Algorithme EM

Colin Coërchon

18-12-2023

Contents

Introduction sur l'Algorithme EM

L'**algorithme EM** (Expectation-Maximization) est une méthode itérative pour trouver des estimations de maximum de vraisemblance de paramètres dans des modèles statistiques, où le modèle dépend de **variables latentes non observées**. Cet algorithme est particulièrement utile dans les situations où les données observées sont un mélange de plusieurs distributions sous-jacentes mais dont les paramètres sont inconnus.

Explication théorique de l'algorithme

Les données du problème

Dans le cadre de l'algorithme EM, les données du problème sont constituées de :

- X : Les données observées.
- Z : Les variables latentes, qui représentent les informations cachées ou non observables directement.
- Θ : Les paramètres du modèle que nous cherchons à estimer.

L'initialisation

Avant de commencer les itérations de l'algorithme EM, il est nécessaire d'initialiser les paramètres Θ. Cette initialisation peut se faire de manière aléatoire, ou basée sur une certaine heuristique choisie.

Le cœur de l'algorithme

L'algorithme EM alterne entre deux étapes principales :

- **Étape E (Expectation)** : Calcule l'espérance de la log-vraisemblance en tenant compte des variables latentes, en utilisant les valeurs actuelles des paramètres estimés. Cette étape génère une fonction, souvent appelée la fonction Q, qui est une moyenne pondérée de la log-vraisemblance des données complètes (observées et latentes).

$$Q(\Theta, \Theta^{(q)}) = \mathbb{E}_{Z \mid X, \Theta^{(q)}} \left[\log \mathbb{P}(X, Z; \Theta)\right]$$

- **Étape M (Maximization)** : Met à jour les paramètres en maximisant cette fonction Q. Cette mise à jour vise à améliorer les estimations des paramètres en maximisant la vraisemblance des données observées.

$$\Theta^{(q+1)} = \arg\max_{\Theta} Q(\Theta, \Theta^{(q)})$$

L'algorithme procède à ces deux étapes jusqu'à ce que **la convergence soit atteinte**, c'est-à-dire que les changements dans les paramètres entre les itérations successives deviennent négligeables.

Un mélange de Gaussiennes

Dans le contexte des mélanges de distributions, notamment des mélanges gaussiens, l'algorithme EM permet d'estimer les moyennes, les variances et les proportions de chaque distribution gaussienne dans le mélange. Les variables latentes, dans ce cas, représentent **l'appartenance de chaque observation à une distribution spécifique du mélange**.

En notant $t_{ik}^{(q)}$ la quantité donnée par $t_{ik}^{(q)} = \mathbb{E}\left(Z_{ik} = 1 \mid X_i = x_i, \Theta^{(q)}\right)$ à l'étape q, on peut séparer concrètement l'algorithme EM en deux étapes : E et M.

On considère que f correspond à la fonction densité d'une gaussienne, et qu'on cherche à estimer les paramètres donnés à l'étape q par $\Theta^{(q)} = \left\{\pi_k^{(q)}, \mu_k^{(q)}, \sigma_k^{2\,(q)} \middle| k \in [\![1, K]\!]\right\}$.

- **Étape E** : $\forall i \in [\![1, n]\!], \forall k \in [\![1, K]\!]$, calcul des t_{ik} par la règle d'inversion de Bayes :

$$\begin{aligned} t_{ik}^{(q)} &= \mathbb{P}\left(Z_{ik} = 1 \mid X_i = x_i, \Theta^{(q)}\right) \\ &= \frac{\mathbb{P}\left(Z_{ik} = 1, X_i = x_i, \Theta^{(q)}\right)}{\mathbb{P}\left(X_i = x_i, \Theta^{(q)}\right)} \\ &= \frac{\mathbb{P}(Z_{ik} = 1, \Theta^{(q)})\, \mathbb{P}(X_i = x_i \mid Z_{ik} = 1, \Theta^{(q)})}{\sum_{\ell=1}^{K} \mathbb{P}(Z_{i\ell} = 1, \Theta^{(q)})\, \mathbb{P}(X_i = x_i \mid Z_{i\ell} = 1, \Theta^{(q)})} \\ &= \frac{\pi_k^{(q)} f(x_i; \mu_k^{(q)}, \sigma_k^{2\,(q)})}{\sum_{\ell=1}^{K} \pi_\ell^{(q)} f(x_i; \mu_\ell^{(q)}, \sigma_\ell^{2\,(q)})}. \end{aligned}$$

- **Étape M** : détermination de $\Theta^{(q+1)}$ maximisant :

$$Q\left(\Theta, \Theta^{(q)}\right) = \sum_{i=1}^{n} \sum_{k=1}^{K} t_{ik}^{(q)} \log\left(\pi_k f(x_i; \mu_k, \sigma_k^2)\right).$$

  Les calculs ne sont ici pas détaillés, mais ils l'ont été dans le cours.

De plus, on arrive après calculs à exprimer les différents paramètres de $\Theta^{(q+1)}$ par rapport aux données :

$$\begin{aligned} \forall k \in [\![1, K]\!], \quad & \frac{\partial Q(\Theta, \Theta^{(q)})}{\partial \pi_k} = 0 \quad \Longrightarrow \pi_k = \frac{1}{n} \sum_{i=1}^{n} t_{ik}^{(q)} \\ \forall k \in [\![1, K]\!], \quad & \frac{\partial Q(\Theta, \Theta^{(q)})}{\partial \mu_k} = 0 \quad \Longrightarrow \mu_k = \frac{\sum_{i=1}^{n} t_{ik}^{(q)} x_i}{\sum_{i=1}^{n} t_{ik}^{(q)}} \\ \forall k \in [\![1, K]\!], \quad & \frac{\partial Q(\Theta, \Theta^{(q)})}{\partial \sigma_k^2} = 0 \quad \Longrightarrow \sigma_k^2 = \frac{\sum_{i=1}^{n} t_{ik}^{(q)} (x_i - \mu_k)^2}{\sum_{i=1}^{n} t_{ik}^{(q)}} \end{aligned}$$

On se concentrera ici au **mélange de 2 gaussiennes** seulement.

Implémentation de l'algorithme

Grâce aux formules données juste avant, appliquées au cas $K = 2$, on en déduit assez facilement l'implémentation de l'algorithme EM.

```r
# Fonction pour initialiser les paramètres pour un mélange de deux gaussiennes
initialisation_parametres <- function(X) {
  pi <- 0.5
  mu_1 <- sample(X, 1) # Choisir aléatoirement un point de données pour mu_1
  mu_2 <- sample(X, 1) # Choisir aléatoirement un autre point de données pour mu_2
  sigma2_1 <- 1
  sigma2_2 <- 1
  return(list(pi = pi, mu_1 = mu_1, mu_2 = mu_2, sigma2_1 = sigma2_1, sigma2_2 = sigma2_2))
}

# Étape E
etape_E <- function(X, pi, mu_1, mu_2, sigma2_1, sigma2_2) {
  t1 <- pi * dnorm(X, mean = mu_1, sd = sqrt(sigma2_1))
  t2 <- (1 - pi) * dnorm(X, mean = mu_2, sd = sqrt(sigma2_2))
  sum_t <- t1 + t2
  t1 <- t1 / sum_t
  t2 <- t2 / sum_t
  return(list(t1 = t1, t2 = t2))
}

# Étape M
etape_M <- function(X, t1, t2) {
  pi <- mean(t1)
  mu_1 <- sum(t1 * X) / sum(t1)
  mu_2 <- sum(t2 * X) / sum(t2)
  sigma2_1 <- sum(t1 * (X - mu_1)^2) / sum(t1)
  sigma2_2 <- sum(t2 * (X - mu_2)^2) / sum(t2)
  return(list(pi = pi, mu_1 = mu_1, mu_2 = mu_2, sigma2_1 = sigma2_1, sigma2_2 = sigma2_2))
}

# L'algorithme EM final, dans le cadre d'un mélange de deux gaussiennes
em_algorithme <- function(X, max_iterations = 1000, tolerance = 1e-10) {
  params <- initialisation_parametres(X)
  for (iteration in 1:max_iterations) {
    params_old <- params

    # Étape E
    t_values <- etape_E(X, params$pi, params$mu_1, params$mu_2, params$sigma2_1, params$sigma2_2)

    # Étape M
    params <- etape_M(X, t_values$t1, t_values$t2)

    # Vérifier la convergence
    if (abs(params$mu_1 - params_old$mu_1) < tolerance &&
        abs(params$mu_2 - params_old$mu_2) < tolerance &&
        abs(params$sigma2_1 - params_old$sigma2_1) < tolerance &&
        abs(params$sigma2_2 - params_old$sigma2_2) < tolerance &&
        abs(params$pi - params_old$pi) < tolerance) {
      break
    }
  }

  return(params)
```

```
}
```

Application dans un exemple

On reprend l'exemple vu dans le TD3.

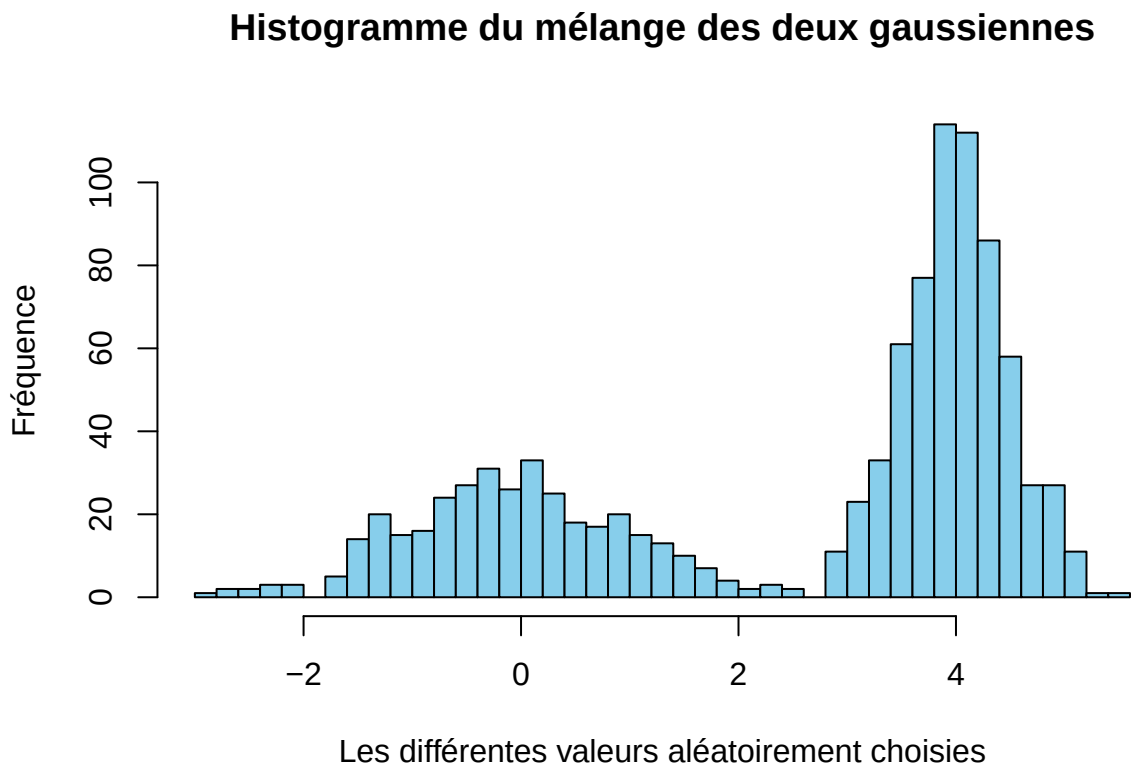
```
mu1 <- 0
mu2 <- 4
sigma1 <- 1
sigma2 <- 1/2
pi1 <- 1/3
pi2 <- 2/3 # 1 - pi1

n <- 1000

labels <- sample(c(1, 2), size = n, replace = TRUE, prob = c(pi1, pi2))

# Si label = 1, on simule un x suivant la première loi normale, sinon il suit la deuxième loi normale.
X <- ifelse(labels == 1,
                        rnorm(n, mean = mu1, sd = sigma1),
                        rnorm(n, mean = mu2, sd = sigma2))

hist(X, breaks = 50, col = 'skyblue', border = 'black',
     main = 'Histogramme du mélange des deux gaussiennes',
     xlab = 'Les différentes valeurs aléatoirement choisies', ylab = 'Fréquence')
```



Application de notre algorithme EM sur cet exemple :

```
params <- em_algorithme(X)
params
```

```
## $pi
## [1] 0.6422018
##
## $mu_1
## [1] 4.000359
##
## $mu_2
## [1] -0.05603529
##
## $sigma2_1
## [1] 0.2287053
##
## $sigma2_2
## [1] 1.004092
```

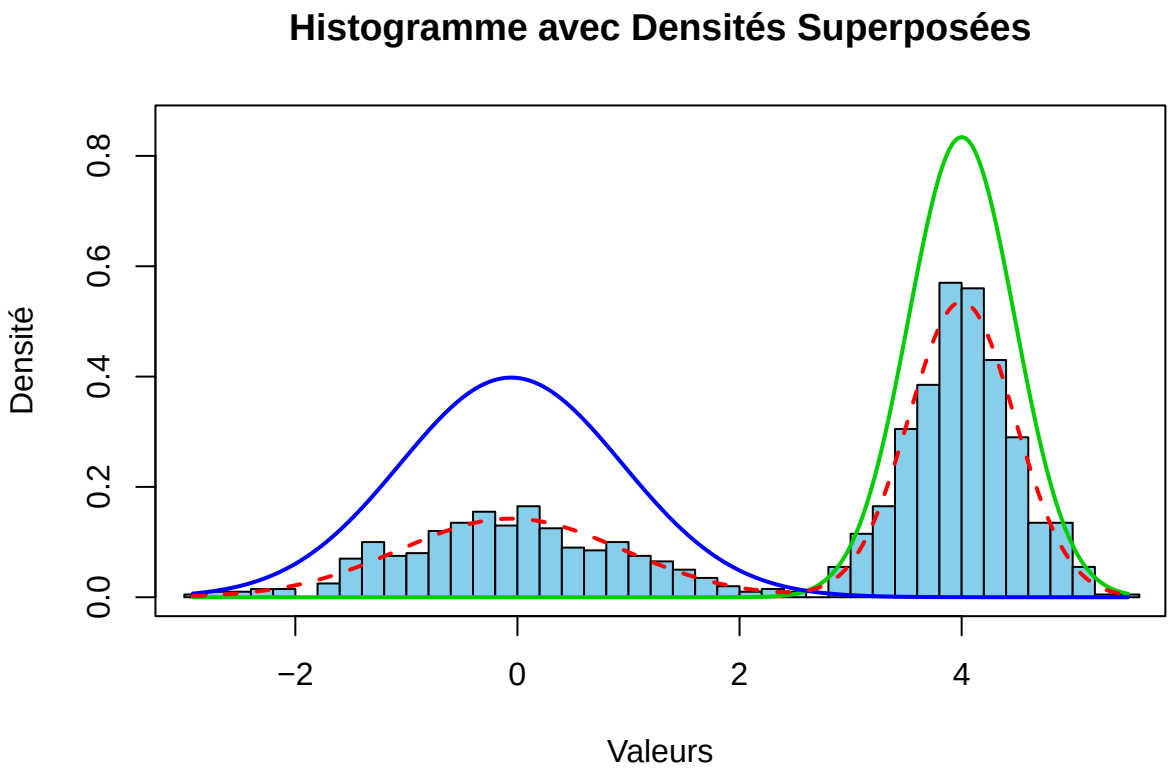
```
x_vals <- seq(min(X), max(X), length.out = 300)
dens1 <- dnorm(x_vals, mean = params$mu_1, sd = sqrt(params$sigma2_1))
dens2 <- dnorm(x_vals, mean = params$mu_2, sd = sqrt(params$sigma2_2))
dens_total <- params$pi * dens1 + (1 - params$pi) * dens2

# Déterminer la plage pour l'axe y
y_max <- max(dens_total) * 1.6  # Augmenter un peu pour avoir de l'espace au-dessus des courbes

# Créer un espace graphique vide avec une plage y étendue
plot(x_vals, type = "n", xlim = range(x_vals), ylim = c(0, y_max), xlab = "Valeurs", ylab = "Densité",
     main = "Histogramme avec Densités Superposées")

# Ajouter l'histogramme
hist(X, breaks = 50, col = 'skyblue', border = 'black', freq = FALSE, add = TRUE)

# Ajouter les courbes de densité
lines(x_vals, dens1, col = 'green3', lwd = 2)
lines(x_vals, dens2, col = 'blue', lwd = 2)
lines(x_vals, dens_total, col = 'red', lwd = 2, lty = 'dashed')
```



Dans ce graphique, les 2 courbes verte et bleue correspondent aux deux densités obtenues par notre algorithme EM. **La courbe en rouge** correspond alors à la densité totale du mélange, qui est la somme pondérée des deux densités gaussiennes.