CARLETON UNIVERSITY

SCHOOL OF
MATHEMATICS AND STATISTICS

HONOURS PROJECT

TITLE: NON-PARAMETRIC BAYESIAN ANALYSIS WITH DIRICHLET PROCESS

AUTHOR: XINYI XU

SUPERVISOR: DR. SONG CAI

DATE: APRIL 2019

Non-parametric Bayesian Analysis with Dirichlet Process

Xinyi Xu

Contents

Abstract

In this project, I will focus on one of the non-parametric techniques, non-parametric hierarchical Bayesian models based on the Dirichlet process. First, I will summarize the basic ideas of the Dirichlet distribution and then the Dirichlet process, including the definitions and some key properties. Then I will discuss about model specification and the computational techniques, following Escobar and West (1995). Escobar and West (1995).

1 Introduction to Bayesian

1.1 Bayesian Inference

In traditional Bayesian inference, a set of data, $X = \{X_1, X_2, \cdots, X_n\}$ is usually assumed to follow some distribution, say $X_i \sim f(X_i|\theta)$, with a given parameter θ, which is further assumed to follow some distribution, i.e., $\theta \sim f(\theta)$. Here f is used as a general notation for density function of its argument. Then $f(\theta)$ is the distribution of the parameter without taking into account the observed data, called the **prior distribution**, while $f(\theta|X)$ is the distribution of the parameter given the observed data, called **posterior distribution**. Using the Bayes' rule, we can obtain the posterior distribution as:

$$f(\theta|X) = \frac{f(\theta, X)}{f(X)} = \frac{f(X|\theta)f(\theta)}{f(X)}, \tag{1}$$

where $f(X)$ is the **marginal likelihood** of the observed data, which can be obtained from

$$f(X) = \int f(X|\theta)f(\theta)d\theta. \tag{2}$$

Since $f(X)$ is a normalizing constant for $f(\theta|X)$, so we can say

$$f(\theta|X) \propto f(X|\theta)f(\theta). \tag{3}$$

The **predictive distribution**(or sometimes is called **posterior predictive distribution**), $f(X_{n+1}|X)$ tells the distribution of a new observed data from the same data distribution given the existed observed data and is given by

$$f(X_{n+1}|X) = \int f(X_{n+1}|\theta, X)f(\theta|X)d\theta. \tag{4}$$

1.2 Non-parametric Bayesian

In parametric Bayesian analysis, the data distribution $f(X_i|\theta)$ is assumed to belong to a specified distribution family with unknown parameter θ. When the assumed distribution family of $f(X_i|\theta)$ deviates from the true underlying data distribution, results of the data analysis would be erroneous. A simple idea to increase the flexibility of Bayesian analysis is to treat the data distribution itself as a parameter and put a prior distribution on it. The implementation of this idea is called the non-parametric Bayesian analysis. In this project, I study the Dirichlet process prior of distributions for non-parametric Bayesian analysis and its application in Bayesian Hierarchical models. Dirichlet process prior combined with hierarchical model makes Bayesian analysis flexible and useful for analyzing complex data.

The rest of the paper is organized as follows. Chapter 3 introduces Dirichlet distribution. Chapter 4 introduces Dirichlet process and summarizes related key results for Bayesian analysis. Chapter 5 give details of non-parametric Bayesian analysis using Dirichlet process and Bayesian hierarchical model. An application using synthetic data is presented in Chapter 6. The relevant computer code is included in the Appendix.

2 Dirichlet Distribution

To understand the Dirichlet process, we start from the finite case, the Dirichlet distribution.

Consider a random variable X with finite support $\{X_1, X_2, \cdots, X_k\}$, which follows some unspecified discrete distribution P where P has the probability mass function (PMF), $p_i = Pr(X_i = x_i)$, $i = 1, \cdots, k$. The space of the data distribution can be parameterized by $\mathbb{F} = \{P = (p_1, p_2, \cdots, p_k) : p_i \geq 0 \text{ for } i = 1, \cdots, k, \text{ and } \sum_{i=1}^{k} p_i = 1\}$.

Consider for n independent trials, each leads to a success for exactly one of k categories and each category has a fixed success probability, p_i. The multinomial distribution describes the probability that for each category, a particular number of success is achieved. The probability measure on X is essentially a multinomial distribution of size one. And the Dirichlet distribution is the conjugate prior of multinomial distribution.

Definition 2.1 *Let $P = (p_1, p_2, \cdots, p_k) \in \mathbb{F}$ where $\mathbb{F} = \{P = (p_1, p_2, \cdots, p_k) : p_i \geq 0$ for $i = 1, \cdots, k$, and $\sum_{i=1}^{k} p_i = 1\}$. Let $\alpha = (\alpha_1, \cdots, \alpha_k), \alpha_i \in \mathbb{R}$ and $\alpha_i > 0$ for $i = 1, \cdots, k$. Then P has a Dirichlet distribution with parameter α, denoted as $P \sim Dir(\alpha)$, if its density is:*

$$f(P) = \frac{\Gamma\left(\sum_{i=1}^{k} \alpha_i\right)}{\prod_{i=1}^{k} \Gamma(\alpha_i)} \prod_{i=1}^{k} p_i^{\alpha_i - 1} \tag{5}$$

where $\Gamma(\cdot)$ is the gamma function.

Some properties of Dirichlet distribution are discussed below.

Suppose $X_1, X_2, \cdots, X_n$ have finite support $\{x_1, x_2, \cdots, x_k\}$ and are independently and identically (i.i.d) distributed with distribution P, for some $P = (p_1, \cdots, p_k) \in \mathbb{F}$. Let $P \sim Dir(\alpha)$, for some hyperparameter $\alpha = (\alpha_1, \cdots, \alpha_k)$. Let $\alpha_0 = \sum_{i=1}^{k} \alpha_i$.

Property 2.1 *The marginal distribution of each X_i is*

$$\begin{aligned} f(X_i) &= \int f(X_i|P)Dir(\alpha)dP \\ &= (\frac{\alpha_1}{\alpha_0}, \cdots, \frac{\alpha_k}{\alpha_0}) = \bar{\alpha} \end{aligned}$$

Also $E(P) = \bar{\alpha}$. It tells that $f(X_i)$ is discrete with support $\{x_1, x_2, \cdots, x_k\}$

Property 2.2 *The posterior distribution of P given the i.i.d data is:*

$$P|X_{1:n} \sim \prod_{i=1}^{k} p_i^{\alpha_i - 1 + n_i} = Dir(\alpha + \sum_{j=1}^{n} \delta_{X_i}) \tag{6}$$

where n_i is the number that $X_j = x_i$ for $j = 1, \cdots, n$ and $\delta_{X_i} = (\delta_{X_i}(1), \cdots, \delta_{X_i}(k))$ with $\delta_{X_i}(l) = 1$ if $X_i = x_l$ and 0 otherwise.

This property shows that Dirichlet distribution is a conjugate prior when the support of X_i is finite.

Property 2.3 *The predictive distribution of X_{n+1} given the i.i.d data is*

$$(X_{n+1}|X_{1:n}) \sim \frac{\alpha + \sum_{j=1}^{n} \delta_{X_i}}{\alpha_0 + n} \tag{7}$$

This can be derived from the equation(4), where $f(X_{n+1}|P, X_{1:n})$ is essentially $f(X_{n+1}|P)$ here due to conditional independence of the data given P.

3 Dirichlet Process

We have discussed the case where the support of data is finite, i.e. $\{x_1, x_2, \cdots, x_k\}$. Now, we consider the case where the support of the data is $\mathbb{R}$. In this case, a generalization of Dirichlet distribution, the Dirichlet process, will be used.

Ferguson (1973) showed the existence of a prior distribution on distributions, called Dirichlet process. Ghosh and Ramamoorthi (2003) further proved the uniqueness of the Dirichlet process. Due to them, we introduce the existence theorem:

Theorem 3.1 *Let α be a positive real number and H be a probability distribution. There exists a random distribution P called Dirichlet process such that for every finite measurable partition $B_1, \cdots, B_k$ of $\mathbb{R}$,*

$$P(B_1), \cdots, P(B_k) \sim Dir\left(\alpha H(B_1), \cdots, \alpha H(B_k)\right).$$

This theorem shows the existence of the Dirichlet process prior. Instead of construction of a prior for a parameter, it is possible to construct a prior for a distribution.

A Dirichlet process with given α and H is denoted as $DP(\alpha, H)$. Suppose that $\pi_1, \cdots, \pi_n$ are i.i.d random variables coming from some distribution G, and let G be distributed as Dirichlet Process with some positive number α and some base probability distribution H. We can write:

$$\{\pi_i\}_{i=1}^{n} \overset{i.i.d.}{\sim} G$$
$$G \sim DP(\alpha, H)$$

As stated before, the finite-support Dirichlet sampling can be treated as a Multinomial distribution. For the infinite-support Dirichlet sampling, we pick up a distribution instead of a function. Here, the distribution H is a base distribution to be specified and α is a scaling parameter with a positive real value indicating the strength of prior information. A small α represents a weak prior belief, while a large α represent a strong prior belief.

Generalizing the properties of Dirichlet distribution, we can further make the following statements about Dirichlet process with the settings given above.

Corollary 3.1 *Denote* $\pi^{(i)} = \{\pi_1, \cdots, \pi_{i-1}, \pi_{i+1}, \cdots, \pi_n\}$. *Then, the conditional distribution for* $(\pi_i|\pi^{(i)})$ *is*

$$(\pi_i|\pi^{(i)}) \sim \alpha a_{n-1} H(\pi_i) + a_{n-1} \sum_{j=1, j\neq i}^{n} \delta_{\pi_j}(\pi_i). \tag{8}$$

where $\delta_{\pi_j}(\pi_i)$ *is the Kronecker delta function, i.e.* $\delta_{\pi_j}(\pi_i) = 1$ *if* $\pi_j = \pi_i$ *and* $\delta_{\pi_j}(\pi_i) = 0$ *otherwise, and* $a_n = 1/(\alpha + n)$

Corollary 3.2 *The posterior distribution of G given* $\pi_1, \cdots, \pi_n$ *is*

$$(G|\pi_1, \cdots, \pi_n) \sim DP(\alpha + n, \frac{\alpha}{\alpha + n} H + \frac{\sum_{i=1}^{n} \delta_{\pi_i}}{\alpha + n}). \tag{9}$$

This can be derived from Property 2.2. We can see that the Dirichlet process is a conjugate prior under the assumed model.

Corollary 3.3 *The predictive distribution of* $(\pi_{n+1}|\pi_{1:n})$ *is given by*

$$(\pi_{n+1}|\pi_{1:n}) \sim \alpha a_n H(\pi_{n+1}) + a_n \sum_{i=1}^{n} \delta_{\pi_i}(\pi_{n+1}). \tag{10}$$

or we can say

$$\pi_{n+1}|\pi_{1:n} = \begin{cases} \pi_i \text{ with probability } a_n \\ \text{new value from } H \text{ with probability } \alpha a_n \end{cases}$$

Suppose there are $k\,(< n)$ *distinct values among* $\pi_1, \cdots, \pi_n$, *denoted as* $\pi_1^*, \cdots, \pi_k^*$, *and there are* n_j *occurrences of* π_j^* *for* $j = 1, \cdots, k$, *with* $n_1 + \cdots + n_k = n$. *Then the predictive distribution is actually equivalent to:*

$$(\pi_{n+1}|\pi_{1:n}) \sim \alpha a_n H(\pi_{n+1}) + a_n \sum_{j=1}^{k} \delta_{\pi_j^*}(\pi_{n+1}) \tag{11}$$

Usually the size of "active" parts, k, is much smaller than n. Antoniak (1974) concluded a theorem about the prior distribution for k, which mainly depends on α. In this paper, the details about k will not be discussed.

4 Estimation

4.1 Normal mixture models and prediction

The density estimation of the predictive distribution can be proceed with the Markov chain Monte Carlo approximation. Here I will go through the approximation described by Escobar and West (1995) for normal mixture models.

Suppose that data $X_1, \cdots, X_n$ are conditionally independent and normally distributed given parameters $\pi_1, \cdots, \pi_n$, each $\pi_i = (\mu_i, V_i)$ for $i = 1, \cdots, n$ coming from some prior distribution $G(\cdot)$. Assume that G has a Dirichlet process defined by a positive scalar α and G_0, where G_0 is the prior expectation of $G(\cdot)$. We can write this hierarchical model as follows:

$$\begin{aligned} \{X_i|\pi_i\}_{i=1}^n &\sim N(\mu_i, V_i), \text{ independently} \\ \{\pi_i\}_{i=1}^n &\overset{i.i.d.}{\sim} G \\ G &\sim DP(\alpha G_0) \end{aligned}$$

We focus on estimating the predictive density, $P(X_{n+1}|X_{1:n})$, of a future observation X_{n+1} given the observed data $X_1, \cdots, X_n$.

First of all, we need to specify the prior mean, G_0. Usually we choose a conjugate distribution for convenience. Here we consider the inverse-gamma distribution to get G_0 a conjugate prior:

$$\begin{aligned} V_i &\sim Inverse - Gamma(s/2, S/2) \\ (\mu_i|V_i) &\sim N(m, \tau V_i) \end{aligned}$$

where s, S, m and τ should also be specified. The specification of these parameters will be discussed later.

We introduced the conditional prior, $(\pi_i|\pi^{(i)})$, in corollary 3.1, and so the condi-

tional posterior for $(\pi_i|\pi^{(i)}, X_{1:n})$ is then given by:

$$(\pi_i|\pi^{(i)}, X_{1:n}) \sim q_0 G_i(\pi_i) + \sum_{j=1, j\neq i}^{n} q_j \delta_{\pi_j}(\pi_i) \tag{12}$$

where $G_i(\pi_i)$ is the bi-variate normal/inverse-gamma distribution with $\pi_i = (\mu_i, V_i)$ that

$$V_i \sim IG(\frac{1+s}{2}, \frac{S_i}{2}) \text{ with } S_i = S + \frac{(x_i - m)^2}{1+\tau}$$
$$(\mu_i|V_i) \sim N(y_i, YV_i) \text{ with } Y = \frac{\tau}{1+\tau} \text{ and } y_i = \frac{m + \tau x_i}{1+\tau}$$

and q_j are the constant weights for $j = 1, \cdots, n, j \neq i$ with $q_0 + \cdots + q_{i-1} + q_{i+1} + \cdots + q_n = 1$.

$$q_0 \propto \alpha \mathbf{c}(\mathbf{s}) \frac{[\mathbf{1} + (\mathbf{x_i} - \mathbf{m})^{\mathbf{2}}/(\mathbf{sM})]^{-(\mathbf{1+s})/\mathbf{2}}}{\mathbf{M^{1/2}}}$$
$$\text{with } \mathbf{c}(\mathbf{s}) = \mathbf{\Gamma}(\frac{\mathbf{1+s}}{\mathbf{2}})\mathbf{\Gamma}(\frac{\mathbf{s}}{\mathbf{2}})^{-\mathbf{1}}\mathbf{s}^{-\mathbf{1/2}}$$

and

$$q_j \propto \exp\{-\frac{(x_i - \mu_j)^2}{2V_j}\}(2V_j)^{-2/2}.$$

Then the predictive distribution is given by

$$f(X_{n+1}|X_{1:n}) = \int f(X_{n+1}|X_{1:n},\, \pi_{1:n},\, m,\, \tau) dP(\pi_{1:n},\, m,\, \tau|X_{1:n}). \tag{13}$$

We know that,

$$f(X_{n+1}|X_{1:n},\, \pi_{1:n},\, m,\, \tau) = f(X_{n+1}|\pi_{1:n},\, m,\, \tau) \tag{14}$$

In addition,

$$f(X_{n+1}|\pi_{1:n},\, m,\, \tau) = \int f(X_{n+1}|\pi_{n+1}) dP(\pi_{n+1}|\pi_{1:n},\, m,\, \tau). \tag{15}$$

Now, $f(X_{n+1}|\pi_{n+1})$ is the normal distribution, and $P(\pi_{n+1}|\pi_{1:n},\, m,\, \tau)$ can be obtained using Corollary 3.3. Thus,

$$(X_{n+1}|\pi_{1:n},\, m,\, \tau) \sim \alpha a_n T_s(m, M) + a_n \sum_{j=1}^{k} n_j N(\mu_j^*, V_j^*), \tag{16}$$

where $T_s(m, M)$ is the Student-t distribution with s degrees of freedom, mode m and scale factor $M^{1/2}$ and $M = (1+\tau)S/s$.

4.2 Computing predictive distribution

To carry out computation, we still need to specify the s, S, m and τ. Consider a higher level of independent priors on m and τ with some specified hyperparameters a, A, w, and W, of the form:

$$m \sim N(a, A)$$
$$\tau \sim IG(w/2, W/2)$$

and m and τ are both conditionally independent of data $X_{1:n}$. Then

$$(m|\tau, \pi) \sim N((1-x)a + x\bar{V}\sum_{j=1}^{k}((V_j^*)^{-1}\mu_j^*), x\tau\bar{V}) \tag{17}$$

$$(\tau|m, \pi) \sim IG(\frac{w+k}{2}, \frac{W+K}{2}) \tag{18}$$

where $x = A/(A + \tau\bar{V}), \bar{V} = 1/\sum_{j=1}^{k}(V_j^*)^{-1}$ and $K = \sum_{j=1}^{k}(\mu_j^* - m)^2/V_j^*$.

τ is a very important hyperparameter in the approximation and we will see the effect later in the application.

Here we state the approximation algorithm following the description by Escobar and West (1995):

Algorithm

Step 1. Initialize m, τ, w, W and π (i.e. μ and V)

Step 2. Update m and τ

Step 3. Update μ_i and V_i

Step 4. Return to Step 2, and iterate until convergence

Let N be the sample size. Then the approximate predictive distribution can be obtained by

$$f(X_{n+1}|X_{1:n}) \sim \frac{1}{N}\sum_{r=1}^{N} f(X_{n+1}|\pi(r), m(r), \tau(r)). \tag{19}$$

5 Application to Synthetic Data

Here we implement the algorithm described above and apply to synthetic data. We will show the effect of the two hyperparameters- the strength α and mean of τ. First, we generate a sample of size 150 from the normal mixture distribution

$$0.2N(3,4) + 0.3N(15,1) + 0.5N(22,6). \tag{20}$$

The histogram of the generated sample along with the density curve of this distribution are shown in Figure 1.

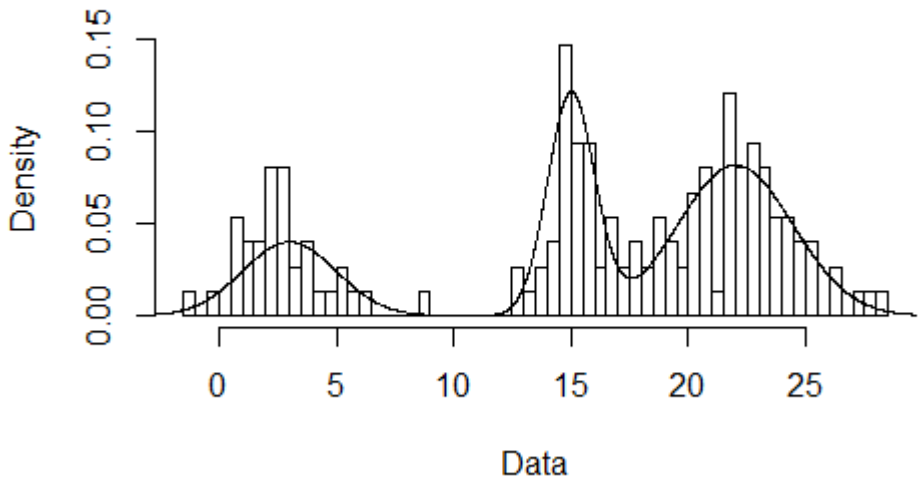


Figure 1: The histogram of the generated sample and density curve of the mixture model(21)

Assume that we only know a rough range of the data $(-10, 35)$, and the variance of the data is around 3. We specify hyperparamters in the hierarchical model as follows:

For m, the mean of μ_i, we want it to cover the data range, so $a = 12.5$, $A = 8^2$, i.e., $m \sim N(12.5, 8^2)$. For τ, since we have a weak belief of it , we specify a small

w ($w = 1$) and a large W ($W = 100$ and also try a way large $W = 200$) i.e., $\tau \sim IG(1/2, 50)$ and $\tau \sim IG(1/2, 100)$, so that τ tends to be large. For V_i, we want it to be around 3, so we specify $s = 1/2$, $S = 8$, i.e., $V_i \sim IG(1/4, 4)$. Finally, since we have weak information about the data, we experiment with a few small α values: α, $\alpha = 2$, $\alpha = 1$ and $\alpha = 0.5$.

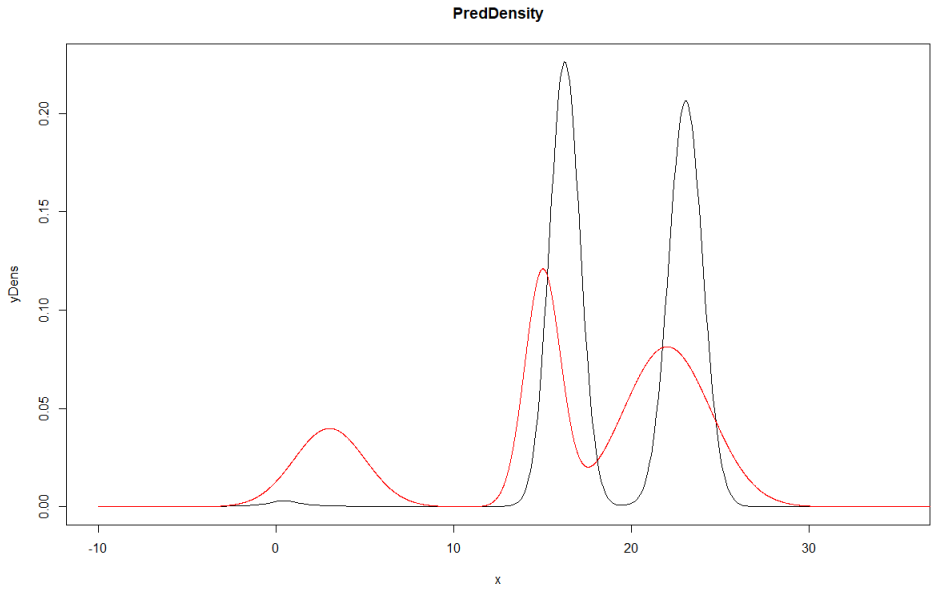
We run the Gibbs sampler with 20,000 iterations, with the first 10,000 iterations as the burn-in period. Using the predictive distribution given in Section 4.2, the frequency histograms and density graph are shown in the figures below. The figures will show the comparison between the density of approximated predictive distribution and that of the underlying true distribution, as well as the sampling histogram of the predictive distribution.

The approximated predictive distribution with different value combinations of W and α are shown below. Each figure contains three sub-graphs, including a comparison between the density curve of the approximated predictive distribution (black curve) and that of the underlying generating distribution (red curve), a histogram of predictive data , and a smoothed density curve of the predictive distribution. The later two graphs are generated according to Equation (16) and Equation (19), and the detailed generating code is included in the Appendix.

When $W = 100$ and $\alpha = 2$, Figure 2 shows that the predictive distribution captures two modes of the underlying distribution with some slight deviation and a very weak mode around 1. For the cases of $\alpha = 1$ and $\alpha = 0.5$ shown in Figure 3 and Figure 4 respectively, only the middle mode centered around 15 is captured. When we change W from 100 to 200, the graphs with different values of α are shown in Figure 5, 6 and 7 with the same format as above. We can see that the predictive distribution with $W = 200$ captures the middle mode around 15 with a slight shift from the true mode.

In general, the middle mode centered at 15 is the most likely one to be captured and, with some W values, the right mode centered at 22 may be captured. However, we did not find any case where the predictive distribution clearly captures the left

mode centered at 3.



Black curve: approximated predictive distribution density curve

Red curve: underlying generating distribution density curve

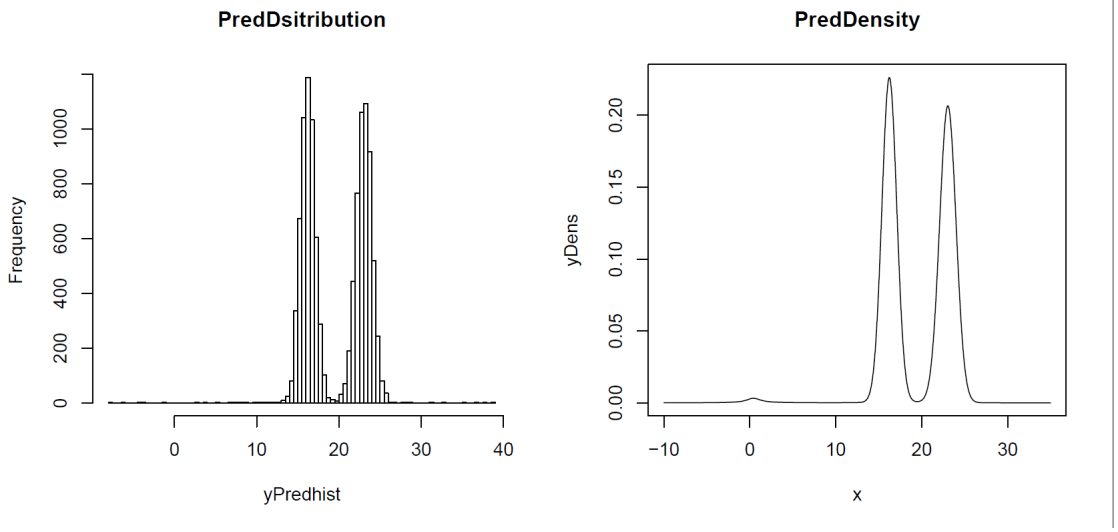
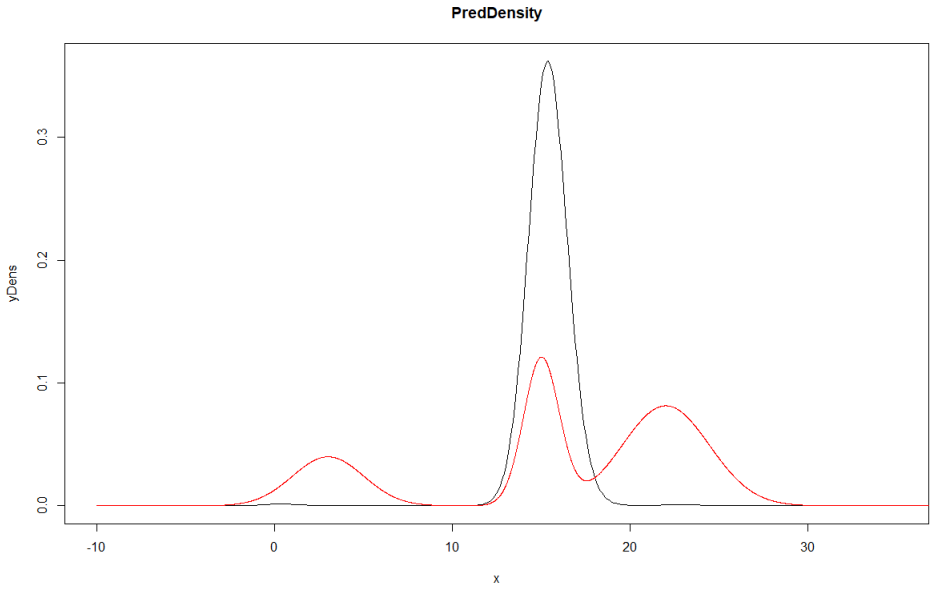


Figure 2: Approximated predictive distribution with $W = 100$ and $\alpha = 2$



Black curve: approximated predictive distribution density curve

Red curve: underlying generating distribution density curve

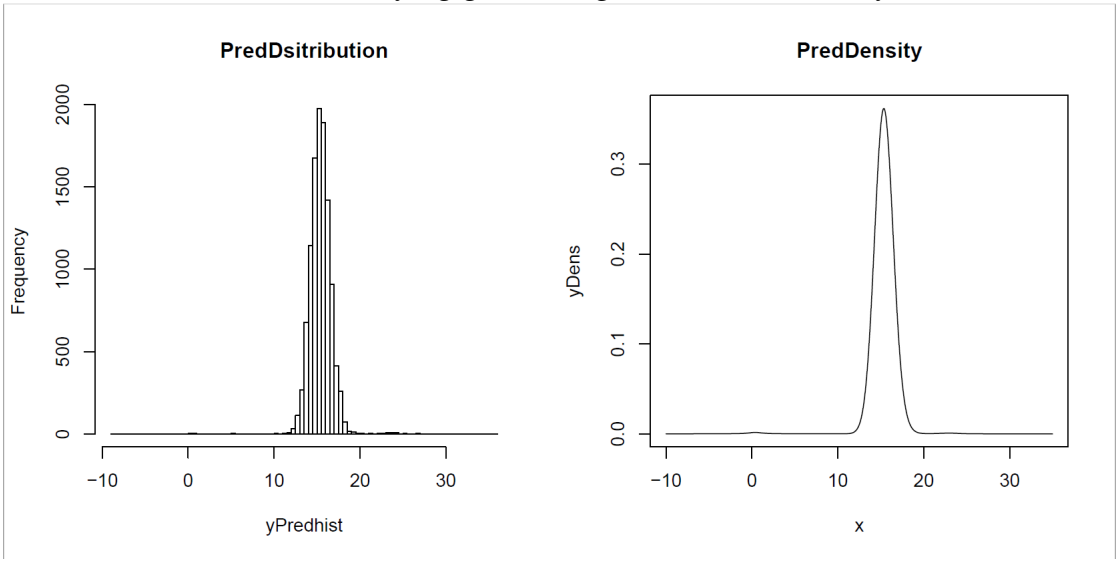
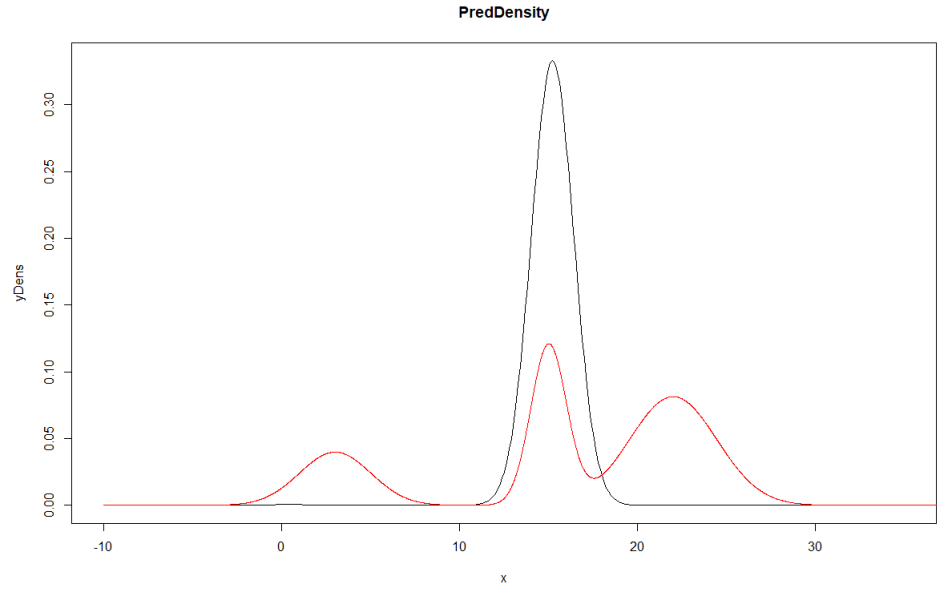


Figure 3: Approximated predictive distribution with $W = 100$ and $\alpha = 1$



Black curve: approximated predictive distribution density curve

Red curve: underlying generating distribution density curve

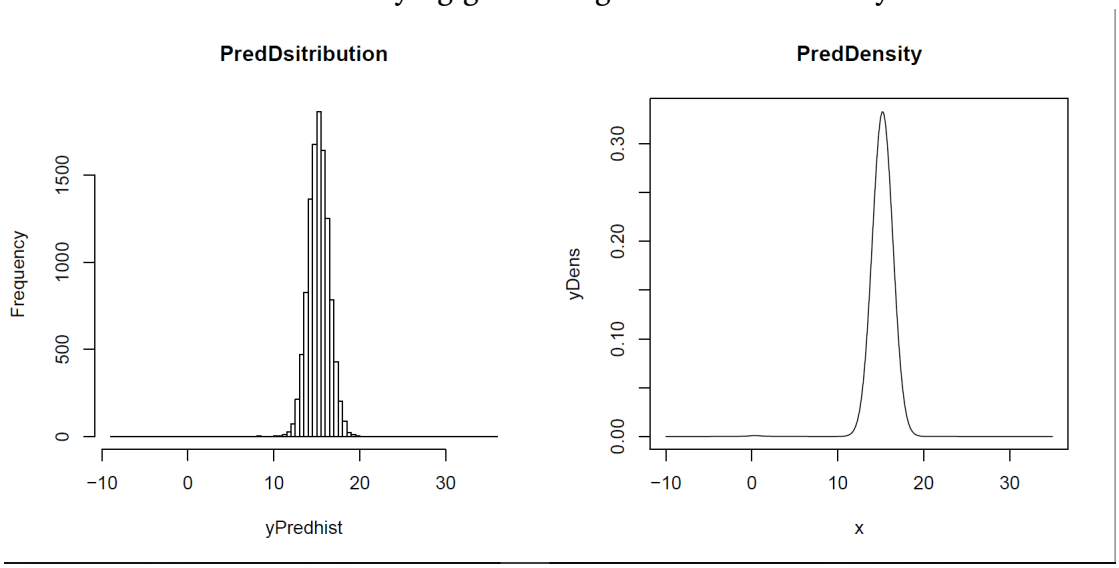
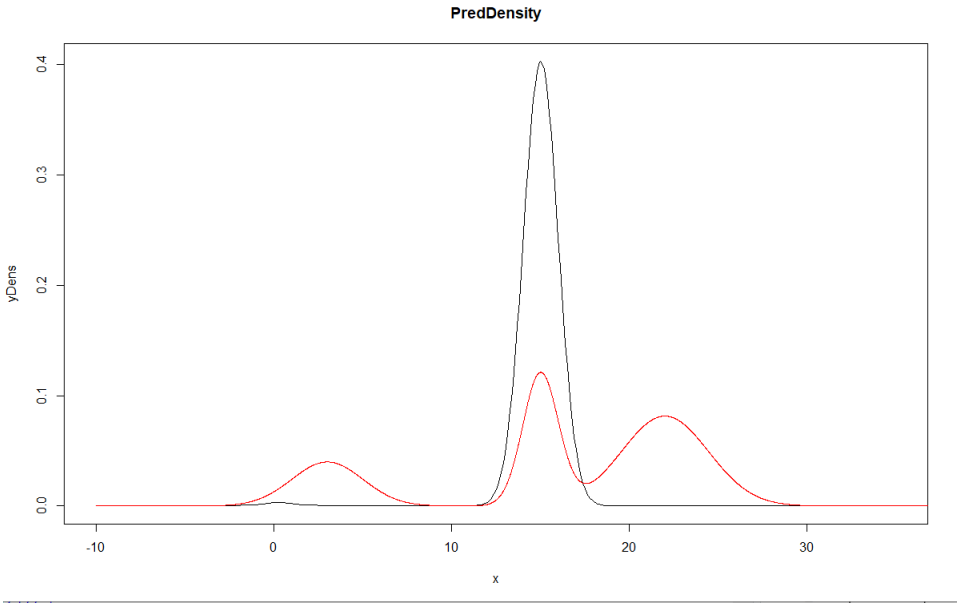


Figure 4: Approximated predictive distribution with $W = 100$ and $\alpha = 0.5$



Black curve: approximated predictive distribution density curve

Red curve: underlying generating distribution density curve

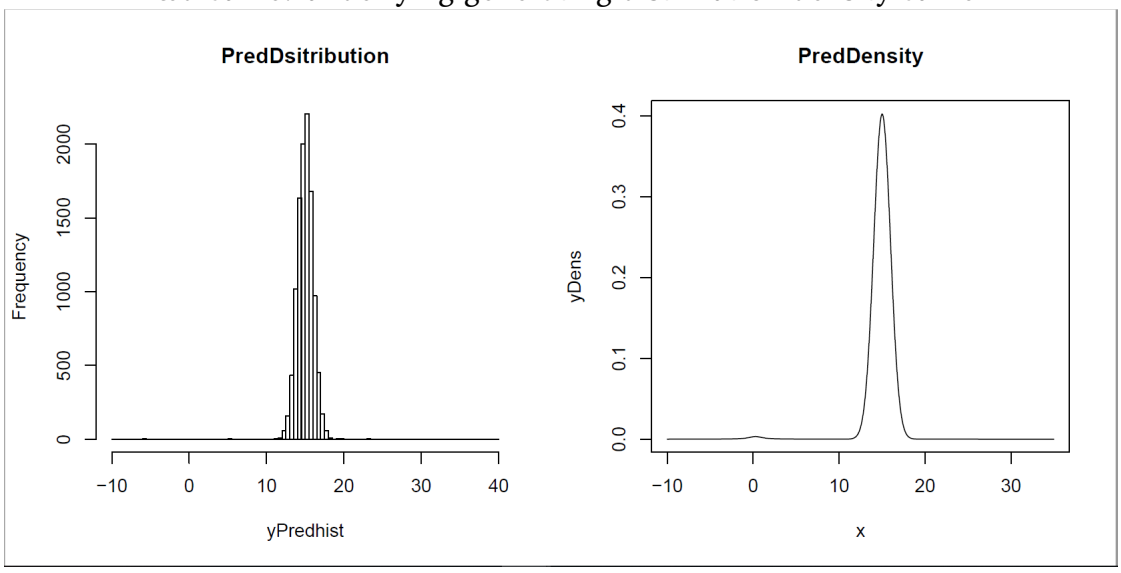
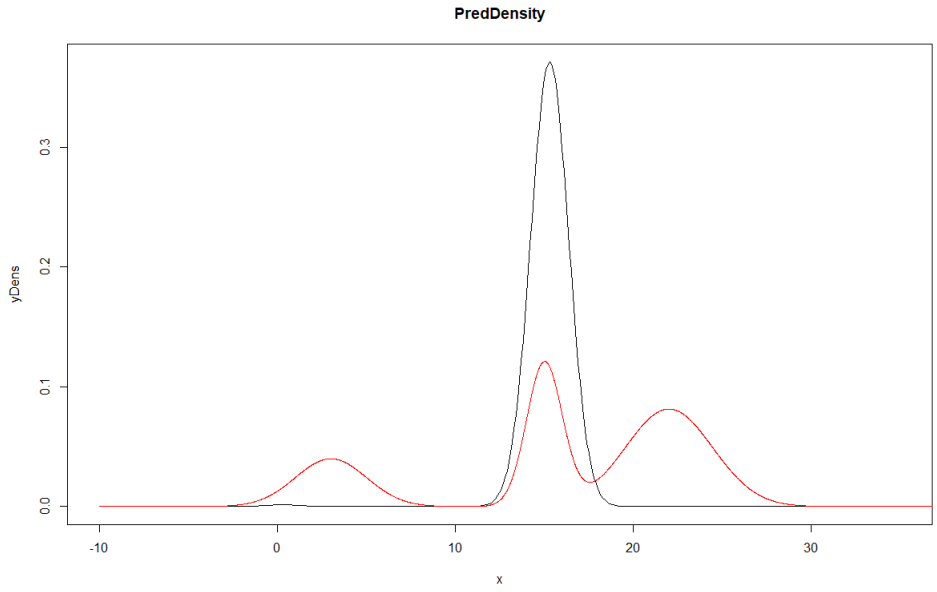


Figure 5: Approximated predictive distribution with $W = 200$ and $\alpha = 2$



Black curve: approximated predictive distribution density curve

Red curve: underlying generating distribution density curve

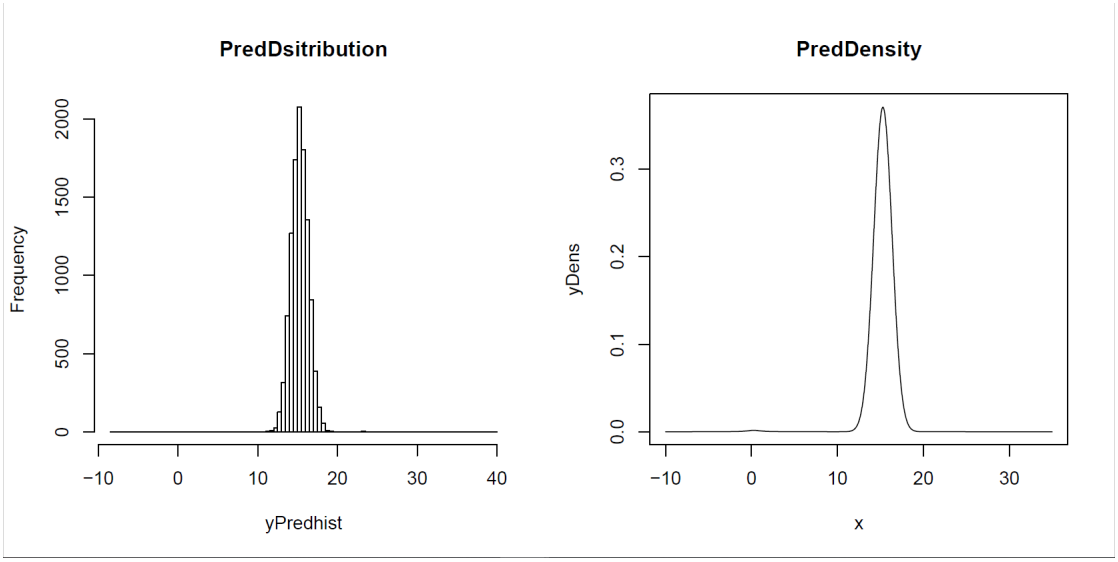
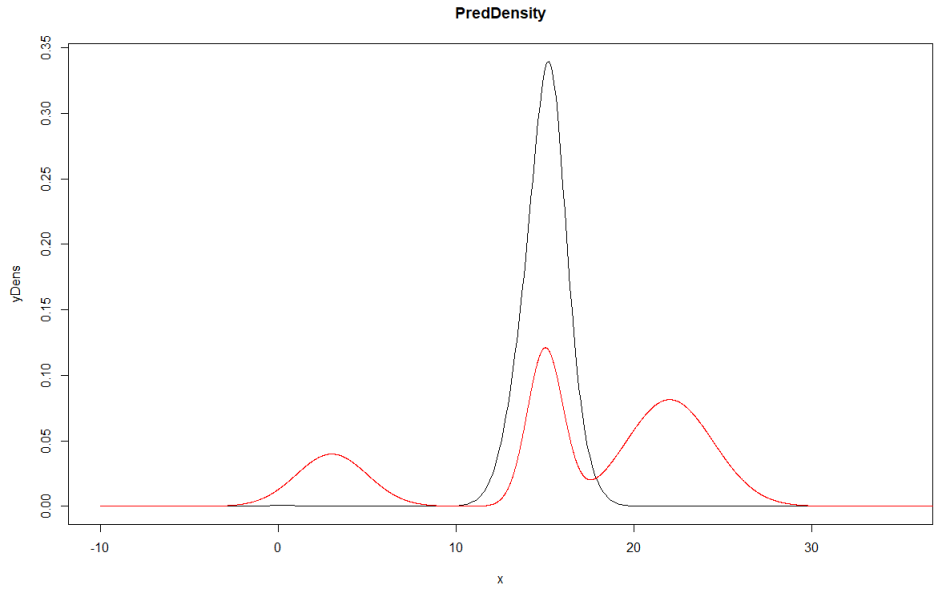


Figure 6: Approximated predictive distribution with $W = 200$ and $\alpha = 1$



Black curve: approximated predictive distribution density curve

Red curve: underlying generating distribution density curve

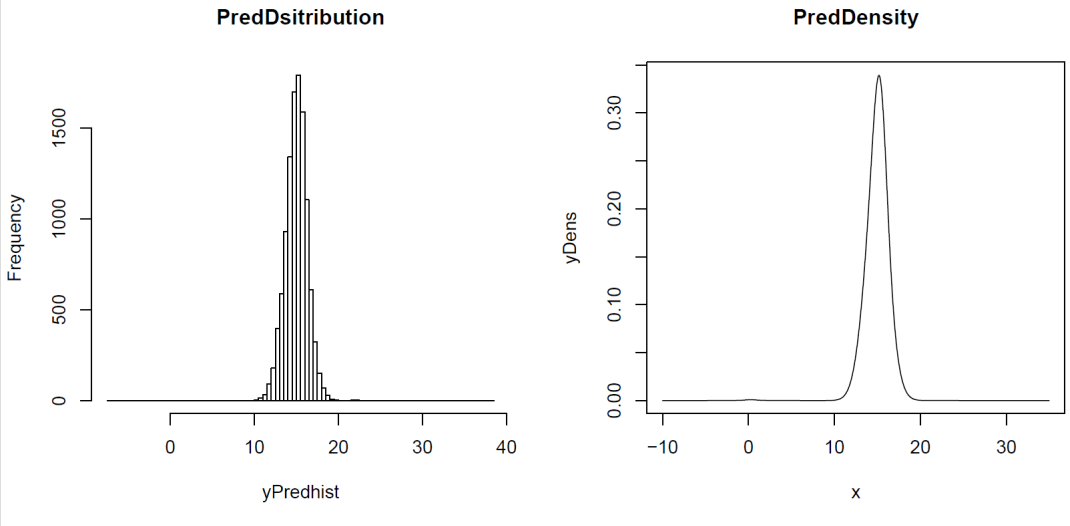


Figure 7: Approximated predictive distribution with $W = 200$ and $\alpha = 0.5$

6 Appendices

DataGen.R

The R code to generate 150 observations from the normal mixture distribution:

```
#Mixture of normal
# 0.2N(3,4)+0.3N(15,1)+0.5N(22,6)
p <- c(0.2, 0.3, 0.5)
mui <- c(3, 15, 22)
Vi <- c(4, 1, 6)
N <- 150
Y <- numeric(N)
for(i in 1:N) {
  kernelId <- which(rmultinom(1, 1, p)==1)
  Y[i] <- rnorm(1, mui[kernelId], sqrt(Vi[kernelId]))
}
hist(Y, 20)
mix1Data <- Y
x <- seq(-10, 40, 0.01)
f <- p[1]*dnorm(x, mui[1], sqrt(Vi[1])) +
  p[2]*dnorm(x, mui[2], sqrt(Vi[2])) +
  p[3]*dnorm(x, mui[3], sqrt(Vi[3]))
plot(x, f, type="l")
```

Dirichlet.R

The R code for computatoin of the Gibbs sampling:

```
Y <- mix1Data
N <- length(Y)
####### Prior Choice pp76 Hoff #######
```

mix1Data

range (-10, 35) – mean 12.5, sd 11.25

mu0 <- 12.5

sigma20 <- 11.25$\hat{2}$

nu0 <- 1

k0 <- 1

s = nu0 = 1; S = nu0*sigma20 <- 11.25^2

m = mu0 = 12.5; tau = 1/k0 = 1

s <- 1; S <- 11.25^2

s <- 1/2

S <- 8

prior for m

a <- 12.5

A <- 8^2

w <- 1

W <- 200

nSim <- 21000

nBurnIn <- 10000

#alpha <- 0.5 ## strength

#alpha <- 1 ## strength

alpha <- 2 ## strength

####### Step 1: #######

NOTE: Only s and S are fixed.

initial of m and tau

mInit <- 12.5

tauInit <- 1

tmpm <- mInit

tmptau <- tauInit

```
XX <- tmptau/(1+tmptau)
xi <- (tmpm+tmptau*Y)/(1+tmptau)
Si <- S + (Y-tmpm)^2/(1+tmptau)
## initial values of mui and Vi
ViInit <- numeric(N)
muiInit <- numeric(N)
for (i in 1:N)
  ViInit[i] <- 1/rgamma(1, shape=(1+s)/2, rate=Si[i]/2)
  muiInit[i] <- rnorm(1, xi[i], sqrt(XX*ViInit[i]))
tmpVi <- ViInit
tmpmui <- muiInit
####### Step 2: #######
## Burnin period
for (isim in 1:nBurnIn) {
  parMat <- matrix(c(tmpmui, tmpVi), nrow=2, ncol=N, byrow=TRUE)
  parMatUni <- unique(parMat, MARGIN=2)
  kKern <- dim(parMatUni)[2]
  nKern <- numeric(kKern)
  for (iKern in 1:kKern) {
    nKern[iKern] <- length(unique(which(parMat==
      parMatUni[,iKern], arr.ind=TRUE)[,2]))
}
  ## Key: parMatUni – unique par matrix
  ## nKern – numbers of each unique par
  ## Update m
  Vbar <- 1/(sum(1/parMatUni[2,]))
  Vmubar <- sum(parMatUni[1,]/parMatUni[2,])
  kai <- A/(A+tmptau*Vbar)
```

```
tmpm <- rnorm(1, (((1-kai)*a)+kai*Vbar*Vmubar), sqrt(kai*tmptau*Vbar))
## Update tau
KK <- sum((parMatUni[1,]-tmpm)^2/parMatUni[2,])
tmptau <- 1/rgamma(1, (w+kKern)/2, (W+KK)/2)
rm(parMat, parMatUni, kKern, nKern)
## Update par (mui and Vi)
#isim ¡- 1 ## debug symbol
XX <- tmptau/(1+tmptau)
xi <- (tmpm+tmptau*Y)/(1+tmptau)
Si <- S + (Y-tmpm)^2/(1+tmptau)
M <- (1+tmptau)*S/s
cs ¡- gamma((1+s)/2)/(gamma(s/2)*sqrt(s))
q0 <- alpha*cs/((1+(Y-tmpm)^2/(s*M))^((1+s)/2)*sqrt(M))
for (k in 1:N) {
  #k <- 2 ## debug symbol
  YMinK <- Y[-k]
  tmpViMink <- tmpVi[-k]
  tmpmuiMink <- tmpmui[-k]
  qtmp <- c(q0[k], (exp(-(YMinK-tmpmuiMink)^2/(2*tmpViMink)))
    /(sqrt(2*tmpViMink)))
  q <- qtmp/(sum(qtmp))
  qSingle <- q[2:length(q)]
  #kernelId <- which(rmultinom(1, 1, q)==1)
  #if (kernelId==1) {
    #tmpVi[k] <- 1/rgamma(1, shape=(1+s)/2, rate=Si[k]/2)
    #tmpmui[k] <- rnorm(1, xi[k], sqrt(XX*tmpVi[k]))
  #} else {
    #tmpVi[k] <- tmpViMink[kernelId-1]
```

```
    #tmpmui[k] <- tmpmuiMink[kernelId-1]
  #}
  parMat <- matrix(c(tmpmuiMink, tmpViMink), nrow=2,
   ncol=length(tmpmuiMink), byrow=TRUE)
  parMatUni <- unique(parMat, MARGIN=2)
  kKern <- dim(parMatUni)[2]
  qCluster <- numeric((1+kKern))
  qCluster[1] <- q[1]
  nKern <- numeric(kKern)
  for (iKern in 1:kKern) {
    KernId <- unique(which(parMat==parMatUni[,iKern],
     arr.ind=TRUE)[,2])
    qCluster[iKern+1] <- sum(qSingle[KernId])
    #nKern[iKern] <- length(unique(which(parMat==
     parMatUni[,iKern], arr.ind=TRUE)[,2]))
    rm(KernId)
  }
  kernelId <- which(rmultinom(1, 1, qCluster)==1)
  if (kernelId==1) {
    tmpVi[k] <- 1/rgamma(1, shape=(1+s)/2, rate=Si[k]/2)
    tmpmui[k] <- rnorm(1, xi[k], sqrt(XX*tmpVi[k]))
  } else {
    tmpVi[k] <- parMatUni[2, kernelId-1]
    tmpmui[k] <- parMatUni[1, kernelId-1]
  }
  rm(parMat, parMatUni, kKern, nKern, iKern)
  rm(qtmp, q)
}
```

```
}
## tmpVi & tmpmui are known
mSim <- numeric(nSim-nBurnIn)
tauSim <- numeric(nSim-nBurnIn)
Vi <- matrix(N*(nSim-nBurnIn), nrow=N, ncol=(nSim-nBurnIn))
mui <- matrix(N*(nSim-nBurnIn), nrow=N, ncol=(nSim-nBurnIn))
postkKernel <- numeric(nSim-nBurnIn)
for (isim in 1:(nSim-nBurnIn)) {
  parMat <- matrix(c(tmpmui, tmpVi), nrow=2, ncol=N, byrow=TRUE)
  parMatUni <- unique(parMat, MARGIN=2)
  kKern <- dim(parMatUni)[2]
  nKern <- numeric(kKern)
  for (iKern in 1:kKern) {
    nKern[iKern] <- length(unique(which(parMat==parMatUni[,iKern],
      arr.ind=TRUE)[,2]))
  }
  ## Key: parMatUni – unique par matrix
  ## nKern – numbers of each unique par
  ## Update m
  Vbar <- 1/(sum(1/parMatUni[2,]))
  Vmubar <- sum(parMatUni[1,]/parMatUni[2,])
  kai <- A/(A+tmptau*Vbar)
  tmpm <- rnorm(1, (((1-kai)*a)+kai*Vbar*Vmubar), sqrt(kai*tmptau*Vbar))
  mSim[isim] <- tmpm
  ## Update tau
  KK <- sum((parMatUni[1,]-tmpm)^2/parMatUni[2,])
  tmptau <- 1/rgamma(1, (w+kKern)/2, (W+KK)/2)
  tauSim[isim] <- tmptau
```

```
rm(parMat, parMatUni, kKern, nKern)
## Update par (mui and Vi)
#isim <- 1 ## debug symbol
XX <- tmptau/(1+tmptau)
xi <- (tmpm+tmptau*Y)/(1+tmptau)
Si <- S + (Y-tmpm)^2/(1+tmptau)
M <- (1+tmptau)*S/s; cs i- gamma((1+s)/2)/(gamma(s/2)*sqrt(s))
q0 <- alpha*cs/((1+(Y-tmpm)^2/(s*M))^(1 + s)/2)*sqrt(M))
for (k in 1:N) {
  #k <- 1 ## debug symbol
  YMinK <- Y[-k]
  tmpViMink <- tmpVi[-k]
  tmpmuiMink <- tmpmui[-k]
  qtmp <- c(q0[k],
  (exp(-(YMinK-tmpmuiMink)^2/(2*tmpViMink)))/(sqrt(2*tmpViMink)))
  q <- qtmp/(sum(qtmp))
  qSingle <- q[2:length(q)]
  #kernelId <- which(rmultinom(1, 1, q)==1)
  #if (kernelId==1)
    #tmpVi[k] <- 1/rgamma(1, shape=(1+s)/2, rate=Si[k]/2)
    #tmpmui[k] <- rnorm(1, xi[k], sqrt(XX*tmpVi[k]))
  # else
    #tmpVi[k] <- tmpViMink[kernelId-1]
    #tmpmui[k] <- tmpmuiMink[kernelId-1]
  #
  parMat <- matrix(c(tmpmuiMink, tmpViMink), nrow=2,
   ncol=length(tmpmuiMink), byrow=TRUE)
  parMatUni <- unique(parMat, MARGIN=2)
```

```
    kKern <- dim(parMatUni)[2]
    qCluster <- numeric((1+kKern))
    qCluster[1] <- q[1]
    nKern <- numeric(kKern)
    for (iKern in 1:kKern) {
      KernId <- unique(which(parMat==parMatUni[,iKern],
        arr.ind=TRUE)[,2])
      qCluster[iKern+1] <- sum(qSingle[KernId])
      #nKern[iKern] <- length(unique(which(parMat==
       parMatUni[,iKern], arr.ind=TRUE)[,2]))
      rm(KernId)
    }
    kernelId <- which(rmultinom(1, 1, qCluster)==1)
    if (kernelId==1) {
      tmpVi[k] <- 1/rgamma(1, shape=(1+s)/2, rate=Si[k]/2)
      tmpmui[k] <- rnorm(1, xi[k], sqrt(XX*tmpVi[k]))
    } else {
      tmpVi[k] <- parMatUni[2, kernelId-1]
      tmpmui[k] <- parMatUni[1, kernelId-1]
    }
    rm(parMat, parMatUni, kKern, nKern, iKern)
    Vi[k, isim] <- tmpVi[k]
    mui[k, isim] <- tmpmui[k]
    rm(qtmp, q)
  }
parMat <- matrix(c(tmpmui, tmpVi), nrow=2, ncol=N, byrow=TRUE)
parMatUni <- unique(parMat, MARGIN=2)
postkKernel[isim] <- dim(parMatUni)[2]
```

```
rm(parMat, parMatUni)
}
```

PredDensity.R

The R code for construction of predictive distribution.

```
predDist <- function(alpha, s, S, m, tau, mui, Vi) {
  n <- dim(mui)[1]
  nGen <- dim(mui)[2]
  aN <- 1/(alpha + n)
  p1 <- alpha*aN
  p2 <- aN
  M <- (1+tau)*S/s
  yPred <- numeric(nGen)
  for (igen in 1 : nGen) {
    parMat <- matrix(c(mui[,igen], Vi[,igen]), nrow=2, ncol=n, byrow=TRUE)
    parMatUni <- unique(parMat, MARGIN=2)
    kKern <- dim(parMatUni)[2]
    nKern <- numeric(kKern)
    for (iKern in 1:kKern) {
      nKern[iKern] <- length(unique(which(parMat==
      parMatUni[,iKern],arr.ind=TRUE)[,2]))
    }
    p <- c(p1, p2*nKern)
    #(X_{n+1}|X_{1:n}) –coro5.3
    kernelId <- which(rmultinom(1, 1, p)==1)
    if (kernelId==1) {
      #random t-dist
      yPredtmp <- rt(1, df=s)
```

```
      yPred[igen] <- m[igen] + sqrt(M[igen])*yPredtmp
      rm(yPredtmp)
    } else {
      #X_i
      yPred[igen] <- rnorm(1, parMatUni[1, (kernelId-1)],
        sqrt(parMatUni[2, (kernelId-1)]))
    }
    rm(parMat, parMatUni, kKern, nKern, iKern, p)
  }
  return(yPred)
}
predDensity <- function(x, alpha, s, S, m, tau, mui, Vi) {
  n <- dim(mui)[1]
  nGen <- dim(mui)[2]
  aN <- 1/(alpha + n)
  p1 <- alpha*aN
  p2 <- aN
  M <- (1+tau)*S/s
  yDen <- numeric(length(x))
  for (igen in 1:nGen) {
    parMat <- matrix(c(mui[,igen], Vi[,igen]), nrow=2, ncol=n, byrow=TRUE)
    parMatUni <- unique(parMat, MARGIN=2)
    kKern <- dim(parMatUni)[2]
    nKern i<-- numeric(kKern)
    for (iKern in 1:kKern) {
      nKern[iKern] <- length(unique(which(parMat==
        parMatUni[,iKern], arr.ind=TRUE)[,2]))
    }
```

```
      #(X_{n+1}|pai, m, tau)~ alpha * a_n * T_s(m, M) + a_n sum(n_j * N(mui*_j, V*_j)
      #alpha * a_n * T_s(m, M)
      yDentmp <- p1*dt(x-m[igen]/sqrt(M[igen]), df=s)
      #a_n sum(n_j * N(mui*_j, V*_j)
      for (k in 1:kKern) {
        yDentmp <- yDentmp + nKern[k]*p2*dnorm(x,
          parMatUni[1,k], sqrt(parMatUni[2,k]))
        #rm(parMat, parMatUni, kKern, nKern, iKern)
      }
      yDen <- yDen + yDentmp
    }
    #P(X_{n+1}|X_{1:n}) 1/Nsum{P(X_{n+1|pai,n,tau})}
    yDen <- yDen/nGen
    return(yDen)
}
####### For Mix Data #######
## Vi (N*nSim); mui (N*nSim), kKernel(nSim)
N <- dim(Vi)[1]
nSim <- dim(Vi)[2]
#s <- 1; S <- 1^2
yPred <- predDist(alpha=alpha, s=s, S=S,
m=mSim, tau=tauSim, mui=mui, Vi=Vi)
pk <- numeric(15)
for (i in 1:15) {
  pk[i] <- length(which(postkKernel==i))/length(postkKernel)
}
hist(yPred, 40)
mean(yPred)
```

```
#var(yPred)
qlow <- quantile(yPred, 0.01)
qhigh <- quantile(yPred, 0.99)
#rm(alpha, i, mui, N, nSim, pk, S, s, Vi, yPred)
## density
x <- seq(-10, 35, 0.1)
yDens <- predDensity(x, alpha=alpha, s=s, S=S, m=mSim,
tau=tauSim, mui=mui, Vi=Vi)
#dev.new(height=5, width=20)
opar <- par()
par(mfrow=c(1,2))
yPredhist <- yPred[which(yPred ¿= -10 & yPred ¡= 40)]
hist(yPredhist,80,,main = "PredDsitribution")
mean(yPredhist)
var(yPredhist)
plot(x, yDens,type="l",main = "PredDensity")
```

References

Charles E. Antoniak. Mixtures of dirichlet processes with applications to bayesian nonparametric problems. *The Annals of Statistics*, 2(6):1152–1174, 1974.

Michael D. Escobar and Mike West. Bayesian density estimation and inference using mixtures. *Journal of the American Statistical Association*, 90(430):577–588, 1995.

Thomas S. Ferguson. A bayesian analysis of some nonparametric problems. *The Annals of Statistics*, 1(2):209–230, 1973.

Jayanta K. Ghosh and R.V. Ramamoorthi. *Bayesian Nonparametrics*. Springer, 2003.