Project 2. EM Algorithm

March 6, 2024

Contents

In this problem, you will implement the EM algorithm for the coin toss problem in R.

Below we provide you with a skeleton of the algorithm. You can either fill this skeleton with the required functions or write your own version of the EM algorithm. If you choose to do the latter, please also present your results using Rmarkdown in a clear fashion.

```
library(ggplot2)
set.seed(42)
```

(a) Load data

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

```
# read the data into D
D <- read.csv("coinflip.csv")
D <- as.matrix(D)
# 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.

```
# Number of coins
k <- 2
# Mixture weights (a vector of length k)
lambda <- rep(1, k) / k
# 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.

```
##' 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
```

```r
##' @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)
  # responsibilities
  gamma <- matrix(0, nrow = k, ncol = N)
  # keep track of lambda and theta during the optimisation
  lambda_all <- lambda
  theta_all <- theta
  # iteration number
  t <- 1

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

    ############## E-step ##############

    ### YOUR CODE STARTS ###

    c_P_obs_i_given_CA <- c()
    c_P_obs_i_given_CB <- c()
    # Compute the responsibilities
    for (i in 1:ncol(gamma)) {
      nheads <- sum(D[i,])
      ntails <- ncol(D) - sum(D[i,])
      gamma_num_CA <- lambda[1]*(theta[1]^nheads)*((1-theta[1])^ntails)
      gamma_num_CB <- lambda[2]*(theta[2]^nheads)*((1-theta[2])^ntails)

      # Saving likelihoods for the next step
      c_P_obs_i_given_CA <- append(c_P_obs_i_given_CA, (theta[1]^nheads)*((1-theta[1])^ntails))
      c_P_obs_i_given_CB <- append(c_P_obs_i_given_CB, (theta[2]^nheads)*((1-theta[2])^ntails))

      P_obs <- gamma_num_CA+gamma_num_CB

      # Compute the responsibilities for each observation
      gamma[, i] <- c(gamma_num_CA/P_obs, gamma_num_CB/P_obs)
    }

    # Update expected complete-data (hidden) log-likelihood
    ll_hid_new <- 0
    for (i in 1:ncol(gamma)) {
      ll_hid_new <- ll_hid_new+gamma[1,i]*log(lambda[1])+gamma[1,i]*log(c_P_obs_i_given_CA[i])
      ll_hid_new <- ll_hid_new+gamma[2,i]*log(lambda[2])+gamma[2,i]*log(c_P_obs_i_given_CB[i])
    }
```

```
    ll_hid <- c(ll_hid, ll_hid_new) # keep track of this quantity

    # Update observed log-likelihood
    l_q <- mean(log(c(gamma[1,], gamma[2,])))
    ll_obs_new <- ll_hid_new - l_q
    ll_obs <- c(ll_obs, ll_obs_new) # keep track of this quantity

    # Recompute difference between two iterations
    diff <- abs(ll_obs[t]-ll_obs[t+1])

    ### YOUR CODE ENDS ###

    ############## M-step ##############

    ### YOUR CODE STARTS ###

    # Recompute priors (mixture weights)
    lambda[1] <- mean(gamma[1,])
    lambda[2] <- mean(gamma[2,])

    lambda_all <- rbind(lambda_all, lambda) # keep track of this quantity

    # Recompute probability of heads for each coin

    # Summing gammas at the positions, where the coin turned heads
    exp_head_counts_CA <- 0
    exp_tail_counts_CA <- 0
    exp_head_counts_CB <- 0
    exp_tail_counts_CB <- 0
    for (i in 1:nrow(D)) {
      nheads <- sum(D[i,])
      ntails <- ncol(D) - sum(D[i,])
      exp_head_counts_CA <- exp_head_counts_CA + gamma[1,i]*nheads
      exp_head_counts_CB <- exp_head_counts_CB + gamma[2,i]*nheads
      exp_tail_counts_CA <- exp_tail_counts_CA + gamma[1,i]*ntails
      exp_tail_counts_CB <- exp_tail_counts_CB + gamma[2,i]*ntails
    }

    theta[1] <- exp_head_counts_CA/(exp_head_counts_CA+exp_tail_counts_CA)
    theta[2] <- exp_head_counts_CB/(exp_head_counts_CB+exp_tail_counts_CB)

    theta_all <- rbind(theta_all, theta) # keep track of this quantity

    ### YOUR CODE ENDS ###
    t <- t+1

  }

  return(list(ll_hid = ll_hid, ll_obs = ll_obs, lambda = lambda, theta = theta, gamma = gamma, lambda_a

}
```

Run the EM algorithm:

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

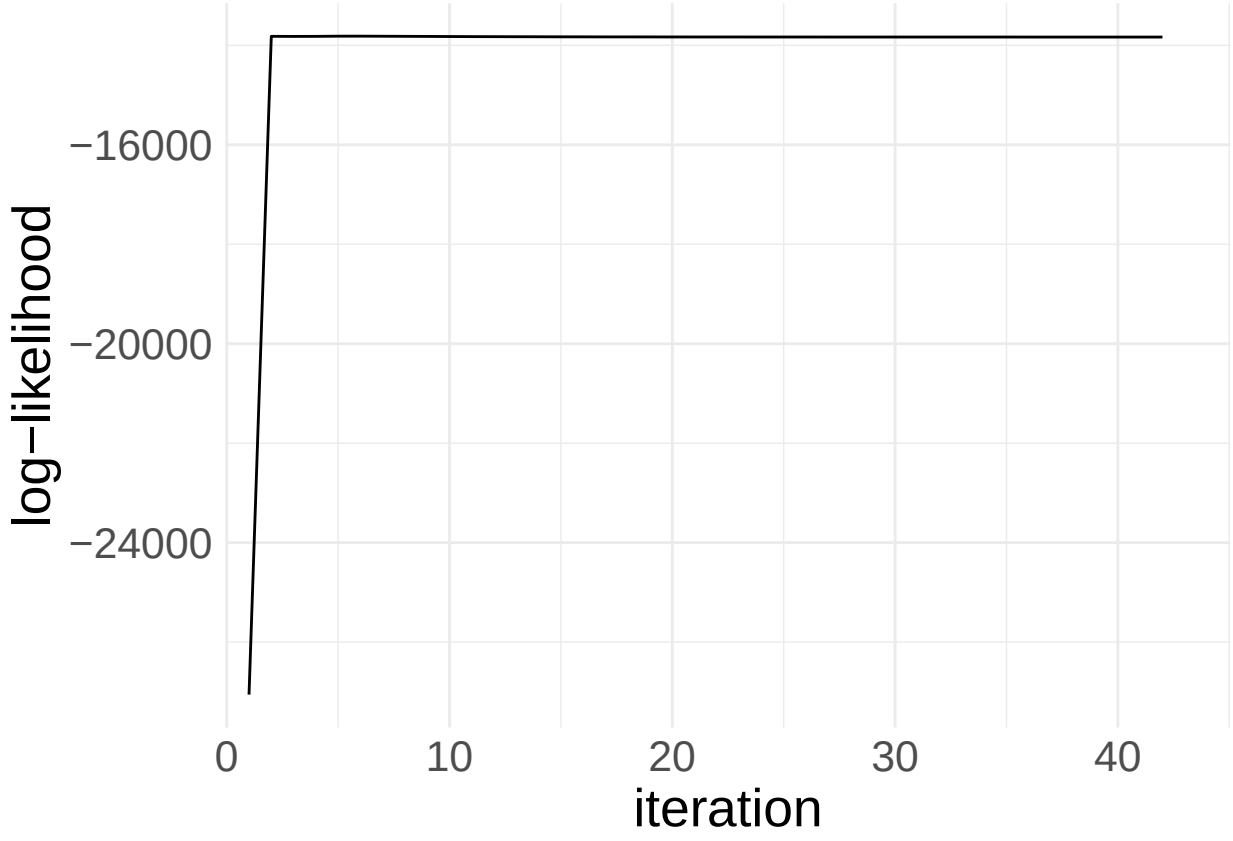
(d) Results

Log-likelihood through iterations

Log-likelihood rapidly increased after the first iteration. Over the remaining iterations it converged to the value of around -13000.

```
data <- data.frame(iterations=1:(length(res$ll_obs)-1),
  ll_obs=res$ll_obs[2:length(res$ll_obs)]
)

ggplot(data, aes(x=iterations, y=ll_obs)) +
  geom_line() +
  labs(x="iteration", y="log-likelihood") +
  theme_minimal() +
  coord_cartesian(xlim=c(2, length(res$ll_obs))) +
  theme(text = element_text(size=20))
```



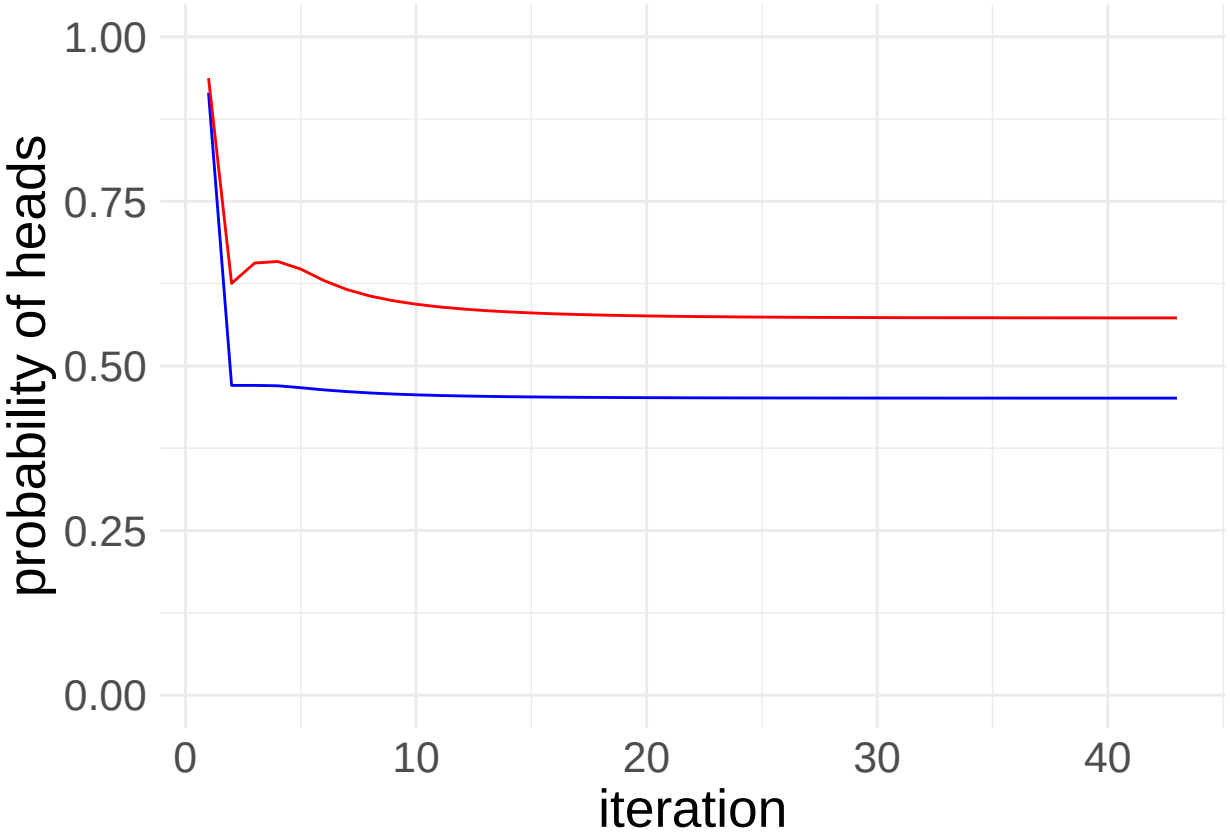
Probability of heads through iterations

Blue curve represents coin A, red curve represents coin B.

The probability of heads for both coins fastly decreased over the first iterations (up to around 10). After 10th iteration probability of coin A to turn heads converged to be around 0.45 and of coin B - 0.67.

```
theta_data <- data.frame(iterations=1:(length(res$theta_all[,1])),
  theta_CA=res$theta_all[,1], theta_CB=res$theta_all[,2]
)

ggplot(theta_data, aes(x=iterations)) +
  geom_line(aes(y=theta_CA), color="blue", linetype="solid") +
  geom_line(aes(y=theta_CB), color="red", linetype="solid") +
  labs(x="iteration", y="probability of heads") +
  theme_minimal() +
  theme(text=element_text(size=20)) +
  ylim(0, 1.0)
```



Mixture weights through iterations

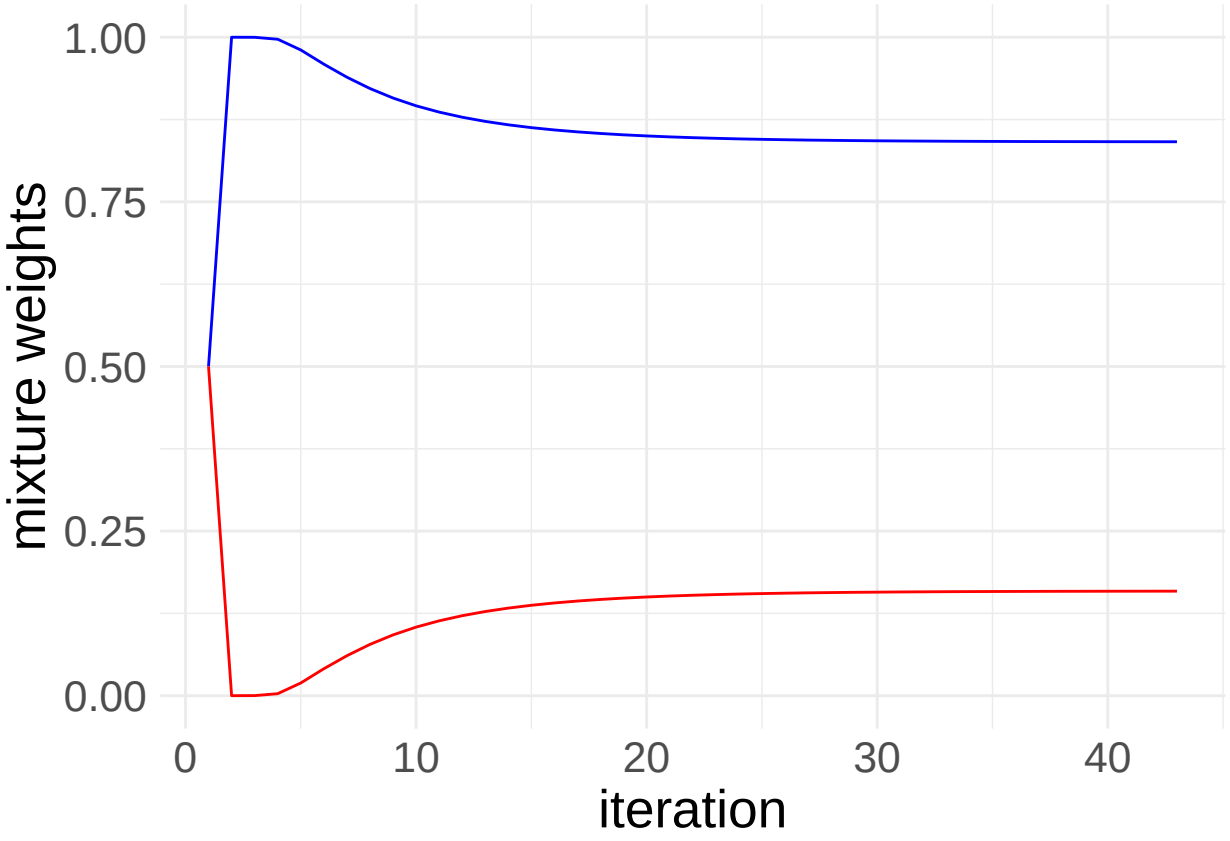
Blue curve represents coin A, red curve represents coin B.

The weight of coin A increased in the first iterations to be close to 1. In the next 10 iterations the weight of coin A converged to around 0.85 (weight of coin B converged symmetricaly to around 0.15).

```
lambda_data <- data.frame(iterations=1:(length(res$lambda_all[,1])),
  lambda_CA=res$lambda_all[,1], lambda_CB=res$lambda_all[,2]
)

ggplot(lambda_data, aes(x=iterations)) +
  geom_line(aes(y=lambda_CA), color="blue", linetype="solid") +
  geom_line(aes(y=lambda_CB), color="red", linetype="solid") +
  labs(x="iteration", y="mixture weights") +
```

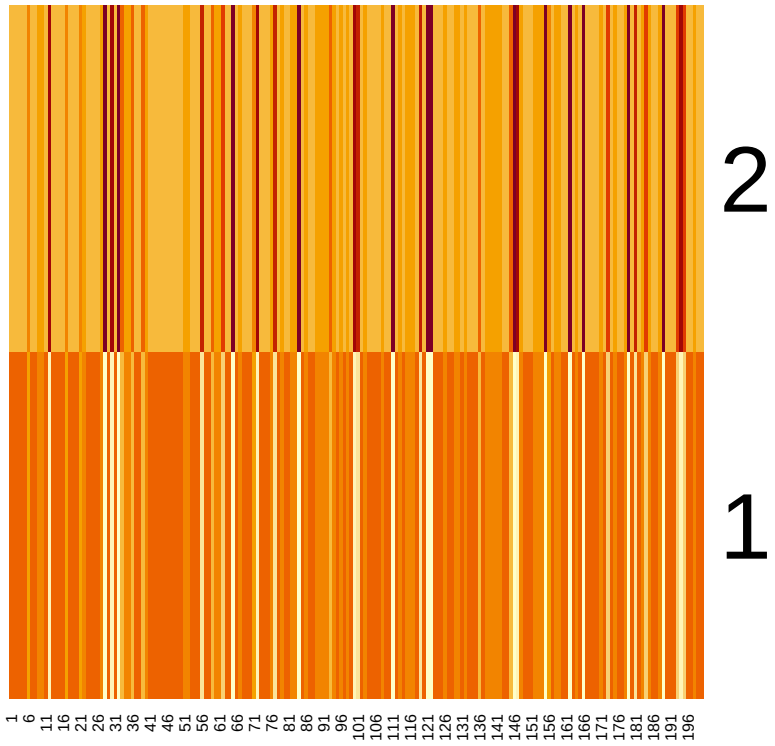
```
  theme_minimal() +
  theme(text=element_text(size=20))
```



Overall, it seems that it took up to 20 iterations for the algorithm to converge to the state similar to the end state (after 47 executed iterations).

Heatmap of responsibilities at final iteration:

```
heatmap(res$gamma, Colv=NA, Rowv=NA)
```



How many observations belong to each coin?

```
nobs_CA <- sum(res$gamma[1,] > res$gamma[2,])
nobs_CB <- sum(res$gamma[1,] < res$gamma[2,])
print(paste("Number of observations belonging to coin A: ", nobs_CA))
```

```
## [1] "Number of observations belonging to coin A:  171"
```

```
print(paste("Number of observations belonging to coin B: ", nobs_CB))
```

```
## [1] "Number of observations belonging to coin B:  29"
```

Render this .rmd into a pdf

```
library(rmarkdown)
render("2.Rmd", pdf_document(TRUE), "2.pdf")
```