EXERCISE 9

WEIYU LI

1. For the dataset ***faithful*** **in R, use Gaussian mixture model to estimate the joint density function of** ***eruptions*** **and** ***waiting*****. Write down your EM algorithm, and compare your result with** ***normalmixEM*** **in R package** ***mixtools*****.**

Solve. Suppose that the density of the gaussian mixture model is

$$f(x) = \sum_{i=1}^{C} \pi_i \phi(x; \mu_i, \Sigma_i),$$

where $\phi(x; \mu_i, \Sigma_i)$ is the density of 2-dim Gaussian with mean μ_i and covariance matrix Σ_i. Note that $\{\pi_i, \mu_i, \Sigma_i\}, i = 1, \ldots, C$ are parameters. Let $x_j, j = 1, \ldots, n$ be n samples, and $z_{ij} = P(x_j \text{ belongs to the } i\text{-th mixing part}), i = 1, \ldots, C, j = 1, \ldots, n$ be missing data. At iteration time t, given $\pi_i^t, \mu_i^t, \Sigma_i^t, i = 1, \ldots, C$, the E-step becomes

$$\begin{aligned} z_{kl}^t =& E[z_{kl} | \pi_i^t, \mu_i^t, \Sigma_i^t, x_j, i = 1, \ldots, C, j = 1, \ldots, n] \\ =& P(x_l \text{ belongs to the } k\text{-th mixing part} | \pi_i^t, \mu_i^t, \Sigma_i^t, x_l, i = 1, \ldots, C) \\ =& \frac{\pi_k^t \phi(x_l; \mu_k^t, \Sigma_k^t)}{\sum_{i=1}^{C} \pi_i^t \phi(x_l; \mu_i^t, \Sigma_i^t)}. \end{aligned}$$

It follows by the M-step (or MLE estimators)

$$\begin{aligned} \pi_i^{t+1} =& \frac{1}{n} \sum_{j=1}^{n} z_{ij}^t, \\ \mu_i^{t+1} =& \frac{\sum_{j=1}^{n} z_{ij}^t x_j}{\sum_{j=1}^{n} z_{ij}^t}, \\ \Sigma_i^{t+1} =& \frac{\sum_{j=1}^{n} z_{ij}^t (x_j - \mu_i^{t+1})(x_j - \mu_i^{t+1})^\top}{\sum_{j=1}^{n} z_{ij}^t}. \end{aligned}$$

Then we can iteratively perform the two steps until stopping criteria met.

```
library('mixtools')
library('mvtnorm')
# initialize
iter <- 20 # count for iteration times
x <- as.matrix(faithful)
plot(x)
# from here, we have an first impression on the dataset
# we guess there's a mixture of C=2 Gaussian distributions
# and we can approximate the initial value of means and variances
```

Date: 2019/11/04.

liweiyu@mail.ustc.edu.cn.

```
mv <- mvnormalmixEM(x) #results by 'mixtools' package
n <- nrow(x)
p <- ncol(x)
C <- 2
# initialize Z, mu, Sig, pi
Z <- matrix(c(rep(1, n), rep(0, (n - 1) * C)), n, C)
mu <- matrix(c(2, 50, 4, 80),C, p, byrow = T)
Sig <- matrix(rep(c(1, 0, 0, 1), C), ncol = p, byrow = T)
pi <- rep(1 / C, C)

# start EM here
for (t in 1:iter) {
  for (k in 1:n) {
    for (l in 1:C) {
      Z[k,l] <- pi[l] * dmvnorm(x[k,], mu[l,], Sig[(p * (l - 1) + 1):(p * l),])
    }
    Z[k,] <- Z[k,] / sum(Z[k,])
  }
  # update Z (E-step)
  pi <- colMeans(Z)
  # update pi (M-step-1)
  for (i in 1:C) {
    mu[i,] <- t(Z[,i]) %*% x / sum(Z[,i])
    # update mu (M-step-2)
    sumsig <- Z[1,i] * (x[1,] - mu[i,]) %*% t(x[1,] - mu[i,])
    for (k in 2:n) {
      sumsig <- sumsig + Z[k,i] * (x[k,] - mu[i,]) %*% t(x[k,] - mu[i,])
    }
    Sig[(p * (i - 1) + 1):(p * i),] <- sumsig / sum(Z[,i])
    # update Sigma (M-step-3)
  }
}
```

And we have the following outputs (copied from the console). Note that each row of mu denotes the mean of each Gaussian part, and each corresponding square matrix of Sig denotes the corresponding covariance matrix.

```
> # results for means
> mu
         [,1]     [,2]
[1,] 2.036388 54.47852
[2,] 4.289662 79.96812
> mv$mu
[[1]]
[1]  2.036389 54.478517

[[2]]
```

```
[1]  4.289662 79.968116

> # results for covariances/variances
> Sig
           [,1]       [,2]
[1,] 0.06916767  0.4351676
[2,] 0.43516762 33.6972821
[3,] 0.16996844  0.9406093
[4,] 0.94060932 36.0462113
> mv$sigma
[[1]]
           [,1]       [,2]
[1,] 0.06916774  0.4351683
[2,] 0.43516828 33.6972866

[[2]]
          [,1]       [,2]
[1,] 0.1699683  0.9406082
[2,] 0.9406082 36.0461985
```

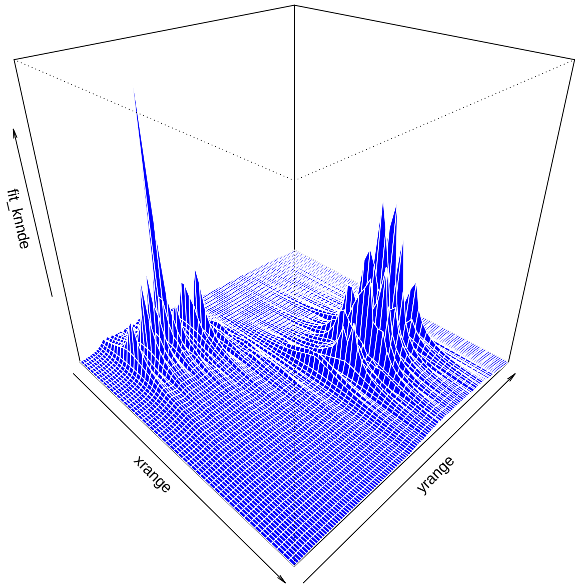
Therefore, our result is similar to the result given by the existing function. □

2. Write a kde function via *knn.dist* in R package *FNN*, and perform the k-nearest neighbor density estimation on the data in 1.

```
library('FNN')
x <- as.matrix(faithful)
cat(range(x[,1]), range(x[,2])) # print ranges of each column of x
# from this output, we give the ranges of our density estimation
xrange <- seq(from = 1, to = 6, by = 0.1)
yrange <- seq(from = 40, to = 100, by = 0.5)

knnde <- function(x, k, xrange, yrange){
  # input the data
  # output estimated points and their correspoding densities
  p <- ncol(x)
  n <- nrow(x)
  est_pt <- expand.grid(xrange, yrange)
  distance <- knnx.dist(x, est_pt, k)
  est_de <- matrix(k / (2 * n * distance[,k]), nrow = length(xrange))
  return(est_de)
}
k <- 5
fit_knnde <- knnde(x, k, xrange, yrange)
persp(xrange, yrange, fit_knnde, phi = 30, theta = 45, col = 'blue', border=0)
```

The figure is shown in the next page. □


fit_knnde
xrange
yrange