

# TP R : modèle de mélange et algorithme EM

Fabrice Rossi

## 1 Rappels

### 1.1 Modèle de mélange

On se donne  $N$  observations dans  $\mathbb{R}^d$ , les  $(x_i)_{1 \leq i \leq N}$ , qu'on suppose des réalisations indépendantes d'une même loi inconnue  $P_X$ . On cherche à modéliser la densité de cette loi sous la forme d'un mélange. Pour cela, on se donne  $K$  densités paramétriques sur  $\mathbb{R}^d$ , les  $p_k(x|w_k)$ , chaque  $w_k$  désignant l'ensemble des paramètres de la densité  $p_k$ . Le modèle génératif retenu est donné par

$$p(x|\pi, w) = \sum_{k=1}^K \pi_k p_k(x|w_k), \quad (1)$$

où  $w$  regroupe tous les  $w_k$  et où  $\pi = (\pi_1, \dots, \pi_K)$  représente les proportions du mélange, avec  $\pi_k \in [0, 1]$  et  $\sum_{k=1}^K \pi_k = 1$ .

Pour ajuster le modèle, on maximise la log-vraisemblance par rapport aux paramètres, c'est-à-dire qu'on cherche à résoudre le problème d'optimisation sous contraintes suivant :

$$(\pi^*, w^*) = \arg \max_{w, \pi \in \mathcal{P}} \sum_{i=1}^N \log \left( \sum_{k=1}^K \pi_k p_k(x_i|w_k) \right), \quad (2)$$

où  $\mathcal{P}$  désigne l'ensemble des valeurs possibles pour  $\pi$  (en tenant compte des contraintes indiquées ci-dessus).

### 1.2 Algorithme EM

Pour résoudre le problème d'optimisation, on utilise l'algorithme EM qui procède de la façon suivante :

1. initialisation aléatoire de  $w^{(0)}$  et  $\pi^{(0)}$  ;
2. à l'itération  $t$  :
  - (a) calcul des responsabilités des composantes du mélange :

$$\gamma_{ik}^{(t)} = \frac{\pi_k^{(t-1)} p_k(x_i|w_k^{(t-1)})}{\sum_{j=1}^K \pi_j^{(t-1)} p_j(x_i|w_j^{(t-1)})} \quad (3)$$

- (b) mise à jour des proportions du mélange par :

$$\pi_k^{(t)} = \frac{\sum_{i=1}^N \gamma_{ik}^{(t)}}{N} \quad (4)$$

- (c) mise à jour des autres paramètres en résolvant les problèmes d'optimisation suivants :

$$w_k^{(t)} = \arg \max_{w_k} \sum_{i=1}^N \gamma_{ik}^{(t)} \log p_k(x_i|w_k) \quad (5)$$

3. retour en 2 jusqu'à convergence (c'est-à-dire stabilisation des paramètres et/ou de la vraisemblance).

## 2 Modèle *Zero inflated Poisson*

On étudie dans un premier temps le modèle simple de la loi *Zero inflated Poisson* qui est un mélange entre une loi discrète telle que  $\mathbb{P}(X = 0) = \pi$  (avec une proportion  $\pi$ ) et une loi de Poisson classique de paramètre  $\lambda$  (avec une proportion  $1 - \pi$ ).

### 2.1 Spécialisation de l'algorithme

- Écrire la vraisemblance des paramètres d'une loi ZIP pour les observations  $(x_i)_{1 \leq i \leq N}$  supposées issues d'une telle loi.
- Réécrire l'algorithme de la section 1.2 dans le cas spécifique de la loi ZIP.

### 2.2 Mise en œuvre en R

1. Écrire une fonction R `rzip` qui simule des réalisations indépendantes d'une loi ZIP de paramètres `pi` et `lambda` (désignant, respectivement, la proportion  $\pi$  de la loi chargeant uniquement 0 et,  $\lambda$  le paramètre de la partie Poisson). On s'appuiera sur la fonction `rpois` qui simule des réalisations d'une loi Poisson.
2. Écrire une fonction R `emzip` qui estime les paramètres  $\pi$  et  $\lambda$  d'une loi ZIP à partir d'un échantillon en maximisant la vraisemblance par l'algorithme EM.
3. En utilisant la fonction `optim` de R, construire une autre fonction `directzip` qui estime les paramètres  $\pi$  et  $\lambda$  d'une loi ZIP à partir d'un échantillon en maximisant la vraisemblance de façon directe (sans passer par l'algorithme EM).

## 3 Estimation de densité dans $\mathbb{R}$

On étudie maintenant un modèle plus générique dans  $\mathbb{R}$ , un mélange de Gaussiennes univariées, c'est-à-dire des composantes de la forme

$$p_k(x|\mu_k, \sigma_k) = \frac{1}{\sigma_k \sqrt{2\pi}} e^{-\frac{(x-\mu_k)^2}{2\sigma_k^2}}. \quad (6)$$

### 3.1 Spécialisation de l'algorithme

Montrer que l'étape (c) de l'algorithme EM devient dans ce cas :

$$\mu_k^{(t)} = \frac{\sum_{i=1}^N \gamma_{ik}^{(t)} x_i}{\sum_{i=1}^N \gamma_{ik}^{(t)}} \quad (7)$$

$$\sigma_k^{(t)} = \sqrt{\frac{\sum_{i=1}^N \gamma_{ik}^{(t)} (x_i - \mu_k^{(t)})^2}{\sum_{i=1}^N \gamma_{ik}^{(t)}}} \quad (8)$$

### 3.2 Mise en œuvre en R

1. Écrire une fonction R `densiteMélange` qui prend pour paramètres les vecteurs `x` (les données), `mu`, `sigma` et `prop` (les paramètres, `prop` désignant le vecteur  $\pi$ ), et qui renvoie la densité du modèle de mélange en chaque observation (sous forme d'un vecteur).
2. Écrire une fonction R `logVR` qui prend pour paramètres les vecteurs `x`, `mu`, `sigma` et `prop`, et qui renvoie la log vraisemblance des données pour le modèle correspondant aux paramètres.
3. Écrire une fonction R `responsabilites` qui prend pour paramètres des vecteurs `x`, `mu`, `sigma` et `prop`, et qui renvoie une matrice contenant les  $\gamma_{ik}$ .

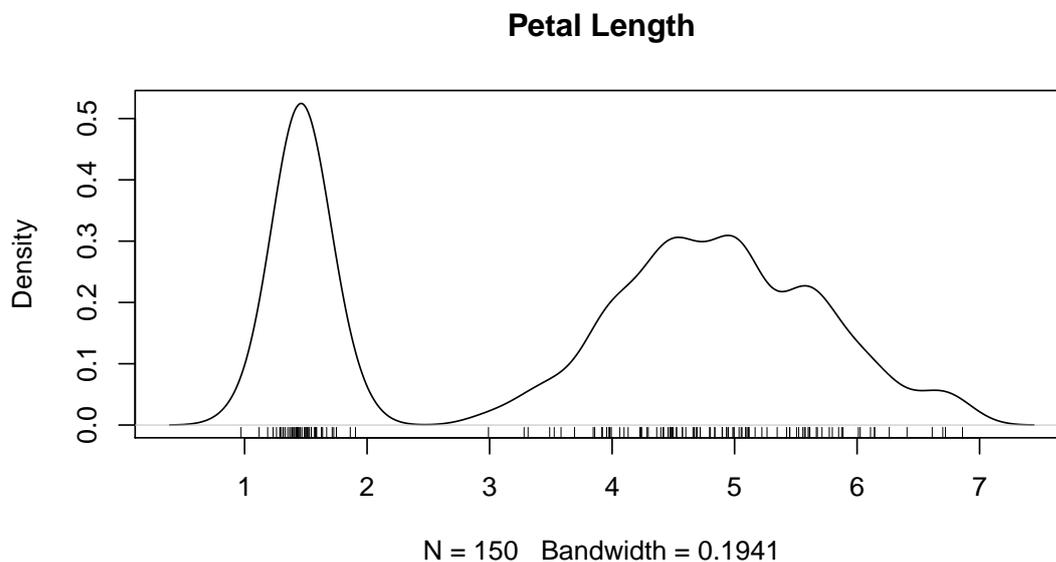


FIGURE 1 – Estimation de densité par méthode à noyaux

4. Écrire une fonction R `params` qui prend pour paramètres un vecteur de données `x` et une matrice de responsabilités `gamma`, et qui renvoie une liste contenant les nouvelles valeurs des paramètres du mélange.
5. Écrire fonction R `unidimEM` qui prend pour paramètres un vecteur de données `x` et un entier `K` et qui met en œuvre l'algorithme EM sur les données avec un mélange de `K` Gaussiennes. On pourra dans un premier temps réaliser un nombre fixe d'itérations et afficher l'évolution de la vraisemblance pour contrôler le bon comportement de l'implémentation.

### 3.3 Test de l'implémentation

Pour tester l'implémentation, on peut étudier les données iris, chargées par

```
data(iris)
```

Attention, ces données comportent des valeurs en double, ce qui peut parfois poser des problèmes numériques. Il est donc utile d'ajouter un peu de bruit, par exemple en faisant

```
iris$Petal.Length <- iris$Petal.Length+runif(-0.05,0.05,n=nrow(iris))
```

La variable `Petal.Length` est multimodale, comme le montre l'estimation de sa densité par un modèle à noyaux (cf la figure 1), obtenue par le code suivant :

```
plot(density(iris$Petal.Length,bw="SJ"),main="Petal Length")
rug(iris$Petal.Length)
```

En appliquant le code proposé au dessus avec  $K = 2$ , on doit obtenir une estimation de densité de la forme de celle de la figure 2.

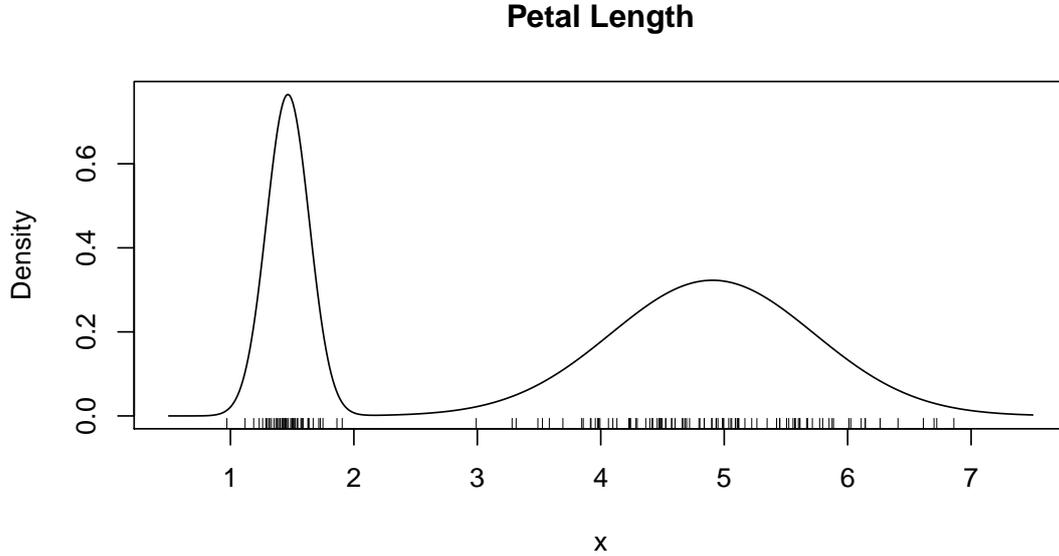


FIGURE 2 – Estimation de densité par un mélange de deux gaussiennes

### 3.4 Choix de modèle

Dans le contexte des modèles de mélange, on utilise en général le critère BIC pour comparer deux modèles. Pour un modèle  $M$ , le critère est donné par

$$BIC(M) = -2 \log p(D|M) + |M| \log |D|, \quad (9)$$

où  $p(D|M)$  désigne la vraisemblance des données  $D$  pour ce modèle,  $|M|$  le nombre de paramètres indépendants du modèle et  $|D|$  le nombre d'observations (ici  $N$ ). Le meilleur modèle est celui dont le  $BIC$  est le plus faible. Dans notre contexte de mélange de gaussiennes,  $|M| = K - 1 + 2K = 3K - 1$ . En effet, on a 2 paramètres par gaussiennes (la moyenne et la variance), et  $K$  paramètres pour décrire les proportions. Cependant, seulement  $K - 1$  paramètres sont indépendants  $\pi$  en raison de la contrainte de somme à 1.

On complétera l'implémentation en proposant une fonction R `unidimEMAuto` prenant comme paramètres le vecteur  $\mathbf{x}$  et un intervalle de recherche pour  $K$ , dont l'objectif sera d'estimer le meilleur modèle sur un jeu d'observations en utilisant le critère BIC. La figure 3 représente le BIC en fonction du nombre de composantes dans l'exemple choisi. On constate que le choix de deux composantes est le plus adapté ici.

## 4 Extension à $\mathbb{R}^d$ pour $d > 1$

Pour estimer une densité dans  $\mathbb{R}^d$ , et donc faire (indirectement) de la classification, on utilisera un mélange de gaussiennes multivariées, donc des composantes de la forme

$$p_k(x|\mu_k, \Sigma_k) = \frac{1}{(2\pi)^{d/2} |\Sigma_k|^{1/2}} e^{-\frac{1}{2}(x-\mu_k)^T \Sigma_k^{-1} (x-\mu_k)}, \quad (10)$$

où  $|\Sigma_k|$  désigne le déterminant de la matrice de covariance, et  $^T$  l'opération de transposition.

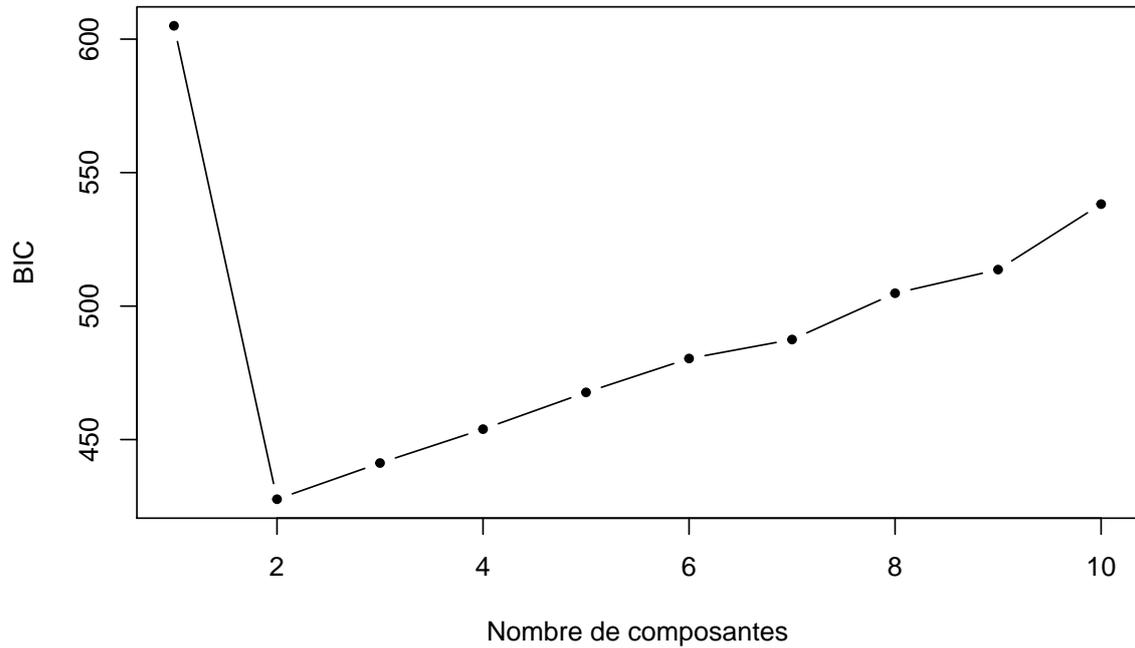


FIGURE 3 – Valeurs du critère BIC pour un mélange de 1 à 10 gaussiennes utilisé pour estimer la densité de la variable `Petal.Length`

#### 4.1 Spécialisation de l’algorithme

On montre que l’étape (c) de l’algorithme EM devient ici :

$$\mu_k^{(t)} = \frac{\sum_{i=1}^N \gamma_{ik}^{(t)} x_i}{\sum_{i=1}^N \gamma_{ik}^{(t)}} \quad (11)$$

$$\Sigma_k^{(t)} = \frac{\sum_{i=1}^N \gamma_{ik}^{(t)} (x_i - \mu_k^{(t)})(x_i - \mu_k^{(t)})^T}{\sum_{i=1}^N \gamma_{ik}^{(t)}} \quad (12)$$

#### 4.2 Mise en œuvre en R

On suivra la stratégie d’implémentation proposée au dessus, en utilisant les éléments suivants :

- R propose une fonction `det` pour calculer le déterminant d’une matrice ;
- l’inverse d’une matrice est obtenue grâce à la fonction `solve` ;
- la fonction `cov.wt` peut être utilisée pour calculer une matrice de covariance pour des données pondérées.