Assignment 3

STAT 440/840 - CM 761

A G component finite mixutre of multivariate-normal is given by

$$g\left(\mathbf{x} \mid \boldsymbol{\theta}\right) = \sum_{g=1}^{G} \pi_g \, \phi_p\left(\mathbf{x} \mid \boldsymbol{\mu}_g, \boldsymbol{\Sigma}_g\right).$$

Note, the parameters are

$$\boldsymbol{\theta} = \left(\pi_1, \ldots, \pi_G, \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_G, \boldsymbol{\Sigma}_1 \ldots, \boldsymbol{\Sigma}_G\right).$$

1. **[8 Marks]** Properties of the model

   a) (1 Mark) To apply the EM we take the component membership for each observation as missing data denoted by Z_{ig}. What is the marginal distribution of the missing data?

- Defining the missing data as

$$Z_{ig} = \begin{cases} 1 & \text{if observation } i \text{ is from group } g \\ 0 & \text{otherwise} \end{cases}$$

and then let

$$\mathbf{Z}_i = \left(Z_{i1}, Z_{i2}, \ldots, Z_{iG}\right)$$

then the mariginal distribution of the missing data is multinomial with probabilities $\pi_1, \pi_2, \ldots, \pi_G$, i.e.

$$\mathbf{Z}_i \sim \text{Multinomial}\left(n = 1, \pi_1, \pi_2, \ldots, \pi_G\right)$$

b) (1 Mark) What is the conditional distribution of the observed data given the missing data?

$$\mathbf{X}_i \mid \mathbf{Z}_{ig} = 1 \sim \text{MVN}_p\left(\boldsymbol{\mu}_g, \boldsymbol{\Sigma}_g\right)$$

OR

$$f(x) = \prod_{g=1}^{G} \left[\phi_p\left(\boldsymbol{\mu}_g, \boldsymbol{\Sigma}_g\right)\right]^{Z_{ig}}$$

c) (2 Marks) What is the distribution of the missing data given the observed data.

$$\mathbf{Z}_i \mid \mathbf{X} = x \sim \text{Multinomial}\left(n = 1, p_1, p_2, \ldots, p_G\right)$$

where

$$p_g = \frac{\pi_g \phi_g\left(\mathbf{x}_i \mid \boldsymbol{\mu}_g, \boldsymbol{\Sigma}_g\right)}{\sum_{k=1}^{G} \pi_k \, \phi_p\left(\mathbf{x}_i \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right)}$$

OR to perform the EM we only need

$$Pr(\mathbf{Z}_{ig} = 1 \mid \mathbf{x}_i) = \frac{\phi_g\left(\mathbf{x}_i \mid \boldsymbol{\mu}_g, \boldsymbol{\Sigma}_g\right)}{\sum_{k=1}^{G} \pi_k \, \phi_p\left(\mathbf{x}_i \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right)}$$

d) (1 Mark) What is the expected value of the missing data given the observed data.

Since the missing data given the observed data is Multinomial

$$\mathbb{E}\left[\mathbf{Z}_{ig} \mid \mathbf{X}_i = \mathbf{x}_i\right] = n \times p_g = 1 \times p_g = \frac{\phi_g\left(\mathbf{x}_i \mid \boldsymbol{\mu}_g, \boldsymbol{\Sigma}_g\right)}{\sum_{k=1}^{G} \pi_k \, \phi_p\left(\mathbf{x}_i \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right)}$$

OR since the missing given the observed data is binary then

$$\mathbb{E}\left[\mathbf{Z}_{ig} \mid \mathbf{X}_i = \mathbf{x}_i\right] = \Pr\left(\mathbf{Z}_{ig} = 1 \mid \mathbf{X}_i = \mathbf{x}_i\right) = \frac{\phi_g\left(\mathbf{x}_i \mid \boldsymbol{\mu}_g, \boldsymbol{\Sigma}_g\right)}{\sum_{k=1}^{G} \pi_k \, \phi_p\left(\mathbf{x}_i \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right)}$$

e) (3 Marks) Give an algorithm and a `R` function to generate data from a G component finite mixture of a multivariate-normals.

```
library(MASS)
rgmix <- function(n = NULL, gpar = NULL) {
    G = length(gpar$pi)
    p = length(gpar[[1]]$mu)

    group = sample(1:G, size = n, replace = TRUE, prob = gpar$pi)

    datax = matrix(0, nrow = n, ncol = p)

    for (g in 1:G) {
        tempg = group == g
        ng = sum(tempg)
        if (ng > 0)
            datax[tempg, ] = rmvnorm(ng, mean = gpar[[g]]$mu, sigma = gpar[[g]]$sigma)

    }

    return(datax)
}
```

2. **[20 Marks]** An EM algorithm,

a) (1 Mark) Give the observed log-likelihood function.

$$\mathcal{L}\left(\theta\right) = \prod_{i=1}^{n} g\left(\mathbf{x}_i \mid \theta\right) = \prod_{i=1}^{n} \left[\sum_{g=1}^{G} \pi_g \, \phi_p\left(\mathbf{x}_i \mid \boldsymbol{\mu}_g, \boldsymbol{\Sigma}_g\right)\right]$$

$$l\left(\theta\right) = \sum_{i=1}^{n} \log g\left(\mathbf{x}_i \mid \theta\right) = \sum_{i=1}^{n} \log \left[\sum_{g=1}^{G} \pi_g \, \phi_p\left(\mathbf{x}_i \mid \boldsymbol{\mu}_g, \boldsymbol{\Sigma}_g\right)\right]$$

b) (2 Marks) Write a R function which takes the parameters and data as an input and ouput the observed log-likelihood.

```
loglik = function(data = NULL, gpar = NULL) {
    # output is a G x nrow(data) matrix
    G = length(gpar$pi)
    w = matrix(0, nrow = nrow(data), ncol = G)
    for (i in 1:G) w[, i] = dmvnorm(data, mean = gpar[[i]]$mu, sigma = gpar[[i]]$sigma)

    w = apply(w, 1, function(z, wt) {
        sum(z * wt)
    }, wt = gpar$pi)
    val = sum(log(w))
    return(val)
}
```

c) (1 Marks) To begin the derivation of the EM algorithm, give the complete data log-likelihood.

$$\mathcal{L}_c(\theta) = \prod_{i=1}^{n} \prod_{g=1}^{G} \left[\pi_g \, \phi_p\left(\mathbf{x}_i \mid \boldsymbol{\mu}_g, \boldsymbol{\Sigma}_g\right)\right]^{Z_{ig}}$$

$$l_c(\theta) = \sum_{i=1}^{n} \sum_{g=1}^{G} Z_{ig} \left[\log \pi_g + \log \phi_p\left(\mathbf{x}_i \mid \boldsymbol{\mu}_g, \boldsymbol{\Sigma}_g\right)\right]$$

d) (4 Marks) E-step: Derive the expected complete data log-likelihood, denoted by $\mathcal{Q}$.

$$\mathcal{Q}\left(\theta \mid \theta^{(t)}\right) = \mathbb{E}\left[l_c(\theta) \mid \mathbf{x}_1, \ldots, \mathbf{x}_n, \theta^{(t)}\right]$$

$$\mathcal{Q}\left(\theta \mid \theta^{(t)}\right) = \sum_{i=1}^{n} \sum_{g=1}^{G} \mathbb{E}\left[Z_{ig} \mid \mathbf{x}_i, \theta^{(t)}\right] \left[\log \pi_g \; + \log \phi_p\left(\mathbf{x}_i \mid \boldsymbol{\mu}_g, \boldsymbol{\Sigma}_g\right)\right]$$

let

$$\widehat{z}_{ig} := \mathbb{E}\left[Z_{ig} \mid \mathbf{x}_i, \theta^{(t)}\right] = \frac{\pi_g \, \phi_p\left(\mathbf{x}_i \mid \boldsymbol{\mu}_g, \boldsymbol{\Sigma}_g\right)}{\sum_{k=1}^{G} \pi_k \, \phi_p\left(\mathbf{x}_i \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right)}$$

$$\mathcal{Q}\left(\theta \mid \theta^{(t)}\right) = \sum_{i=1}^{n} \sum_{g=1}^{G} \widehat{z}_{ig} \left[\log \pi_g \; + \log \phi_p\left(\mathbf{x}_i \mid \boldsymbol{\mu}_g, \boldsymbol{\Sigma}_g\right)\right]$$

e) (4 Marks) Write a R function which takes the parameters and data as an input and ouput the expected value of the missing data given the observed data.

```
e.step <- function(data = NULL, gpar = NULL) {
    G = length(gpar$pi)
    if (G > 1) {
        zlog = matrix(0, nrow = nrow(data), ncol = length(gpar$pi))
        for (k in 1:G) zlog[, k] = dmvnorm(data, mean = gpar[[k]]$mu, sigma = gpar[[k]]$sigma,
            log = FALSE)
        w = t(apply(zlog, 1, function(z, wt, v) {
            x = z * wt
            x = x/sum(x)
            return(x)
        }, wt = gpar$pi, v = v))
```

```
    } else w = matrix(1, nrow = nrow(data), ncol = G)
    return(w)
}
```

f) (4 Marks) M-step: Find the parameter updates by maximizing the expected complete data log-likelihood. Similar to the multivariate normal but now with weights.

$$\widehat{\pi}_g^{(t+1)} = \widehat{\pi}_g = \frac{\sum_{i=1}^n \widehat{z}_{ig}}{n} = \frac{n_g}{n}$$

$$\widehat{\boldsymbol{\mu}}_g^{(t+1)} = \widehat{\boldsymbol{\mu}}_g = \overline{\mathbf{x}}_g = \frac{\sum_{i=1}^n \widehat{z}_{ig}\mathbf{x}_i}{n_g} = \frac{\sum_{i=1}^n \widehat{z}_{ig}\mathbf{x}_i}{\sum_{i=1}^n \widehat{z}_{ig}}$$

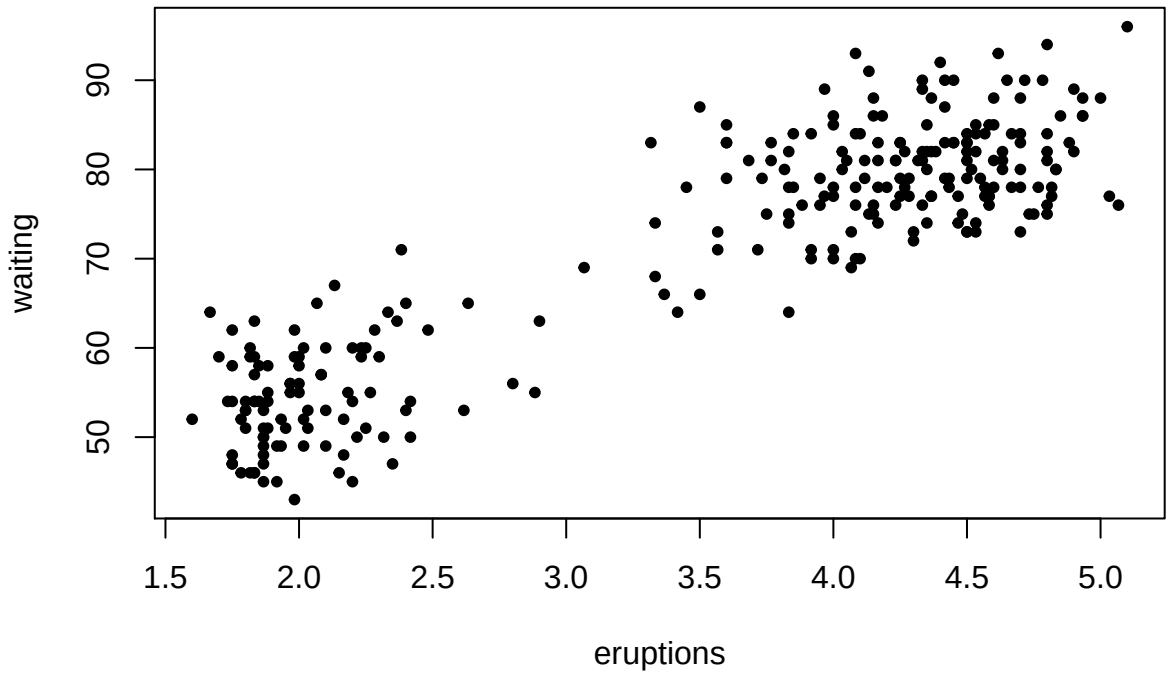
$$\widehat{\boldsymbol{\Sigma}}_g^{(t+1)} = \widehat{\boldsymbol{\Sigma}}_g = \frac{1}{n_g}\sum_{i=1}^n \widehat{z}_{ig}\left(\mathbf{x}_i - \widehat{\boldsymbol{\mu}}_g\right)\left(\mathbf{x}_i - \widehat{\boldsymbol{\mu}}_g\right)' = \frac{1}{n_g}\sum_{i=1}^n \widehat{z}_{ig}\left(\mathbf{x}_i - \widehat{\boldsymbol{\mu}}_g\right)\left(\mathbf{x}_i - \widehat{\boldsymbol{\mu}}_g\right)'$$

g) (4 Marks) Write a R function which takes the expected value of the missing data and returns the updated parameter values.

```
m.step <- function(data = NULL, w = NULL) {
    G = ncol(w)
    gpar = list()

    for (k in 1:G) {
        gpar[[k]] = list()

        temp = cov.wt(data, wt = w[, k], center = TRUE, method = "ML")
        gpar[[k]]$mu = temp$center
        gpar[[k]]$sigma = temp$cov
    }
    gpar$pi = apply(w, 2, mean)

    return(gpar)
}
```

3. **[24 Marks]** Implementing a EM algorithm, for the the old faithful dataset in R.

```
data(faithful)
plot(faithful, pch = 20)
```



a) (12 Marks) Implement a EM algorithm to fit a two component ($G = 2$) multivariate-normal finite mixture to the old faithful dataset in R. Use the following starting value

$$\pi_1 = \frac{1}{10}, \quad \pi_2 = \frac{9}{10},$$

$$\boldsymbol{\mu}_1 = \begin{pmatrix} 2 \\ 60 \end{pmatrix}, \quad \boldsymbol{\mu}_2 = \begin{pmatrix} 2 \\ 50 \end{pmatrix},$$

$$\boldsymbol{\Sigma}_1 = \begin{pmatrix} 0.1 & 0 \\ 0 & 0.1 \end{pmatrix}, \quad \text{and} \quad \boldsymbol{\Sigma}_2 = \begin{pmatrix} 10 & 0 \\ 0 & 10 \end{pmatrix}.$$

As part of the results give R code for the EM,

```
EMn <- function(data = NULL, gpar0 = NULL, G = 2, n = 10) {
    val = list()
    if (is.null(gpar0))
        val$gpar = igpar(data = data, g = G, covtype = covtype) else val$gpar = gpar0
    val$loglik = numeric(n)

    for (i in 1:n) {
        tempw = e.step(data = data, gpar = val$gpar)
        val$gpar = m.step(data = data, w = tempw)

        val$loglik[i] = loglik(data = data, gpar = val$gpar)
    }

    return(val)
}
```

obtain the MLE while using this strange starting value

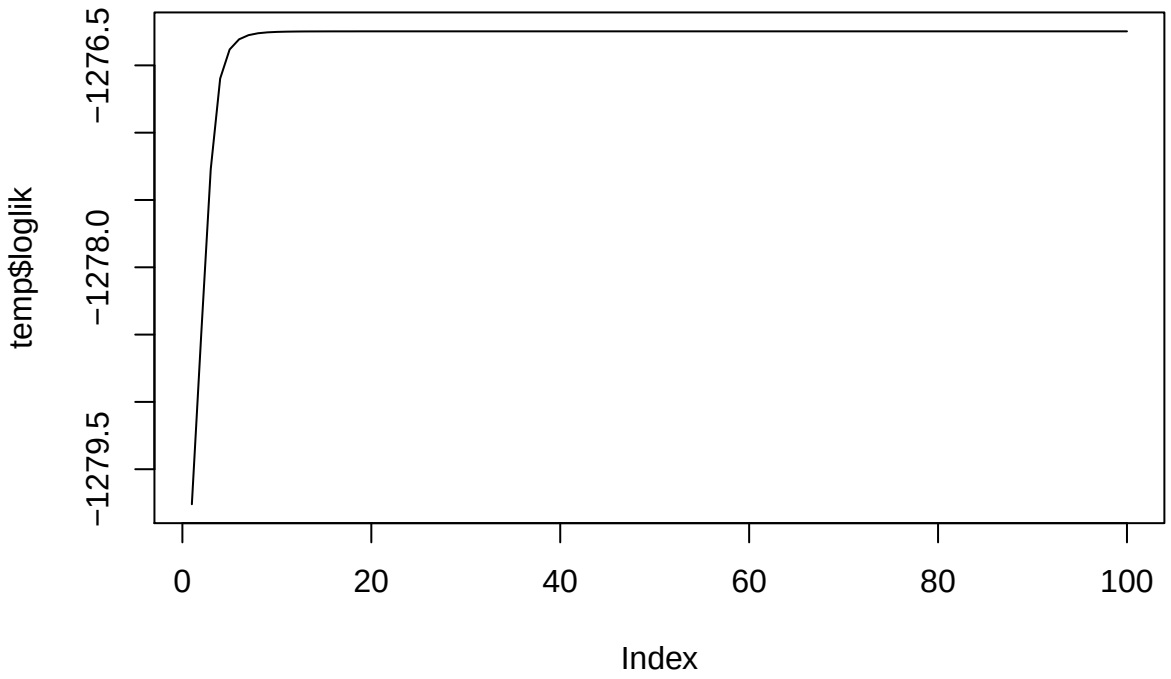
```
library(mvtnorm)
gpar0 = list()
gpar0[[1]] = list()
gpar0[[1]]$mu = c(2, 60)
```

```
gpar0[[1]]$sigma = diag(2) * 0.1
gpar0[[2]] = list()
gpar0[[2]]$mu = c(2, 50)
gpar0[[2]]$sigma = diag(2) * 10
gpar0$pi = c(1, 9)/10

temp = EMn(data = faithful, gpar0 = gpar0, G = 2, n = 100)
temp$gpar
```

```
## [[1]]
## [[1]]$mu
## eruptions   waiting
##  1.963811 58.992353
##
## [[1]]$sigma
##            eruptions   waiting
## eruptions 0.02984192 0.0722365
## waiting   0.07223650 0.7509790
##
##
## [[2]]
## [[2]]$mu
## eruptions   waiting
##  3.568321 71.526190
##
## [[2]]$sigma
##           eruptions   waiting
## eruptions  1.235731  13.64913
## waiting   13.649128 185.95020
##
##
## $pi
## [1] 0.0501946 0.9498054
```

plot the observed log-likelihood from each iteration,

```
plot(temp$loglik, type = "l")
```



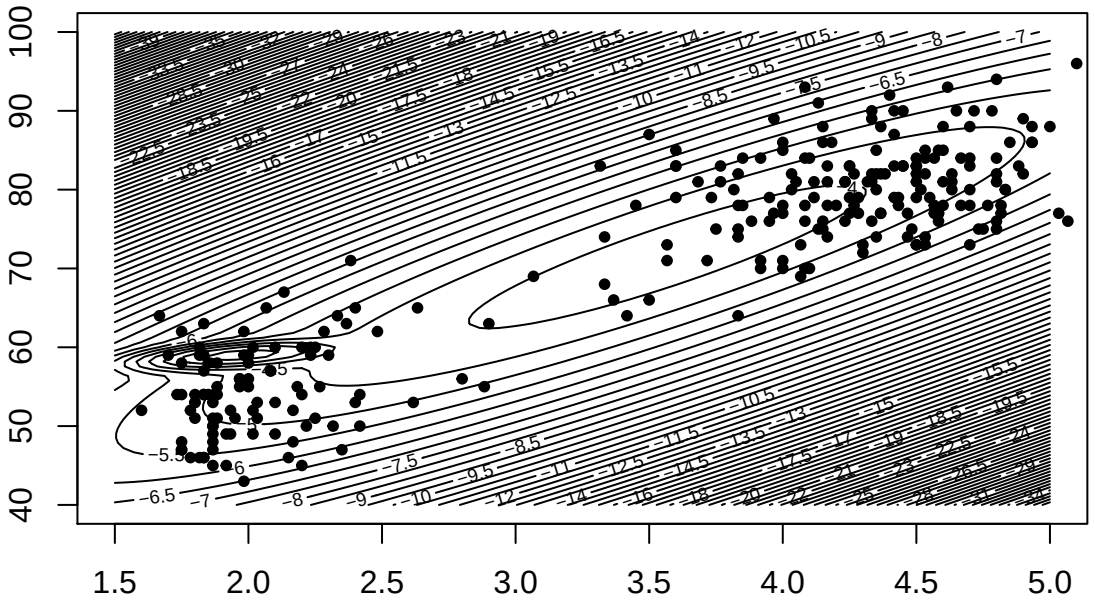
a contour plot of the fitted density along with the points, comment on the fit.

First we need the density

```
dfnorm <- function(data = NULL, gpar = NULL) {
    # output is a G x nrow(data) matrix
    G = length(gpar$pi)
    w = matrix(0, nrow = nrow(data), ncol = G)
    for (i in 1:G) w[, i] = dmvnorm(data, mean = gpar[[i]]$mu, sigma = gpar[[i]]$sigma)

    w = apply(w, 1, function(z, wt) {
        sum(z * wt)
    }, wt = gpar$pi)
    val = log(w)
    return(val)
}
```

then the contour is

```
# par(mfrow=c(1,2), mar=2.5*c(1,1,1,0.1))

x1 = seq(1.5, 5, length.out = 101)
x2 = seq(40, 100, length.out = 100)
zz = matrix(0, nrow = length(x1), ncol = length(x2))
for (i in 1:length(x1)) zz[i, ] = dfnorm(cbind(x1[i], x2), gpar = temp$gpar)

contour(x1, x2, zz, nlevels = 100)
points(faithful, pch = 20)
```



b) (2 Marks) Write a R function to generate random parameters values to use as starting values.

```
rgpar <- function(data = NULL, g = NULL) {
    w = matrix(rexp(nrow(data) * g), nrow = nrow(data), ncol = g)
    w = matrix(t(apply(w, 1, function(z) {
        z/sum(z)
    })), nrow = nrow(data), ncol = g)

    gpar = m.step(data = data, w = w)

    return(gpar)
}
```

c) (6 Marks) Start the EM from a 100 different starting values. As part of the results

- report the solution with the highest log-likelihood,

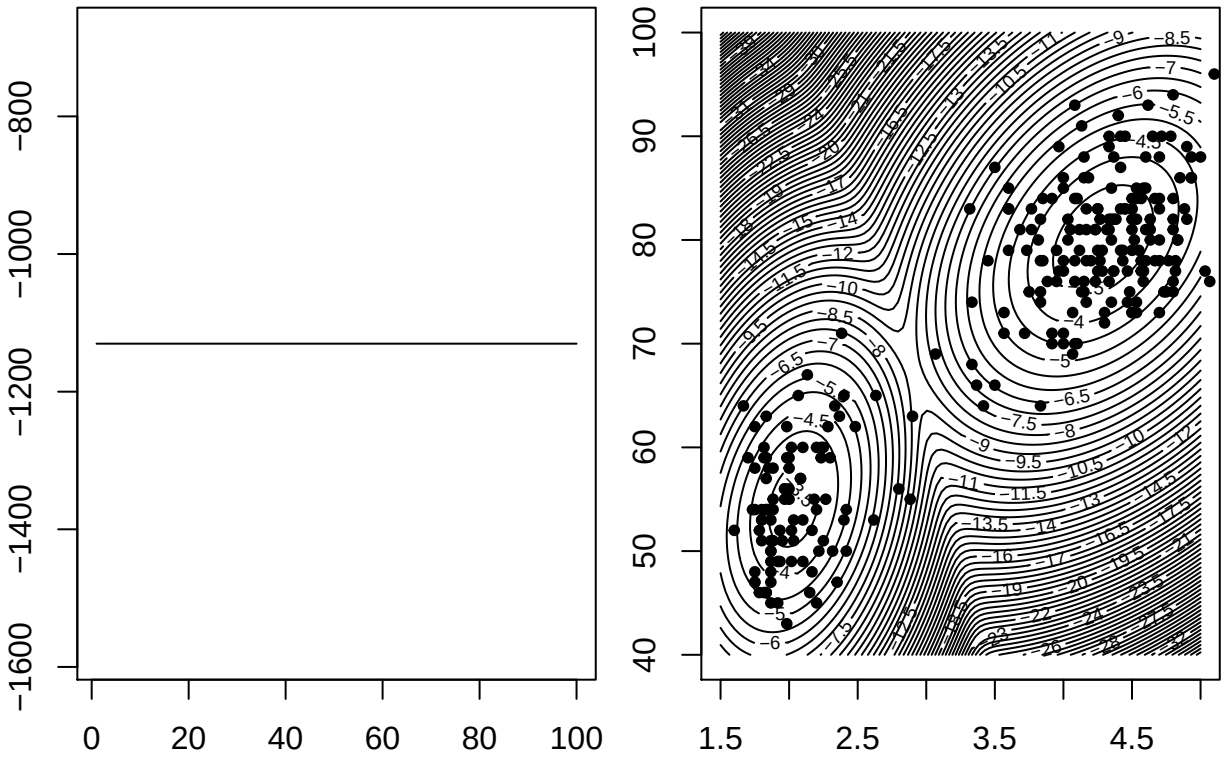
```
set.seed(1)
z = numeric(100)
tz = list()
te = list()
for (i in 1:length(z)) {
    tz[[i]] = rgpar(faithful, g = 2)
    te[[i]] = EMn(data = faithful, gpar0 = tz[[i]], G = 2, n = 100)
    z[i] = max(te[[i]]$loglik)
}
te[[1]]$gpar
```

```
## [[1]]
## [[1]]$mu
## eruptions   waiting
##  2.036388 54.478516
```

```
##
## [[1]]$sigma
##             eruptions    waiting
## eruptions 0.06916767  0.4351676
## waiting   0.43516762 33.6972821
##
##
## [[2]]
## [[2]]$mu
## eruptions   waiting
##  4.289662 79.968115
##
## [[2]]$sigma
##           eruptions    waiting
## eruptions 0.1699684  0.9406093
## waiting   0.9406093 36.0462113
##
##
## $pi
## [1] 0.3558729 0.6441271
```

- give contour plot of the fitted density along with the points and

```
par(mfrow = c(1, 2), mar = 2.5 * c(1, 1, 1, 0.1))
plot(z, type = "l")

x1 = seq(1.5, 5, length.out = 101)
x2 = seq(40, 100, length.out = 100)
zz = matrix(0, nrow = length(x1), ncol = length(x2))
for (i in 1:length(x1)) zz[i, ] = dfnorm(cbind(x1[i], x2), gpar = te[[1]]$gpar)

contour(x1, x2, zz, nlevels = 100)
points(faithful, pch = 20)
```



comment why this is different than the solution in 3a).

- the solution 3a) is a local mode, this solution seems to be more likely to be the global mode because we obtained the same solution from many different starting values.

d) (4 Marks) When fitting finite mixture models it often of interest to know the predicted component memembership are given by the a posteriori probabilities (expected values). These are typically done with we using maximum *a posteriori* probabilities (MAP) given by

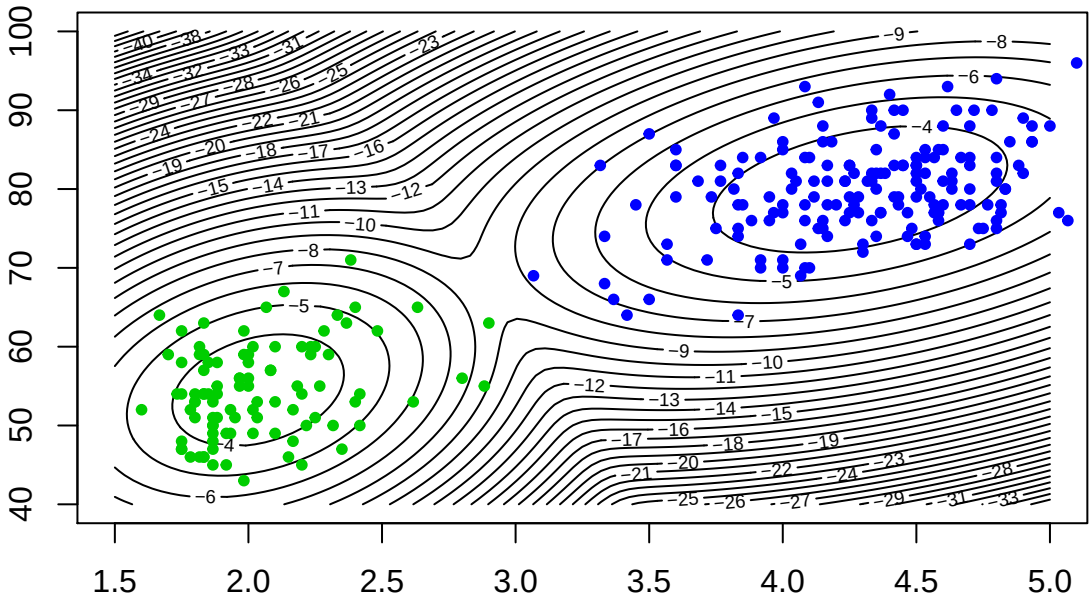
$$\text{MAP}(\widehat{z}_{ig}) = \begin{cases} 1 & \text{if } \max_h\{\widehat{z}_{ih}\} \text{ occurs in component } g, \\ 0 & \text{otherwise.} \end{cases}$$

Provide `R` code to calculate the MAP estimates.

```
MAP <- function(data, gpar) {
    w = e.step(data = data, gpar = gpar)
    z = apply(w, 1, function(z) {
        z = (1:length(z))[z == max(z)]
        return(z[1])
    })
    z = as.numeric(z)
    return(z)
}
```

Then plot the data and colour the different MAP estimates.

```
contour(x1, x2, zz, nlevels = 50)
points(faithful, pch = 20, col = MAP(faithful, gpar = te[[1]]$gpar) + 2)
```



4. [**12 Marks**] EM extensions,

a) (6 marks) Implement the Incremental EM with $m = 20$. Using the starting value given in 3a) perform 100 iterations and plot the observed log-likelihood from each iteration. Comment on the result.

Incremental EM updates only a subset of the n observations.

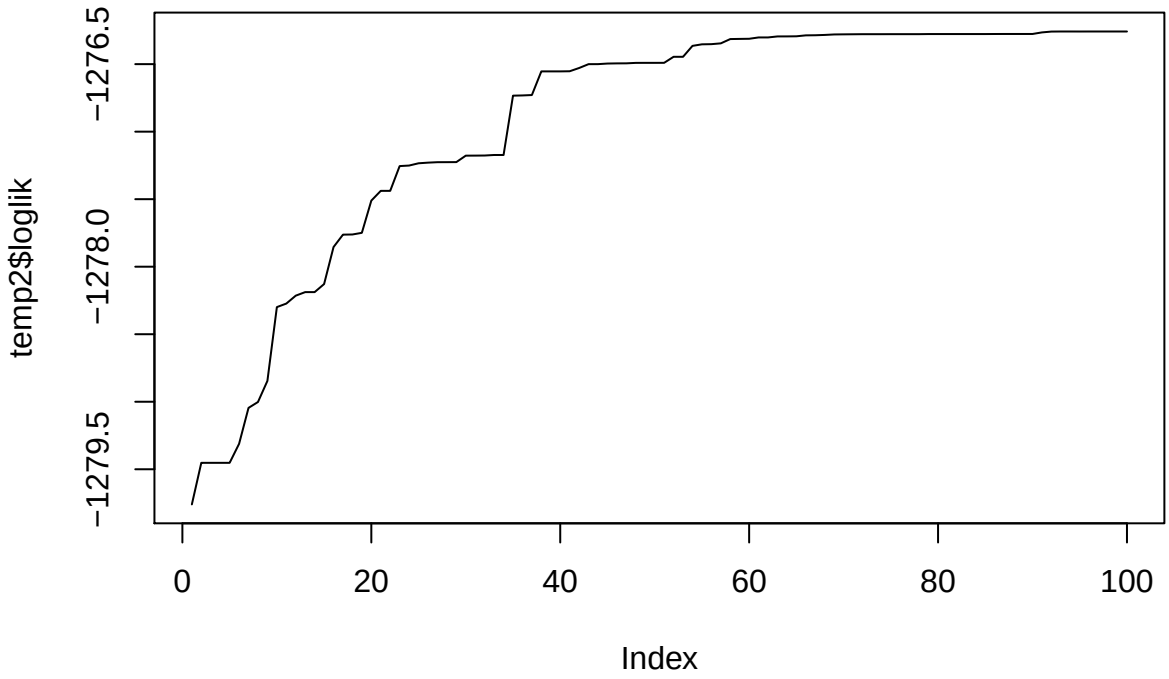
```
EMn2 <- function(data = NULL, gpar0 = NULL, G = 2, n = 10, m = 2) {
    val = list()
    if (is.null(gpar0))
        val$gpar = igpar(data = data, g = G, covtype = covtype) else val$gpar = gpar0
    val$loglik = numeric(n)

    # Initialize the weight matrix
    tempw = e.step(data = data, gpar = val$gpar)

    for (i in 1:n) {
        sub1 = sample(nrow(data), m)
        tempw[sub1, ] = e.step(data = data[sub1, ], gpar = val$gpar)
        val$gpar = m.step(data = data, w = tempw)

        val$loglik[i] = loglik(data = data, gpar = val$gpar)
    }

    return(val)
}
```

the strange starting value

```
gpar0 = list()
gpar0[[1]] = list()
gpar0[[1]]$mu = c(2, 60)
```

```
gpar0[[1]]$sigma = diag(2) * 0.1
gpar0[[2]] = list()
gpar0[[2]]$mu = c(2, 50)
gpar0[[2]]$sigma = diag(2) * 10
gpar0$pi = c(1, 9)/10
```

```
temp2 = EMn2(data = faithful, gpar0 = gpar0, G = 2, n = 100, m = 20)
plot(temp2$loglik, type = "l")
```



- The log-likelihood is monotonic but still gets stuck in the local mode.

b) (6 marks) Implement the Stochastic EM. Using the starting value given in 3a) perform 100 iterations and plot the observed log-likelihood from each iteration. Comment on the result.

Stochastic EM, randomly generates the E-step

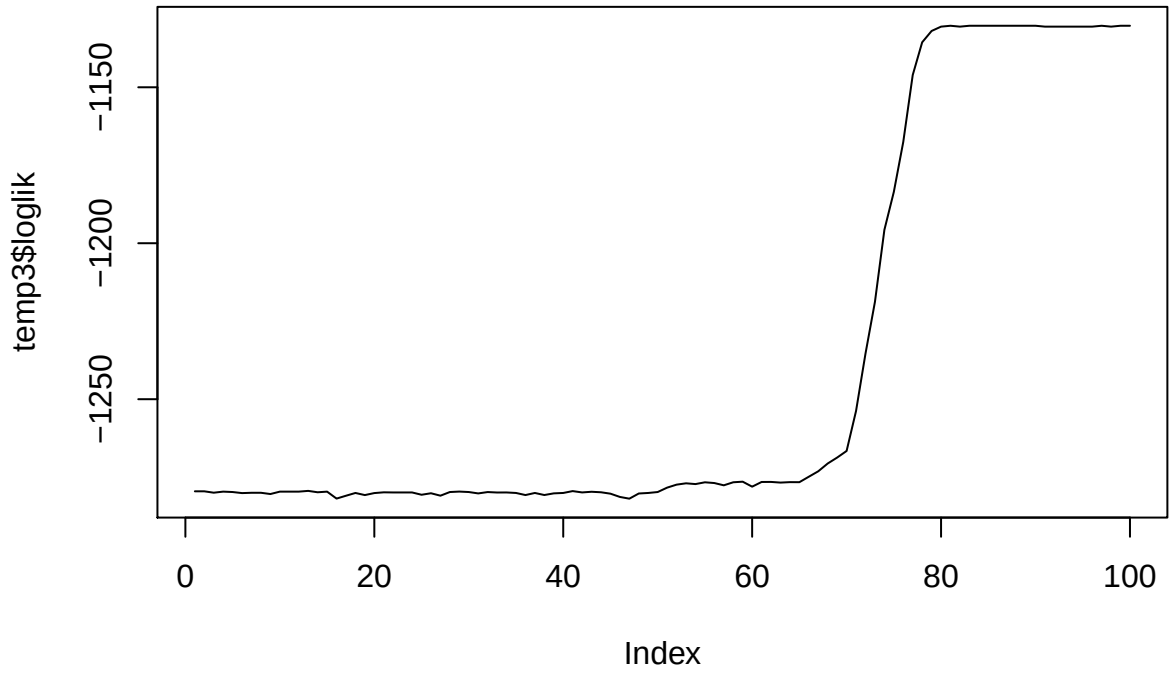
```
EMn3 <- function(data = NULL, gpar0 = NULL, G = 2, n = 10) {
    val = list()
    if (is.null(gpar0))
        val$gpar = igpar(data = data, g = G, covtype = covtype) else val$gpar = gpar0
    val$loglik = numeric(n)

    tempw = e.step(data = data, gpar = val$gpar)

    for (i in 1:n) {
        tempw = e.step(data = data, gpar = val$gpar)
        genz = apply(tempw, 1, function(z) {
            sample(z, size = 1, prob = z)
        })
```

```
        tempw = sweep(tempw, 1, genz, "==")

        val$gpar = m.step(data = data, w = tempw)

        val$loglik[i] = loglik(data = data, gpar = val$gpar)
    }

    return(val)
}
```

the strange starting value

```
gpar0 = list()
gpar0[[1]] = list()
gpar0[[1]]$mu = c(2, 60)
gpar0[[1]]$sigma = diag(2) * 0.1
gpar0[[2]] = list()
gpar0[[2]]$mu = c(2, 50)
gpar0[[2]]$sigma = diag(2) * 10
gpar0$pi = c(1, 9)/10
```

```
set.seed(440)
temp3 = EMn3(data = faithful, gpar0 = gpar0, G = 2, n = 100)
plot(temp3$loglik, type = "l")
```




```
max(temp3$loglik)
```

```
## [1] -1130.283
```

- The log-likelihood is not longer monotonic but does not gets stuck in the local mode.