TD2 - The EM algorithm for Gaussian mixtures

Nicolas Jouvin

```r
library(dplyr)
library(ggplot2)
library(knitr)
```

Maths part

Recall the Gaussian mixture model where $X = \{x_i\}_{i=1}^n \subset \mathbb{R}$ are **i.i.d** with p.d.f.

$$p_\theta(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x \mid \mu_k, \sigma_k^2) \tag{1}$$

The model parameters are $\theta = \{\pi_k, \mu_k, \sigma_k^2\}$, with $\sum_l \pi_l = 1$.

This model can be interpreted as a **latent variable model** with unobserved latent variables $Z = \{z_i\}_{i=1}^n$, where $z_i \in \{0,1\}^K$ is a binary vector encoding for the cluster asignment of x_i.

The complete data are $\{x_i, z_i\}_{i=1}^n$.

Questions (on paper)

1. Write the complete-data log-likelihood of the model

$$\log p_\theta(X, Z)$$

2. Derive the maximum **complete** likelihood estimators π_k^C, μ_k^C and $\sigma_k^{2,C}$ solution of

$$\arg\max_\theta \log p_\theta(X, Z)$$

3. **Bonus** redo the same calculations in dimension d. *Hint*: For A a positive matrix, $\nabla_A \log\det(A) = A^{-1}$.

4. Let $Z = \{z_i\}_i \sim q$, with q any distribution on $\{0,1\}^{Kn}$. Denote $q_{ik} = q(z_{ik} = 1)$. Use the preceding questions to give the solution of

$$\arg\max_{\theta} \mathbb{E}_{Z\sim q}\left[\log p_\theta(X,Z)\right]$$

in terms of q_{ik}.

Programmation part : implementing your own EM in 1-d

About this exercise

The goal of this practictal exercise is to implement your own EM algorithm. You will be guided step-by-step throughout the question.
Hint: I strongly advise against using ChatGPT or any equivalent LLM to "help" you. One needs to make mistake to learn the caveats of coding such algorithms !
We will apply it on the following synthetic dataset 1-dimension.

```r
data<-data.frame(Var=c(-3.3,-4.4,-1.9,3.3,2.5,3.2,0.3,0.1,-0.1,-0.5),
                 partition1=c(1,1,1,2,2,2,2,2,1,1),
                 partition2=c(1,3,2,1,3,2,1,3,2,1)
                 )
t(data)
```

```
           [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
Var        -3.3 -4.4 -1.9  3.3  2.5  3.2  0.3  0.1 -0.1  -0.5
partition1  1.0  1.0  1.0  2.0  2.0  2.0  2.0  2.0  1.0   1.0
partition2  1.0  3.0  2.0  1.0  3.0  2.0  1.0  3.0  2.0   1.0
```

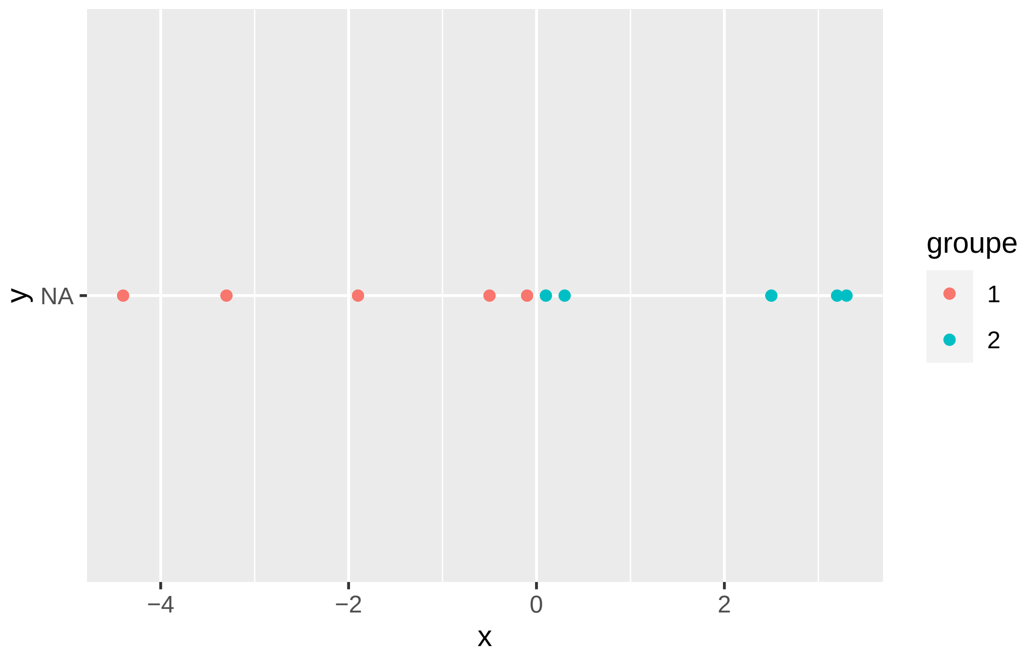
Introductory question

How do you interpret the two following graphic: does one partition seems better than the other ?

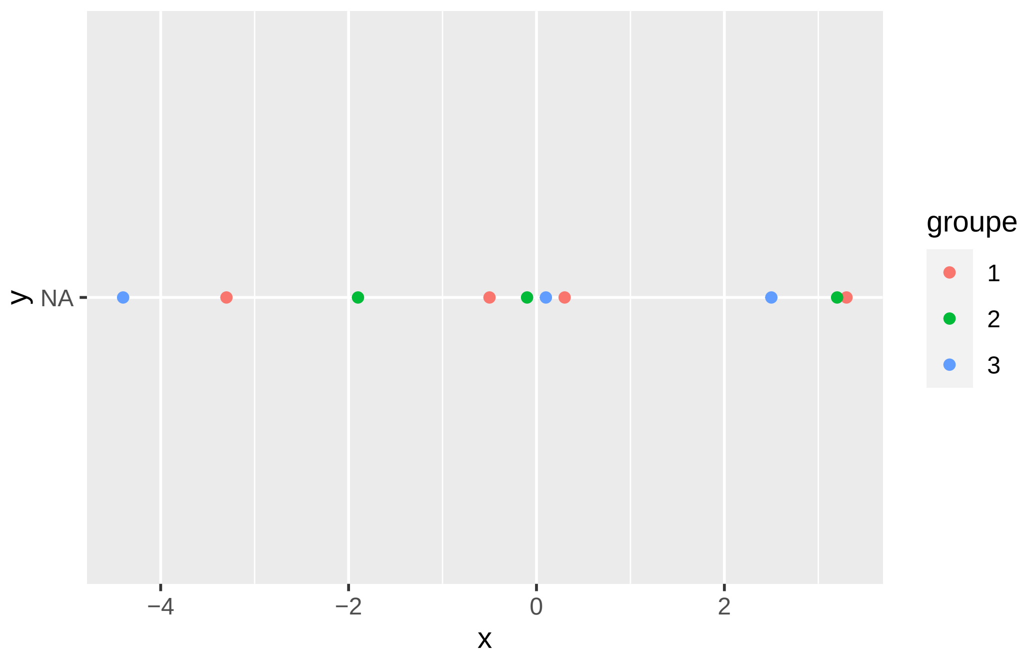
```r
library(ggplot2)

plot_data <- function(x, partition) {
  # function that plot the 1D data vector x with color
  # according to an argument partition
  #
  # return: a ggplot graph
  df = data.frame(x = x, groupe = factor(partition))
  gg = ggplot(df) + geom_point(aes(x=x, y = NA, color=groupe))
  return(gg)
}

gg_part1 = plot_data(data$Var, data$partition1)
gg_part2 = plot_data(data$Var, data$partition2)
print(gg_part1)
```




```r
print(gg_part2)
```



Question 1 : initialisation de l'algorithme

Coder la fonction `initEM(x, partition)` qui retourne une liste `param` avec slots

- `param$pi` : l'init $\pi^{(0)}$
- `param$theta` une liste avec slot
  - `param$theta$mu`: l'init $\mu^{(0)}$
  - `param$theta$sigma2` : l'init ${\sigma^2}^{(0)}$

Question 2: Etape E

Coder une fonction `Estep(x, param)` qui calcule et renvoie les $\tau_{ik} \propto \pi_k \mathcal{N}(x_i \mid \mu_k, \sigma_k^2)$. **Astuce** calculer en log-space pour mieux représenter les petites quantités (numériquement, $e^{-1000} \approx 0$ tandis que $\log(e^{-1000}) = -1000$).

$$\log \tau_{ik} = \log \pi_k + \log \mathcal{N}(x_i \mid \mu_k, \sigma_k^2) - cte_i$$

La constante de normalisation peut-être calculée comme suit

1. *Méthode naïve (instable):* $cte_i = \log \sum_l \exp\{\log \tau_{il}\}$
2. *LogSumExp trick:* $cte_i = m_i + \log \sum_l \exp\{\log \tau_{il} - m_i\}$ avec $m_i := \max_l \log \tau_{il}$. This ensures an $exp(0)$ somewhere in the sum, hence avoiding numerical underflows.

Question 3: étape M

En utilisant les formules données dans les slides, coder

- une fonction `compute_PI(tau)` qui calcule $\hat{\pi}$.
- une fonction `compute_mu(tau, x)` qui calcule $\hat{\mu}_k$ pour tout k.
- une fonction `compute_sigma2(tau, mu, x)` qui calcule $\hat{\sigma}^2$.

Les assembler dans une fonction `Mstep(x, tau)` qui fait la M-step de l'algorithme EM.

Question 4: calcul de la vraisemblance marginale

Coder une fonction `compute_mixture_llhood = function(x, param)` qui retourne la log-vraisemblance marginale des observations :

$$\log p(x_1, \dots, x_n \mid \theta, \pi) = \sum_i \log \sum_k \pi_k \mathcal{N}(x_i \mid \mu_k, \sigma_k^2)$$

Astuce: utiliser la vectorisation de `dnorm()` pour éviter une boucle sur les observations.

Question 4 (BONUS): vérifier l'égalité de l'ELBO

D'après le cours, on sait que l'ELBO $\mathcal{L}(q, \theta) = \mathbb{E}_q [\log p_\theta(X, Z)] + \mathcal{H}(q)$ est égale à la vraisemblance marginale ssi

$$q = p_\theta(Z \mid X).$$

Le vérifier numériquement sur cet exemple.

Question 5: algorithme EM

Mettre toute ces fonctions ensemble dans une fonction `EMgauss1D(X, K, partition_init, max.iter, rtol)`. L'algorithme s'arrête après un nombre fixé `max.iter` d'itérations ou quand la différence relative entre les vraisemblances successive à t et $t+1$ est inférieur au seuil `rtol` :

$$|\frac{\log p_{\theta^{(t+1)}}(X) - \log p_{\theta^{(t)}}(X)}{\log p_{\theta^{(t)}}(X)}| < rtol.$$

La fonction devra retourner une liste avec les slots

- `logliks` : la valeur successive des vraisemblance le long des itération (pour l'afficher dans un graphique par exemple)
- `param` : les paramètre finaux à la fin de l'algorithme

- `tau` : les probabilité a posteriori d'appartenir à chacuns des groupes. **Attention**, elles doivent calculées avec les valeurs des paramètres **finales**.

Tester votre fonction avec les hyper-paramètres suivants:

```
max.iter = 20
rtol = 1e-6
K = 2
partition_init = data$partition1
```

Note On peut jouer avec le paramètre `max.iter` et `rtol` pour la convergence de la vraisemblance. Ne pas oublier que l'on converge uniquement vers un maximum **local** (en fait pire : un point selle) de cette dernière. On peut ensuite visualiser les estimateur des paramètre $(\hat{\pi_k}, \hat{\mu_k}, \hat{\Sigma_k})_k$

Question 7: affichage et diagnostic

1. Afficher l'évolution de la log-vraisemblance en fonctions des itérations de l'algorithme.
2. Afficher les paramètres estimés dans chacuns des clusters.

Clustering (partitionnement) à la fin de l'EM

On affecte les points selon leurs probabilité à posteriori après convergence de l'algorithme à l'itération (T). Cela revient à estimer $z_i, \forall i$:

$$\forall i, \quad \hat{z}_{ik^\star} = 1 \text{ où : } k^\star = \arg\max_{k=1,\dots,K} p(z_{ik} = 1 \mid x_i, \theta^{(T)}) = \frac{\pi_k^{(T)} \mathcal{N}(x_i, \mu_k^{(T)}, \Sigma_k^{(T)})}{\sum_l \pi_l^{(T)} \mathcal{N}(x_i, \mu_l^{(T)}, \Sigma_l^{(T)})}$$

Dans notre code, cela revient à faire un argmax par ligne sur la matrix `res_em$tau`.

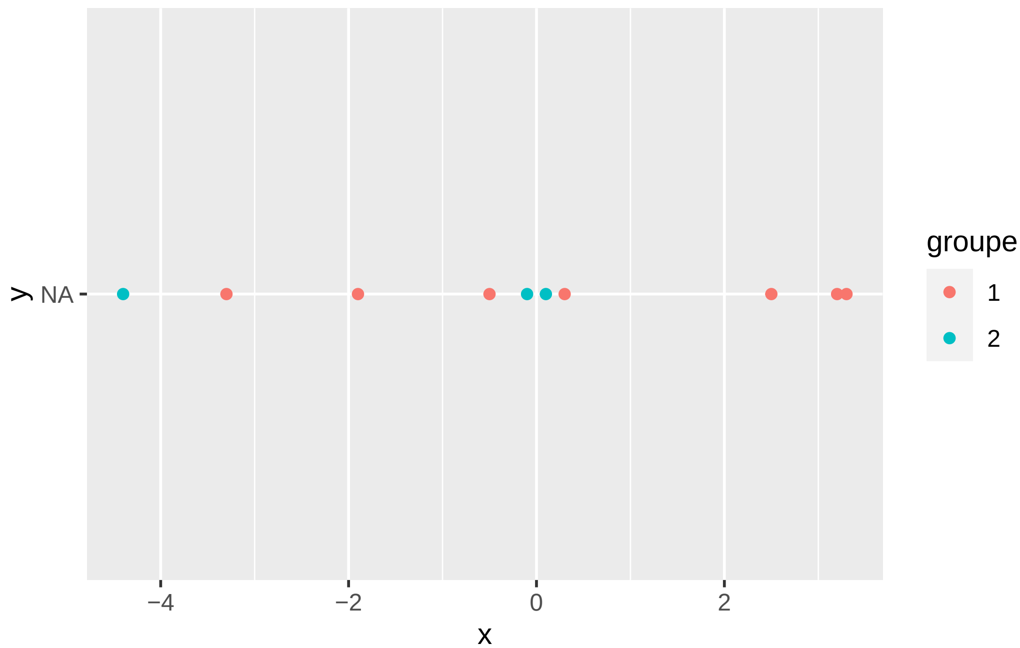
Question 8: visualisation

Calculer la partition obtenue grâce à cette méthode et faire un graphique avec les point coloré selon cette partition (utiliser `plot_data()`).

Question 9: Impact de l'initialisation

Refaites un test avec une initialisation aléatoire (donc probablement mauvaise). La log vraisemblance va-t-elle augmenter à chaque itération dans ce cas de figure ? Qu'en est-il des paramètres estimés ?

```
partition_init = sample(1:K, size=10, replace=T)
plot_data(data$Var, partition_init)
```



Pour aller plus loin :

Relancer la procédure avec $K = 3$ en partant d'une partition initiale choisie au hasard ou celle de l'exercice au choix.