03_EM_iteration

Frances Lin

11/24/2020

```r
library(here)
library(tidyverse)
library(tibble)
library(stats)
library(ggplot2)
library(gridExtra)
devtools::load_all()
```

Load data

```r
data <- readRDS(here("results", "data.rds"))
head(data)
```

```
## # A tibble: 6 x 2
##   component  value
##   <chr>      <dbl>
## 1 A         -1.21
## 2 A          0.277
## 3 A          1.08
## 4 A         -2.35
## 5 A          0.429
## 6 A          0.506
```

Load initial values and rename to df

```r
df_summary_kmeans <- readRDS(here("results", "df_summary_kmeans.rds"))
df <- df_summary_kmeans
df
```

```
## # A tibble: 2 x 6
##   cluster   mean   var    sd  size    pi
##     <int>  <dbl> <dbl> <dbl> <int> <dbl>
## 1       1  4.03  1.01  1.00    102  0.51
## 2       2 -0.229 0.814 0.902    98  0.49
```

Putting it all together

E-step: Calculate posterior probability (or soft labelling) to pass it to M-step using Bayes Rule and obtain log likelihood to check for convergence

```
#e_step(x = data$value, mu = df$mean, sd = df$sd, pi = df$pi)
#good
```

Delete this section if the e_step() can run

```
# E-step
#
# At the beginning of E-step, we pass
# 1. x = data$value --- results from 00_simulated_data
# 2. mu = df$mean   --- results from 01_initialization_kmeans
# 3. sd = df$sd
# 4. pi = df$pi
#
# At the end of E-step, we return
# 1. log likelihood for checking convergence
# 2. prior probability for M-step

e_step <- function(x, mu, sd, pi){
  # top of the posterior equation (or complete likelihood)
  prob_1 <- dnorm(x, mu[1], sd[1]) * pi[1]
  prob_2 <- dnorm(x, mu[2], sd[2]) * pi[2]

  # bottom of the posterior equation (or normalizing constant)
  normalizer <- prob_1+ prob_2

  # posterior probability
  post_1 <- prob_1 / normalizer
  post_2 <- prob_2 / normalizer

  # log likelihood
  #likelihood <- prob_1 + prob_2
  #log_likelihood <- log(prob_1 + prob_2)
  result <- sum(log(prob_1 + prob_2)) #result <- sum(log(normalizer))

  # return log likelihood and posterior probability
  list(
    "log_likelihood" = result,
    "posterior_prob" = cbind(post_1, post_2)
  )
}

# Recall that we want to set
  # x = data$value for cluster 1 for example
  # mu = df$mean[1]
  # sd = df$sd[1]
  # pi = df$pi[1]

#e_step(x = data$value, mu = df$mean, sd = df$sd, pi = df$pi)
E_step <- e_step(x = data$value, mu = df$mean, sd = df$sd, pi = df$pi)
#E_step
```

M-step: Replace hard labelling with posterior probability (or soft labelling) and optimize the parameters using MLE and return the final estimates if convergence happens

```
E_step <- e_step(x = data$value, mu = df$mean, sd = df$sd, pi = df$pi)
#E_step
```

```
#m_step(x = data$value, posterior = E_step$posterior_prob)
#good
```

Delete this section if the m_step() can run

```
# M-step
#
# At the beginning of E-step, we pass
# 1. x = data$value --- results from 00_simulated_data
# 2. posterior_prob --- results from e_step()
#
# At the end of M-step, we return
# 1. mu current
# 2. sd current
# 3. pi current

m_step <- function(x, posterior){
  # Obtain mean
  mean_1 <- sum(posterior[, 1] * x) / sum(posterior[, 1])
  mean_2 <- sum(posterior[, 2] * x) / sum(posterior[, 2])

  # Obtain variance
  var_1 <- sum(posterior[, 1] * (x - mean_1) ^ 2) / sum(posterior[, 1])
  var_2 <- sum(posterior[, 2] * (x - mean_2) ^ 2) / sum(posterior[, 2])

  # Obtain pi
  pi_1 <- sum(posterior[, 1]) / length(x)
  pi_2 <- sum(posterior[, 2]) / length(x)

  # Return parameters (mu, sd, pi)
  list(
    "mu" = c(mean_1, mean_2),
    "sd" = c(sqrt(var_1), sqrt(var_2)),     # store sd instead of var
    "pi" = c(pi_1, pi_2)
  )
}

#m_step(x = data$value, posterior = E_step$posterior_prob)
M_step <- m_step(x = data$value, posterior = E_step$posterior_prob)
#M_step
```

Iterations until convergence (i.e. change is minimal) using log likelihood

```r
# Set the #s of iterations
iterations <- 50

for(i in 1:iterations){
  if (i == 1){
  # Initialization
  # Pass the initial parameters as a result of kmeans
  e_out <- e_step(x = data$value, mu = df$mean, sd = df$sd, pi = df$pi)
  m_out <- m_step(x = data$value, posterior = e_out$posterior_prob)

  # Set to current log likelihood
  current_log_likelihood <- e_out$log_likelihood

  # Store log likelihood vector for plotting
  log_likelihood <- e_out$log_likelihood

  } else {
  # Repeat E and M steps until convergence
  # Pass the parameters as a result of the 1st (and on) EM iteration
  e_out <- e_step(x = data$value, mu = m_out$mu, sd = m_out$sd, pi = m_out$pi)
  m_out <- m_step(x = data$value, posterior = e_out$posterior_prob)

  # Incrementally store log likelihood vector for plotting
  log_likelihood <- c(log_likelihood, current_log_likelihood)

  # Check for convergence
  # Compare current log likelihood to current + 1 log likelihood
  check <- abs(current_log_likelihood - e_out$log_likelihood)

  if(check < 1e-3){
    # Converge
    break
  } else {
    # Do not converge
    # Reset current + 1 to current and repeat E and M steps
    current_log_likelihood <- e_out$log_likelihood
  }
  }
}

# Return log likelihood vector for plotting
log_likelihood
```

```
##  [1] -407.5568 -407.5568 -407.3297 -407.2470 -407.1993 -407.1712 -407.1545
##  [8] -407.1446 -407.1387 -407.1351 -407.1330 -407.1317
```

```r
# Return current (or final) log likelihood element for checking
current_log_likelihood
```

```
## [1] -407.1317
```

```r
# Return #s of iteractions for plotting
n_iterations <- length(log_likelihood)
n_iterations
```

```
## [1] 12
```

```r
# Return convergence for checking
check
```

```
## [1] 0.000784191
```

```r
# Return for reporting
#e_out

# Return for reporting
m_out
```

```
## $mu
## [1]  3.9457404 -0.2821921
##
## $sd
## [1] 1.087458 0.863983
##
## $pi
## [1] 0.5261278 0.4738722
```

Estimates improve with N(0,1) and N(4, 1), as compared to N(0,1) and N(2,1)

```r
# Combine EM results
result_1_parameters <- tibble(
  "mean" = c(m_out$mu[1], m_out$mu[2]),
  "sd" = c(m_out$sd[1], m_out$sd[2]),
  "pi" = c(m_out$pi[1], m_out$pi[2])
)
result_1_parameters
```

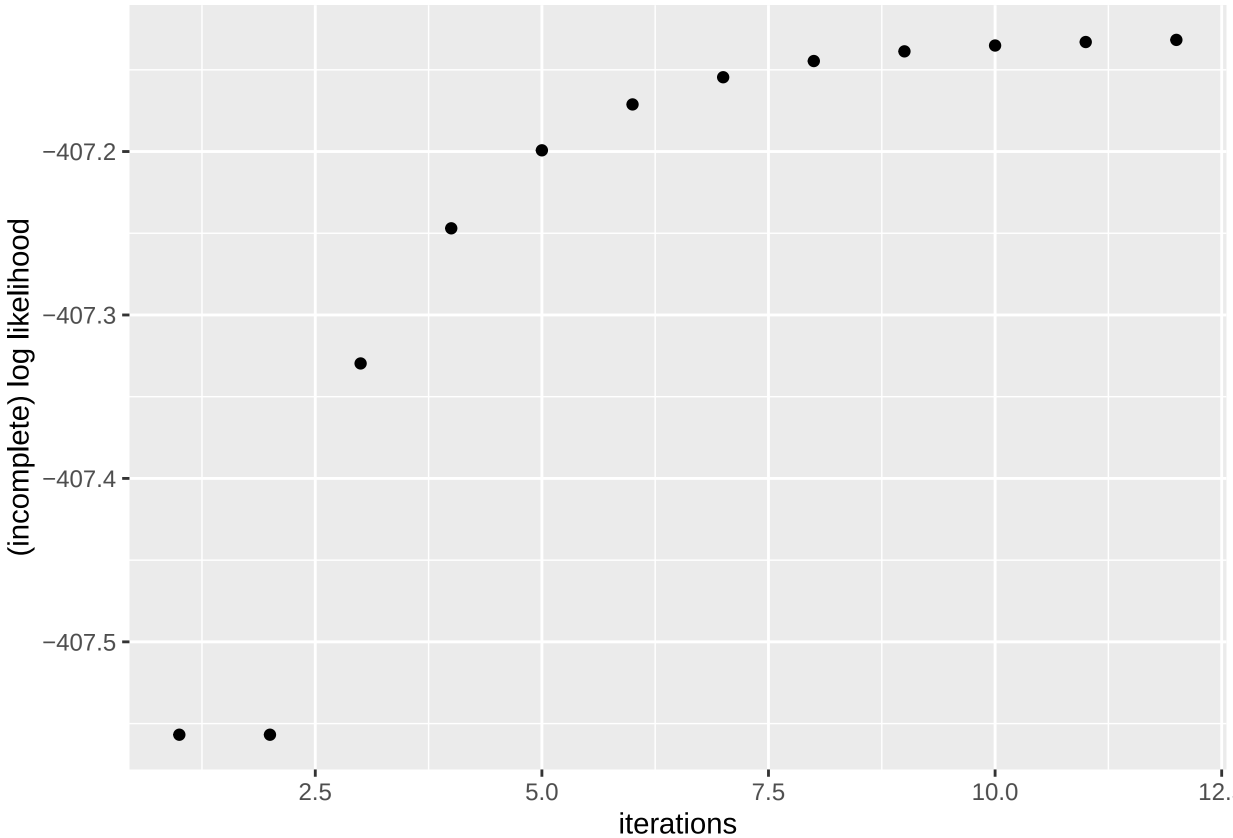
```
## # A tibble: 2 x 3
##     mean    sd    pi
##    <dbl> <dbl> <dbl>
## 1  3.95  1.09  0.526
## 2 -0.282 0.864 0.474
```

Converge now at 12nd iteraction, as compared to 31st iteraction

```r
# Combine EM results
result_2_max_log_like <- tibble(
  "max_log_likelihood" = current_log_likelihood,
  "#s of iteractions" = n_iterations
)
result_2_max_log_like
```

```
## # A tibble: 1 x 2
##   max_log_likelihood `#s of iteractions`
##                <dbl>               <int>
## 1              -407.                  12
```

```r
# Plot (incomplete) log likelihood
result_3_plot_log_likelihood <- qplot(x = 1:n_iterations, y = log_likelihood,
                                      xlab = "iterations",
                                      ylab = "(incomplete) log likelihood")
result_3_plot_log_likelihood
```



If time permits: Plot simulated data in histogram and overlay a density curve

Save out results

```r
write_rds(result_1_parameters, here("results", "result_1_parameters.rds"))
write_rds(result_2_max_log_like, here("results", "result_2_max_log_like.rds"))
write_rds(result_3_plot_log_likelihood, here("results", "result_3_plot_log_likelihood.rds"))
write_rds(e_out, here("results", "e_out.rds"))
write_rds(m_out, here("results", "m_out.rds"))

write_rds(log_likelihood, here("results", "log_likelihood.rds")) # fix reporting error
write_rds(current_log_likelihood, here("results", "current_log_likelihood.rds"))
write_rds(n_iterations, here("results", "n_iterations.rds"))
```

```r
write_rds(check, here("results", "check.rds"))

#write_rds(plot_EM, here("results", "result_4_plot_EM.rds"))
```