# Lab1_Block2_Assignment2

2022-12-06

## Assignment 2. Mixture models

The EM algorithm was implemented for the Bernoulli mixture model. Note that the R code was adapted slightly from the provided template to produce output that better aligned with the R markdown format (though it produces the same results as using the provided template from the beginning for each value of $M$).

The Bernoulli mixture model is:

$$p(\boldsymbol{x}) = \sum_{m=1}^{M} \pi_m Bern(\boldsymbol{x}|\boldsymbol{\mu}_m)$$

where $\boldsymbol{x} = \{x_1, x_2, ..., x_D\}$ is a $D$-dimensional binary random vector, $\pi_m = p(y = m)$ and

$$Bern(\boldsymbol{x}|\boldsymbol{\mu}_m) = p(\boldsymbol{x}|y = m) = \prod_{d=1}^{D} \mu_{m,d}^{x_d}(1 - \mu_{m,d})^{(1-x_d)}$$

where $\boldsymbol{\mu} = (\mu_{m,1}, ..., \mu_{m,D})$ is a $D$-dimensional vector of probabilities.

The log likelihood of the dataset $\{\boldsymbol{x}_i^n\}$ is:

$$\sum_{i=1}^{n} \log p(\boldsymbol{x}_i)$$

The weights in the EM algorithm are calculated as $p(y = m|\boldsymbol{x}_i, \hat{\boldsymbol{\mu}})$

We have that $p(\boldsymbol{x}, y) = p(\boldsymbol{x}|y)p(y)$ and that $p(\boldsymbol{x}, y) = p(y|\boldsymbol{x})p(x)$, thus

$$p(y|\boldsymbol{x}) = \frac{p(y)p(\boldsymbol{x}|y)}{p(x)}$$

The weights then become:

$$p(y = m|\boldsymbol{x}_i) = \frac{p(y = m)p(\boldsymbol{x}|y = m)}{p(\boldsymbol{x})} = \frac{\hat{\pi}_m Bern(\boldsymbol{x_i}|\hat{\boldsymbol{\mu}}_m)}{\sum_{m=1}^{M} \hat{\pi}_m Bern(\boldsymbol{x_i}|\hat{\boldsymbol{\mu}}_m)}$$

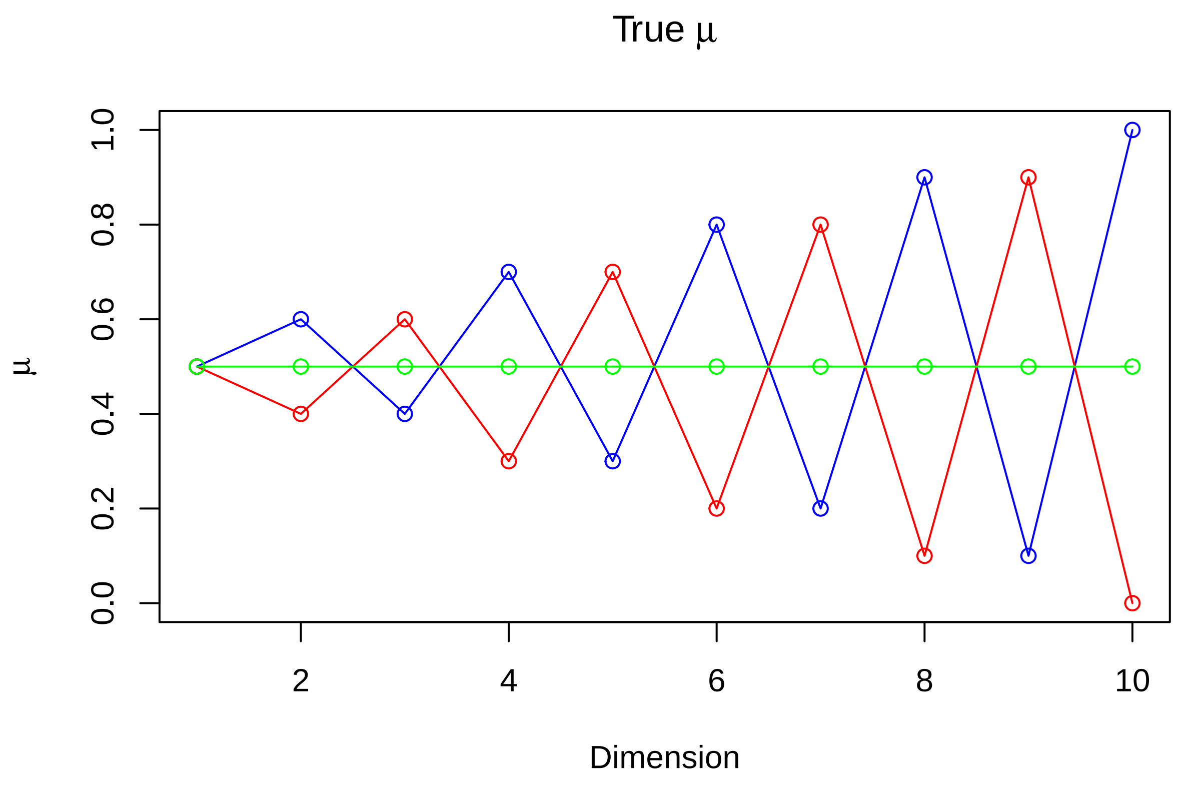
Finally, the parameters are updated according to:

$$\hat{\pi}_m = \frac{1}{n} \sum_{i=1}^{n} w_i(m),$$

$$\hat{\boldsymbol{\mu}}_{\boldsymbol{m}} = \frac{1}{\sum_{i=1}^{n} w_i(m)} \sum_{i=1}^{n} w_i(m)\boldsymbol{x}_i$$

The true parameter values are displayed below.

True $\mu$:

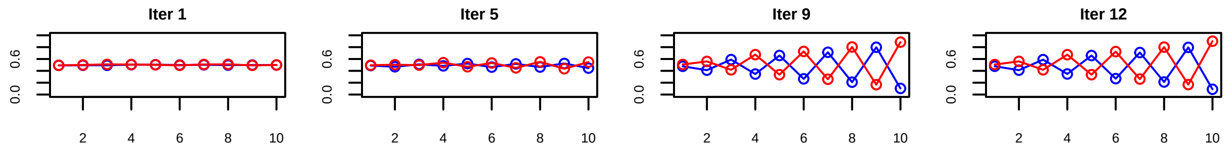


True $\pi$:

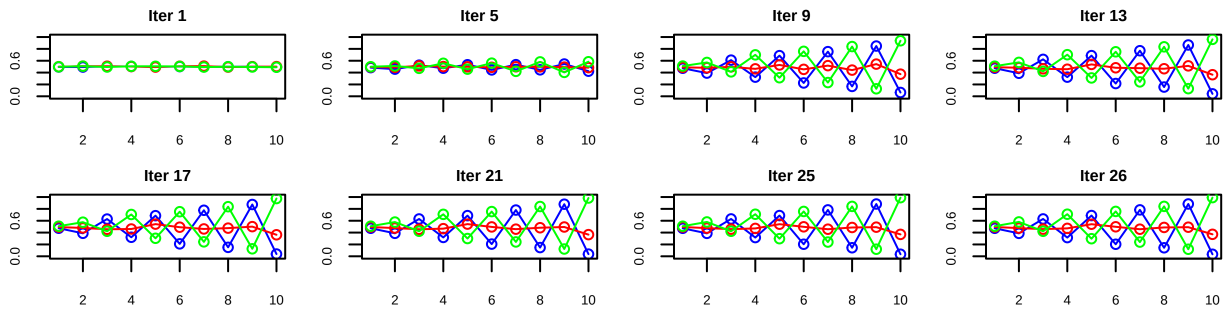
```
#> [1] 0.3333333 0.3333333 0.3333333
```

The EM algorithm was run for $M = 2, 3$ and 4. Below, estimates of $\mu$ are displayed for the first, then every fourth iteration as well as after the final iteration. For overview and completeness, the final $\mu$ are also displayed after the iteration plots, along with the estimated values for $\pi$.

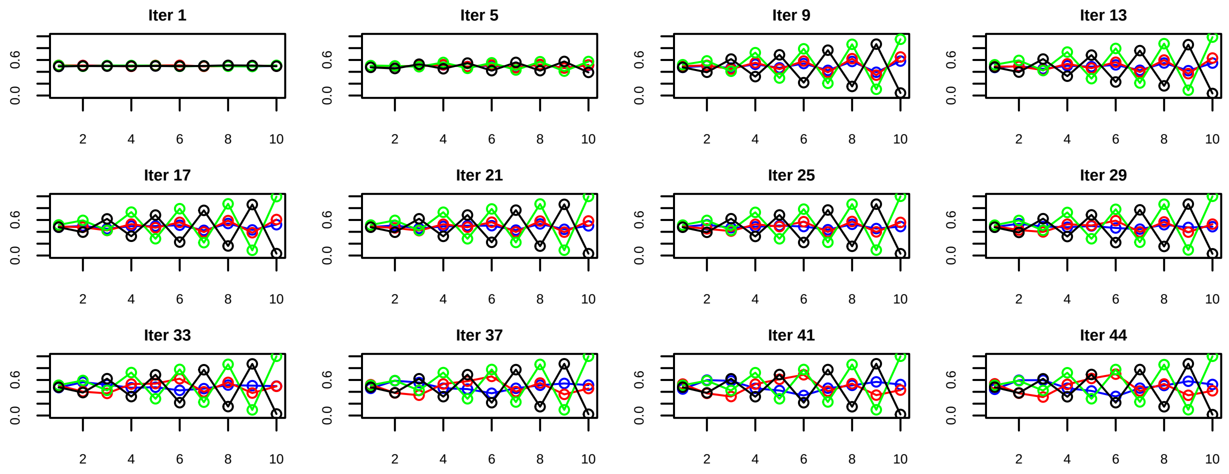
```
#> M = 2, start values of pi (rounded):0.49921, 0.50079
```



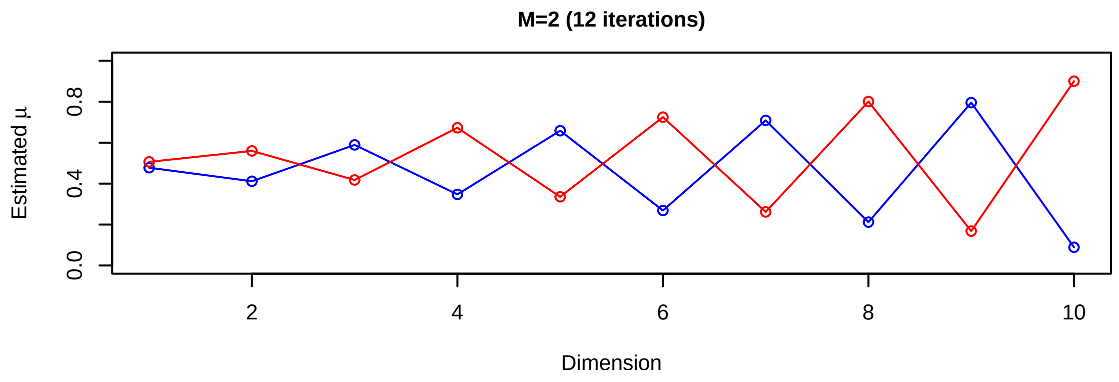
```
#> M = 3, start values of pi (rounded):0.33261, 0.33366, 0.33374
```

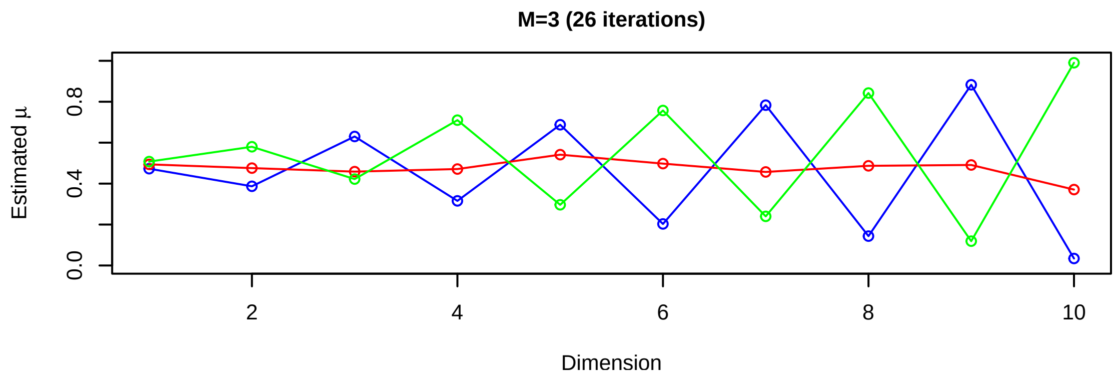


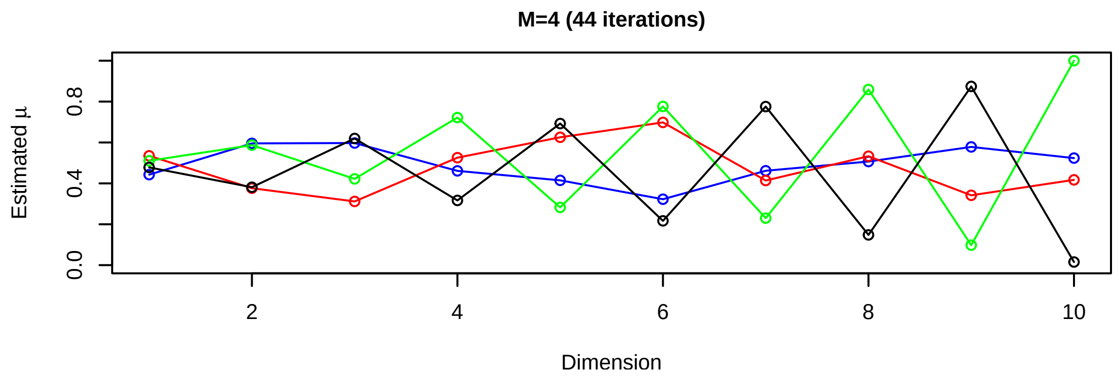
```
#> M = 4, start values of pi (rounded):0.24918, 0.24997, 0.25003, 0.25082
```



The values of the estimated $\mu$ and $\pi$ are displayed below:



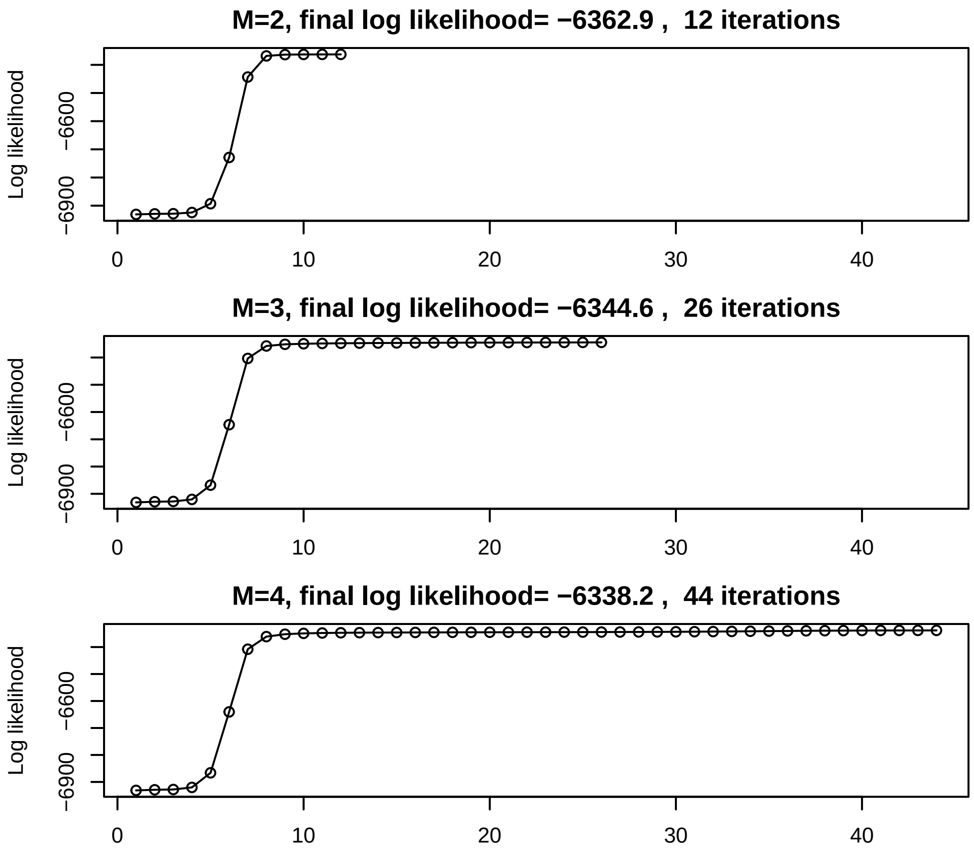




Estimated $\pi$:

```
#> Estimated pi M=2:

#> 0.497125 0.502875

#> Estimated pi M=3:

#> 0.3416794 0.2690298 0.3892909

#> Estimated pi M=4:

#> 0.1547196 0.1418652 0.3514089 0.3520062
```

The log likelihood was also plotted versus the iteration number:



It can be seen that the number of iterations of the EM algorithm increased with increasing $M$, and that the maximum log likelihood was obtained for $M = 4$, even though the true $M = 3$. This can be explained by that the higher the number of clusters ($M$), the better fit can be obtained to the data.

## Appendix: All code for this report

```
knitr::opts_chunk$set(comment = "#>",
                      echo = FALSE)
#-------------------------------------------------------------------------------#
#-------------------------------------------------------------------------------#
#- Assignment 2
#-------------------------------------------------------------------------------#
#-------------------------------------------------------------------------------#

#-------------------------------------------------------------------------------#
# Functions
#-------------------------------------------------------------------------------#

#- Function to calculate p(x|y) for given class m, with param mu
# mu is D-dimensional vector
# x is matrix of dimension nxD
bern <- function(x, mu) {
  D <- dim(x)[2]
  stopifnot(length(mu)==D)
  n <- dim(x)[1]
  res <- numeric(n)
  for (i in 1:n) {
    prod_d <- 1
    for (d in 1:D) {
      term_d <- (mu[d]^x[i,d])*((1-mu[d])^(1-x[i,d]))
      prod_d <- term_d*prod_d
    }
    res[i] <- prod_d
  }
  return(res)
}

#- Function to calculate p(x)
# mu_v - vector of mu
# pi_v -vector with pi
px <- function(x, mu_v, pi_v) {
  M <- dim(mu_v)[1]
  sum_px <- 0
  for (m in 1:M) {
    sum_px <- sum_px + (bern(x, mu_v[m,])*pi_v[m])
  }
  return(sum_px)
}
#-------------------------------------------------------------------------------#
# Main program
#-------------------------------------------------------------------------------#

#set.seed(1234567890)

max_it <- 100 # max number of EM iterations
min_change <- 0.1 # min change in log lik between two consecutive iterations
n=1000 # number of training points
```

```r
D=10 # number of dimensions
x <- matrix(nrow=n, ncol=D) # training data

true_pi <- vector(length = 3) # true mixing coefficients
true_mu <- matrix(nrow=3, ncol=D) # true conditional distributions
true_pi=c(1/3, 1/3, 1/3)
true_mu[1,]=c(0.5,0.6,0.4,0.7,0.3,0.8,0.2,0.9,0.1,1)
true_mu[2,]=c(0.5,0.4,0.6,0.3,0.7,0.2,0.8,0.1,0.9,0)
true_mu[3,]=c(0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5)
plot(true_mu[1,], type="o", col="blue", ylim=c(0,1), xlab = "Dimension",
     ylab=expression(mu), main=expression(paste("True ",mu)))
points(true_mu[2,], type="o", col="red")
points(true_mu[3,], type="o", col="green")

#- Note: this is done in loop to obtain same seed as would have been obtained
#  when running the original script separately for each M
# #- Producing the training data
# for(i in 1:n) {
#   m <- sample(1:3,1,prob=true_pi)
#   for(d in 1:D) {
#     x[i,d] <- rbinom(1,1,true_mu[m,d])
#   }
# }

true_pi

#M=3 # number of clusters
llik <- matrix(nrow=max_it, ncol=3) # log likelihood of the EM iterations
mu_list <- list() # save final mu for each M
pi_list <- list() # save final pi for each M
iter_list <- list() # save number of iter for each M

for (M in 2:4) {

  cat(paste0("M = ",M))
  set.seed(1234567890)
  #- Producing the training data
  for(i in 1:n) {
    m <- sample(1:3,1,prob=true_pi)
    for(d in 1:D) {
      x[i,d] <- rbinom(1,1,true_mu[m,d])
    }
  }

  w <- matrix(nrow=n, ncol=M) # weights
  pi <- vector(length = M) # mixing coefficients
  mu <- matrix(nrow=M, ncol=D) # conditional distributions
  #llik <- vector(length = max_it) # log likelihood of the EM iterations

  #- Random initialization of the parameters
  pi <- runif(M,0.49,0.51)
  pi <- pi / sum(pi)
  for(m in 1:M) {
```

```
  mu[m,] <- runif(D,0.49,0.51)
}
#pi
#mu
pi_str <- toString(round(pi,5))
cat(paste0(", start values of pi (rounded):", pi_str))
par(mfrow=c(3,4),mai = c(0.3, 0.25, 0.2, 0.15))
for(it in 1:max_it) {
  #- Plot 1st then each 4th iter

  if(it==1 | ((it-1)%%4==0)) {
    plot(mu[1,], type="o", col="blue", ylim=c(0,1), main=paste0("Iter ", it),
         cex.main=0.8, cex.axis=0.6)
    points(mu[2,], type="o", col="red")
    if (dim(mu)[1] >= 3) {
      points(mu[3,], type="o", col="green")
    }
    if (dim(mu)[1] == 4) {
      points(mu[4,], type="o", col="black")
    }
  }
  Sys.sleep(0.5)

  #- E-step: Computation of the weights
  ###############################################################################
  #- Your code here
  sum_px <- px(x=x, mu_v=mu, pi_v=pi)
  for (m in 1:M) {
    w[,m] <- (pi[m]*bern(x, mu[m,]))/sum_px
  }
  ###############################################################################

  #- Log likelihood computation.
  ###############################################################################
  #- Your code here
  llik[it,M-1] <- sum(log(sum_px))

  ###############################################################################

  #cat("iteration: ", it, "log likelihood: ", llik[it,M-1], "\n")
  #flush.console()

  #-  Stop if the lok likelihood has not changed significantly
  ###############################################################################
  #- Your code here
  # The value of the observed data log-likelihood increases at each iteration of
  # the procedure, unless it has reached a stationary point
  if (it > 1) {
    llik_ch <- llik[it,M-1]-llik[it-1,M-1]
    if (llik_ch < min_change) {
      break
    }
  }
```

```
    ###############################################################################

    #- M-step: ML parameter estimation from the data and weights
    ###############################################################################
    #- Your code here
    for (m in 1:M) {
      pi[m] <- (1/n)*sum(w[,m])
      mu[m,] <- (1/sum(w[,m]))*colSums(w[,m]*x)
    }
    ###############################################################################
  }

  # - Plot likelihood vs iterations
  #plot(llik[1:it,M-1], type="o")

  #- Plot last iteration

  plot(mu[1,], type="o", col="blue", ylim=c(0,1), main=paste0("Iter ", it),
       cex.main=0.8, cex.axis=0.6)
  points(mu[2,], type="o", col="red")
  if (dim(mu)[1] >= 3) {
    points(mu[3,], type="o", col="green")
  }
  if (dim(mu)[1] == 4) {
    points(mu[4,], type="o", col="black")
  }

  #pi
  #mu

  mu_list[[M-1]] <- mu
  pi_list[[M-1]] <- pi
  iter_list[[M-1]] <- it
}

# - Plot final parameters
par(mfrow=c(3,1), mai=c(0.5,0.8,0.35,0.5))
for (M in 2:4) {
  mu <- mu_list[[M-1]]
  it <- iter_list[[M-1]]
  plot(mu[1,], type="o", col="blue", ylim=c(0,1),xlab="Dimension",
       ylab=expression(paste("Estimated ",mu)),
       main=paste0("M=", M, " (",  it, " iterations)"), cex.main=1)
  points(mu[2,], type="o", col="red")
  if (dim(mu)[1] >= 3) {
    points(mu[3,], type="o", col="green")
  }
  if (dim(mu)[1] == 4) {
    points(mu[4,], type="o", col="black")
  }
}
```

```r
cat("Estimated pi M=2:\n")
cat(pi_list[[1]],"\n")


cat("Estimated pi M=3:\n")
cat(pi_list[[2]],"\n")

cat("Estimated pi M=4:\n")
cat(pi_list[[3]],"\n")

# - Plot likelihood vs iterations
par(mfrow=c(3,1), mai=c(0.3,1,0.3,1))
plot(llik[1:it,1], type="o", xlab="Iteration", ylab="Log likelihood",
     main=paste("M=2, final log likelihood=",
                round(llik[iter_list[[1]],1],1), ", ",
                iter_list[[1]], "iterations"))
plot(llik[1:it,2], type="o", xlab="Iteration", ylab="Log likelihood",
     main=paste("M=3, final log likelihood=",
                round(llik[iter_list[[2]],2],1), ", ",
                iter_list[[2]], "iterations"))
plot(llik[1:it,3], type="o", xlab="Iteration", ylab="Log likelihood",
     main=paste("M=4, final log likelihood=",
                round(llik[iter_list[[3]],3],1), ", ",
                iter_list[[3]], "iterations"))
```