Coding Assignment 4

Due Monday, October 31

Part I: EM Algorithm for Gaussian Mixtures (3-pt)

Implement the EM algorithm for a p-dimensional Gaussian mixture model with G components:

$$\sum_{k=1}^{G} p_k \cdot \mathsf{N}(x; \mu_k, \Sigma).$$

Your function should output the estimated parameters as well as the log-likelihood evaluated with the estimated parameters.

Store the output as a list in R with four components

- prob: G-dimensional probability vector $(p_1, \dots, p_G)$
- mean: p-by-G matrix with the k-th column being μ_k, the p-dimensional mean for the k-th Gaussian component;
- Sigma: p-by-p covariance matrix Σ shared by all G components;
- loglik: a number equal to $\sum_{i=1}^{n} \log \Big[\sum_{k=1}^{G} p_k \cdot \mathsf{N}(x; \mu_k, \Sigma) \Big]$.

Structure of your code should look like the following.

```
Estep <- function(data, G, para){
  # Return the n-by-G probability matrix
  }

Mstep <- function(data, G, para, post.prob){
  # Return the updated parameters
  }

loglik <- function(data, G, para){
  # compute loglikelihood
}

myEM <- function(data, itmax, G, para){
  for(t in 1:itmax){
        post.prob <- Estep(data, G, para)
        para <- Mstep(data, G, para, post.prob)
      }
  # update para$loglik
  return(para)
  }
 }
```

Test your code on the `faithful` data from `R` package `mclust` with $G = 2$ and $G = 3$. The result from your function should agree with the result from `mclust` after 20 iterations, up to the precision level specified in the sample code.

Implement all the computation using your own code. Students are not allowed to use built-in functions to evaluate multivariate Gaussian densities; implement that calculation by yourself.

In addition to the necessary R/Python code and output, your submission should include the following:

1. expression of the marginal (or the incomplete) likelihood function $\prod_{i=1}^n p(x_i \mid p_{1:G}, \mu_{1:G}, \Sigma)$ or its log, which is the objective function we aim to maximize;
2. expression of the complete likelihood function $\prod_{i=1}^n p(x_i, Z_i \mid p_{1:G}, \mu_{1:G}, \Sigma)$ or its log, which is the function we work with in the EM algorithm;
3. expression of the distribution of Z_i's at the E-step;
4. expression of the objective function you aim to maximize (or minimize) at the M-step;
5. derivation and the updating formulae for $p_{1:G}$, $\mu_{1:G}$, and Σ at the M-step.

I like to use uppercase letters for latent variables, such as Z_i, but it's just a personal preference; feel free to use lowercase letters.

If you do not know how to include math formulae in R Markdown, you can write your derivation on a piece of paper, take a photo, and then insert it into your HTML file or save it as a separate PDF file.

Part II: EM Algorithm for HMM (2-pt)

Consider a hidden Markov model (HMM) whose outcome is a sequence of discrete random variables taking **Three** unique values. Implement the Baum-Welch (i.e., EM) algorithm to estimate the parameters and the Viterbi algorithm to output the most probable latent sequence.

Test your code with the data provided on Campuswire with **Two** hidden states. The result from your function should agree with the result from R package `HMM` after 100 iterations, up to the precision level specified in the sample code.

More details are provided in the sample code posted on Campuswire.

Different from Part I, you do not need to include any derivations for Part II.

What do you need to submit?

- A Markdown (or Notebook) file in HTML format, which should contain all necessary code and the corresponding output/results.
- Set the seed at the beginning of your code to be the last 4-dig of your University ID.
- Name your file starting with `Assignment_4_xxxx_netID..`, where "xxxx" is the last 4-dig of your University ID and make sure the same 4-dig is used as the seed in your code.