02_EM_step

Frances Lin

11/18/2020

Description

This .Rmd file contains the very rough draft of how the `e_step()` and the `m_step()` functions are created and can be skipped. Details (descriptions, codes, and usages) of these functions can be found in the R folder.

Load packages

```
library(here)
library(tidyverse)
library(tibble)
library(stats)
```

Load data

```
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

```
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  2.10  0.737 0.859   110  0.55
## 2       2 -0.471 0.445 0.667    90  0.45
```

E-step: Obtain posterior probability (or soft labelling) using Bayes Rule

```
# prior probability (or mixing weight)
pi_1 <- df$pi[1]
pi_2 <- df$pi[2]

# top of the posterior equation (or incomplete likelihood and prior probability)
prob_1 <- dnorm(x = data$value, mean = df$mean[1], sd = df$sd[1]) * pi_1
prob_2 <- dnorm(x = data$value, mean = df$mean[2], sd = df$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
```

E-step: Log-likelihood of the first interaction

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

```
## [1] -354.7605
```

Check to see if it makes sense

```
data <- data %>%
  mutate(
    post_1 = post_1,
    post_2 = post_2
  )
# since value 1 to 6 belongs to component A
# should assign more weights to the 1st component
# should assign more weights to the 2nd component for value 195 to 200
# also recall that
# mu_1 <- 0 for component A (or 1)
# sd_1 <- 1
# mu_2 <- 4 for component B (or 2)  # mu_2 <- 2 for component B (or 2)
# sd_2 <- 1
head(data)
```

```
## # A tibble: 6 x 4
##   component  value    post_1 post_2
##   <chr>      <dbl>     <dbl>  <dbl>
## 1 A         -1.21  0.00105    0.999
## 2 A          0.277 0.158      0.842
```

```
## 3 A           1.08  0.878      0.122
## 4 A          -2.35  0.0000750  1.000
## 5 A           0.429 0.263      0.737
## 6 A           0.506 0.332      0.668
```

```
tail(data)
```

```
## # A tibble: 6 x 4
##   component   value post_1   post_2
##   <chr>       <dbl>  <dbl>    <dbl>
## 1 B           4.17   1.000  6.06e-10
## 2 B           2.50   1.000  5.76e- 5
## 3 B           2.62   1.000  2.75e- 5
## 4 B           1.03   0.849  1.51e- 1
## 5 B           2.16   1.000  4.35e- 4
## 6 B          -0.0782 0.0435 9.57e- 1
```

M-step: Optimize the parameters using maximum likelihood

```
# Obtain mean
mean_1 <- sum(post_1 * data$value) / sum(post_1)
mean_2 <- sum(post_2 * data$value) / sum(post_2)

# Obtain variance
var_1 <- sum(post_1 * (data$value - mean_1) ^ 2) / sum(post_1)
var_2 <- sum(post_2 * (data$value - mean_2) ^ 2) / sum(post_2)

# Obtain pi
pi_1 <- sum(post_1) / length(data$value)
pi_2 <- sum(post_2) / length(data$value)
```

Parameters of the first interaction

```
df_EM <- tibble(
  custer = c(1, 2),
  mean = c(mean_1, mean_2),
  var = c(var_1, var_2),
  sd = c(sqrt(var_1), sqrt(var_2)),
  pi = c(pi_1, pi_2)
)
df_EM
```

```
## # A tibble: 2 x 5
##   custer   mean   var    sd    pi
##    <dbl>  <dbl> <dbl> <dbl> <dbl>
## 1      1  2.03  0.841 0.917 0.565
## 2      2 -0.476 0.485 0.696 0.435
```