Assignment_2

Jieran Sun, Hui Jeong (HJ) Jung, Gumdmundur Björgvin Magnusson

2023-03-05

Problem 4

$$P(C_i|D_j) = \frac{P(D_j|C_i)P(C_i)}{P(D_j)} \tag{1}$$

Knowing that the two coins C_A and C_B each have a probability of 0.7 and 0.4 for obtaining heads, we can calculate each probability $P(C_i|D_j)$ with the above formula.

$$\begin{aligned} P(C_A|D_1) &= \frac{(0.3^6 * 0.7^4) * 0.6}{(0.3^6 * 0.7^4) * 0.6 + (0.6^6 * 0.4^4) * 0.4} && (2) \\ &= 0.1802056 \approx 0.18 && (3) \end{aligned}$$

To explain, $P(D_1|C_A) = 0.3^6 * 0.7^4$ as there were 4 heads and 6 tails in D_1. We are given that $P(C_A) = 0.6$. $P(D_1)$ can be obtained by calculating $P(D_1|C_A)P(C_A) + P(D_1|C_B)P(C_B)$.

$$\begin{aligned} P(C_A|D_2) &= \frac{(0.3^2 * 0.7^8) * 0.6}{(0.3^2 * 0.7^8) * 0.6 + (0.6^2 * 0.4^8) * 0.4} && (4) \\ &= 0.9705765143 \approx 0.97058 && (5) \\ P(C_B|D_1) &= \frac{(0.6^6 * 0.4^4) * 0.4}{(0.3^6 * 0.7^4) * 0.6 + (0.6^6 * 0.4^4) * 0.4} && (6) \\ &= 0.8197943509 \approx 0.8198 && (7) \\ P(C_B|D_2) &= \frac{(0.6^2 * 0.4^8) * 0.4}{(0.3^2 * 0.7^8) * 0.6 + (0.6^2 * 0.4^8) * 0.4} && (8) \\ &= 0.02942348571 \approx 0.0294 && (9) \end{aligned}$$

The updated mixture weights are

$$\begin{aligned} P(C_A) &= (P(C_A|D_1) + P(C_A|D_2))/(P(C_A|D_1) + P(C_A|D_2) + P(C_B|D_1) + P(C_B|D_2)) && (10) \\ &= 0.57539105715 \approx 0.57539 && (11) \\ P(C_B) &= (P(C_B|D_1) + P(C_B|D_2))/(P(C_A|D_1) + P(C_A|D_2) + P(C_B|D_1) + P(C_B|D_2)) && (12) \\ &= 0.4246089183 \approx 0.42461 && (13) \end{aligned}$$

Problem 5

```r
set.seed(2023)
```

(a) Load data

We first read the data stored in the file "coinflip.csv".

```r
# read the data into D
D <- data.table::fread("coinflip.csv")
# check the dimension of D
all(dim(D) == c(200, 100))
```

```
## [1] TRUE
```

(b) Initialize parameters

Next, we will need to initialize the mixture weights and the probabilities of obtaining heads. You can choose your own values as long as they make sense.

```r
# Number of coins
k <- 2
# Mixture weights (a vector of length k)
lambda <- runif(k)
lambda <- lambda/sum(lambda)
# Probabilities of obtaining heads (a vector of length k)
theta <- runif(k)
lambda <- runif(k)
lambda <- lambda/sum(lambda)
# Probabilities of obtaining heads (a vector of length k)
theta <- runif(k)
```

(c) The EM algorithm

Now we try to implement the EM algorithm. Please write your code in the indicated blocks.

```r
##' This function implements the EM algorithm for the coin toss problem
##' @param D Data matrix of dimensions 100-by-N, where N is the number of observations
##' @param k Number of coins
##' @param lambda Vector of mixture weights
##' @param theta Vector of probabilities of obtaining heads
##' @param tolerance A threshold used to check convergence
coin_EM <- function(D, k, lambda, theta, tolerance = 1e-2) {

  # expected complete-data (hidden) log-likelihood
  ll_hid <- -Inf
  # observed log-likelihood
  ll_obs <- -Inf
  # difference between two iterations
  diff <- Inf
  # number of observations
  N <- nrow(D)
```

```r
  # responsibilities
  gamma <- matrix(0, nrow = k, ncol = N)

  # run the E-step and M-step until convergence
  while (diff > tolerance) {

    # store old likelihood
    ll_obs_old <- ll_obs

    ############# E-step #############
    # Compute the responsibilities
    PofXgivenThetaTimesLambda <- gamma
    for (i in 1:N) {
      for (k_i in 1:k) {
          PofXgivenThetaTimesLambda[k_i,i] <-
            prod(D[i,]*theta[k_i] + (D[i,]-1)*(theta[k_i]-1))*lambda[k_i]
      }
    }
    gamma <- apply(PofXgivenThetaTimesLambda, 2, function(x) x/sum(x))
    # Update expected complete-data (hidden) log-likelihood

    ll_hid <- sum(gamma * log(PofXgivenThetaTimesLambda))

    # Update observed log-likelihood

    ll_obs <- sum(log(apply(PofXgivenThetaTimesLambda, 2, sum)))

    # Recompute difference between two iterations

    diff <- abs(ll_obs - ll_obs_old) # abs shouldnt be needed

    ### YOUR CODE ENDS ###

    ############# M-step #############
    # Recompute priors (mixture weights)

    lambda <- rowMeans(gamma)

    # Recompute probability of heads for each coin
    for (k_i in 1:k) {
          theta[k_i] <- sum(gamma[k_i,] * rowSums(D == 1))/(100*sum(gamma[k_i,]))
    }
        # Recompute probability of heads for each coin
    theta <- c(sum(gamma[1,] * rowSums(D == 1))/(100*sum(gamma[1,])),
               sum(gamma[2,] * rowSums(D == 1))/(100*sum(gamma[2,])))

    ### YOUR CODE ENDS ###

  }
  return(list(ll_hid = ll_hid, ll_obs = ll_obs, lambda = lambda, theta = theta, gamma = gamma ))
}
```

Run the EM algorithm:

```r
res <- coin_EM(D, k, lambda, theta)
```

(d) Results

Probability of heads:

```r
cat(sprintf(
  "The probability of heads are:
  coin 1: %.3f
  coin 2: %.3f",
  res$theta[1], res$theta[2]))
```

```
## The probability of heads are:
##   coin 1: 0.450
##   coin 2: 0.570
```

Mixture weights:

```r
cat(sprintf(
  "The mixture weights are:
  coin 1 : %.3f
  coin 2 : %.3f ",
  res$lambda[1], res$lambda[2]
))
```

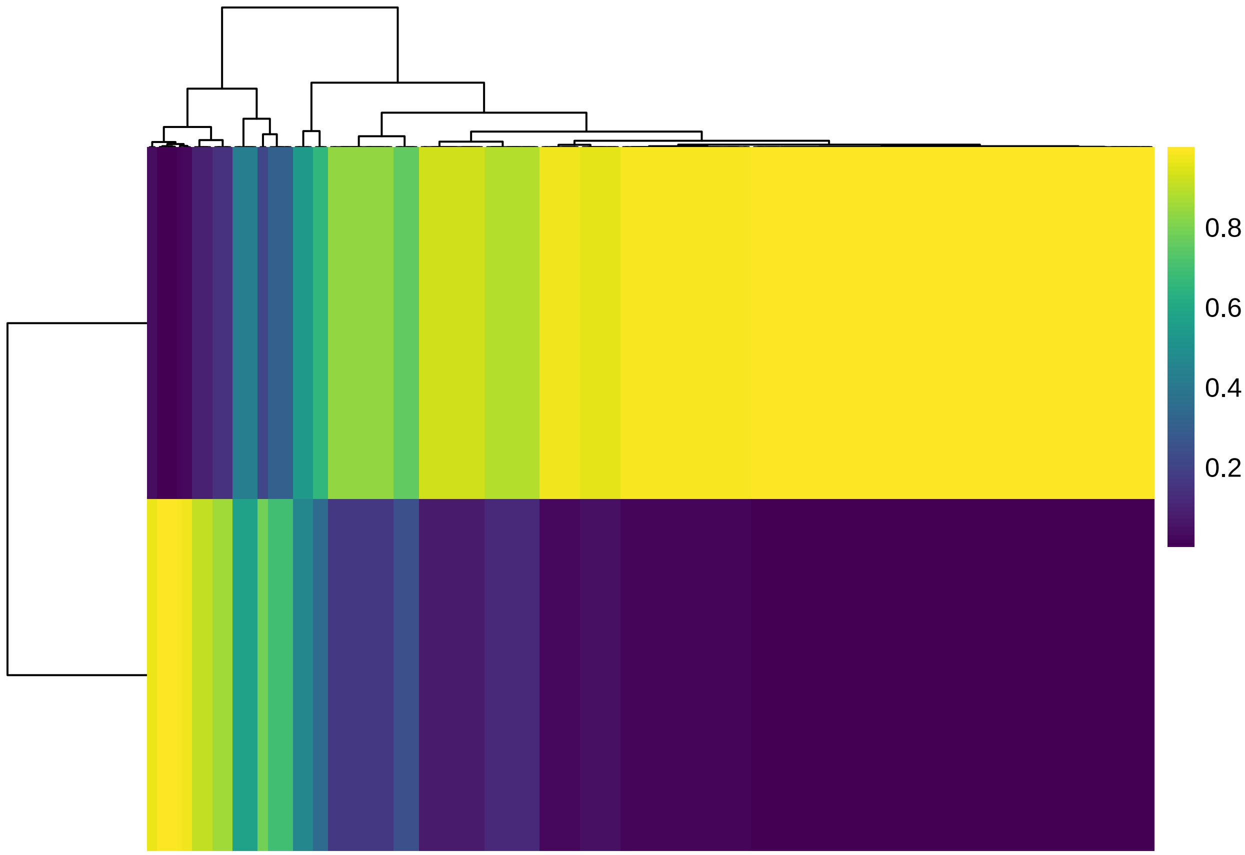
```
## The mixture weights are:
##   coin 1 : 0.833
##   coin 2 : 0.167
```

Heatmap of responsibilities:

```r
library(viridis)
```

```
## Loading required package: viridisLite
```

```r
pheatmap::pheatmap(res$gamma, color = viridis_pal(option = "D")(100))
```



How many observations belong to each coin?

```
res$gamma <- data.table::as.data.table(res$gamma)
cat("The number of observation for each coin is predicted below: \n")
```

```
## The number of observation for each coin is predicted below:
```

```
knitr::kable(table(as.numeric(res$gamma[, lapply(.SD, which.max)])))
```

Var1	Freq
1	171
2	29