a special case of a mixture model we can easily identify some of the shortcomings of the algorithm and suggest possible ways in which they can be addressed. For example, it should be clear that k-means expects clusters to be spherical, of about the same size, and to be more or less equally represented in the sample. Many data sets you will observe in real life do not satisfy these requirements, and for those tsetse k-means clustering will be a poor tool. One way to address this is to use a more general location-scale mixture model that allows non-uniform weights as well as different variance-covariance matrices for each of the components.

We illustrate the use of mixture models for clustering using the Fisher's famous `iris` data set. The data set gives the measurements in centimeters of the length and width of both the sepal and petal for 50 flowers from each of 3 species of iris plants (*Iris setosa*, *Iris versicolor*, and *Iris virginica*). Pair plots of the raw data can be seen in Figure 18.

We fit a location and scale mixture of tetra-variate Gaussian distributions using an EM algorithm (recall Section 2.3), and compare the results with those generated by the `R` function `kmeans`. The resulting partitions of the

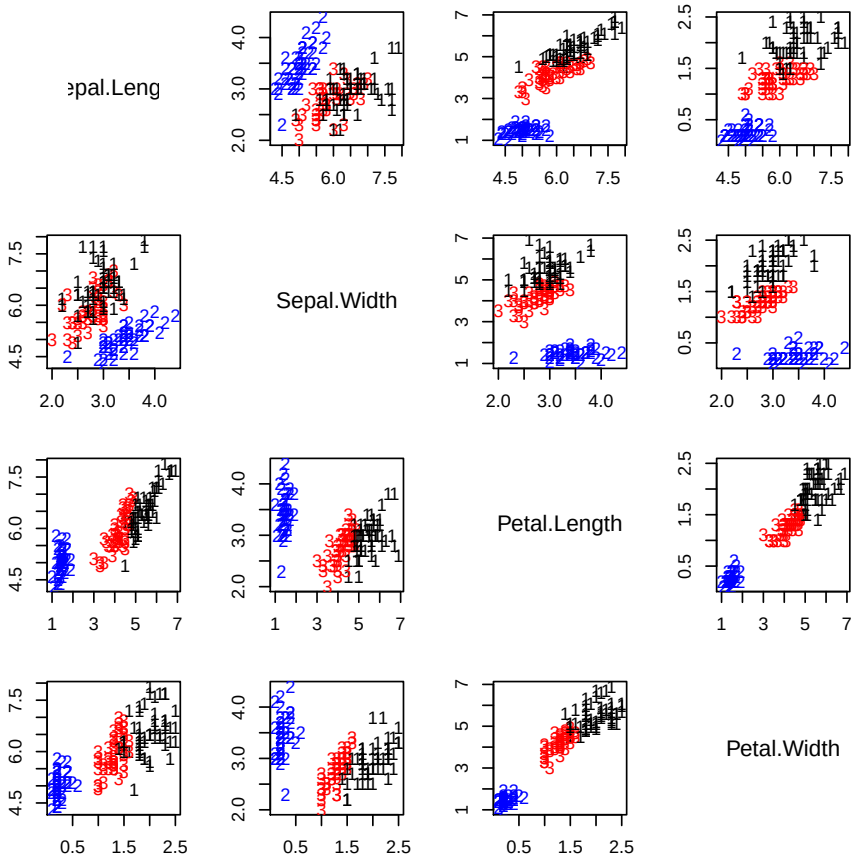

Figure 19: Results for the clustering procedure for the `iris` data set based on a location and scale mixture of multivariate Gaussian distributions fitted using the EM algorithm. Numbers correspond to the inferred cluster labels, $\hat{c}_1, \ldots, \hat{c}_n$.

data are compared using the adjusted Rand index, which is implemented by the function `adjustedRandIndex` of the package `mclust`. This index has zero expected value in the case of random partitions, and it is bounded above by 1 in the case of perfect agreement between two partitions. Note that, for both approaches, the results presented here correspond to the best fit from 15 independent runs of the algorithm. The reason for this multiple runs is discussed in more detail in Chapter 5.

Figures 19 and 20 present the results generated by the EM algorithm and a k-means clustering procedure, respectively. We can see that the group of flowers corresponding to *setosa* is correctly picked out by both algorithms. On the other hand, both procedures struggle to accurately separate *virginica* and *versicolor*. This is not surprising since there is substantial overlap between these two species. Nonetheless, the more flexible mixture model does a much job of separating these two groups. In particular, note that the value of the adjusted Rand index with respect to the true partition is larger for the mixture model (0.9039) than for k-means clustering (0.7302).

Label switching is a major issue when using mixture models for clustering applications. Note, for example, that the same solution to the clustering problem can be represented by K! different labeling schemes. Hence, comparisons among multiple solutions of the problem need to carried out carefully. To illustrate the problem in more detail consider two of 15 solutions

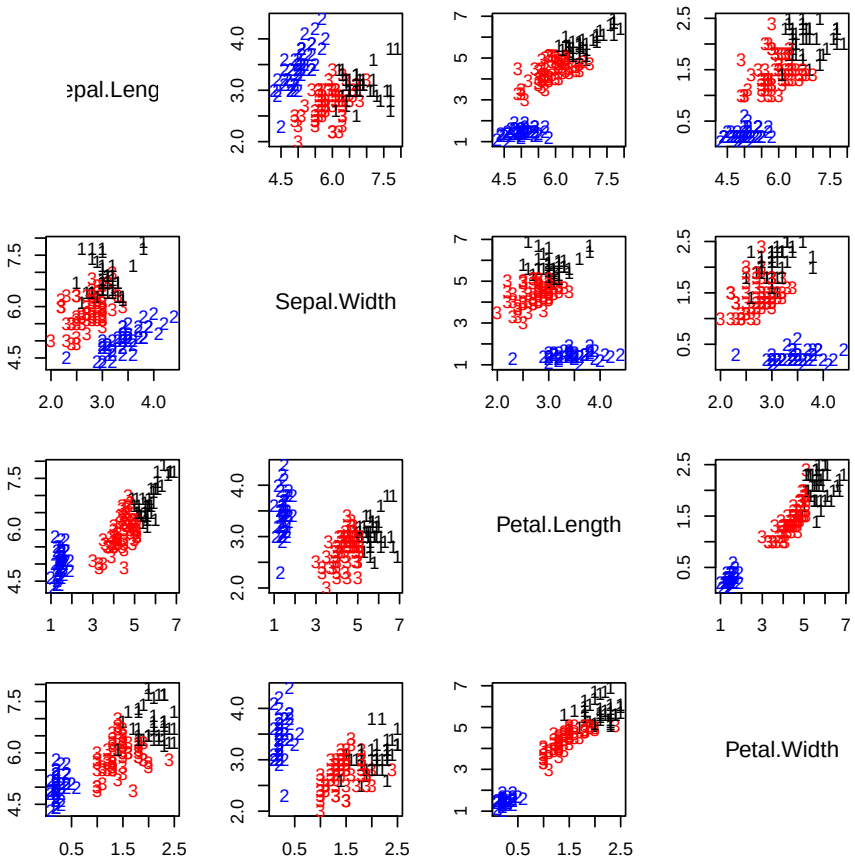

Figure 20: Results for the k-means clustering algorithms for the `iris` data set. Numbers correspond to the inferred cluster labels $s_1, \ldots, s_n$.

produced by our previous run. Both runs converge to the same value of the Q function (-185.0587), which is also the highest value over all 15 runs. The two solutions are identical from the point of view of the way the observations are grouped, a fact that is confirmed if we compute the adjusted Rand index between them (which takes the value of 1 in this case). However, if we look at the estimated $\hat{c}_i$s associated with both solutions we note that they differ in that the labels for all three groups have been switched around!

```
QQ.sum[8]

## [1] -185.0587

cc1 = apply(v.sum[8,,], 1 ,which.max)
QQ.sum[15]

## [1] -185.0587

cc2 = apply(v.sum[15,,], 1 ,which.max)
adjustedRandIndex(cc1, cc2)

## [1] 1

cc1[1:15]

##  [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2

cc2[1:15]

##  [1] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3

table(cc1,cc2)

##    cc2
## cc1  1  2  3
##   1  0 55  0
##   2  0  0 50
##   3 45  0  0
```

In general, because of label switching, partitions cannot be compared by simply looking at the inferred labels. Comparisons of partitions need to be done using metrics that are invariant to permutations of the labels, such as the adjusted Rand index. This is also true when employing MCMC algorithms to fit Bayesian mixture models, where posterior summaries over the partition structure need to account for the fact that different iterations of the MCMC algorithm might be label-switched.

4.3 (SUPERVISED) CLASSIFICATION

The goal in classification tasks is to identify to which of a set of categories a batch of new observations (usually called the *test set*) belong on the basis of a labeled *training set*. This task is similar to clustering (unsupervised classification), except that now a training set that provides information about the characteristics of the groups is available. For notational convenience let $x_1, \ldots, x_n$ be the n observations in the training set, with $c_1, \ldots, c_n$ being their **known** labels, and let $x_{n+1}, \ldots, x_{n+m}$ denote the m observations in

the test set with **unknown** labels $c_{n+1}, \ldots, c_{n+m}$ that we are interested in predicting.

For example, we might want to predict whether a group of people suffers from a disease using the results from p medical tests. The training set corresponds to measurements taken on two groups of individuals: one that contains only healthy patients, and one that contained only diseased individuals. The test set corresponds in this case of all new patients coming into a doctor's office and whose disease status is unknown.

A common class of methods for classification are based on a direct application of Bayes theorem (sometimes called *naive Bayes* classifier):

$$\Pr(x_i \in \text{class } k) = \frac{\omega_k g_k(x_i \mid \theta_k)}{\sum_{l=1}^{K} \omega_l g_l(x_i \mid \theta_l)},$$

with

$$\tilde{c}_i = \arg\max_k \Pr(x_i \in \text{class } k).$$

for $i = n+1, \ldots, n+m$. A well known example of naive Bayes classification is linear discriminant analysis. Assume that $K = 2$ (like in our motivating example) and set

$$g_k(x_i) = \left(\frac{1}{\sqrt{2\pi}}\right)^q |\Sigma|^{-1/2} \exp\left\{-\frac{1}{2}(x_i - \mu_k)^T \Sigma^{-1/2}(x_i - \mu_k)\right\},$$

i.e., the distribution of each class is q-variate normal distribution with its own mean μ_k but common variance-covariance Σ. In this case

$$\Pr(x_i \in \text{class } 1) \propto \omega \exp\left\{-\frac{1}{2}(x_i - \mu_1)^T \Sigma^{-1}(x_i - \mu_1)\right\},$$

while

$$\Pr(x_i \in \text{class } 2) \propto \omega \exp\left\{-\frac{1}{2}(x_i - \mu_2)^T \Sigma^{-1}(x_i - \mu_2)\right\}.$$

Hence, $\tilde{c}_i = 1$ if and only if

$$\omega \exp\left\{-\frac{1}{2}(x_i - \mu_1)^T \Sigma^{-1}(x_i - \mu_1)\right\} > (1-\omega) \exp\left\{-\frac{1}{2}(x_i - \mu_2)^T \Sigma^{-1}(x_i - \mu_2)\right\},$$

which, after some algebra, leads to

$$(x_i - \mu_1)^T \Sigma^{-1}(x_i - \mu_1) - (x_i - \mu_2)^T \Sigma^{-1}(x_i - \mu_2) < -2 \log \frac{\omega}{1-\omega}.$$

This expression can be further simplified by noting that

$$\begin{aligned}
&(x_i - \mu_1)^T \Sigma^{-1}(x_i - \mu_1) - (x_i - \mu_2)^T \Sigma^{-1}(x_i - \mu_2) \\
&\quad = \cancel{x_i^T \Sigma^{-1} x_i} - 2 x_i \Sigma^{-1} \mu_1 + \mu_1^T \Sigma^{-1} \mu_1 - \cancel{x_i^T \Sigma^{-1} x_i} \\
&\qquad + 2 x_i \Sigma^{-1} \mu_2 - \mu_2^T \Sigma^{-1} \mu_2 \\
&\quad = 2 x_i \Sigma^{-1}(\mu_2 - \mu_1) + \mu_1^T \Sigma^{-1} \mu_1 - \mu_2^T \Sigma^{-1} \mu_2.
\end{aligned}$$

Hence, $\tilde{c}_i = 1$ if and only if

$$x_i \Sigma^{-1}(\mu_2 - \mu_1) < T^*,$$

where $T^* = \frac{1}{2}\mu_2^T\Sigma^{-1}\mu_2 - \frac{1}{2}\mu_1^T\Sigma^{-1}\mu_1 - \log\frac{1-\omega}{\omega}$ is a threshold that is independent of the observation x_i. This formula just states that a linear combination of the features in observation x_i can be used to separate the two classes. Hence, the name *Linear Discriminant Analysis*. In practice μ_1, μ_2 and Σ are unknown and estimated from the training set using the corresponding maximum likelihood estimators.

Let's explore now the relationship between mixture models and linear discriminant analysis. In particular, consider a three-step non-iterative algorithm that resembles the EM algorithm for a mixture of two Gaussian distributions:

- **"E step":** For the observations in the training set (i.e., for $i = 1, \ldots, n$) let
$$v_{i,k} = \begin{cases} 1 & c_i = k \\ 0 & \text{otherwise.} \end{cases}$$
- **"M step:"** Estimate
$$\hat{\mu}_k = \frac{1}{\sum_{i=1}^n v_{i,k}} \sum_{i=1}^n v_{i,k} x_i, \qquad k = 1, 2,$$
$$\hat{\Sigma} = \frac{1}{n} \sum_{i=1}^n \sum_{k=1}^2 v_{i,k}(x_i - \mu_k)(x_i - \mu_k)^T,$$
$$\hat{\omega} = \frac{1}{n} \sum_{i=1}^n v_{i,1}.$$
- **"Post-processing step:"** For the observation in the test set (i.e., for $i = n+1, \ldots, n+m$) compute
$$v_{i,1} = 1 - v_{i,2} = \frac{1}{v_{i,\cdot}} \hat{\omega} \exp\left\{-\frac{1}{2}(x_i - \hat{\mu}_1)^T \hat{\Sigma}^{-1}(x_i - \hat{\mu}_1)\right\},$$
where
$$v_{i,\cdot} = \hat{\omega} \exp\left\{-\frac{1}{2}(x_i - \hat{\mu_1})^T \hat{\Sigma}^{-1}(x_i - \hat{\mu_1})\right\} + (1 - \hat{\omega}) \exp\left\{-\frac{1}{2}(x_i - \hat{\mu}_2)^T \hat{\Sigma}^{-1}(x_i - \hat{\mu}_2)\right\},$$
and set
$$\hat{c}_i = \arg\max_k v_{i,k}.$$

This algorithm is equivalent to the linear discrimination algorithm we discussed above. Note that there is no need for iterative steps in this case: the labels $c_1, \ldots, c_n$ in the training set are known so the E step only needs to be carried out once. Furthermore, given the labels, the MLEs $\hat{\mu}_1$, $\hat{\mu}_2$ and $\hat{\Sigma}$ (which are obtained in the M step) can be obtained explicitly in a single step. Also note that the algorithm uses only the information in the training set (observations 1 to n) to compute $\hat{\mu}_1$, $\hat{\mu}_2$ and $\hat{\Sigma}$.

A variant of this algorithm that can be considered a semi-supervised classification algorithm uses all observations to estimate the MLEs of μ_1, μ_2 and Σ instead:

- **E step:** For the observations in the training set (i.e., for $i = 1, \ldots, n$) let

$$v_{i,k}^{(t+1)} = \begin{cases} 1 & c_i = k, \\ 0 & \text{otherwise,} \end{cases},$$

while for the observation in the test set (i.e., for $i = n+1, \ldots, n+m$) compute

$$v_{i,1}^{(t+1)} = 1 - v_{i,2}^{(t+1)} = \frac{1}{v_{i,\cdot}^{(t+1)}} \omega^{(t)} \exp\left\{ -\frac{1}{2} (x_i - \mu_1^{(t)})^T \left[\Sigma^{(t)}\right]^{-1} (x_i - \mu_1^{(t)}) \right\},$$

where

$$v_{i,\cdot}^{(t+1)} = \omega^{(t)} \exp\left\{ -\frac{1}{2} (x_i - \mu_1^{(t)})^T \left[\Sigma^{(t)}\right]^{-1} (x_i - \mu_1^{(t)}) \right\} + \left(1 - \omega^{(t)}\right) \exp\left\{ -\frac{1}{2} (x_i - \mu_2^{(t)})^T \left[\Sigma^{(t)}\right]^{-1} (x_i - \mu_2^{(t)}) \right\}.$$

- **M step:** Estimate

$$\hat{\omega}^{(t+1)} = \frac{1}{n} \sum_{i=1}^{n+m} v_{i,1}^{(t+1)},$$

$$\hat{\mu}_k^{(t+1)} = \frac{1}{\sum_{i=1}^{n+m} v_{i,k}^{(t+1)}} \sum_{i=1}^{n+m} v_{i,k}^{(t+1)} x_i,$$

$$\hat{\Sigma}^{(t+1)} = \frac{1}{n} \sum_{i=1}^{n+m} \sum_{k=1}^{K} v_{i,k}^{(t+1)} (x_i - \mu_k)(x_i - \mu_k)^T.$$

- **Post-processing step:** Once the algorithm has converged, classify observations in the test set by setting

$$\hat{c}_i = \arg\max_k v_{i,k}$$

(Note that all sums in the M step go from 1 to $n+m$ rather than just 1 to n, and that this is an iterative algorithm for which the E and M steps need to be repeated until convergence). When the test set is small compared with the training set, the supervised and semi-supervised algorithms yield almost identical results. On the other hand, when the training set is small compared with the test set and the classes indeed satisfy the assumptions of the model (i.e., the features are distributed a normal distributions with different means but the same variance), the semi-supervised algorithm tends to yield more accurate results.

By placing linear discriminant analysis in the context of mixture models we open the door for a number of natural extensions. First, classifications with more than two classes can be accommodated using a general mixture with K components. Secondly, by using location and scale mixtures that allow each class to have its own variance matrix we obtained the so-called *quadratic discriminant analysis*. Furthermore, alternative families for g_k can accommodate other features of the classes such as skewness. A particularly interesting example arises when g_k is itself a mixture with L components, leading to *mixture discriminant analysis*. Finally, using Bayesian methods to

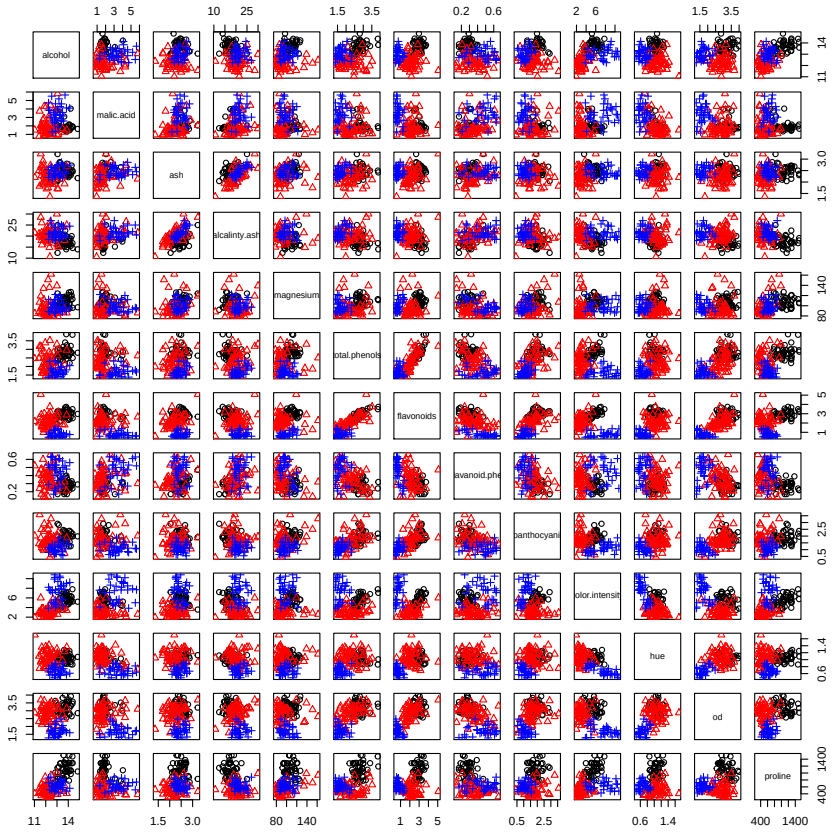

Figure 21: Pair plots for the training set of `wine` dataset. Colors/symbols correspond to the cultivar of each sample (the true known classification labels).

estimate the mixture leads to Bayesian versions of linear/quadratic/mixture discriminant analysis.

To illustrate the use of mixture models for classification, we consider a data set that is the result of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each wine (see Figure 21). Two files are provided, `wine_training.txt` and `wine_test.txt`, which contain the training set (132 observations, 44 in the first class, 58 in the second and 30 in the third) and the test set (46 observations, 15 in group 1, 13 in group 2 and 18 in group 3), respectively.

We employ the variant of the EM algorithm for semi-supervised classification discussed above and contrast the results against those produced by the `lda` and `qda` functions in the `R` package `MASS`. The code is provided in the file `winesclas.R`. In this case both our semi-supervised method and quadratic discriminant analysis misclassify one observation in the test set, while linear discriminant analysis performs perfectly

Label switching is not usually an issue in classification applications because the training set provides information that anchors the interpretation of the labels we infer in the test set.

5 PRACTICAL CONSIDERATIONS

5.1 ENSURING NUMERICAL STABILITY WHEN COMPUTING CLASS PROBABILITIES

All the algorithms we have discussed to fit mixture models require that we compute probability weights of the form

$$\frac{z_k}{\sum_{l=1}^{K} z_l}$$

where $z_k = \omega_k g_k(x \mid \theta_k)$ is a positive number for every k. When computing these weights, much care needs to be exercised to avoid computational issues arising from finite machine precision. In particular, if calculations are not done carefully, you might end up trying to compute $0/0$!

Let's employ a simple example to illustrate the point. Consider a mixture of two Gaussian distributions with weights $\omega_1 = \omega_2 = 1/2$, means $\mu_1 = 0$ and $\mu_2 = 1$, and common variance $\sigma = 1$, and assume that you observe a value $x_1 = 50$. Let's use the R function `dnorm` to evaluate the terms $z_1 = \frac{1}{\sqrt{2\pi}} \exp\left\{-\frac{50^2}{2}\right\}$ and $z_2 = \frac{1}{\sqrt{2\pi}} \exp\left\{-\frac{49^2}{2}\right\}$ that would be required to, for example, compute the weight $v_{1,1}$ and $v_{1,2}$.

```
dnorm(50, 0, 1)

## [1] 0

dnorm(50, 1, 1)

## [1] 0

dnorm(50, 0, 1)/(dnorm(50, 0, 1) + dnorm(50, 1, 1))

## [1] NaN
```

Note that `R` reports both values being equal to zero, and if we attempt to compute the normalized weight, we get `NaN` as the response (recall that this is `R`'s way of telling you that you are trying to carry out an undefined operation). The reason for this apparent nonsense is that the result of exponentiating a negative number with even a moderately large absolute value is smaller than the smallest number that the computer can represent using its finite precision. In other words, the numbers are so small that the computer cannot tell them apart from zero!

The simplest way to address this numerical issue is to work in the logarithmic scale. In particular, note that

$$\begin{aligned}
\frac{z_k}{\sum_{l=1}^{K} z_l} &= \frac{\exp\{\log z_k\}}{\sum_{l=1}^{K} \exp\{\log z_l\}} && \text{(exp and log are inverse functions)} \\
&= \frac{\exp\{-b\}\exp\{\log z_k\}}{\exp\{-b\}\sum_{l=1}^{K} \exp\{\log z_l\}} && \text{(multiply and divide by the same quantity, } e^{-b}\text{)} \\
&= \frac{\exp\{\log z_k - b\}}{\sum_{l=1}^{K} \exp\{\log z_l - b\}} && (11)
\end{aligned}$$

for any number b. In words, if we add any constant to the logarithms of the unnormalized weights, the value of the normalized weights remain unchanged.

To empirically verify the identity above let's go back to our motivating example involving a mixture of two normals, but let assume that the observation is $x_2 = 3$:

```
## Direct calculation
z1 = dnorm(3, 0, 1)
z2 = dnorm(3, 1, 1)
z1/(z1+z2)

## [1] 0.07585818

## Compute in the logarithm scale, add b
## to all values, and then exponentiate before standardizing
lz1 = dnorm(3, 0, 1, log=3)
lz2 = dnorm(3, 1, 1, log=3)
b = 3
exp(lz1+b)/(exp(lz1+b) + exp(lz2+b))

## [1] 0.07585818
```

Although (11) is valid for any b, it should be clear that some values will work better than others for the purpose of avoiding a $0/0$ calculation. In particular, we are interested in choosing a value b that makes at least one of the terms in the denominator different from zero after exponentiation. One such choice is $b = \max_{l=1,\dots,K} \log z_l$:

$$\sum_{l=1}^{K} \exp\left\{\log z_l - \max_{l=1,\dots,K} \log z_l\right\} = 1 + \sum_{l:l\neq l^*} \exp\left\{\log z_l - \max_{l=1,\dots,K} \log z_l\right\}$$

where $l^* = \arg\max_{l=1,\dots,K}\{\log z_l\}$ is the index corresponding to the largest value of $\log z_l$. One key advantage of this choice is that all the terms in the sum are less or equal than one, which ensures that we do not overflow when computing $\exp\left\{\log z_l - \max_{l=1,\dots,K} \log z_l\right\}$. Returning to our original example with $x_1 = 50$:

```
## Compute in the logarithm scale,
## add b to all values, and then exponentiate
lz1 = dnorm(50, 0, 1, log=TRUE)
lz2 = dnorm(50, 1, 1, log=TRUE)
b = max(lz1, lz2)
exp(lz1-b)/(exp(lz1-b) + exp(lz2-b))

## [1] 3.179971e-22
```

Note that, for this trick to work, it must be implemented carefully and operations need to be carried out directly in the logarithm scale. If the evaluation of kernel involves any exponentiation, we want to avoid carrying out that exponentiation in the first place!

```
## Wrong
lz1 = log(dnorm(50, 0, 1))
lz2 = log(dnorm(50, 1, 1))
b = max(lz1, lz2)
exp(lz1-b)/(exp(lz1-b) + exp(lz2-b))

## [1] NaN

## Wrong
lz1 = log(exp(-0.5*50^2)/sqrt(2*pi))
lz2 = log(exp(-0.5*49^2)/sqrt(2*pi))
b = max(lz1, lz2)
exp(lz1-b)/(exp(lz1-b) + exp(lz2-b))

## [1] NaN

## Right
lz1 = dnorm(50, 0, 1, log=TRUE)
lz2 = dnorm(50, 1, 1, log=TRUE)
b = max(lz1, lz2)
exp(lz1-b)/(exp(lz1-b) + exp(lz2-b))

## [1] 3.179971e-22

## Also right (just more cumbersome)
lz1 = -0.5*log(2*pi) - 0.5*50^2
lz2 = -0.5*log(2*pi) - 0.5*49^2
b = max(lz1, lz2)
exp(lz1-b)/(exp(lz1-b) + exp(lz2-b))

## [1] 3.179971e-22
```

5.2 NUMERICAL CONSEQUENCES OF MULTIMODALITY

One challenge involved in working with mixture models is their likelihoods tend to be highly multimodal. The effect of this multimodality is often relatively mild in low-dimensional settings in which the components of the mixture are well separated, but can become quite severe when there is substantial overlap among the components or the dimensionality of the observations is high.

To illustrate the multimodality of the likelihood function, recall our clustering example from Section 4.2. In that example we fitted a location and scale mixture of three multivariate Gaussian distributions to Fisher's `iris` data set. As you might recall, when implementing that example we ran our algorithm multiple times using different (randomly generated) starting locations for the cluster centers, and reported the results based on the best solution. We did this because the EM algorithm is only guaranteed to converge to a local mode.

How bad are the other solutions? Figure 22 shows a boxplot of the value of the Q function at convergence for all 15 initial values, while Figures 23 and 24 show the partition structure for the data associated with the best and worst solutions, respectively. (Note that Figure 23 is identical to Figure 19). From the boxplot it is clear that the value of the Q function can vary wildly depending on the starting value of the algorithm. Given that the steps of

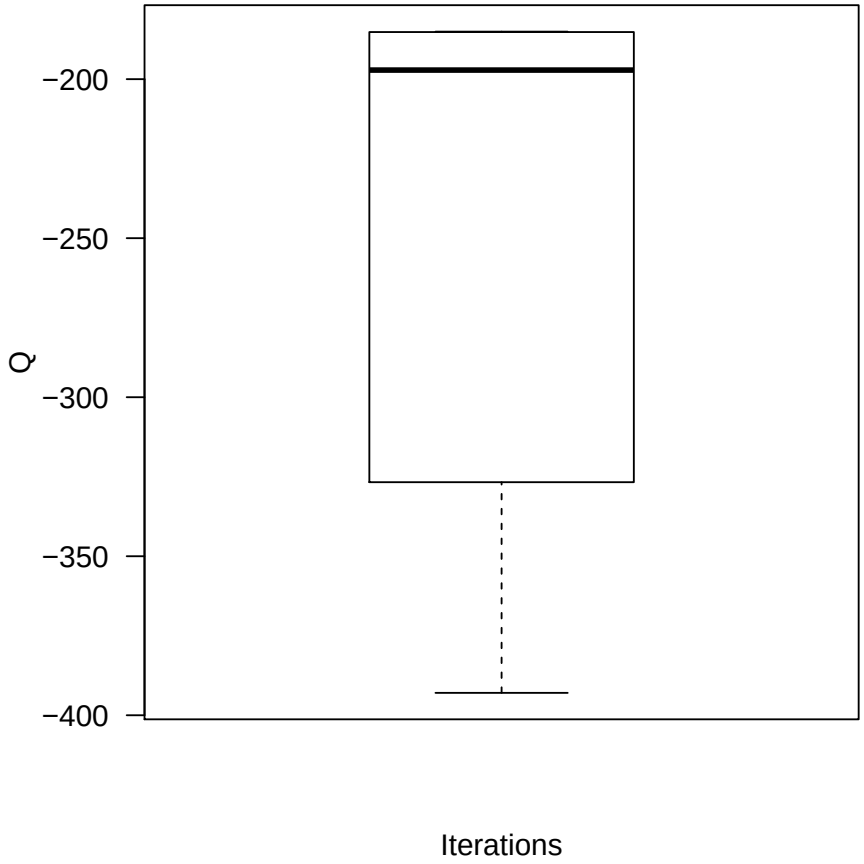

Figure 22: Boxplot of the value of the Q function at convergence for 15 runs of the EM algorithm for a location and scale mixture of multivariate Gaussian distributions for the `iris` data set.

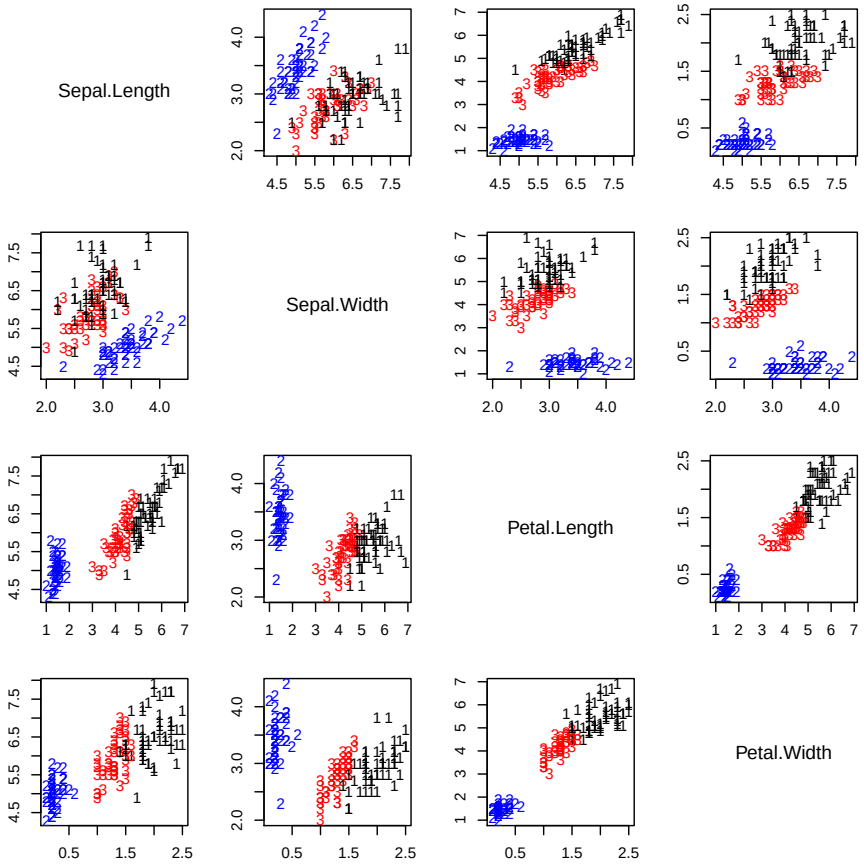

Figure 23: Best result (out of 15 runs) for the clustering procedure for the `iris` data set based on a location and scale mixture of multivariate Gaussian distributions fitted using the EM algorithm. Numbers correspond to the infered cluster labels, $\hat{c}_1, \ldots, \hat{c}_n$.

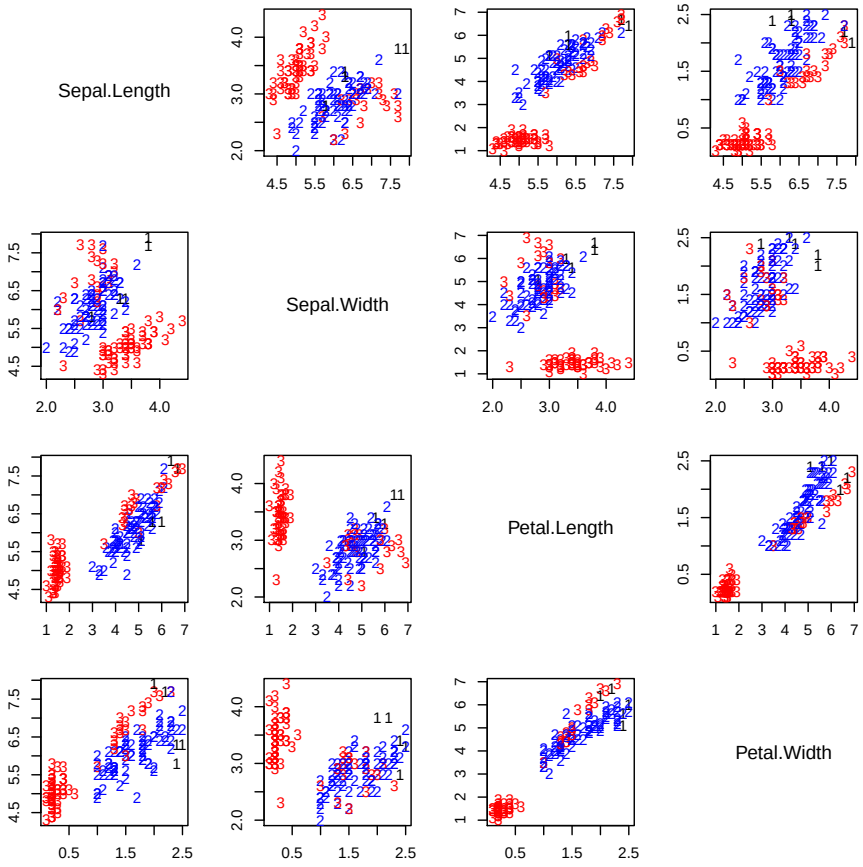

Figure 24: Worst result (out of 15 runs) for the clustering procedure for the `iris` data set based on a location and scale mixture of multivariate Gaussian distributions fitted using the EM algorithm. Numbers correspond to the infered cluster labels, $\hat{c}_1, \ldots, \hat{c}_n$.

the EM algorithm always improve on the value of Q, this is clear evidence of multimodality. Furthermore, note that the quality of the worst solution is really low. Indeed, recall that Setosa flowers have features that are quite different from the other two species and therefore a reasonable solution should clearly allocate them to a cluster of their own. Against this intuition, the worst solution creates at least one cluster that includes observation from all three species.

Multimodality is not an exclusive problem of EM algorithm. As you might have guessed, the k-means clustering algorithm suffers from identical issues. Multimodality is also a problem when working with Markov chain Monte Carlo algorithms for finite mixture models, although the issues tend to be less severe than with the EM algorithm. Indeed, MCMC algorithms are theoretically guaranteed to converge if run for long enough (and, therefore, to jump across modes according to their relative importance). Nonetheless, in practice it might take an impractically large number of iterations to do so. This means that the MCMC algorithm might get "stuck" exploring the posterior distribution around a local mode for long periods of time and appear to converge, but fail to discover the global one. Therefore, it is good practice to also perform multiple runs of your MCMC algorithm when carrying out a Bayesian analysis for mixture models. If different runs appear to coverage to qualitatively different solutions then only the samples from the chain with the largest unnormalized log posterior should be used for inference.

Another issue that might arise when implementing the EM algorithm that is related to multimodality (and to the lack of identifiability of mixture models) is that some of the modes of the likelihood function might involve partitions that utilize only a subset of the K components in the mixture (i.e., modes for which there is at least one index k such that $\nu_{i,k} \approx 0$ for all $i = 1, \ldots, n$). This is an issue because the partial MLEs $\theta_1^{(t)}, \ldots, \theta_K^{(t)}$ do not exist in that case.

As before, we illustrate this issue in the context of clustering the observations in the `iris` data set. Rather than using a random initial point for the algorithm, we initialize it in such a way that the first component completely covers the support of the observed data, while the other two fall completely outside it (see Figure 25 and the code contained in `clusteringiris_broken.R`). If you attempt to execute the EM algorithm you will get an error of the form:

```
## Error in if (abs(QQn - QQ)/abs(QQn) < epsilon) {: missing value
where TRUE/FALSE needed
```

This error arises because the evaluation of the Q function at the end of the iteration resulted in a `NaN` value, leading to an undefined logical operation when carrying out the convergence check.

```
QQn

## [1] NaN

if(abs(QQn-QQ)/abs(QQn)<epsilon){
    sw=TRUE
}

## Error in if (abs(QQn - QQ)/abs(QQn) < epsilon) {: missing value
where TRUE/FALSE needed
```

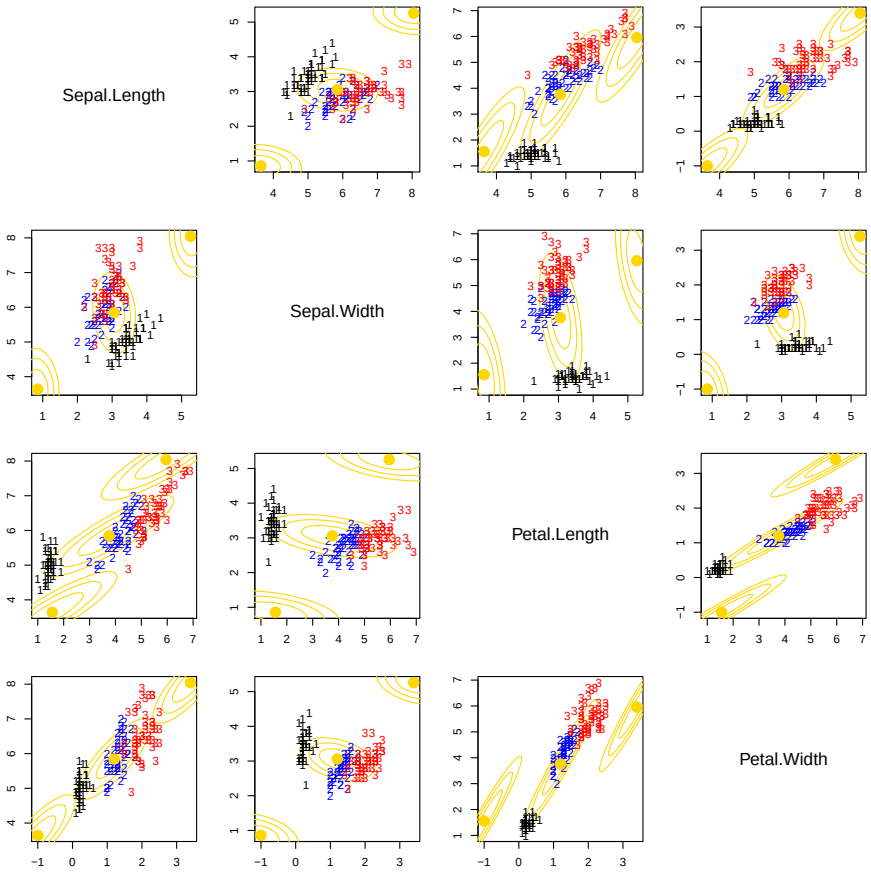

Figure 25: An initial state for the EM algorithm in the `iris` dataset that fails to lead to convergence because it leads to two components.

Unlike the previously discussed numerical issues (multimodality and approximation errors due to finite computer precision), this is a fundamental problem with maximum likelihood estimation rather than an implementation issue. Empty components are fine in the context of Bayesian inference as long as proper priors are used (as recommended in Chapter 3). In practice, any solution of the EM algorithm that seems to converge to a mode of this type should be discarded outright, and the algorithm restarted with a new initial value. Fortunately, this is not an extremely common situation.

5.3 SELECTING THE NUMBER COMPONENTS: BIC

Selecting the right number of components in a mixture is a critical and difficult task, particularly for clustering and density estimation tasks. In fact, determining the number of cluster in the mixture is often an important scientific question of its own. For example, in the `iris` data set, the number of clusters in the sample corresponds to the number of distinct plant species in the sample, and discriminating between $K = 2$ and $K = 3$ allows us to determine whether *Iris versicolor* and *Iris virginica are* indeed different species.

One of the most common approaches to model selection that is useful for mixture models is the Bayesian Information Criteria (BIC). Given a collection of J models to be compared, the BIC for model j is given by the formula

$$\mathrm{BIC}_j = -2 \log L_j(\hat{\eta}_j) + r_j \log(n), \tag{12}$$

where L_j corresponds to the likelihood of model j, $\hat{\eta}_j$ is the maximum likelihood estimator for its parameters, r_j is the number of free parameters associated with model j, and n in the number of observations in the sample. The optimal model is the one with the lowest BIC.

You can interpret the first term in (12) as measuring goodness of fit: lower values for $-2 \log L_j(\hat{\eta}_j)$ implies that model j fits the data better. On the other hand, the second term is a complexity penalty: the more parameters in the model, the higher the penalty. Hence, BIC explicitly trades off goodness of fit with complexity.

In the case of mixture models, j corresponds to K (the number of components in the mixture), $\eta_K = (\omega_1, \ldots, \omega_K, \theta_1, \ldots, \theta_K)$, and the likelihood takes the form

$$L_K(\omega_1, \ldots, \omega_K, \theta_1, \ldots, \theta_K) = \prod_{i=1}^{n} \sum_{k=1}^{K} \omega_k g_k(x_i \mid \theta_k)$$

(as discussed in Section 1.4). On the other hand, since we only have $K - 1$ independent weights, the number of parameters is given by

$$r_K = (K - 1) + \sum_{k=1}^{K} \dim \theta_k,$$

where the number of parameters associated with the kernel, $\sum_{k=1}^{K} \dim \theta_k$, depends on the specific choice of kernel. For example, in the case of location mixtures of q-variate normal distributions we have

$$\sum_{k=1}^{K} \dim \theta_k = Kq + \frac{q(q+1)}{2}$$

(there are K mean vectors of dimension q, and a single variance-covariance matrix with of dimension $q \times q$, which has $q(q+1)/2$ free parameters). Similarly, for location and scale mixture of q-variate normal distributions with completely free covariance matrices for each dimension we have

$$\sum_{k=1}^{K} \dim \theta_k = K\left(q + \frac{q(q+1)}{2}\right)$$

To illustrate the performance of BIC in model selection consider again the `galaxies` data set we discussed in Section 4.1. We previously used a location mixture of 6 univariate Gaussian distributions to estimate the density associated with the velocity of the galaxies. The choice of 6 components was based on the fact that the observations were collected from 6 well-separated conic sections. Here we use BIC to determine whether six in indeed the optimal number of components for this problem, as well as to illustrate some of the identifiability issues we had originally introduced in Section 1.5.

We fit location mixtures of Gaussian distributions between $K = 2$ and $K = 20$ components using the EM algorithm. The code is available in the file `galaxies_BIC.R`. Figure 26 presents the value of the BIC as a function of K. Note that $K = 6$ is indeed the optimal choice for the number of component. BIC is relatively high for lower values of K (where the model does not provide a good fit to the data) as well as for higher values of K (where there is a good fit but also a large number of parameters).

To provide some intuition on these results we compare in Figure 27 the kernel density estimate for $K = 6$ (the optimal value) against those for $K = 4$ and $K = 5$, while in Figure 28 we compare the estimate for $K = 6$ against those for $K = 7$ and $K = 8$. From Figure 27 we see that the estimates change substantially when we go from $K = 4$ to $K = 6$. The biggest change can be seen in the estimate of the central area of the distribution: for $K = 4$ this area is estimated as clearly unimodal, while for $K = 5$ and $K = 6$ it becomes multimodal. On the other hand, from Figure 28 we see that the estimate does not change much when we add additional components over 6. In particular, the density estimator for $K = 7$ is identical to that for $K = 6$. In fact, the estimate we obtain $K = 7$ just splits one component into two that have about the same mean but different weights that add up to the original.

The above results agrees with our intuition about the behavior of BIC. Adding a six component to the model allows for a substantially better fit to the data, and therefore BIC decreases is minimized at $K = 6$. On the other hand, adding a seventh (or an eight!) component does not improve the fit of the model enough to justify the additional complexity, and therefore the BIC increases.

We can also use this example to demonstrate the trade offs between the number of components in the mixture and the standard deviation (bandwidth) of the kernels in density estimation (see Section 4.1). In Figure 29 we present the maximum likelihood estimate for the standard deviation σ as a function of the number of components K. Note that, as our intuition had suggested, the standard deviation tends to decrease as the number of components in the mixture increases. Hence, as the number of components increase they become more localized, and the density estimate becomes less smooth.

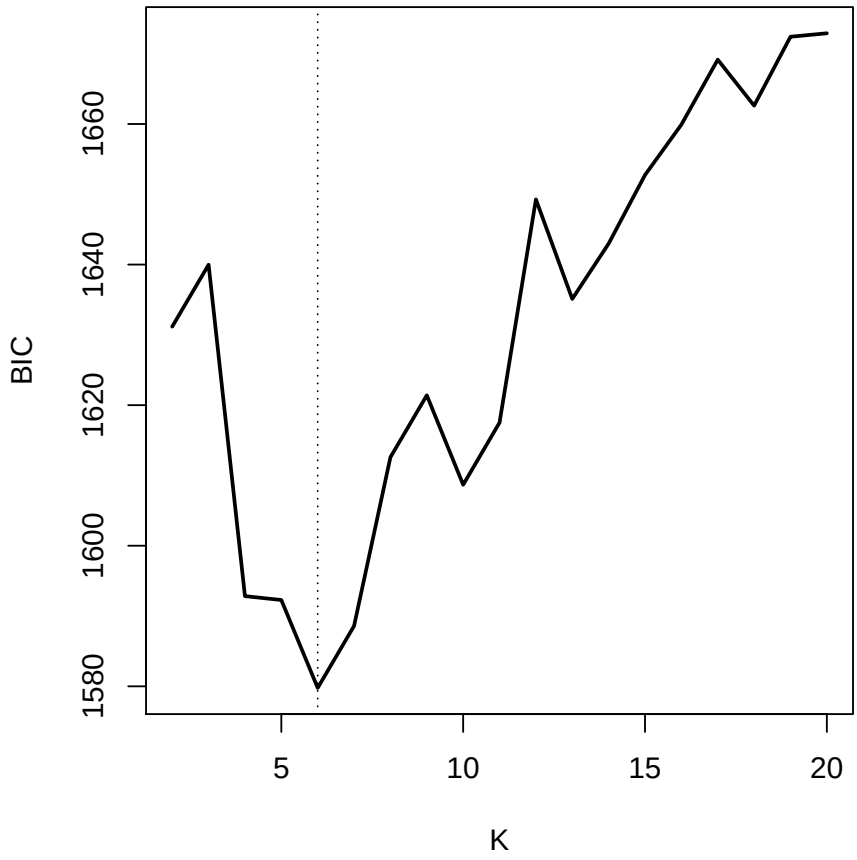

Figure 26: BIC as a function of the number of components K in a location mixture of Gaussian distributions for the `galaxies` dataset.

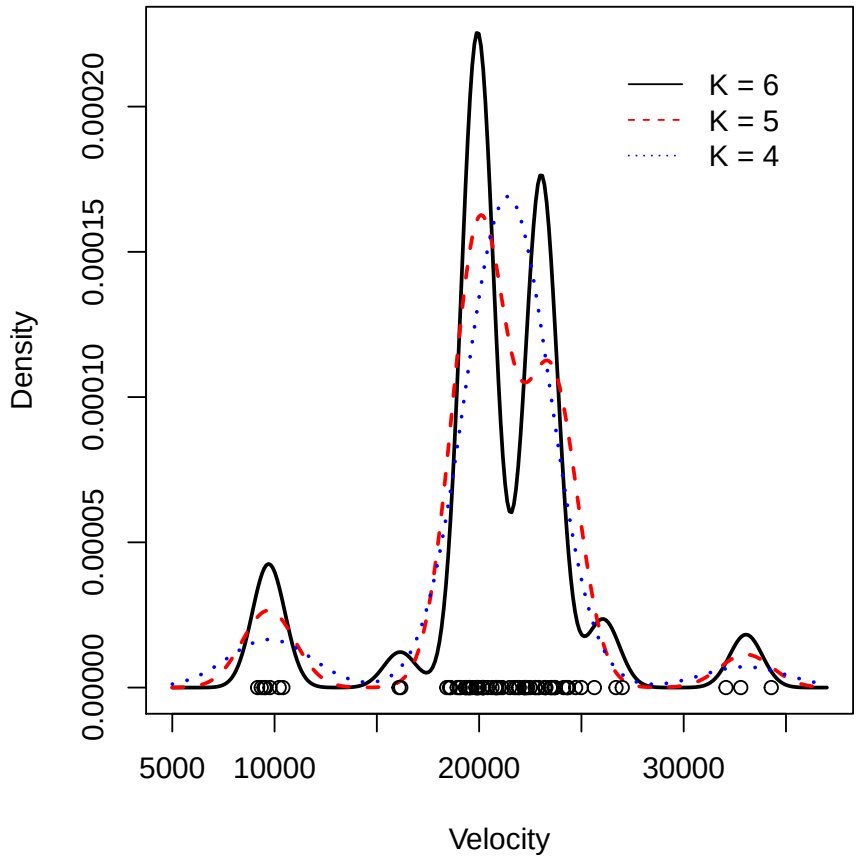

Figure 27: Density estimates for the `galaxies` dataset based on location mixtures of Gaussian distributions for $K = 6$ (optimal according to BIC), $K = 5$ and $K = 4$.

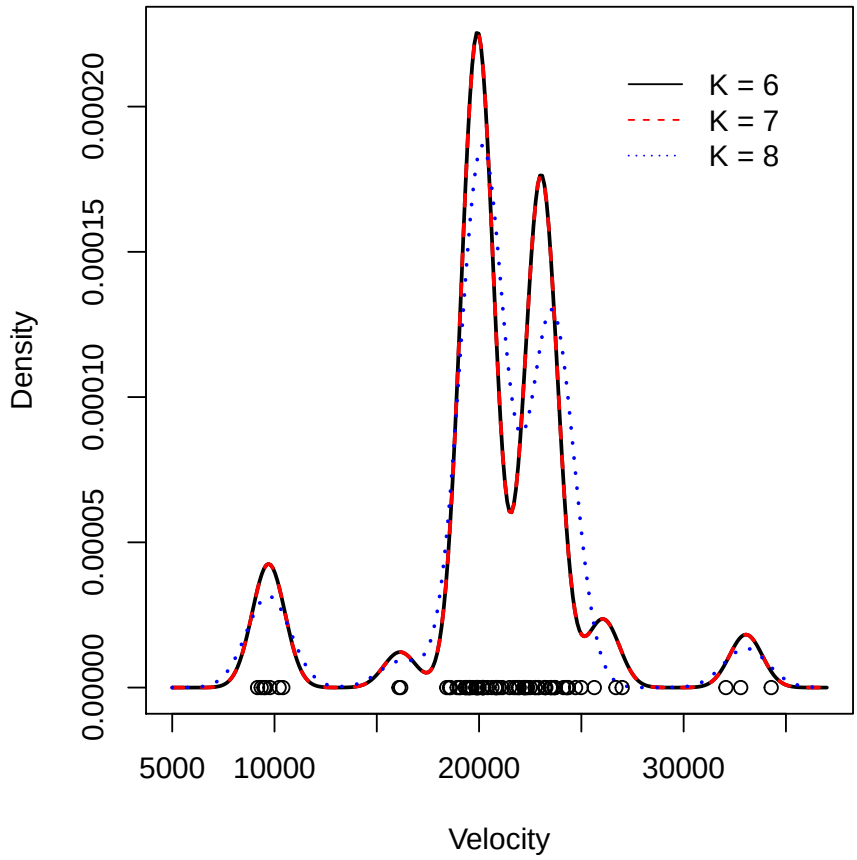

Figure 28: Density estimates for the `galaxies` dataset based on location mixtures of Gaussian distributions for $K = 6$ (optimal according to BIC), $K = 7$ and $K = 8$. Note that the graphs for $K = 6$ and $K = 7$ overalp.

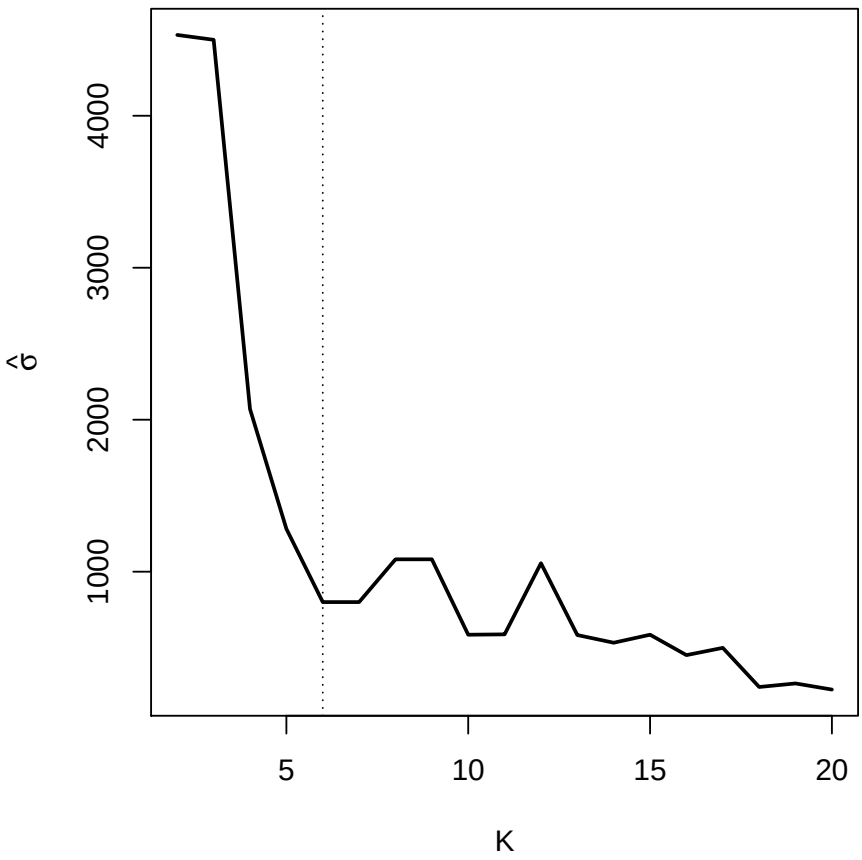

Figure 29: Maximum likelihood estimate for the standard deviation σ as a function of the number of components K in a location mixture of Gaussian distributions for the `galaxies` dataset.

5.4 FULLY BAYESIAN INFERENCE ON THE NUMBER OF COMPONENTS

As we discussed in Section 1.5, a mixture model with K^* components can be represented as another mixture with $K > K^*$ components, with the additional $K - K^*$ components having zero weights,

$$f(x) = \sum_{k=1}^{K^*} \omega_k g_k(x \mid \theta_k) + \sum_{k=K^*+1}^{K} 0 g_k(x \mid \theta_k).$$

In a Bayesian context, this observation can be exploited to devise a mechanism to perform inferences on the number of components in the model. This is done by noting that K, the *maximum* number of components we are willing to tolerate in the mixture (and which we might set to a relatively large number), can potentially be different from the actual number of components used by the data $K^* = \sum_{k=1}^{K} \mathbb{1}(m_k \geqslant 1)$ where $m_k = \sum_{i=1}^{n} \mathbb{1}(c_i = k)$ is the number of observations in the sample coming from component k. In practice, this translate into fitting mixtures with relatively large number of components K and then using summaries of the posterior distribution over the assignments $c_1, \ldots, c_n$ to determine the *effective* number of components that are supported by the data.

For this strategy to be effective, however, care needs to be exercised in the choice of a prior for the weights $\omega_1, \ldots, \omega_K$. Indeed a uniform prior on the K-dimensional simplex (our default choice so far) is such that the expected number of clusters K^* grows without bounds as K increases. This is clearly unappealing since the posterior distribution would then be substantially affected by the choice of the upper bound K. Instead, it would be desirable to select a prior on the weights that is independent of the choice K, or at least one where the value of K^* converges to a finite limit when K goes to infinity. We can obtain such a prior by making the parameters of the Dirichlet prior depend on K,

$$(\omega_1, \ldots, \omega_K) \sim \text{Dirichlet}\left(\frac{1}{K}, \ldots, \frac{1}{K}\right),$$

so that the prior probability of any single component decreases as the maximum number of components we allow increases. Under this prior, the a priori expected value of K^* is such that

$$\lim_{K \to \infty} \mathbb{E}(K^*) = \sum_{i=1}^{n} \frac{\alpha}{\alpha + i - 1} \approx \alpha \log \frac{n + \alpha - 1}{\alpha}. \tag{13}$$

(The approximation exploits the fact that we can reinterpret the sum as a Riemman approximation to the integral, i.e., $\sum_{i=1}^{n} \frac{\alpha}{\alpha+i-1} \approx \int_1^n \frac{1}{x+\alpha-1} dx$.)

Equation 13 can be used to complete the prior elicitation process. In particular, we start with the number of observations n in the sample and a rough prior estimate of the number of clusters. We use these two values to solve (13) for the corresponding value of α. Then we set K to be "large" (in practice, values between 20 and 50 are usually enough). Finally, we run your algorithm and study the posterior distribution of K^*. If that posterior distribution puts substantial mass close to the maximum number of components allowed, we increase K and run your algorithm again. Otherwise, we use the resulting posterior samples for inference.

To illustrate our approach consider again the `galaxies` data set. In that case $n = 82$ and our prior information suggests that we expect around 6 clusters in this data. Hence, the value of α should satisfy

$$6 = \alpha \log \frac{82 + \alpha - 1}{\alpha}.$$

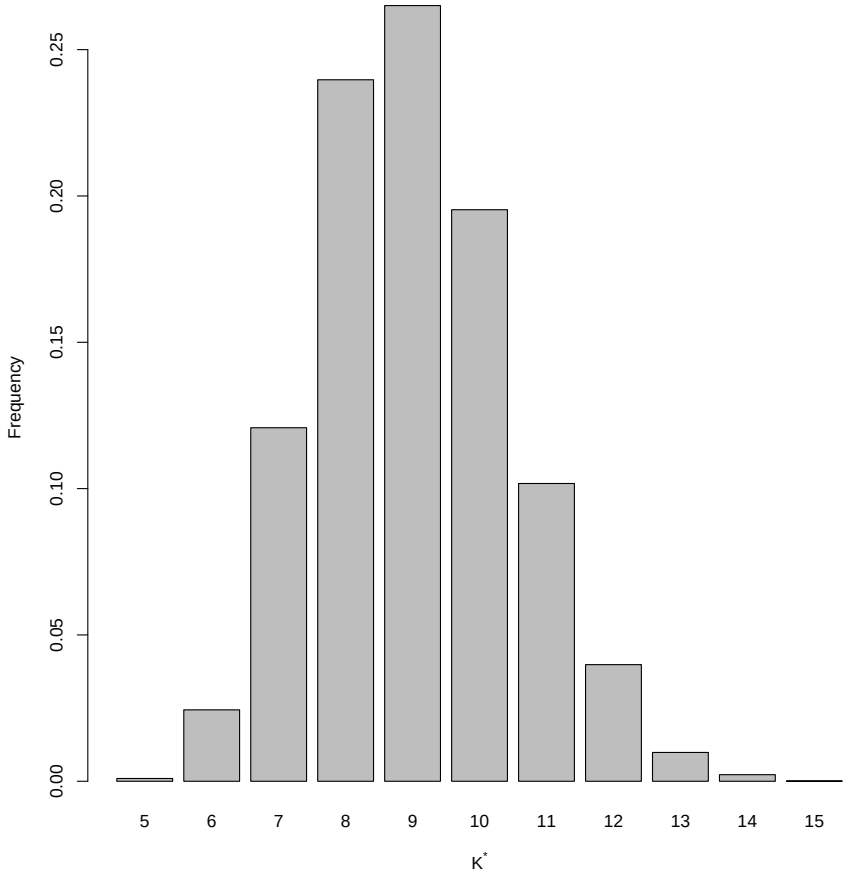

Figure 30: Posterior distribution on the number of occupied components K^* for the `galaxies` dataset under a Dirichlet prior for weights with hyperparameters $\alpha_1 = \cdots = \alpha_K = 1.5/K$ and $K = 30$.

This is a nonlinear equation, but it can be easily solve using R:

```
ff = function(alpha)   alpha*log((82 + alpha - 1)/alpha) - 6
alph = uniroot(ff, c(0.01, 20))
alph$root

## [1] 1.496393
```

Hence, we proceed first to fit a mixture model with a Dirichlet prior on the weights $\omega_1, \ldots, \omega_K$ with parameters $\alpha_1 = \alpha_2 = \ldots = \alpha_K = \alpha/K$ with $\alpha = 1.5$ and $K = 25$ (see the file `galaxies_bayesian.R`). Figure 30 shows the posterior distribution on the number of occupied components K^* under this model. Note that the Bayesian procedure suggests a mixture with between 8 and 10 components, with the posterior mode being 9. This is somewhat larger than the six suggested by the scientists who collected the data and the BIC procedure we discussed in Section 5.3.

The posterior distribution we just presented is unaffected by increases in the maximum number of components K (test this yourself by running the code on your own with, for example, $K = 60$). On the other hand, the posterior distribution is affected by changes in the hyperparameter α, but only modestly as long as the prior distribution remains reasonable. To illustrate this, Figure 31 shows the posterior distribution for $\alpha = 2$ (which, according

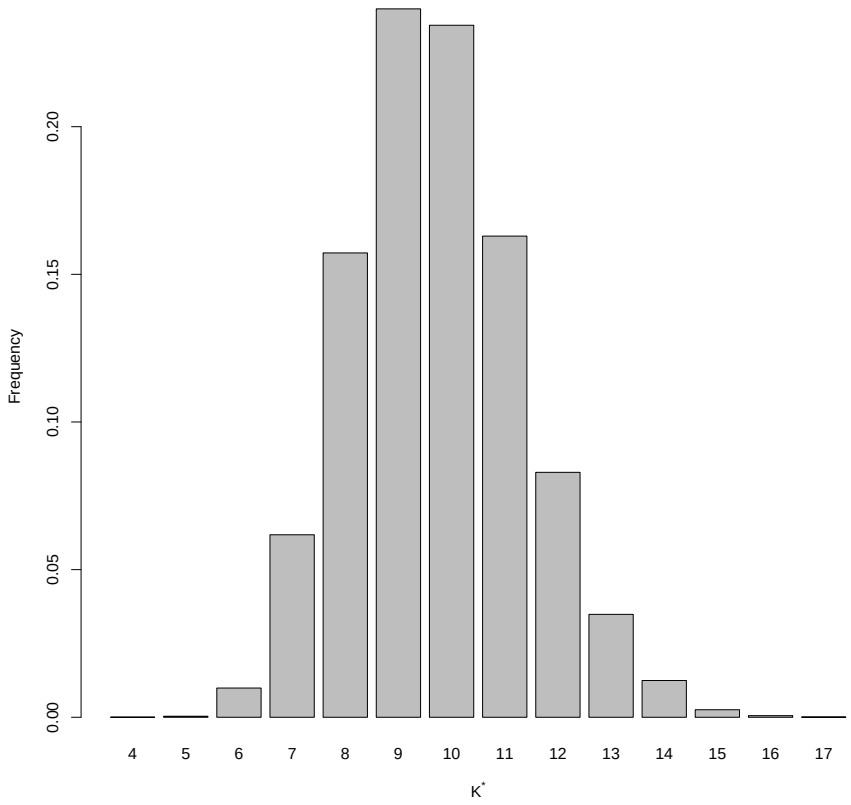

Figure 31: Posterior distribution on the number of occupied components K^* for the `galaxies` data set under a Dirichlet prior for weights with hyperparameters $a_1 = \cdots = a_K = 2/K$ and $K = 30$.

to equation (13), implies $E(K^*) \approx 7.5$ a priori). This posterior still favors between 8 and 10 components and the mode is still at 9 components, but 10 components receives comparatively more weight under this alternative prior. On the other hand, Figure 32 shows the posterior for $\alpha = 1$ (which corresponds to $E(K^*) \approx 4.4$ a priori). In this case the posterior suggests between 7 and 9 components, with the posterior mode being located at 8.

As a concluding remark, it should be reemphasized that this approach of using a larger number of components in the mixture and letting the data decide how many to use is feasible in a Bayesian context but not in a frequentist one. Indeed, recall from Section 5.2 that the MLEs for mixtures with empty components do not exist and lead to a number of numerical issues. On the other hand, as long as you use proper priors for $\theta_1, \ldots, \theta_K$, empty components are not an issue in a Bayesian context.

5.5 FULLY BAYESIAN INFERENCE ON THE PARTITION STRUCTURE

The number of components in the mixture is one potentially interesting summary of the posterior distribution over partitions, but it is not the only one. In this section we explore tools that can be used to provide more general summaries, including uncertainty measures.

One particular challenge associated with summarizing the posterior distribution in the context of mixture models is the lack of identifiability of the indicators $c_1, \ldots, c_n$ due to label switching. Hence, any summary measure we employ must necessarily be invariant to permutations in the labels. One

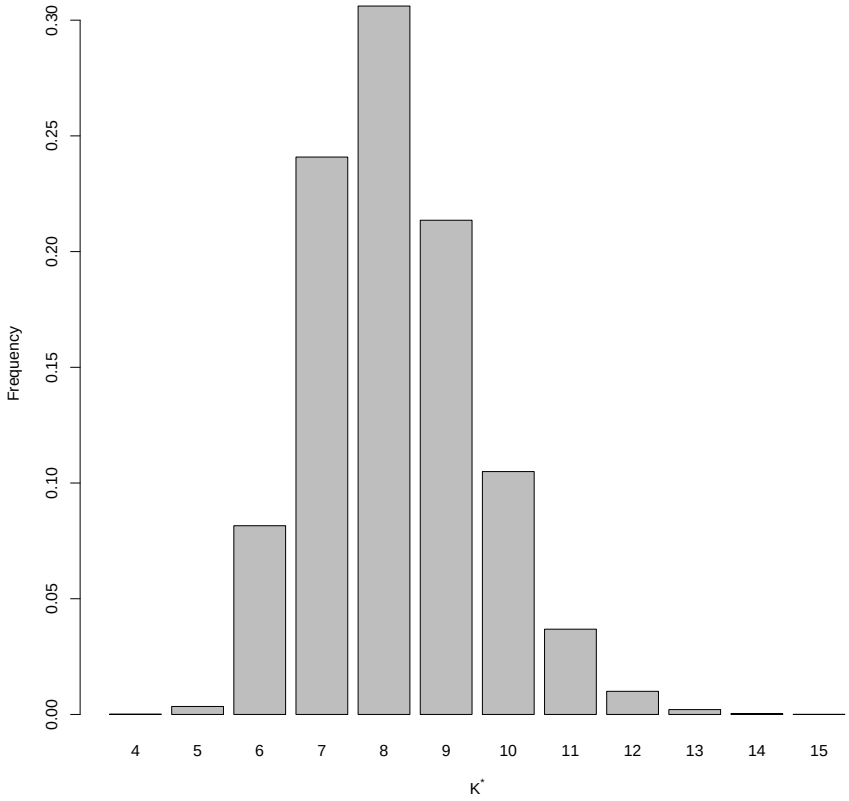

Figure 32: Posterior distribution on the number of occupied components K^* for the `galaxies` data set under a Dirichlet prior for weights with hyperparameters $\alpha_1 = \cdots = \alpha_K = 1/K$ and $K = 30$.

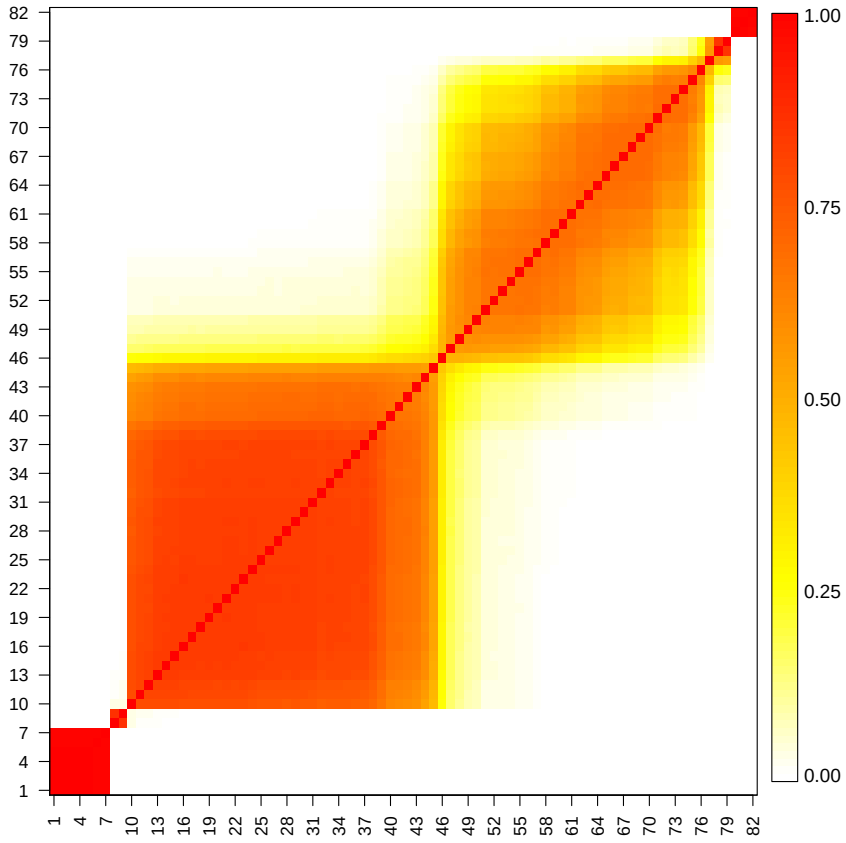

Figure 33: Heatmap of the pairwise co-clustering matrix for the `galaxies` dataset under a Dirichlet prior for weights with hyperparameters $\alpha_1 = \cdots = \alpha_K = 1.5/K$ and $K = 30$.

such metric that is quite useful is the $n \times n$ pairwise co-clustering matrix D with entries

$$D_{i,j} = \Pr(c_i = c_j \mid \text{data}).$$

The value of $\Pr(c_i = c_j \mid \text{data})$ can be easily approximated from the posterior samples as the frequency by which observations i and j are allocated to the same component.

To illustrate the construction and use of this summary, Figure 33 shows the pairwise co-clustering matrix associated with the `galaxies` data set. Note that, since observations are ordered in the sample according to their value, groups of observations that are often clustered together in the posterior samples form red squares in the graph. In particular, we can clearly see three tight, well-separated clusters (two located at the bottom left corner of the graph that include 7 and 2 observations, respectively, and one at the top right corner that includes 3 observations). These three clusters correspond to the modes at velocities around 10,000, 16,000 and 32,000 in the kernel density estimate from Figure 34. We also observe at least three additional red blocks in the heatmap, although they are not as well defined. In particular, the model is quite uncertain about the clustering structure associated with observations 49 to 76. The model appears to provide support for various alternative explanations of the data, from a single big cluster in this area, to up to 3 smaller ones.

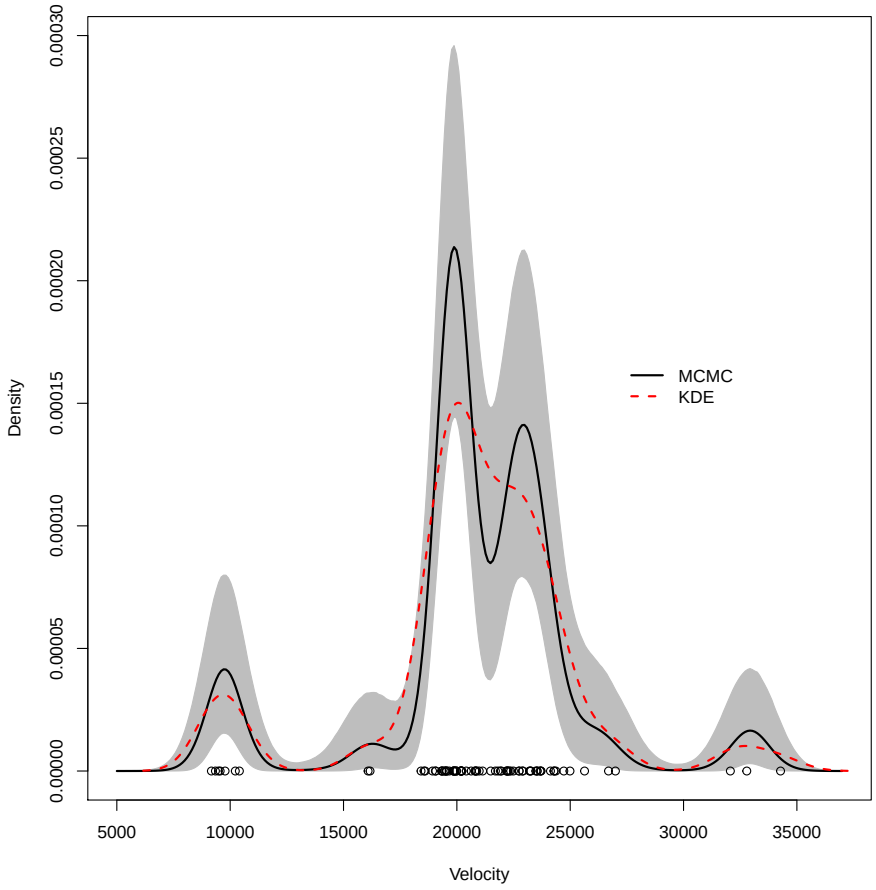

Figure 34: Bayesian kernel density estimated for the `galaxies` dataset under a Dirichlet prior for weights with hyperparameters $a_1 = \cdots = a_K = 1.5/K$ and $K = 30$.

It is important to reiterate that the pairwise incidence plot looks so appealing in this case because we are working with univariate observations that have been sorted in increasing order. In more general settings (and, particularly, when working with multivariate observations for which there is no natural ordering) the graph might be unreadable unless the data is reorganized. One simple approach is to use any posterior sample and make sure that observations that are clustered together in that sample appear in adjacent positions in the heatmap.

The pairwise co-clustering matrix not only provides us with a visual tool to quantify uncertainty in the clustering structure, but it can also be used to generate point estimates of the partition structure of the data (and, as a byproduct, an alternative point estimate for the number of clusters in the sample). As is common in Bayesian approaches to deriving point estimators, we start with a loss function, which in this case takes the form:

$$L(c, \hat{c}) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \gamma_1 \mathbb{1}(c_i = c_j)\mathbb{1}(\hat{c}_i \neq \hat{c}_j) + \gamma_2 \mathbb{1}(c_i \neq c_j)\mathbb{1}(\hat{c}_i = \hat{c}_j).$$

This loss function includes two terms. The first terms counts the number of pairs of observations that the estimator classifies as being in separate clusters when in reality they belong together, and penalizes each of these instances by a common factor γ_1. Similarly, the second term counts the number of pairs of observations that the estimators assigns to the same cluster while in truth they belong to different ones, and penalizes it by a constant γ_2.

Since the true partition c is unknown, we compute the expected utility with respect to the posterior distribution,

$$\begin{aligned}
L^*(\hat{c}) &= E\Big[\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \gamma_1 \mathbb{1}(c_i = c_j)\mathbb{1}(\hat{c}_i \neq \hat{c}_j) \\
&\qquad\qquad + \gamma_2 \mathbb{1}(c_i \neq c_j)\mathbb{1}(\hat{c}_i = \hat{c}_j) \Big] \\
&= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \gamma_1 E\left[\mathbb{1}(c_i = c_j)\right] \left(1 - \mathbb{1}(\hat{c}_i = \hat{c}_j)\right) \\
&\qquad\qquad + \gamma_2 \left(1 - E\left[\mathbb{1}(c_i = c_j)\right]\right) \mathbb{1}(\hat{c}_i = \hat{c}_j) \\
&= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \gamma_1 \Pr(c_i = c_j \mid \text{data}) - \gamma_1 \Pr(c_i = c_j \mid \text{data})\mathbb{1}(\hat{c}_i = \hat{c}_j) \\
&\qquad\qquad + \gamma_2 \mathbb{1}(\hat{c}_i = \hat{c}_j) - \gamma_2 \Pr(c_i = c_j \mid \text{data})\mathbb{1}(\hat{c}_i = \hat{c}_j) \\
&= \gamma_1 \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \Pr(c_i = c_j \mid \text{data}) + \\
&\qquad (\gamma_1 + \gamma_2) \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \left\{ \frac{\gamma_2}{\gamma_1 + \gamma_2} - \Pr(c_i = c_j \mid \text{data}) \right\} \mathbb{1}(\hat{c}_i = \hat{c}_j)
\end{aligned}$$

Now, note that the term $\gamma_1 \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \Pr(c_i = c_j \mid \text{data})$ in $L^*(\hat{c})$ does not depend on the point estimator $\hat{c}$ and can therefore be treated as a constant for any given data set. Furthermore, the point at which a function is maximized does not change if we add and/or multiply the the function by positive constant, and multiplying a -1 just changes the minimization

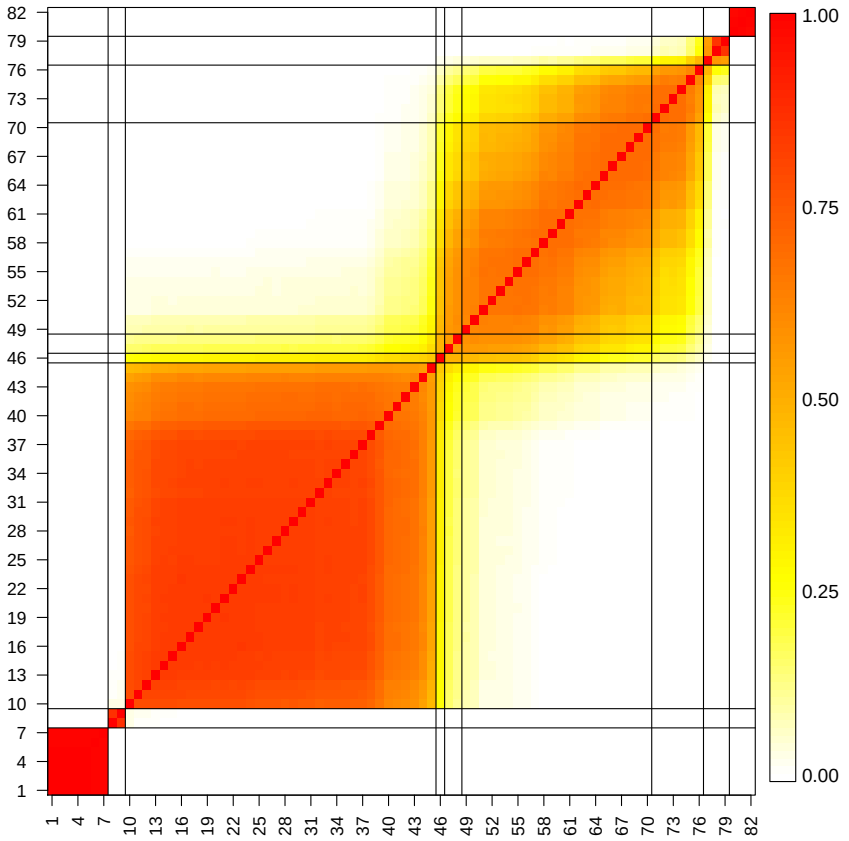

Figure 35: Heatmap plot for the co-clustering matrix and optimal clustering under $\bar{D} = 0.5$ for the `galaxies` dataset. Black lines identify the limits of each cluster under the optimal partition.

problem into a maximization one (and viceversa). Hence, minimizing $L^*(\hat{c})$ is equivalent to maximizing:

$$L^{**}(\hat{c}) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \left\{ D_{i,j} - \frac{\gamma_2}{\gamma_1 + \gamma_2} \right\} \mathbb{1}(\hat{c}_i = \hat{c}_j). \tag{14}$$

Since both γ_1 and γ_2 need to be non-negative, the ratio $\tilde{D} = \frac{\gamma_2}{\gamma_1+\gamma_2}$ lives in the $[0, 1]$ interval. $\tilde{D}$ acts as a threshold that, roughly speaking, controls how often two data points need to be observed in the same cluster in the posterior distribution before they are put together by the point estimator. Note that setting $\bar{D} = 1$ (which corresponds to $\gamma_1 = 0$) means that no penalty is incurred if we wrongly split clusters. This choice leads to an optimal cluster allocation in which each observation is its own cluster, no matter the value of the matrix D. For analogous reasons, $\bar{D} = 0$ (which corresponds to $\gamma_2 = 0$) leads to a point estimate in which all observations are allocated to a single cluster. Solutions for other values of $\bar{D}$ need to be obtained numerically, typically using an iterative algorithm that reallocates one observation at a time. This algorithm typically converges to a global maximum if the initial configuration corresponds to one of the configurations sampled by the MCMC algorithm. A common choice of $\bar{D}$ is $\bar{D} = 1/2$, which corresponds to weighting both types of clustering errors equally.

Code for computing and maximizing L^{**} in the context of the `galaxies` data set can be found at the end of the file `galaxies_bayesian.R`. Figure 35 presents the optimal clustering structure for $\bar{D} = 1/2$ superimposed on the same heatmap from Figure 34. In this case, the optimal partition involves 9 clusters. Four of these correspond to relatively small but well-defined clusters located at the two extremes of the velocity range. The approach also identifies observations 10 to 45 as a single cluster. However, it breaks the set that comprises observations 46 to 76 into 4 groups: a large one (observations 49 to 70), a medium-sized one (observations 71 to 76), and two tiny ones (one comprising observations 47 and 48, and one comprising only observation 46).

A few parting comments on the use of point estimators generated by optimizing (14) are in order. First, note that the optimal partition may or may not correspond to one of the samples from the posterior distribution generated by the MCMC algorithm. Secondly, larger values of $\bar{D}$ tend to lead to a smaller number of clusters. However, the structure of the optimization problem is such that we can partition the unit interval into an exhaustive and non-overlapping set of intervals $[0, \lambda_1), [\lambda_1, \lambda_2), \ldots, [\lambda_N, 1]$ (typically with $N < n$) such that the optimal partition is constant on each interval $[\lambda_j, \lambda_{j+1})$. Moreover, there is usually some level of continuity in these partitions, in the sense that partitions associated with adjacent intervals are identical except for a small number of clusters, which get split/merged. This simplifies the process of assessing the sensitivity of the solution to the choice of $\bar{D}$. As an illustration, Figures 36 and 37 show the optimal partitions for the `galaxies` data set for $K = 0.4$ and $K = 0.53$, respectively. When comparing Figure 36 with 35 we note that the number of clusters in the optimal partition has gone down from 9 to 8, due exclusively to observations 46 to 76 now being partitioned into 3 clusters rather than 4. Similarly, when comparing Figure 37 with 35 we see that the optimal partition involves now 10 clusters, which are obtained by reallocating the four old clusters comprising observations 46 to 76 into five clusters.

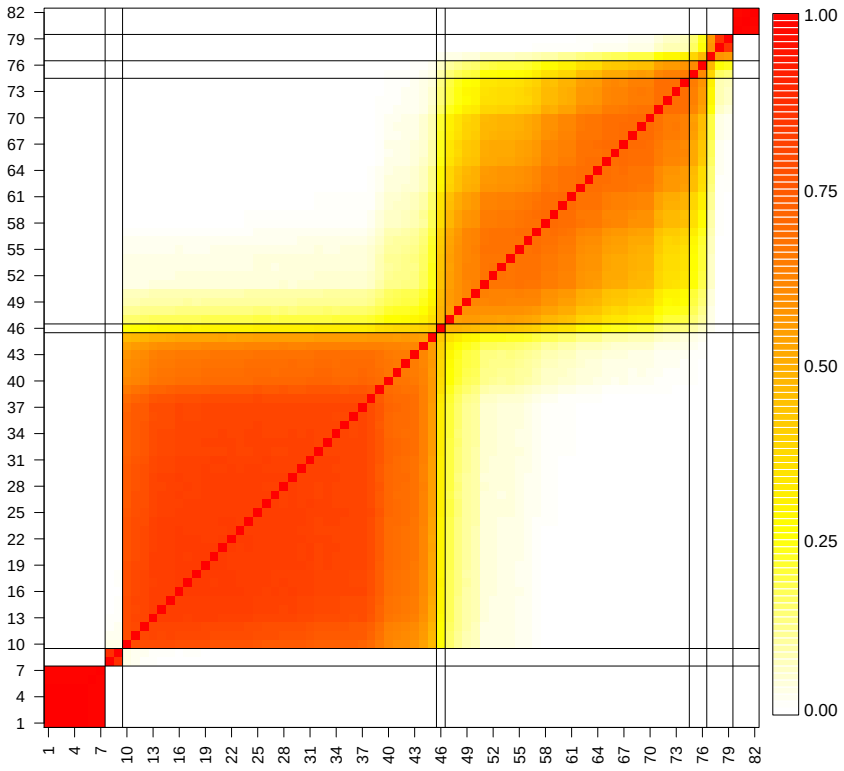

Figure 36: Heatmap plot for the co-clustering matrix and optimal clustering under $\bar{D} = 0.4$ for the `galaxies` data set. Black lines identify the limits of each cluster under the optimal partition.

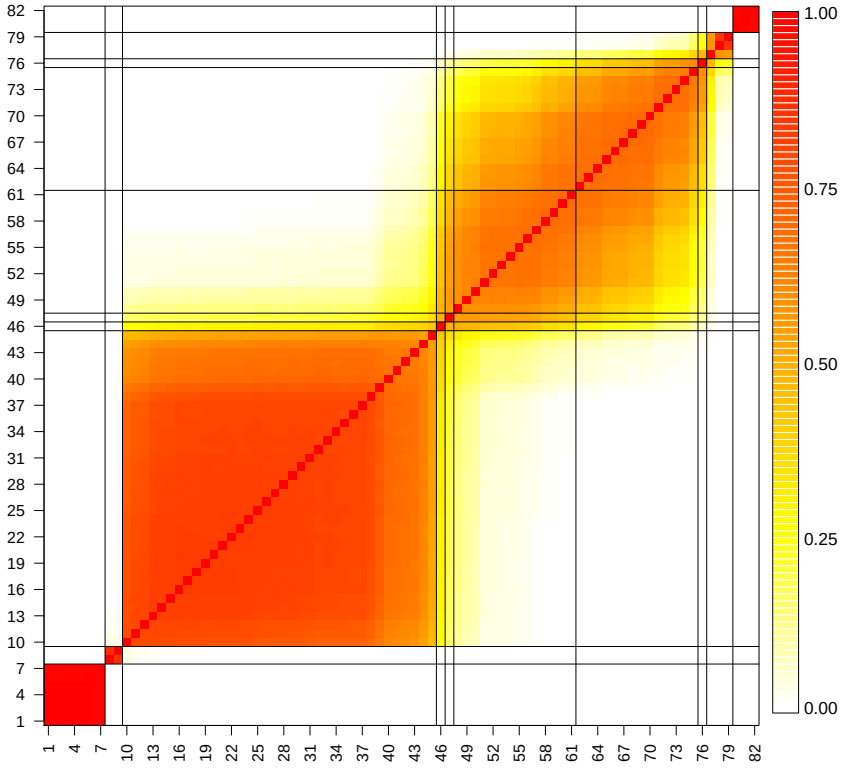

Figure 37: Heatmap plot for the co-clustering matrix and optimal clustering under $\bar{\mathrm{D}} = 0.53$ for the `galaxies` data set. Black lines identify the limits of each cluster under the optimal partition.