Rapport TER

Aymeric Van Mael - Julien Gheysens

Encadré par Adrien Hardy

Vendredi 26 mai 2017

Table des matières

1 Introduction

La classification en analyse de données *exploratoire* et *décisionelle* est un grand challenge de nos jours. On cherche en effet de plus en plus à regrouper des personnes/objets selon certaines caractéristiques que ce soit dans le domaine médical, afin de definir une thérapeutie adaptée à un type particulier de malades, ou en astronomie pour catégoriser les étoiles selon des critères de température ou de luminosité.

L'objectif de l'analyse *exploratoire* est de créer une partition en regroupant les données en fonction de leurs similarités et de leurs dissimilarités. Cette partie de l'analyse de données est la classication dite *non supervisée*. Concernant la partie décisionelle, on s'intéresse plutôt à affecter à une des classes préalablement définie, toute nouvelle observation. Cette partie est representée par la classification *supervisée*.

Pour cela, considérer que les données sont issues d'une loi de mélanges finis de distributions est devenue très populaire ces dernières années. En effet les modèles de mélanges, notamment gaussiens, bénéficient de proprietés avantageuses permettant de traiter les données efficacement : critères de choix de modèles permettant de définir le nombre de lois du mélange, estimations paramétriques multivariées grâce à des algorithmes de recherche de paramètres des lois du mélange.

Nous ferons tout d'abord des rappels concernant les mélanges gaussiens et nous expliquerons comment simuler algorithmiquement ces mélanges. La section suivante abordera la classification non supervisée. Nous supposerons dans un premier temps connu le nombre g de classes composant le mélange et nous expliquerons le fonctionnement de deux algorithmes d'estimation des paramètres de la loi. Cependant dans la pratique le nombre de groupes g n'est pas toujours connu. C'est dans cet objectif que nous présenterons deux points de vue, l'un statisticien et l'autre bayésien, qui permettent d'estimer g à travers des critères comme AIC,BIC ou encore ICL. La section suivante abordera la classification supervisée où nous détaillerons les algorithmes mis en place afin de classer une nouvelle observation, et nous parlerons de la géométrie des groupes, plus précisement de leurs frontières, dans une troisième partie. Nous terminerons par une illustration de nos algorithmes sur un jeu de données *Iris* de RStudio, les observations étant décrites par deux variables : la durée de l'éruption et le temps d'attente entre deux éruptions.

La réalisation de ce rapport a été basé sur l'article de réference [2].

2 Mélange Gaussien

2.1 Rappels

Tout d'abord, nous allons faire quelques rappels pour bien définir ce qu'est un vecteur gaussien. [6] [7]

Définition 1. *Soient $m \in \mathbb{R}$ et $\sigma \leq 0$. Z une variable aléatoire suivant la loi N(0,1). On appelle loi gaussienne de paramètre m et σ^2 la loi de la variable aléatoire $\sigma Z + \mu$. La densité est alors :*

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}$$

On note $N(\mu, \sigma)$ cette loi.

Définition 2. *Un vecteur aléatoire $\boldsymbol{X}$ est gaussien si toute combinaison linéaire de ses composantes est une variable aléatoire gaussienne.*

Les quelques proprietés suivantes seront notamment utiles pour les algorithmes de génération de vecteurs gaussiens.

Théorème 1. *Soit $X_1, ..., X_d$ des variables réelles gaussiennes. Si elles sont mutuellement indépendantes alors le vecteur $\boldsymbol{X} = (X_1, ..., X_d)$ est gaussien*

Démonstration. Soient $X_1, X_2, ..., X_n$ n variables aléatoires mutuellement indépendantes de loi $X_i \sim \mathcal{N}(a_i.\mu_i, a_i^2.\sigma_i^2)$, $i \in \{1, ..., n\}$. Montrons que $\mathbf{X} = (X_1, ..., X_d)$ est un vecteur gaussien, c'est à dire que $a_1.X_1 + a_2.X_2 + ... + a_n.X_n$ suit une loi normale.

Rappel :

Soit X_i suit une loi normale de paramètres μ_i et Σ_i.

Sa fonction caractéristique est donnée par

$$\phi_x(t) = exp(it\mu_i - \frac{\sigma^2 t^2}{2})$$

On sait que $a_i X_i \sim \mathcal{N}(a_i m_i, a_i^2 \sigma_i^2)$ et que sa fonction caractéristique est :

$$\phi_{a_i.X_i}(t) = exp(ia_i\mu_i t - \frac{a_1^2\sigma_i^2 t^2}{2})$$

$$\begin{aligned}
\text{Donc : } \phi_{\sum_{i=1}^n a_i X_i}(t) &= \mathbb{E}[e^{it\sum_{i=1}^n a_i X_i}] \\
&= \mathbb{E}[\prod_{i=1}^n e^{ita_i X_i}] \\
&= \prod_{i=1}^n \phi_{a_i X_i}(t) \\
&= \prod_{i=1}^n exp(ia_i\mu_i t - \frac{a_i^2\sigma_i^2 t^2}{2}) \\
&= exp(it\sum_{i=1}^n a_i\mu_i - \frac{t^2}{2}\sum_{i=1}^n a_i^2\sigma_i^2) \\
&= \phi_{\mathcal{N}(\sum_{i=1}^n a_i\mu_i, \sum_{i=1}^n a_i^2\sigma_i^2)}
\end{aligned}$$

□

Corollaire 1 (Propriété pour l'indépendance)**.** *Soit* $\boldsymbol{X} = (X_1, ..., X_d)$ *un vecteur gaussien. On note* $m = \mathbb{E}(\boldsymbol{X})$ *et* $\Sigma = Var(\boldsymbol{X})$*. On a pour toute matrice* A *possédant* d *colonnes et pour tout vecteur* $b \in \boldsymbol{R}^d$*,*

$$A\boldsymbol{X} + b \sim \mathcal{N}(A\mu + b, A\Sigma^t A)$$

Corollaire 2 (Propriété pour l'indépendance)**.** *Soit* $\boldsymbol{X} = (X_1, ..., X_d)$ *un vecteur gaussien. Pour tout* $i, j \in \{1, ..., d\}^2$ *tel que* $i \neq j$*,* X_i *et* X_j *sont indépendantes si et seulement si* $cov(X_i, X_j) = 0$

Théorème 2. *Soit* $\boldsymbol{X} = (X_1, ..., X_d)$ *un vecteur gaussien. On note* $\mu = \mathbb{E}(\boldsymbol{X})$ *et* $\Sigma = Var(\boldsymbol{X})$*.* $\boldsymbol{X}$ *admet une densité* f *par rapport à la mesure de Lebesgue sur* $\mathbb{R}^d$ *si et seulement si* $det(\Sigma) \neq 0$*.*

- *Si* $det(\Sigma) = 0$ *, la loi de* $\boldsymbol{X} - \mu$ *est presque sûrement portée par un espace vectoriel engendré par les vecteurs propres associés aux valeurs propres non nulles de* Σ*.*
- *Si* $det(\Sigma) \neq 0$*,*

$$\forall x \in \mathbb{R}^d, f(x) = \left(\frac{1}{\sqrt{2\pi}}\right)^d \frac{1}{\sqrt{det(\Sigma)}} e^{-\frac{(x-\mu)^t \Sigma^{-1}(x-\mu)}{2}}$$

Démonstration. La matrice Σ étant symétrique, il existe U, matrice orthogonale et D diagonale telles que :

$\Sigma = U^t.D.U$ où $D = \begin{pmatrix} \lambda_1 & & & & & & \\ & \lambda_2 & & & & & \\ & & \ddots & & & & \\ & & & \lambda_r & & & \\ & & & & 0 & & \\ & & & & & \ddots & \\ & & & & & & 0 \end{pmatrix}$, U matrice orthogonale et r rang de Σ.

Si $det(\Sigma) \neq 0$, on a que $U\sqrt{D}Y + \mu \sim \mathcal{N}(\mu, \Sigma)$ avec $Y \sim \mathcal{N}(0, Id)$.
Donc,

$$\begin{aligned} \mathbb{E}[g(X)] &= \mathbb{E}g(U\sqrt{D}Y + \mu) \\ &= \int_{\mathbb{R}^d} g(U\sqrt{D}Y + \mu).\frac{1}{\sqrt{2.\pi}}^d.exp(-\frac{||y||^2}{2})\mathrm{d}y \\ &= \int_{\mathbb{R}^d} g(x)(\frac{1}{\sqrt{2\pi}})^d.\frac{1}{\sqrt{det(\Sigma)}}.exp(-\frac{(x-\mu)^t\Sigma^{-1}(x-\mu)}{2})\mathrm{d}x \end{aligned}$$

□

Définition 3. *Une loi de mélange fini* p *sur un espace* X *est une loi de probabilité s'exprimant comme une combinaison linéaire de plusieurs lois de probabilité* $p_1, ..., p_g$ *sur* X*. Autrement dit, il existe* g *coefficients* $\pi_1, ..., \pi_g$ *(*$\pi > 0$ *et* $\sum_{k=1}^g \pi_k = 1$*) tels que, pour tout* $x_1 \in X$*,*

$$p(x_1) = \sum_{k=1}^{g} \pi_k p_k(x_1)$$

. Les π_k *et* p_k *sont respectivement appelées proportions et composantes du mélange.*

En l'occurence ici, pour la loi mélange p peut s'interpréter comme la loi marginale de la variable aléatoire X_1 obtenue à partir de la loi du couple de variables aléatoires (X_1, Z_1) où

$$Z_1 \sim Multinomiale(1, \pi_1, ..., \pi_g) \text{ et } X_1|Z_1 = z_1 \sim \mathcal{N}(\mu_1, \Sigma_1)$$

2.2 Simulation/Implémentation

Maintenant que nous avons bien défini les mélanges gaussiens, nous avons cherché à implémenter sur Rstudio des fonctions permettant de simuler des mélanges de vecteurs gaussiens de $\mathbb{R}^d$.

On souhaite simuler un mélange gaussien de g lois, chacune de moyenne $m_j \in \mathbb{R}^d$ et de matrice de covariance Σ_j, $j \in \{1, ..., g\}$.

Pour cela nous allons d'abord créer une fonction r_VG afin de simuler un vecteur gaussien de loi $\mathcal{N}(\mu, \Sigma)$.

Comme Σ_j est symétrique définie positive, on peut la diagonaliser Σ pour obtenir la décomposition $\Sigma = P.D.P^t$(on pourrait également utiliser la décomposition de Cholesky). Avec $D = \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_d \end{pmatrix}$ et λ_k valeurs propres de Σ_j, $k \in \{1, ..., d\}$.

Nous utilisons alors la fonction "rnorm" de RStudio pour simuler d réalisations de d variables gaussiennes Y centrées et de variance λ_k, $k \in \{1, ..., d\}$.

En effet D étant diagonale, pour simuler un vecteur gaussien il suffit de simuler d variables gaussiennes indépdantes chacune de variance λ_k, $k \in \{1, ..., d\}$. Le vecteur $X = (X_1, ..., X_d)$ suivra alors une loi normale de variance D.

Il suffit ensuite d'appliquer la transformation $P.Y + \mu$. D'après le corollaire 1 on aura bien $P.Y + \mu \sim \mathcal{N}(\mu, P.D.P^t)$.

Il est maintenant facile de créer une fonction servant à simuler un échantillon de taille n de loi de mélange gaussien p. Pour cela, on suppose connu les poids π_k, les moyennes μ_k et les variances Σ_k. On "tire" alors au hasard une loi en utilisant *rmultinom* puis on stocke dans une matrice $x \in \mathcal{M}_{n,d}$ la réalisation correspondant la k-ième loi gaussienne p_k.

Et enfin on souhaite avoir une fonction qui nous donne la densité d'un vecteur gaussien. Elle nous sera utile dans la suivante.Pour cela il nous faut effectuer le raisonnement inverse par rapport à r_VG. Ici notre fonction d_VG part d'une réalisation de X de moyenne μ et de variance Σ pour introduire une nouvelle variable Y centrée de variance D grâce à la transformation : $Y = P^t.(X - \mu)$. D étant diagonale, la valeur de la densité de Y sera alors le produit des densités de ces marginales, que l'on calcule avec la fonction "dnorm".

Pour illustrer nos algorithmes, voici quelques exemples d'échantillons avec des mélanges à deux et trois lois gaussiennes. La figure 1 représente un mélange de moyennes $\mu_1 = (-5, 0)$ et $\mu_2 = (1, 1)$, et de variance $\Sigma_1 = \Sigma_2 = Id$. Et la figure 2 montre un échantillon ayant comme paramètres $\mu_1 = (-5, 5)$, $\mu_2 = (4, 4)$ et $\mu_3 = (3, -3)$, et de variance $\Sigma_1 = \begin{pmatrix} 2 & 1.5 \\ 1.5 & 2 \end{pmatrix}$, $\Sigma_2 = Id$ et $\Sigma_3 = \begin{pmatrix} 4 & -0.5 \\ -0.5 & 1 \end{pmatrix}$.

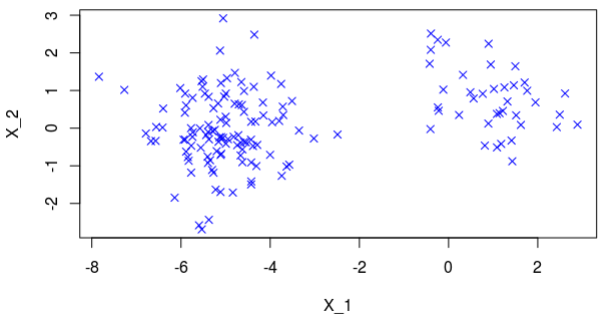


FIGURE 1 – Exemple d'un mélange avec deux lois gaussiennes

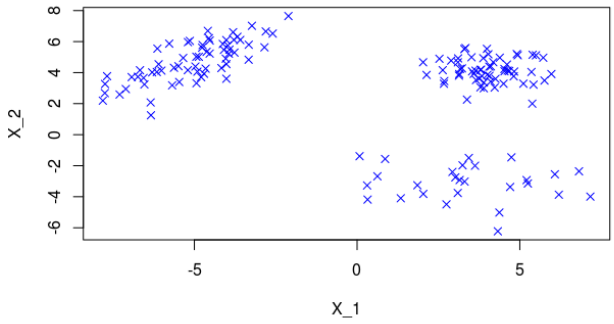


FIGURE 2 – Exemple d'un mélange avec trois lois gaussiennes

3 Classification non supervisée

Appelée aussi classification automatique ou regroupement, le but de la classification non supervisée est de partitionner l'ensemble de nos individus en g groupes $G_1, ..., G_g$. On représente cette partition par un tableau binaire $\mathbf{z}$ comportant n lignes et g colonnes, où n est le nombre d'individu de notre ensemble. Ce tableau est rempli selon la formule suivante :

$$\mathbf{z}_{ik} = \begin{cases} 1 & si \quad x_i \in G_k \\ 0 & sinon \end{cases}$$

En outre, la classification non supervisée nous permet de mieux comprendre notre échantillon en regroupant des comportements similaires de différents individus. Pour certains échantillons, il peut être assez facile de trouver les différents groupes à l'oeil nu s'ils sont bien "séparés" (par rapport à une certaine norme) et si les individus sont définis avec deux variables. Mais si les groupes sont "rapprochés" ou que le nombre de variable est plus important, l'analyse peut s'avérer plus difficile.

3.1 L'algorithme EM

Dans cette section, nous allons développer le fonctionnement de l'algorithme *EM* [1]. Notre but est toujours d'estimer les paramètres des différentes lois de notre mélange gaussien, notons cet ensemble $\theta = \{\pi_1, ..., \pi_g, \mu_1, ..., \mu_g, \Sigma_1, ..., \Sigma_g\}$. Une représentation graphique sera possible lorsque nous disposons d'un jeu de données caractérisé par deux variables avec la fonction *graph_EM* en traçant des régions de confiance autour de chaque groupe.

3.1.1 Définition

L'algorithme EM(espérance-maximisation) est un algorithme itératif permettant de trouver les paramètres du maximum de vraisemblance d'un modèle probabiliste. Cette algorithme repose sur deux grandes étapes que l'on réitère un certain nombre de fois.

- Etape (E) : une estimation de l'espérance, où l'on calcule la probabilité conditionelle qu'un individu x_i soit issu du groupe G_k.
- Etape (M) : une estimation du maximum, où l'on maximise l'espérance de la log vraisemblance observée des paramètres en remplaçant la partition z par l'estimation de l'espérance calculée à l'étape précedente.

3.1.2 Calcul de l'estimation des paramètres

On cherche maintenant à estimer notre paramètre $\theta = \{\pi_1, ..., \pi_g, \mu_1, ..., \mu_g, \Sigma_1, ..., \Sigma_g\}$ avec l'algorithme EM. On note $\hat{\theta_k}$ estimation de θ à l'étape k par cet algorithme.

Proposition 1. *L'étape (M) de l'algorithme EM consiste à choisir θ_{k+1} de la façon suivante :*

$$\mu_j^{(k+1)} = \frac{\sum_{i=1}^n x_i.H_{i,j}(\theta_k)}{\sum_{i=1}^n H_{i,j}(\theta_k)} \qquad 1 \leq j \leq g$$

$$\Sigma_j^{(k+1)} = \frac{\sum_{i=1}^n (x_i - \mu_j).(x_i - \mu_j)^T.H_{i,j}(\theta_k)}{\sum_{i=1}^n H_{i,j}(\theta_k)} \qquad 1 \leq j \leq g$$

$$\pi_j^{(k+1)} = \frac{1}{n}\sum_{i=1}^n H_{i,j}(\theta_k) \qquad 1 \leq j \leq g$$

avec $H_{i,j}$ est la probabilité conditionnelle que l'individu i soit issu du groupe j

Démonstration. [3] La vraisemblance pour une mélange gaussien s'écrit :

$$V(\theta, X) = \prod_{i=1}^n \left(\sum_{j=1}^g \pi_j f_{\mathcal{N}(\mu_j,\Sigma_j)}(x_i) \right)$$

Et donc sa log vraisemblance vaut :

$$L(\theta, X) = \sum_{i=1}^n log \left(\sum_{j=1}^g \pi_j f_{\mathcal{N}(\mu_j,\Sigma_j)}(x_i) \right)$$

Nous considèrons maintenant **z** défini comme dans le début de cette partie. Grâce à cet élément, nous pouvons définir la log vraisemblance complétée de la façon suivante :

$$\begin{aligned} L(\theta, X, Z) &= \sum_{i=1}^n \sum_{j=1}^g z_{ij}\ log\left(\pi_j f_{\mathcal{N}(\mu_j,\Sigma_j)}(x_i)\right) \\ &= \sum_{i=1}^n \sum_{j=1}^g z_{ij}\ log\left(\pi_j \frac{1}{(2\pi)^{\frac{d}{2}}} |\Sigma_j|^{-\frac{1}{2}} e^{-\frac{(x_i-\mu_j)^t \Sigma_j^{-1}(x_i-\mu_j)}{2}}\right) \\ &= \sum_{i=1}^n \sum_{j=1}^g z_{ij}\left(log(\pi_j) - log((2\pi)^{\frac{d}{2}} + log(|\Sigma_j|^{-\frac{1}{2}}) - \frac{(x_i-\mu_j)^t \Sigma_j^{-1}(x_i-\mu_j)}{2}\right) \end{aligned}$$

Nous introduisons $Q(\theta, \theta_k)$ l'espérance conditionnelle de la log vraisemblance complétée sachant X sous θ_k.

$$\begin{aligned} Q(\theta, \theta_k) &= \mathbb{E}_{\theta_k}[L(\theta, X, Z)|X] \\ &= \mathbb{E}_{\theta_k}\left[\sum_{i=1}^n \sum_{j=1}^g z_{ij}\ log\left(\pi_j f_{\mathcal{N}(\mu_j,\Sigma_j)}(x_i)\right)|X\right] \\ &= \sum_{i=1}^n \sum_{j=1}^g \mathbb{E}_{\theta_k}[z_{ij}|X]\ log\left(\pi_j f_{\mathcal{N}(\mu_j,\Sigma_j)}(x_i)\right) \end{aligned}$$

Nous cherchons à maximiser $Q(\theta, \theta_k)$ c'est-à-dire on cherche :

— $\mu_j^{(k+1)}$ tel que $\frac{\partial Q(\theta,\theta_k)}{\partial \mu_j} = 0$

— $\Sigma_j^{(k+1)}$ tel que $\frac{\partial Q(\theta,\theta_k)}{\partial \Sigma_j} = 0$

— $\pi_j^{(k+1)}$ tel que $\frac{\partial Q(\theta,\theta_k)}{\partial \pi_j} = 0$

Nous faisons la notation suivante : $H_{i,j}(\theta_k) = \mathbb{E}_{\theta_k}[z_{ij}|X]$

Calcul de $\mu_j^{(k+1)}$:

$$\begin{aligned}
\frac{\partial Q(\theta,\theta_k)}{\partial \mu_j} = 0 &\Leftrightarrow \sum_{i=1}^{n} \frac{\partial}{\partial \mu_j}(x_i - \mu_j)^T.\Sigma_j^{-1}(x_i - \mu_j).H_{i,j}(\theta_k) = 0 \\
&\Leftrightarrow \sum_{i=1}^{n}(x_i - \mu_j)^T.\Sigma_j^{-1}.H_{i,j}(\theta_k) = 0 \\
&\Leftrightarrow \sum_{i=1}^{n}\mu_j^T.\Sigma_j^{-1}.H_{i,j}(\theta_k) = \sum_{i=1}^{n} x_i^T.\Sigma_j^{-1}.H_{i,j}(\theta_k) \\
&\Leftrightarrow \sum_{i=1}^{n}\mu_j.H_{i,j}(\theta_k) = \sum_{i=1}^{n} x_i.H_{i,j}(\theta_k) \\
&\text{donc } \boxed{\mu_j^{(k+1)} = \frac{\sum_{i=1}^{n} x_i.H_{i,j}(\theta_k)}{\sum_{i=1}^{n} H_{i,j}(\theta_k)}}
\end{aligned}$$

Calcul de $\Sigma_j^{(k+1)}$ (grâce à [8]) :

$$\begin{aligned}
\frac{\partial Q(\theta,\theta_k)}{\partial \Sigma_j} = 0 &\Leftrightarrow -\frac{1}{2}\sum_{i=1}^{n}\frac{\partial}{\partial \Sigma_j} ln|\Sigma_j|.H_{i,j}(\theta_k) - \frac{1}{2}\sum_{i=1}^{n}\frac{\partial}{\partial \Sigma_j}(x_i - \mu_j)^T\Sigma_j^{-1}(x_i - \mu_j).H_{i,j}(\theta_k) = 0 \\
&\Leftrightarrow \sum_{i=1}^{n}\Sigma_j^{-1}.H_{i,j}(\theta_k) - \sum_{i=1}^{n}(\Sigma_j^{-1})^T.(x_i - \mu_j).(x_i - \mu_j)^T(\Sigma_j^{-1})^T.H_{i,j}(\theta_k) = 0 \\
&\Leftrightarrow \sum_{i=1}^{n} H_{i,j}(\theta_k) - \sum_{i=1}^{n}\Sigma_j^{-1}.(x_i - \mu_j).(x_i - \mu_j)^T.H_{i,j}(\theta_k) = 0 \\
&\Leftrightarrow \Sigma_j.\sum_{i=1}^{n} H_{i,j}(\theta_k) = \sum_{i=1}^{n}(x_i - \mu_j).(x_i - \mu_j)^T.H_{i,j}(\theta_k) \\
&\text{donc } \boxed{\Sigma_j^{(k+1)} = \frac{\sum_{i=1}^{n}(x_i - \mu_j).(x_i - \mu_j)^T.H_{i,j}(\theta_k)}{\sum_{i=1}^{n} H_{i,j}(\theta_k)}}
\end{aligned}$$

Calcul de π_j^{k+1} :
On a comme contrainte d'égalité : $\sum_{j=1}^{g} \pi_j = 1$
On pose $u(\pi_1, ..., \pi_g) = \sum_{j=1}^{g} -1$
Le Lagrangien devient alors : $L(\theta, \theta_k) = Q(\theta, \theta_k) - \lambda.u(\pi_1, ..., \pi_g)$ avec $\lambda \in \mathbb{R}$

$$\frac{\partial Q(\theta, \theta_k)}{\partial \pi_j} - \frac{\partial u(\theta, \theta_k)}{\partial \pi_j} = 0 \Leftrightarrow \frac{\sum_{i=1}^{n} H_{i,j}(\theta_k)}{\pi_j} - \lambda = 0$$
$$\Leftrightarrow \pi_j^{(k+1)} = \frac{\sum_{i=1}^{n} H_{i,j}(\theta_k)}{\lambda}$$

or les $\pi_j^{(k+1)}$ doivent satisfaire $\sum_{j=1}^{g} \pi_j = 1$

$$\sum_{j=1}^{g} \pi_j = 1 \Leftrightarrow \frac{1}{\lambda}. \sum_{i=1}^{n} \sum_{j=1}^{g} H_{i,j}(\theta_k) = 1$$
$$\Leftrightarrow \frac{1}{\lambda}. \sum_{i=1}^{n} 1 = 1$$
$$\Leftrightarrow n = \lambda$$
$$\text{Donc, } \boxed{\pi_j^{(k+1)} = \frac{1}{n} \sum_{i=1}^{n} H_{i,j}(\theta_k)}$$

□

3.1.3 Influence de l'initialisation

Après plusieurs exécutions de notre algorithme EM, on a pu remarquer qu'il nous retourne quelques fois une mauvaise solution sur un même exmple, surtout si deux groupes sont plutôt rapprochés dans notre exmple. Ceci s'explique par le fait que notre convergence est en fait une convergence locale. Donc l'initialisation de notre algorithme joue un rôle sur la convergence de notre algorithme. Pour remédier à ce problème, nous allons faire un algorithme ($MULTI\_EM$) qui va exécuter plusieurs fois notre algorithme EM avec une initilisation différente. Si on s'intéresse à notre choix d'initialisation, on peut faire différents choix possibles. Le choix que nous avons développer est le suivant.

- Pour chaque moyenne μ_j, on choisit aléatoirement un individu de notre ensemble (avec la fonction "sample") et on l'affecte à μ_j. Ainsi, on aura des moyennes de départ qui ont tendance à être plus proches d'un certain groupe par rapport aux autres moyennes.
- Pour les variances Σ_j, on les choisit toutes égales à une matrice diagonale qui a pour coefficients la variance empirique de chaque variable.
- Pour les π_j, on les rend tous identiques entre eux dans un premier temps, puis on affecte à tour de rôle l'un des poids d'une valeur plus importante.

C'est donc le choix de π_j qui nous permet d'avoir plusieurs tests de l'algorithme EM. Maintenant que nous avons fait plusieurs fois l'algorithme, il nous faut trouver quelle solution est la meilleure. Comme nous cherchons à faire des groupes, notre idée est de choisir la solution qui possède une variance intra minimale. La variance intra d'un groupe est un indice qui permet de connaître l'effet de dispersion des individus de ce même groupe. Si cette variance est minimale on a donc des groupes de plus petites tailles possibles. Et donc nous n'aurrons pas un groupe regroupant potentiellement plusieurs ensembles d'individus et des autres groupes regoupant que quelques individus. C'est dans cet objectif que vous avons implémenter l'algorithme Multi EM présent dans l'annexe. Nous avons utiliser la définition de la variance intra suivante.

Définition 4. *Soit un groupe d'individu* $\Gamma = \{x_i; i = 1, ..., n\}$ *et* μ *son individu moyen. On note par* $||.||_2$ *la norme euclidienne. L'inertie* $I(\Gamma)$ *de ce groupe est :*

$$I(\Gamma) = \sum_{i=1}^{n} \frac{1}{n} ||x_i - \mu||_2^2$$

Si l'on dispose d'un ensemble disposant de g groupes $\Gamma_1, ..., \Gamma_g$*, la somme des inerties de chaque groupe est appelée inertie intraclasse.*

$$I_{intra} = I(\Gamma_1) + ... + I(\Gamma_g)$$

3.1.4 Implémentation de l'algorithme EM

Pour l'implémentation de l'algorithme *EM*, nous avons utilisé les techniques et les outils developpés précédemment [A : Outils pour les algorithmes EM et CEM]. Nous allons maintenant discuté comment fonctionne notre algorithme EM, celui-ci est présent en annexe page [A : L'algorithme EM]. Après l'initialisation de nos paramètres, nous avons commencé par définir les valeurs de départ de nos paramètres (π_j, μ_j, Σ_j) présent dans l'ensemble θ_0. Dans la boucle de l'algorithme EM, nous calculons tout d'abord les probabilités conditionnelles que l'individu i appartienne au groupe j à partir des valeurs de (π_j, μ_j, Σ_j) calculées précédemment. Ensuite, nous redéfinissont nos paramètres (π_j, μ_j, Σ_j) avec les formules de la proposition 1. Nous réalisons cette boucle tant

que la précision *eps* de nos paramètres n'est pas atteinte. A la fin de la boucle, nous calculons l'inertie intra de notre jeu de données à partir des paramètres (π_j, μ_j, Σ_j) finaux pour l'algorithme *Multi_EM*. Finalement, *EM* nous renvoie une liste comportant toutes les valeurs (π_j, μ_j, Σ_j) stockées dans un vecteur pour le π_j et dans deux matrices différentes pour les μ_j et les Σ_j, le nombre d'itération et la valeur de l'inertie intra.

Pour l'implémentation de l'algorithme *Multi_EM*, nous avons comparé les valeurs d'inertie intra de plusieurs réalisations de l'algorithme *EM*. Cet algorithme est présent en annexe [A : MULTI_EM et MULTI_CEM]. Tout d'abord, nous réalisons une premiere itération avec un poids identique pour chaque classe de notre ensemble. Puis nous réalisons g itérations (le nombre de la classe) où pour chaque itération, un groupe a un poids plus important que les autres. A chaque fois, nous comparons les valeurs d'inertie intra pour sélectionner le résultat où l'inertie intra est la plus faible. Ainsi, *Multi_EM* nous renvoie une liste comme l'algorithme *EM* dont l'inertie intra est la plus faible parmis les différents résultats des algorithmes réalisés.

3.2 Région de confiance

Maintenant que nous avons nos résultats, nous souhaitons faire une représentation grahique. Nous avons donc décidé de tracer des *régions de confiance* autour des groupes que nous avons identifiés. Cette représentation nécessite les valeurs des moyennes et des matrices de variance de chaque groupe, ce qui nous apporte un autre moyen de vérifier la cohérance de nos résultats lorsque l'on dispose de deux variables. Au delà, une représentation graphique peut s'avérer plus difficile.

Tout d'abord, on rappelle qu'avec une seule variance $X \sim N(m, \sigma)$ le plus petit intervalle tel que $\mathbb{P}(X \in [a; b]) = \alpha$ est un intervalle centré autour de la moyenne(la preuve se situe en "ANNEXE" page 26).

Avec un vecteur gaussien $X = (X_1, X_2) \sim N(\mu, \Sigma)$, notre région de confiance est toujours une courbe de niveau qui a la forme d'une ellipse (la preuve se situe en "ANNEXE" page 27). En effet, pour tracer la région de confiance d'un groupe, on diagonalise la matrice de variance de ce groupe. On obtient alors deux variables indépendantes (corollaire 2). On peut donc appliquer la formule pour obtenir un intervalle de confiance pour chacune des deux variables. Ce qui nous donne les distances du grand et du petit axe d'une ellipse centrée. Ensuite, on applique la matrice de permutation inverse sur notre ellipse centré et on obtient une région de confiance de la forme d'une ellipse.

Sachant maintenant que notre région de confiance pour un vecteur gaussien est une ellipse, nous aimerions avoir une équation définissant cette région en fonction d'un paramètre α.

Proposition 2. *Soit* $X = (X_1, X_2) \sim N(\mu, \Sigma)$ *où* $\mu = (\mu_1, \mu_2)$ *et* $\Sigma = \begin{pmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{pmatrix}$.

Soit ϵ_k *notre ellipse de confiance d'équation* $(\frac{X_1-\mu_1}{\sigma_1})^2 + (\frac{X_2-\mu_2}{\sigma_2})^2 = k^2$ *. L'équation de l'ellipse de confiance de niveau* α *est :*

$$\frac{1}{-2ln(1-\alpha)} \left(\left(\frac{X_1 - \mu_1}{\sigma_1} \right)^2 + \left(\frac{X_2 - \mu_2}{\sigma_2} \right)^2 \right) = 1$$

Démonstration. En remarquant que $(\frac{X_1-\mu_1}{\sigma_1})^2 + (\frac{X_2-\mu_2}{\sigma_2})^2$ suit une loi du Khi-deux de degré de liberté 2 car c'est la somme de deux carrés de loi normale centrée réduite, on a :

$$\begin{aligned}
\mathbb{P}(X \in \epsilon_k) &= \mathbb{P}((\frac{X_1-\mu_1}{\sigma_1})^2 + (\frac{X_2-\mu_2}{\sigma_2})^2 \leq k^2) \\
&= \int_0^{k^2} \frac{1}{2^{\frac{2}{2}}\Gamma(\frac{2}{2})} t^0 e^{-\frac{t}{2}} \mathrm{d}x \\
&= \int_0^{k^2} \frac{1}{2} e^{-\frac{t}{2}} \mathrm{d}x \\
&= [-e^{-\frac{t}{2}}]_0^{k^2} \\
&= \boxed{1 - e^{-\frac{k^2}{2}}}
\end{aligned}$$

On cherche alors à exprimer le coefficient k en fonction d'un seuil de confiance α.

$$\begin{aligned}
\mathbb{P}(X \in \epsilon_k) = 1 - e^{-\frac{k^2}{2}} &= \alpha \\
&\Leftrightarrow 1 - \alpha = e^{-\frac{k^2}{2}} \\
&\Leftrightarrow -ln(1-\alpha) = \frac{k^2}{2} \\
&\Leftrightarrow k^2 = -2ln(1-\alpha)
\end{aligned}$$

On obtient alors l'équation de l'ellipse de confiance suivante :

$$\boxed{\frac{1}{-2ln(1-\alpha)} \left(\left(\frac{X_1-\mu_1}{\sigma_1} \right)^2 + \left(\frac{X_2-\mu_2}{\sigma_2} \right)^2 \right) = 1}$$

□

Nous savons maintenant tracer une ellipse lorsque nous avons un vecteur gaussien dont les composantes sont indépendantes. Pour pouvoir généraliser notre cas à un vecteur gaussien quelconque de variance Σ, nos devons juste diagonaliser cette matrice $\Sigma = PDP^{-1}$. Ainsi, si l'on trace d'abord l'ellipse de confiance pour un vecteur gaussien de variance D puis nous faisons un changement de base avec la matrice P, nous aurons donc une ellipse de confiance pour notre vecteur gaussien de variance Σ. C'est dans cet objectif que nous avons implémenter la fonction *ellipse_dim2*.

Nous pouvons donc maintenant tracer des ellipses de confiance de niveau α autour de nos moyennes estimées de chaque groupe. Nous avons aussi implémenté la fonction *graph_EM* [A : Graphique EM et CEM] pour avoir directement une représentation graphique de notre mélange gaussien avec les ellipses de confiance de chaque groupe. Voici le résultat obtenue sur le mélange à deux lois et un niveau de confiance de 95% :

Avec les ellipses de confiance, nous pouvons avoir un idée sur l'appartenance d'un individu à un certain groupe. Mais nous aimerons savoir exactement à quel groupe un individu est le plus susceptible d'appartenir. C'est dans cet objectif que nous avons programmé l'algorithme *CEM*.

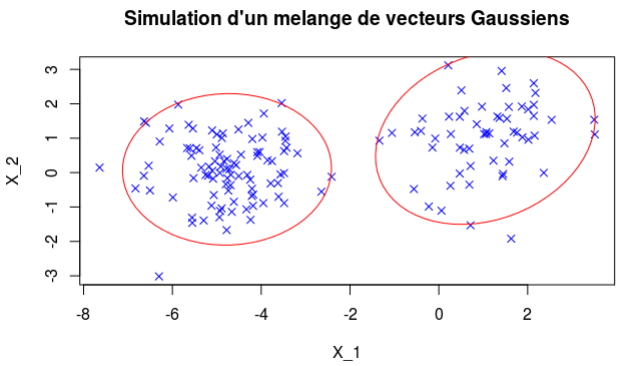


FIGURE 3 – Exemple d'un mélange avec deux lois gaussiennes

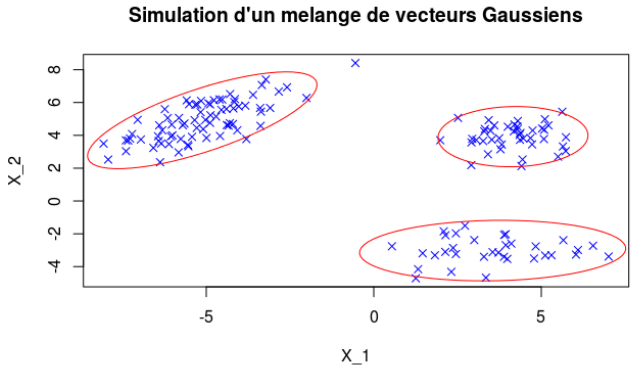


FIGURE 4 – Exemple d'un mélange avec trois lois gaussiennes

3.3 L'algorithme CEM

Dans cette section, nous allons développer le fonctionnement de l'algorithme *CEM*. Notre but est toujours d'estimer notre $\theta = \{\pi_1, ..., \pi_g, \mu_1, ..., \mu_g, \Sigma_1, ..., \Sigma_g\}$ et aussi d'estimer la partition z qui est une matrice de taille $n \times g$ où n est le nombre d'individus de notre jeu de données et g le nombre de groupe. Ainsi, nous pourrons facilement savoir qui composent nos différents groupes. Une représentation graphique sera possible lorsque nous disposons d'un jeu de données caractérisé par deux variables avec la fonction *graph_ CEM*.

3.3.1 Définition

L'algorithme CEM(classification-espérance-maximisation) fonctionne d'une manière similaire à l'algorithme EM. Cet algorithme repose sur trois grandes étapes que l'on réitère un certain nombre de fois.

- Etape (E) : une estimation de l'espérance, où l'on calcule la probabilité conditionelle qu'un individu x_i soit issu du groupe G_k.
- Etape (C) : une estimation de l'appartenance des individus à l'une des classes en calculant $\hat{z}$.
- Etape (M) : une estimation du maximum, où l'on maximise l'espérance de la log vraisemblance observée des paramètres en remplaçant la partition z par l'estimation de la partition $\hat{z}$.

Comme l'initialisation joue un rôle sur notre résultat final, nous avons utilisé le même procédé que pour l'aglorithme *EM* en implémentant *Multi_ CEM*. Nous réalisons plusieurs fois l'algorithme CEM avec une initialisation de nos π_j différentes et on sélectionne le résultat possédant une inertie intra la plus faible.

3.3.2 Classification avec CEM

Nous avons donc grâce à l'algorithme CEM une partition $\hat{z}$ de notre échantillon. Cette partition repose sur le principe MAP (Maximim A Posteriori) affectant chaque individus x_i à la classe de plus grande probabilité conditionnelle estimée :

$$\hat{z_{ik}} = \begin{cases} 1 & si \quad k = argmax_{k'=1,...,g}(\mathbb{P}_{\hat{\theta}}(z_{ik'}|X_i = x_i)) \\ 0 & sinon \end{cases}$$

Nous pouvons utiliser cette partition pour notre représentation graphique. La fonction *graph_ CEM* récupère la partition $\hat{z}$ et associe un symbole à chaque individu x_i en fonction de son appartenance à une classe, c'est-à-dire en fonction de la position du 1 sur la i^{me} lignes de $\hat{z}$. Elle représente aussi les ellipse de confiance avec les paramètres estimés par l'algorithme *CEM*. L'implémentation de l'algorithme *CEM* se fait de la même manière que pour *EM*. Cet algorithme est présent en annexe A : L'algorithme CEM. Dans *CEM*, les paramètres (π_i, μ_i, Σ_i) sont calculés ne sont plus avec les valeurs des probabilités conditionnelles mais avec la partition $\hat{z}$. Le reste de l'algorithme *CEM* est identique à l'algorithme *EM*. De plus, notre l'algorithme *MULTI_ CEM* [A : MULTI_EM et MULTI_CEM], fonctionne aussi de la même façon que *MULTI_ EM* sauf qu'il utilise *CEM* à la place de *EM*.

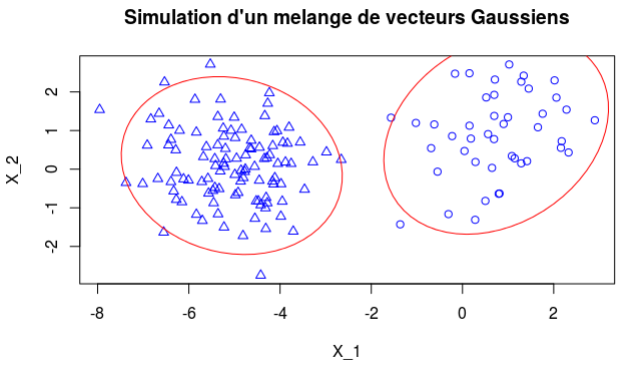


FIGURE 5 – Exemple d'un mélange avec deux lois gaussiennes

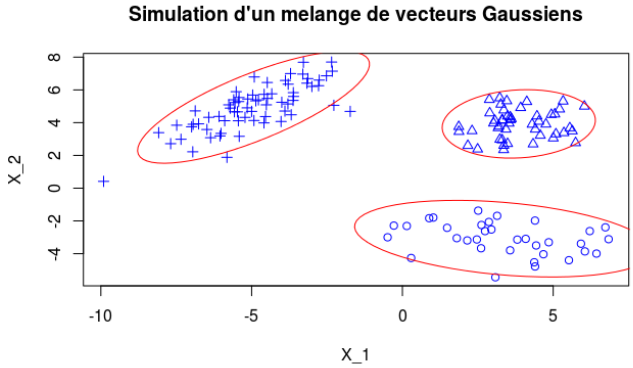


FIGURE 6 – Exemple d'un mélange avec trois lois gaussiennes

3.4 Comparaison entre EM et CEM

Outre le fait que l'algorithme CEM nous donne une sortie supplémentaire par rapport à l'agorithme EM (la partition $\hat{z}$), on peut remarquer quelques différences en pratique entre ces deux algorithmes. En effet, CEM a l'avantage d'être un algorithme stationnaire par rapport à EM qui a un paramètre de précision pour arrêter la boucle. De plus, nous remarquons que ces deux algorithmes convergent localement vers une solution, d'où l'importance des fonctions *Multi_ EM* et *Multi_ CEM*. En pratique, nous remarquons que CEM a tendance à converger plus rapidement que EM. Mais le résultat peut comporter un biais qui est d'autant plus important si les groupes sont fortement imbriqués. Ceci s'explique par le fait que l'agorithme CEM remplace les probabilités conditionnelles à l'étape (E) par notre $\hat{z}$ calculé grâce au principe MAP. Par exemple, avec un jeu de données composé de deux classes, si nous avons que la probabilité conditionnelle d'appartenir au premier groupe est de 0.6 et celle du second de 0.4, EM utilise ces probabilités comme des poids alors que CEM remplace ces probabilités par 1 et 0 pour les premiers et seconds groupes respectivement. Nous pouvons aussi remarquer que CEM apporte une meilleure estimation des paramètres si l'échantillon est petit.

3.5 Choix du modèle

Dans cette section, nous allons chercher le meilleur modèle pour notre jeu de donnée. Nous cherchons donc à définir le nombre de classes optimal g de notre jeu de données. C'est-à-dire nous

cherchons à définir le nombre de classes sans pour autant qu'il ne soit trop execessif. Si nous avons une classe pour chaque individu, notre analyse peut s'avérer inutile. En effet, augmenter le nombre de paramètre améliore nécessairment la qualité de notre analyse. Il nous faut donc un nombre de classe minimale pouvant expliquer au maximun les différentes classes de notre jeu de donnée. Nous appelons ce modèle un modèle parcimonieux. Pour choisir le meilleur modèle répondant à nos attentes, il existe plusieurs critères classiques en statistiques qui se définissent comme une pénalisation de la log-vraisemblance maximale.

$$\star IC_m = L(\hat{\theta}_m; x) - v_m f(n)$$

où $\hat{\theta}_m$ désigne l'estimation du maximum de la vraisemblance sous le modèle m, v_m le nombre de paramêtre à estimer et f est une fonction de n. On retient alors modèle m ayant la plus grande valeur du critère. Il existe plusieurs choix pour la fonction f. Nous allons développer deux entre elles.

3.5.1 Le critère BIC

Le critère BIC [A : Critere AIC ou critere BIC] (Bayesian Information Criterion) propose le choix de f suivant [5].

$$f(n) = \frac{1}{2} log(n)$$

BIC se place dans un contexte bayésien (nos paramètres θ_i et nos différents modèles à tester m_i sont vu comme des variables aléatoires). Ce critère cherche à sélectionner le modèle m_i qui maximise la probabilité *a posteriori* $\mathbb{P}(m_i|X)$:

$$m_{BIC} = \arg\max_{m_i} \mathbb{P}(m_i|X)$$

D'après la formule de Bayes, nous pouvons réécrire $\mathbb{P}(m_i|X)$ comme étant :

$$\mathbb{P}(m_i|X) = \frac{\mathbb{P}(X|m_i)\mathbb{P}(m_i)}{\mathbb{P}(X)}$$

Si l'on suppose que chaque $\mathbb{P}(m_i)$ ait la même valeur quelque soit le modèle, la recherche du meilleur modèle selon BIC dépend seulement du calcul de $\mathbb{P}(X|m_i)$. De plus, avec la méthode d'approximation de Laplace, nous obtenons une valeur approchée de $\mathbb{P}(X|m_i)$.

$$\mathbb{P}(X|m_i) = e^{V(m_i,\theta_i^\star)} \left(\frac{2\pi}{n}\right)^{v_{m_i}/2} |A_{\theta_i^\star}|^{-1/2} + \mathcal{O}(n^{-1})$$

où V désigne la vraisemblance, $\theta_i^\star$ est l'$\arg\max_{\theta_i} \frac{1}{n}\sum_{k=1}^{n} log(V(X_k, \theta_i))$ et $A_{\theta_i^\star}$ l'opposé de la matrice hessienne des dérivées secondes partielles de la fonction $\frac{1}{n}\sum_{k=1}^{n} log(V(X_k, \theta_i))$ en θ_i. En remarquant que $log(\mathbb{P}(X|m_i)) \approx log(V(m_i, \hat{\theta}_i)) - \frac{v_{m_i}}{2} log(n)$, nous pouvons définir BIC_i comme :

$$BIC_i = log(V(m_i, \hat{\theta}_i)) - \frac{v_{m_i}}{2} log(n)$$

Le modèle sélectionné par ce critère est alors :

$$m_{BIC} = \arg\max_{m_i} BIC_i$$

.

3.5.2 Le critère AIC

Le critère AIC [A : Critere AIC ou critere BIC] (Akaike Information Criterion) propose le choix de f suivant [4].

$$f(n) = 1$$

L'AIC est basé sur la théorie de l'information. L'objectif de ce critère est de choisir le modèle vérifiant l'expression suivante.

$$m_{AIC} = \arg\max_{m_i} \mathbb{E}\left[\int log\left(\frac{f(x)}{V(m_i, \theta_i)}\right)\ f(x)\ \ dx\right]$$

Nous pouvons retrouver ce modèle en maximisant la valeur suivante :

$$m_{AIC} = \arg\max_{m_i} log(V(m_i, \hat{\theta}_i)) - v_{m_i}$$

3.5.3 Comparaison entre BIC et AIC

Le choix d'un des deux critères se porte sur le point de vue que nous adoptons sur le problème. Si nous voulons savoir quelles méthodes nous donnent le meilleur modèle en simulant des données à partir d'un modèle m_t, le critère BIC a plus tendance à nous retrouver notre modèle de départ m_t. Mais si nous voulons choisir le critère en fonction de la qualité de la prédiction, le critère AIC s'avère plus performant. En conclusion, nous pouvons dire que le choix d'un critère de sélection de modèles dépend de l'objectif de l'analyse et de la connaissance des données. C'est dans ce but que nous avons décidé de nous intéresser au critère de sélection ICL.

3.5.4 Le critère ICL

Le critère ICL [A : Critere ICL] (Integrated Complete Likelihood) a pour philosophie de reporter l'objectif de la classification de la méthode d'estimation vers la méthode de sélection du modèle. En effet, au lieu de pénaliser la log-vraisemblance comme dans AIC ou BIC, le critère ICL pénalise la log-vraisemblance complétée du même therme que le critère BIC. Ainsi, nous définissons ce critère de la manière suivante :

$$\begin{aligned} ICL_{m_i} =& L(\hat{\theta}_{m_i}, x, \hat{z}_{m_i}) - \frac{v_{m_i}}{2} log(n) \\ =& L(\hat{\theta}_{m_i}, x) - \mathbb{E}(\hat{t}_{m_i}, \hat{z}_{m_i}) - \frac{v_{m_i}}{2} log(n) \\ =& BIC_i - \mathbb{E}(\hat{t}_{m_i}, \hat{z}_{m_i}) \end{aligned}$$

où $\hat{t}_{m_i}$ désigne l'estimation de la probabilité conditionnelle qu'un individu appartienne à un certain groupe avec le modèle m_i.

4 Classification supervisée

4.1 Modélisation

En classification supervisée, nous disposons en théorie du couple (x, z) pour notre échantillon, c'est-à-dire que nous connaissons quel individu appartient à quel groupe. Cet échantillon est appelé *ensemble d'apprentissage*. Et nous souhaitons, à l'instar de la régression linéaire, prévoir pour toute nouvelle observation le groupe auquel elle appartient. En définitive, nous cherchons à estimer une règle de classement r définie par :

$$\begin{aligned} r :&\mathcal{X} \to \{1, ..., g\} \\ &x_{n+1} \mapsto r(x_{n+1}) \end{aligned}$$

D'autre part, nous souhaitons également avoir une description des groupes et plus précisément de la frontière qui les sépare.

Pour résoudre ce problème de classification, nous nous appuyons sur les probabilités conditionnelles pour obtenir une règle de classement r. Cette règle correspond au critère MAP que l'on a vu précédement :

$$\forall x_{n+1} \in \mathcal{X}, r(x_{n+1}) = \underset{k \in \{1,...,g\}}{Argmax}\ \mathbb{P}(Z_{n+1,k} = 1 | X_{n+1} = x_{n+1})$$

4.2 Implémentation des Algorithmes

Afin de visualiser les résultats de ces règles de classement, nous avons créer des algorithmes qui nous trace les frontières de groupe (figure 8) pour notre exemple avec deux lois.

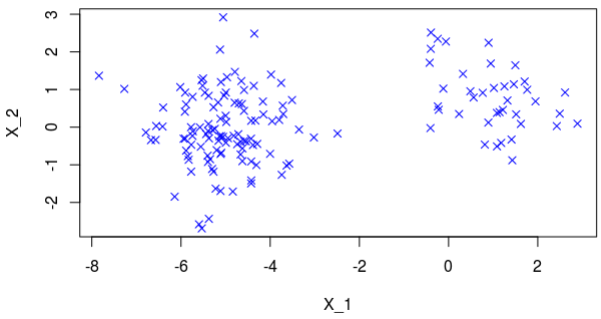


FIGURE 7 – Mélange Gaussien avec deux lois

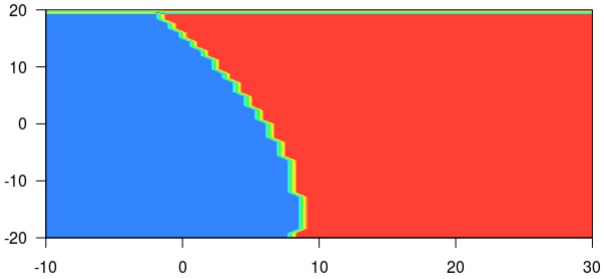


FIGURE 8 – Graphique des frontières

Pour obtenir ce graphique nous avons tracé un maillage de $2(N + 1)$ points du plan. On souhaite stocker chaque point du maillage dans une matrice $A \in \mathcal{M}_{N+1,N+1}$ la valeur du groupe auquel appartient l'individu correspondant. Pour cela nous calculons le critère MAP en chaque $A_{i,j}, \forall \in \{1, ..., N + 1\}$.

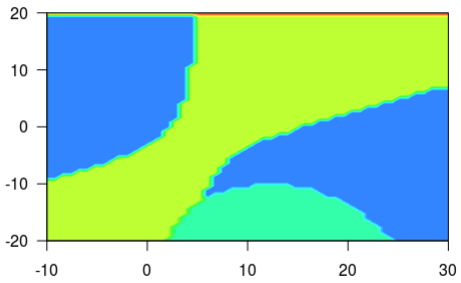
Nous avons donc besoin de deux fonctions, une première nommée *appartenance* qui implémente MAP et une seconde *graph_ coloree* qui crée le maillage et affiche le graphe [A : Graphique colore].

Appartenance va prendre en entrée un individu x_i, le nombre de groupe g ainsi que notre paramètre θ. Ce programme calcule les probabilités conditionnelles $\mathbb{P}_\theta(Z_{i,k} = 1|X_i = x_i), \forall k \in \{1, .., g\}$ que nous stockons dans un vecteur et retourne le numéro de la classe auquel x_i appartient avec la fonction *which.max()*.

Quant à *graph_ coloree*, nous donnons le numéro du groupe le plus probable pour tout point de $\mathbb{R}^2$ dans le pavé défini par les deux paramètres d'entrés *xlim* et *ylim*. Grâce à notre *appartenance*, nous stockons ensuite ces valeurs dans A afin de pouvoir afficher le graphe avec la fonction *filled.contour* de RStudio.

4.3 Frontière entre les groupes

En observant le graphique ci-dessous, on constate que les frontières entre les groupes ont des formes particulières.Nous distinguons une hyperbole (en Jaune et Bleu) et une ellipse (en vert) qui s'ajoute au graphique. Les caractéristiques mathématiques des frontières correspondent à l'égalité des probabilitées conditionelles de classement d'un individu dans un des groupes.



Nous pouvons dans le cas d'un mélange à deux lois, retrouver les équations théoriques régissant les frontières des classes en distinguant deux cas : un premier cas homoscédastique et le second hétéroscédastique.

L'égalité des probabilitées conditionelles s'écrit comme suit :

$$\mathbb{P}(Z_{i1} = 1|X_i = x) = \mathbb{P}(Z_{i2} = 1|X_i = x)$$

La formule de Bayes nous donne ensuite : $\frac{\mathbb{P}(Z_{i1}=1 \cap X_i=x)}{\mathbb{P}(X_i=x)} = \frac{\mathbb{P}(Z_{i2}=1 \cap X_i=x)}{\mathbb{P}(X_i=x)}$

Il en résulte les équivalences suivantes :

$$\begin{aligned}
& \mathbb{P}(Z_{i1} = 1 \cap X_i = x) = \mathbb{P}(Z_{i2} = 1 \cap X_i = x) \\
\Leftrightarrow \quad & \mathbb{P}(X_i = x|Z_{i1}).\mathbb{P}(Z_{i1}) = \mathbb{P}(X_i = x|Z_{i1}).\mathbb{P}(Z_{i1}) \\
\Leftrightarrow \quad & \pi_1.f_1(x) = \pi_2.f_2(x) \\
\Leftrightarrow \quad & \pi_1.|\Sigma_1|^{-1}.e^{-\frac{1}{2}(x-\mu_1)^t.\Sigma_1^{-1}.(x-\mu_1)} = \pi_2.|\Sigma_2|^{-1}.e^{-\frac{1}{2}(x-\mu_2)^t.\Sigma_2^{-1}.(x-\mu_2)} \\
\Leftrightarrow \quad & log(\frac{\pi_1}{\pi_1}) + log[det(\Sigma_1^{-1}).det(\Sigma_2)] = \frac{1}{2}(x-\mu_1)^t.\Sigma_1^{-1}.(x-\mu_1) - \frac{1}{2}(x-\mu_2)^t.\Sigma_2^{-1}.(x-\mu_2)
\end{aligned}$$

— Si $\Sigma_1 = \Sigma_2 = \Sigma$ l'équation devient :

$$\begin{aligned}
2ln(\frac{\pi_1}{\pi_2}) &= (x-\mu_1)^t.\Sigma^{-1}.(x-\mu_1) - (x-\mu_2)^t.\Sigma^{-1}.(x-\mu_2) \\
&= (x-\mu_1-\mu_2+\mu_2)^t.\Sigma^{-1}.(x-\mu_1-\mu_2+\mu_2) - (x-\mu_2)^t.\Sigma^{-1}.(x-\mu_2) \\
&= (x-\mu_2)^t.\Sigma^{-1}.(\mu_2-\mu_1) + (\mu_2-\mu_1)^t.\Sigma^{-1}.(\mu_2-\mu_1) \\
&= 2.x^t.\Sigma^{-1}.(\mu_2-\mu_1) - (\mu_1+\mu_2).\Sigma^{-1}.(\mu_2-\mu_1)
\end{aligned}$$

L'équation décrivant la frontière entre les deux lois est alors une droite :

$$\boxed{ln(\frac{\pi_1}{\pi_2}) = (\mu_2-\mu_1).\Sigma^{-1}.x - \frac{1}{2}.(\mu_1-\mu_2).\Sigma^{-1}.(\mu_2+\mu_1)}$$

— Si $\Sigma_1 \neq \Sigma_2$ (cas hétéroscedastique). L'équation devient une conique :

$$log(\frac{\pi_1}{\pi_1}) + log[det(\Sigma_1^{-1}).det(\Sigma_2)] = \frac{1}{2}(x-\mu_1)^t.\Sigma_1^{-1}.(x-\mu_1) - \frac{1}{2}(x-\mu_2)^t.\Sigma_2^{-1}.(x-\mu_2)$$

Après simplification, on trouve l'équation d'une conique :

$$\boxed{log(\frac{\pi_1}{\pi_1}) + log[det(\Sigma_1^{-1}).det(\Sigma_2)] = \frac{1}{2}.x^t(\Sigma_1^{-1}-\Sigma_2^{-1}).x + x[\Sigma_2^{-1}.\mu_2 - \Sigma_1^{-1}]\frac{1}{2}(\mu_1.\Sigma_1^{-1}.\mu_1 - \mu_2.\Sigma_2^{-1}.\mu_1 2)}$$

4.4 Les deux problèmes pour classer un individu

Deux problèmes quant à l'utilisation de cette méthode apparaissent pour classer une nouvelle observation. En pratique, le paramètre θ nous est inconnu et on l'estime avec une des méthodes en classification non supervisée. Outre les limitations de EM/CEM pour des lois trop imbriquées, le fait de vouloir classer un individu trop proche de la frontière ou trop éloigné des moyennes des groupes peut être problématique.

En effet, pour comprendre la difficulté de classement aux frontière, reprenons l'exemple du mélange à deux lois. La figure ci-dessus nous montre la frontière théorique et celle de droite représente la frontière obtenue avec l'algorithme EM et *grahpe_ coloree*. Nous voyons nettement la difference entre les deux qui est due à l'erreur commise lors de l'approximation de θ. Nous prendrons alors la liberté de ne pas classer ce type d'individus. On parle dans ce cas de *rejet d'ambiguïté*.

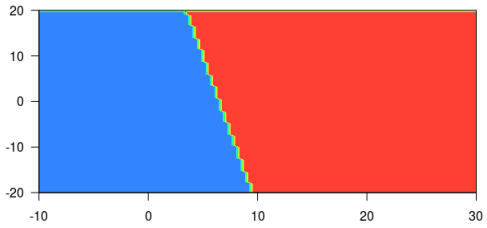


FIGURE 9 – Graphique des frontières théoriques

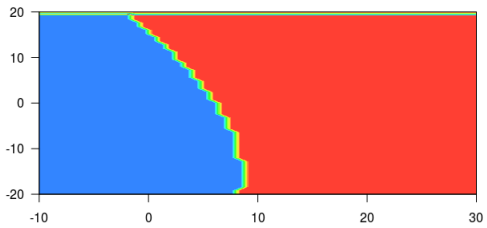


FIGURE 10 – Graphique des frontières avec EM

Une deuxième forme de rejet consiste à ne pas classer une nouvelle observation dite atypique, c'est-à-dire très en dehors des zones de confiance associées aux groupes. La figure ci-dessus affiche en rouge les éllipses de confiances au seuil de 95 %. Si nous prenons par exemple un nouvel individu x_{n+1} de coordonées $(-5, -5)$. D'après la figure 10, la règle du MAP classerait sans hésiter x_{n+1} dans le groupe de gauche. Cependant ce nouveau point étant très éloigé des régions de confiances(fig. **??**), il n'est pas très pertinent de l'affecter à l'une des classes car il pourrait provenir d'une nouvelle classe pas encore obsérvée jusqu'à présent. Ce type de rejet est appelé *rejet de distance*.

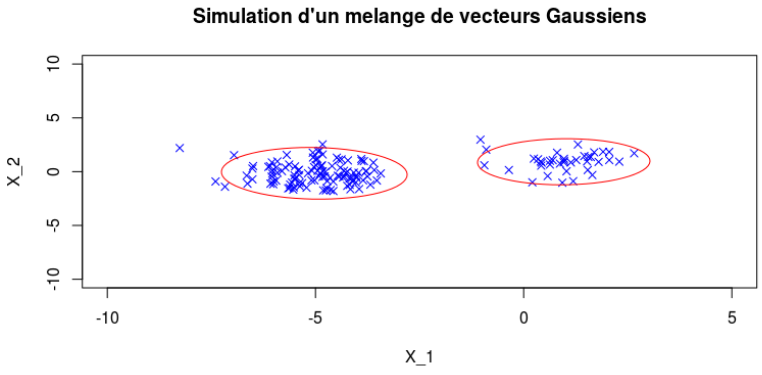


FIGURE 11 – Mélange deux lois avec ellipses confiances à 95%

5 Exemple avec le jeu de données *Iris*

Nous récupérons le jeu de données *Iris* qui contient notamment 5 variables : *Sepal.length* / *Sepal.width* / *Petal.length* / *Petal.witdth* / *Species*. Nous pouvons tracer les deux graphes des largeurs en fonction des longueurs pour les pétales et les sépales. De plus, avec un code couleur nous pouvons voir la réelle appartenance de chaque fleur a son espèce.

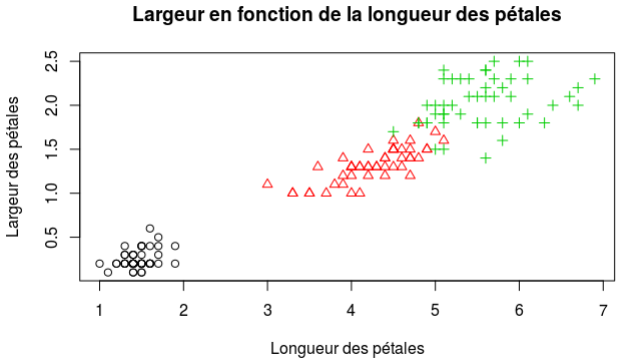


FIGURE 12 – Graphique des individus classés théoriques pour les pétales

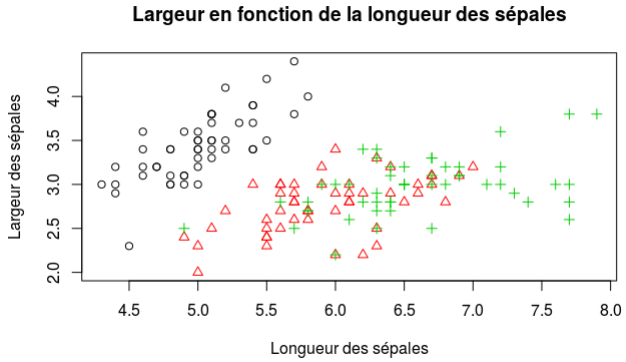


FIGURE 13 – Graphique des individus classés théoriques pour les sépales

Notre objectif est maintenant de retrouver cette classification. Tout d'abord, n'ayant que les valeurs des longueurs et des largeurs, nous cherchons le nombre de classes de l'échantillon. En utilisant *critere_ICL* sur les pétales et les sépales, nous remarquons que celui-ci nous retourne un nombre de classes qui est égale à deux ou trois. De plus, lorsque nous recherchons le nombre de groupe avec les variables des sépales, nous obtenons régulièrement deux. Cela peut s'expliquer par le fait que les classes réprésentées en rouge et en vert sur la figure 13 sont imbriquées.

En gardant la valeur de trois pour le nombre de groupe, nous appliquons dans un premier temps l'algorithme *MULTI_EM* puis *graph_EM* pour nos deux jeux de données.

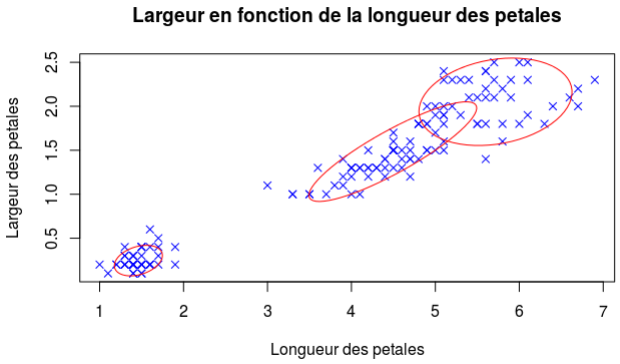


FIGURE 14 – Graphique avec les ellipses de confiance pour les pétales

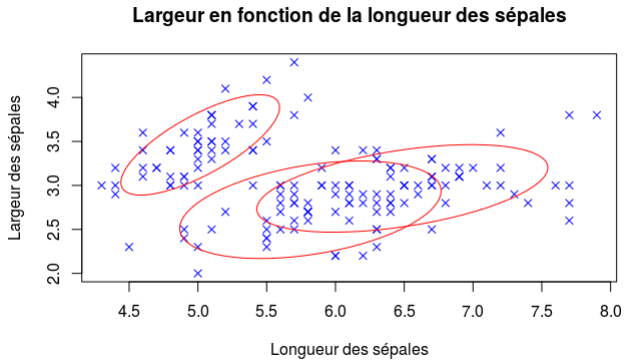


FIGURE 15 – Graphique avec les ellipses de confiance pour les sépales

Nous pouvons dire que l'algorithme *EM* a bien retrouvé les différents groupes de nos ensembles. De plus, nous remarquons d'autant plus le fait que deux classes sont imbriquées au vu du résultat présent sur la figure 15.

Nous pouvons maintenant nous intéresser à l'appartenance de chaque individu à une classe avec l'algorithme *MULTI_ CEM* et *graph_ CEM*.

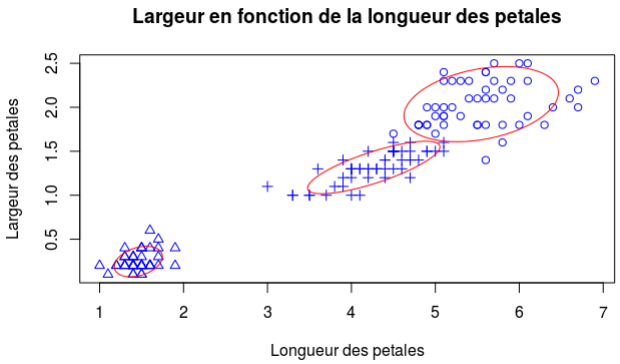


FIGURE 16 – Graphique avec les ellipses de confiance et classification de chaque individu pour les pétales

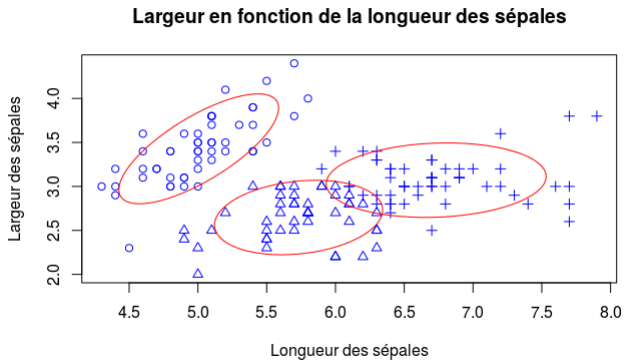


FIGURE 17 – Graphique avec les ellipses de confiance et classification de chaque individu pour les sépales

Cette fois ci, nous constatons que les groupes obtenus par l'algorithme *CEM* sont beaucoup moins imbriqués que pour *EM*. Si nous comparons ces résultats avec les figure 12 et 13, nous voyons que la partition estimée pour les pétales correspond bien au cas théorique. Il doit y avoir environ trois ou quatre erreurs de classement. En revanche pour les sépales, les classes étant plus fortement imbriquées, nous remarquons une erreur de classement plus importante : environ dix individus mal classés.

Finalement, nous nous intéressons à réaliser la classification supervisée de nos deux ensembles grâce à *graph_ colore*.

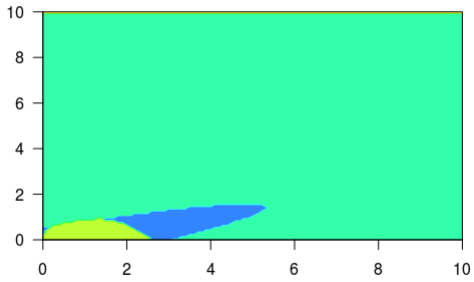


FIGURE 18 – Graphique des frontières calculées empiriques

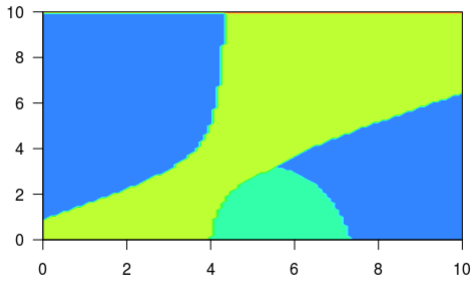


FIGURE 19 – Graphique des frontières calculées empiriques

Concernant les pétales, nous constatons que l'un des groupes, en cyan, est extrêmement dominant. On peut néammoins être critique sur ce résultat en disant de ne pas classer un individu trop éloigné de notre échantillon original(rejet de distance). Si nous devons classer plusieurs individus supplémentaires loin de notre ensemble, ces individus pourraient former une autre classe. Pour les sépales, il est plus facile de savoir à quel groupe un individu supplémentaire peut appartenir.

6 Conclusion

Pour conclure, nous avons introduit divers méthodes et algorithmes pouvant réaliser une classification d'éléments d'un jeu de données. Cet présentation reste néamoins qu'une introduction. Les algorithmes réalisés ne sont pas optimisés mais restent efficaces pour des jeux de données de taille raisonnable. De plus, lors de cette présentation nous avons fait des choix de méthodes aussi bien que pour l'initialisation de *EM* ou *CEM* que pour l'importance accordée à l'inertie des groupes.

Il existe d'autres méthodes faisant d'autres choix qui peuvent s'avérer plus efficaces dans certains cas. Nous avons décidé de nous intéresser à la *classification supervisée* et *non supervisée* mais il existe justement un juste milieu en faisant une *classification semi-supervisée* pour des données de très haute dimension (plusieurs millers de variables) ou encore pour des données partiellement étiquetées.

Références

[1] Wikipédia : Algorithme espérance-maximisation. https ://fr.wikipedia.org/wiki/Algorithme_espérance-maximisation.

[2] Christophe Biernacki. Pourquoi les modèles de mélange pour la classification ? *Université Lille 1*, oct 2009.

[3] Jean Christophe Breton. Estimation d'un mélange avec l'algorithme em. *Université Rennes 1*, 2008.

[4] J. de Leeuw. Akaike(1973) information theory and an extension of the maximun likelihood principale. *University of California at Los Angeles*, 1992.

[5] Tristan Mary-Huard Emilie Lebarbier. Le critère bic : fondements théoriques et interprétation. *Inria*, sept 2004. https ://hal.inria.fr/inria-00070685.

[6] Yoann Gelineau. Vecteurs gaussiens. *Université Lyon 1*, dec 2007.

[7] Daniel Li. Vecteurs aléatoires gaussiens. *Université d'Artois*.

[8] Ruocong Zhang Marc Weber. Formulaire de déerivation matricielle. *Inria*, Oct 2009.

A Annexe

Intervalle de confiance en dimension 1

Dans cette partie, nous allons prouver que pour une variable aléatoire $X \sim N(m, \sigma^2)$ son intervalle de confiance est un intervalle centré autour de la moyenne. Sans perte de généralités, on peut prendre une variable aléatoire $X \sim N(0, 1)$.
On cherche les coefficients $a < 0$ et $b > 0$ tel que :

$$\mathbb{P}\left(X \in [a; b]\right) = \alpha > 0.5$$

On note f la densité de la variable X.

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{\sigma^2}}$$

Supposons que [a,b] est le plus petit intervalle vérifiant $\mathbb{P}\left(X \in [a; b]\right) = \alpha$ tel que $|a| > |b|$.

$$\begin{aligned}
\mathbb{P}\left(X \in [a; b]\right) &= \int_a^b f(x)\mathrm{d}x \\
&= \int_0^b f(x)\mathrm{d}x + \int_a^0 f(x)\mathrm{d}x \\
&= \int_{\frac{|a|+|b|}{2}}^b f(x)\mathrm{d}x + \int_0^{\frac{|a|+|b|}{2}} f(x)\mathrm{d}x + \int_{-\frac{|a|+|b|}{2}}^0 f(x)\mathrm{d}x + \int_a^{-\frac{|a|+|b|}{2}} f(x)\mathrm{d}x \\
&= \int_{-\frac{|a|+|b|}{2}}^{\frac{|a|+|b|}{2}} f(x)\mathrm{d}x - \int_b^{\frac{|a|+|b|}{2}} f(x)\mathrm{d}x + \int_a^{-\frac{|a|+|b|}{2}} f(x)\mathrm{d}x
\end{aligned}$$

On cherche maintenant le signe de $-\int_b^{\frac{|a|+|b|}{2}} f(x)\mathrm{d}x + \int_a^{-\frac{|a|+|b|}{2}} f(x)\mathrm{d}x$.

$$-\int_b^{\frac{|a|+|b|}{2}} f(x)\mathrm{d}x + \int_a^{-\frac{|a|+|b|}{2}} f(x)\mathrm{d}x = -\int_b^{\frac{|a|+|b|}{2}} f(x)\mathrm{d}x + \int_{\frac{|a|+|b|}{2}}^{-a} f(x)\mathrm{d}x \quad \textit{car } f \textit{ est pair}$$

On sait que f est décroissante sur $[0, +\infty]$ (f est même unimodale) et $|a| > |b|$.

$$\textit{Donc} \quad \int_b^{\frac{|a|+|b|}{2}} f(x)\mathrm{d}x > \int_{\frac{|a|+|b|}{2}}^{-a} f(x)\mathrm{d}x$$

$$-\int_b^{\frac{|a|+|b|}{2}} f(x)\mathrm{d}x + \int_a^{-\frac{|a|+|b|}{2}} f(x)\mathrm{d}x < 0$$

$$\mathbb{P}\left(X \in [a; b]\right) < \int_{-\frac{|a|+|b|}{2}}^{\frac{|a|+|b|}{2}} f(x)\mathrm{d}x = \mathbb{P}\left(X \in \left[-\frac{|a|+|b|}{2}; \frac{|a|+|b|}{2}\right]\right)$$

Courbes de niveau

Dans cette partie, nous allons expliciter les courbes de niveau de la fonction de densité d'un vecteur gaussien de dimension d.

Pour un vecteur gaussien de dimension d, la fonction de densité est :

$$f(x) = \frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)}$$

On cherche donc une expression en fonction de x tel que :

$$\begin{aligned}
f(x) &= r \\
\frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)} &= r \\
e^{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)} &= r(2\pi)^{\frac{d}{2}}|\Sigma|^{\frac{1}{2}} \\
(x-\mu)^T\Sigma^{-1}(x-\mu) &= -2\log(r(2\pi)^{\frac{d}{2}}|\Sigma|^{\frac{1}{2}}) \\
(x-\mu)^T\Sigma^{-1}(x-\mu) &= R
\end{aligned}$$

Comme vu précédemment, avec cette dernière ligne de calcul on a une équation d'une ellipsoide si R est positif ou nul. En effet :

$$\begin{aligned}
r &\in \left]0; \frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma|^{\frac{1}{2}}}\right] \\
r(2\pi)^{\frac{d}{2}}|\Sigma|^{\frac{1}{2}} &\in]0;1] \\
\log(r(2\pi)^{\frac{d}{2}}|\Sigma|^{\frac{1}{2}}) &\leq 0 \\
-2\log(r(2\pi)^{\frac{d}{2}}|\Sigma|^{\frac{1}{2}}) = R &\geq 0
\end{aligned}$$

Simulation

```
r_VG <- function(MU,SIG){
  # --- nos parametres ---
  d <- length(MU)
  x <- matrix(0,nrow = d,ncol = 1)
  # -------
  # --- Calcul de l'echantillon ---
  x <- rnorm(d,0,sqrt(eigen(SIG)$values))
  x <- eigen(SIG)$vectors %*% x + MU
  # -------
  return(x)
}
d_VG <- function(x,MU,SIG){
  # --- nos parametres ---
  d <- length(x)
  P <- eigen(SIG)$vectors
  y <- t(eigen(SIG)$vectors) %*% (x-MU)
  # -------
  # --- Calcul de la densite ---
  densit <- dnorm(y,0,sqrt(eigen(SIG)$values))
  # -------
  return(prod(densit))
}
r_MelangeGauss <- function(n,PI,MU,SIG){
  # --- nos parametres ---
  d <- length(MU[1,])
  g <- length(PI)
  x <- matrix(0,nrow = n,ncol = d)
  # -------
  # --- Calcul de l'echantillon ---
  for (i in 1:n) {
    # tirage de la loi multinomiale
    m = rmultinom(1,1,PI)
    for (j in 1:g) {
      if (m[j]) {
        x[i,] <- t( r_VG( MU[j,], SIG[((j-1)*d+1):(j*d),]) )
      }
    }
  }
  # -------
  return(x)
}
```

Outils pour les algorithmes EM et CEM

```
H <- function(x,PI,MU,SIG){
  # --- nos parametres --
  n <- length(x[,1])
  g <- length(PI)
  h <- matrix(0,nrow = n,ncol = g)
  # -------
  # --- Calcul de la probabilite conditionnelle ---
  # quand g = 1
  if (g==1){
    h <- matrix(1,nrow = n,ncol = g)
  # autres cas
  } else {
    d <- dim(x)[2]
    df <- 0
    for (i in 1:n){
      df <- 0
      for (j in 1:g){
        df <- df + ( PI[j] * d_VG(x[i,],MU[j,],SIG[((j-1)*d+1):(j*d),]) )
      }
      for (j in 1:g){
        h[i,j] <- (PI[j]*d_VG(x[i,],MU[j,],SIG[((j-1)*d+1):(j*d),])) / df
      }
    }
  }
  # -------
  return(h)
}
norme <- function(x){
  return(sqrt(sum(x**2)))
}
Inertie_intra <- function(x,z,MU,g,n){
  res <- 0
  for(i in 1:n){
    for (j in 1:g){
      res <- res + z[i,j]*norme(x[i,]-MU[j,])**2
    }
  }
  return(res)
}
```

L’algorithme EM

```
EM <- function(x,g,eps,init_p){
  # --- nos parmetres ---
  n <- length(x[,1])
  d <- length(x[1,])
  p <- vector()
  MU <- matrix(0,nrow = g,ncol = d)
  SIG <- matrix(0,nrow = g*d,ncol = d)
  # -------
  # --- Initilisation ---
  p <- init_p
  u <- matrix(0,nrow = d,ncol = d)
  for (j in 1:g){
    MU[j,] <- x[sample.int(n,size = 1),]
  }
  for (i in 1:g){
    for (k1 in 1:d){
      u[k1,k1] <- sd(x[,k1])
    }
    SIG[((i-1)*d+1):(i*d),] <- u
  }
  p_prec <- rep(0,g)
  MU_prec <- matrix(0,nrow = g,ncol = d)
  SIG_prec <- matrix(0,nrow = g*d,ncol = d)
  # -------
  # --- Boucle de l'agorithme EM ---
  k <- 0
  while((norme(p_prec-p)>eps)&&(norme(MU_prec-MU)>eps)&&(norme(SIG_prec-SIG)>eps)){
    p_prec <- p
    MU_prec <- MU
    SIG_prec <- SIG
    h <- H(x,p,MU,SIG)
    SIG = matrix(0,nrow = g*d,ncol = d)
    for (j in 1:g){
      p[j] = mean(h[,j])
      MU[j,] = colSums(h[,j] * x) / sum(h[,j])
      SIG[((j-1)*d+1):(j*d),] = t((h[,j]*t((t(x)-MU[j,])))%*%t(t(x)-MU[j,])
      SIG[((j-1)*d+1):(j*d),] = SIG[((j-1)*d+1):(j*d),] / sum(h[,j])
    }
    k <- k+1
  }
  # -------
  # --- Calcul de l'inertie ---
  z <- matrix(0,nrow=n,ncol=g)
  h <- H(x,p,MU,SIG)
  for (i in 1:n){
    z[i,which.max(h[i,])] = 1
  }
  I <- Inertie_intra(x,z,MU,g,n)
  return(list(Poids=p,Moyenne=MU,Variance=SIG,Nb_iteration=k,Inertie=I))
}
```

L’algorithme CEM

```
CEM <- function(x,g,init_p){
  # --- nos parametres ---
  n <- length(x[,1])
  d <- length(x[1,])
  p <- rep(0,g)
  MU <- matrix(0,nrow = g,ncol = d)
  SIG <- matrix(0,nrow = g*d,ncol = d)
  z <- matrix(0,nrow=n,ncol=g)
  # -------
  # --- Initilisation ---
  p <- init_p
  u <- matrix(0,nrow = d,ncol = d)
  MU <- x[sample.int(n,size = g),]
  for (j in 1:g){
    for (k1 in 1:d){
      u[k1,k1] <- sd(x[,k1])
    }
    SIG[((j-1)*d+1):(j*d),] <- u
  }
  p_prec <- rep(0,g)
  MU_prec <- matrix(0,nrow = g,ncol = d)
  SIG_prec <- matrix(0,nrow = g*d,ncol = d)
  # -------
  # --- Boucle de l'agorithme CEM ---
  k <- 0
  while((norme(p_prec-p)>0)&&(norme(MU_prec-MU)>0)&&(norme(SIG_prec-SIG)>0)){
    p_prec <- p
    MU_prec <- MU
    SIG_prec <- SIG
    h <- H(x,p,MU,SIG)
    z <- matrix(0,nrow=n,ncol=g)
    for (i in 1:n){
      z[i,which.max(h[i,])] = 1
    }
    # Quand une classe ne contient aucun element
    if(min(colSums(z))==0){
      return(list(Inertie=Inf))
    }
    for (j in 1:g){
      p[j] = mean(z[,j])
      MU[j,] = colSums(x[which(z[,j]==1),,drop=FALSE]) / sum(z[,j])
      SIG[((j-1)*d+1):(j*d),] = (t(x[which(z[,j]==1),])-MU[j,])%*%t(t(x[which(z[,j]==1),])-
MU[j,])
      SIG[((j-1)*d+1):(j*d),] = SIG[((j-1)*d+1):(j*d),] / sum(z[,j])
    }
    k <- k+1
  }
  # -------
  # --- Calcul de l'inertie ---
  I <- Inertie_intra(x,z,MU,g,n)
  # -------
  return(list(Poids=p,Moyenne=MU,Variance=SIG,Classification=z,Nb_iteration=k,Inertie=I))
}
```

MULTI_EM et MULTI_CEM

```
MULTI_EM <- function(x,g,eps){
  # --- nos parametres ---
  res_EM_final <- list()
  I <- 0
  I_min <- 0
  init_p <- rep(0,g)
  s <- seq(1/g,1,1/g)
  s <- s/sum(s)
  # --- Premiere iteration avec poids identique pour chaque classe ---
  res_EM_final <- EM(x,g,eps,rep(1/g,g))
  I_min <- res_EM_final[[5]]
  # --- Boucle avec differentes initialisation pour p ---
  for(k in 1:g){
    for(j in 1:g){
      init_p[j] <- s[(j+k)%%g+1]
    }
    res_EM <- EM(x,g,eps,init_p)
    I <- res_EM[[5]]
    if(I<I_min){
      I_min <- I
      res_EM_final <- res_EM
    }
  }
  return(res_EM_final)
}
MULTI_CEM <- function(x,g){
  # --- nos parametres ---
  res_CEM_final <- list()
  I <- 0
  I_min <- 0
  init_p <- rep(0,g)
  s <- seq(1/g,1,1/g)
  s <- s/sum(s)
  # --- Premiere iteration avec poids identique pour chaque classe ---
  res_CEM_final <- CEM(x,g,rep(1/g,g))
  I_min  <- res_CEM_final$Inertie
  # --- Boucle avec differentes initialisation pour p ---
  for(k in 1:g){
    for(j in 1:g){
      init_p[j] <- s[(j+k)%%g+1]
    }
    res_CEM <- CEM(x,g,init_p)
    I <- res_CEM$Inertie
    if(I<I_min){
      I_min  <- I
      res_CEM_final <- res_CEM
    }
  }
  return(res_CEM_final)
}
```

Graphique EM et CEM

```
ellipse_dim2 <- function(x, g, THETA, prec){
  # --- nos parametres ---
  SIG <- THETA$Variance
  MU <- THETA$Moyenne
  n <- dim(x)[1]
  # -------
  # --- Boucle pour tracer une ellipse pour chaque groupe ---
  for (j in 1:g){
    v <- eigen(THETA$Variance[((j-1)*2+1):(j*2),])
    a <-  sqrt(-2*log(1-prec)*v$values[1])
    b <-  sqrt(-2*log(1-prec)*v$values[2])
    t <- seq(0,2*3.2,0.1)
    cor <- matrix(0,nrow = 2,ncol = 65)
    cor[1,] <- a*cos(t)
    cor[2,] <- b*sin(t)
    cor <- v$vectors %*% cor
    cor <- cor + MU[j,]
    lines(t(cor), col="red")
  }
  # -------
}
graph_EM <- function(x,g,THETA,prec){ # prec =0.95
  # --- Representation des points de notre echantillon ---
  plot(x, type = "p",pch = 4,main = "Simulation d'un melange de vecteurs Gaussiens",xlab = "X
_1",ylab = "X_2", col="blue")
  # -------
  # --- Representation de nos ellipses ---
  ellipse_dim2(x, g, THETA, prec)
  # -------
}
graph_CEM <- function(x,g,THETA,prec){
  # --- Representation des points de notre echantillon avec un code pour la representation de
 la partition ---
  plot(x[which(THETA$Classification[,1]==1),], type = "p",pch = 1,main = "Simulation d'un mel
ange de vecteurs Gaussiens",xlab = "X_1",ylab = "X_2", col="blue",xlim =
c(min(x[,1]),max(x[,1])),ylim = c(min(x[,2]),max(x[,2])))
  for (j in 2:g){
    points(x[which(THETA$Classification[,j]==1),],pch = j,col="blue")
  }
  # -------
  # --- Representation de nos ellipses ---
  ellipse_dim2(x, g, THETA, prec)
  # -------
}
```

Outils pour critere de selection du modele

```
log_vraisemblance <- function(donnees,THETA,g){
  # --- nos parametres ---
  ret <- 0
  n <- length(x[,1])
  # -------
  # --- Calcul de notre log vraisemblance ---
  for(i in 1:n){
    res1 <- 0
    for(j in 1:g){
      res1 <- res1 + THETA$Poids[j] * d_VG(donnees[i,],THETA$Moyenne[j,],THETA$Variance[((j-
1)*2+1):(j*2),])
    }
    ret <- ret + log(res1)
  }
  # -------
  return(ret)
}
Entropie <- function(n,g,z,h){
  ret = 0
  # --- Calcul de notre valeur d'entropie pour le critere ICL ---
  for(i in 1:n){
    ret <- ret - log(h[i,which.max(z[i,])])
  }
  # -------
  return(ret)
}
```

Critere AIC ou critere BIC

```
critere_AIC_BIC <- function(x,mod,nb_test,multi){
  # --- nos parametres ---
  n <- length(x[,1])
  d <- length(x[1,])
  valeur <- rep(0,nb_test)
  # -------
  # --- Selection de notre critere ---
  if(mod=="BIC"){
    penalite = 0.5*log(n)
  } else if (mod=="AIC"){
    penalite = 1
  } else {
    stop("la fonction ne reconnait pas ce critere")
  }
  # -------
  # --- Avec un seul groupe ---
  Poids <- 1
  Moyenne <- matrix(colMeans(x),nrow = 1,ncol = d)
  Variance <- var(x)
  THETA <- list(Poids=Poids,Moyenne=Moyenne,Variance=Variance)
  lg_prec <- log_vraisemblance(x,THETA,1) - (2+3)*1* penalite
  valeur[1] <- lg_prec
  nb_groupe_modele <- 1
  # -------
  # --- Avec plusieurs groupes ---
  for(g in 2:nb_test){
    if (multi==TRUE){
      THETA <- MULTI_EM(x,g,0.01)
    }else{
      init_p <- runif(g)
      init_p <- init_p/sum(init_p)
      THETA <- EM(x,g,0.01,init_p)
    }
    lg <- 0
    lg <- log_vraisemblance(x,THETA,g) - (2+3)*g * penalite
    valeur[g] <- lg
    if (lg>lg_prec){
      nb_groupe_modele <- g
      lg_prec <- lg
    }
  }
  # -------
  return(list(nb_groupe_modele=nb_groupe_modele,valeur_critere=valeur))
}
```

Critere ICL

```
critere_ICL <- function(x,nb_test){
  # --- nos parametres ---
  n <- length(x[,1])
  d <- length(x[1,])
  valeur <- vector(length = nb_test)
  # -------
  # --- Avec un seul groupe ---
  Poids <- 1
  Moyenne <- matrix(colMeans(x),nrow = 1,ncol = d)
  Variance <- var(x)
  Classification <- matrix(1,nrow = n,ncol =1)
  THETA <- list(Poids=Poids,Moyenne=Moyenne,Variance=Variance,Classification=Classification)
  lg_prec <- log_vraisemblance(x,THETA,1) - (2+3)*1*0.5*log(n)
  # Comme h = 1 pour tous element  => Entropie est nulle
  valeur[1] <- lg_prec
  lg <- 0
  nb_groupe_modele <- 1
  # -------
  # --- Avec plusieurs groupes ---
  for(g in 2:nb_test){
    THETA <- MULTI_CEM(x,g)
    lg <- log_vraisemblance(x,THETA,g) - (2+3)*g*0.5*log(n)
    h <- H(x,THETA$Poids,THETA$Moyenne,THETA$Variance)
    lg <- lg - Entropie(n,g,THETA$Classification,h)
    if (length(lg)){
      valeur[g] <- lg
      if(lg>lg_prec){
        nb_groupe_modele <- g
        lg_prec = lg
      }
    }else{
      valeur[g]<- -Inf
    }
  }
  # -------
  return(list(nb_groupe_modele=nb_groupe_modele,valeur_critere=valeur))
}
```

Graphique colore

```
appartenance <- function(x,g,THETA){
  # --- nos parametres ---
  MU <- THETA$Moyenne
  SIG <- THETA$Variance
  PI <- THETA$Poids
  h <- rep(0,g)
  df <- 0
  # -------
  # --- Calcul des probabilites d'appartenance d'un point a chaque groupe ---
  for (j in 1:g){
    df <- df + ( PI[j] * d_VG(x,MU[j,],SIG[((j-1)*d+1):(j*d),]) )
  }
  for (j in 1:g){
    h[j] <- (PI[j]*d_VG(x,MU[j,],SIG[((j-1)*d+1):(j*d),])) / df
  }
  # -------
  # --- Selection de la valeur maximal ---
  return(which.max(h))
}
graph_colore <- function(g,THETA,xlim,ylim,N){
  # --- Calcul de l'appartenance de chaque poids du plan discretise ---
  A <- matrix(0,nrow = N+1,ncol = N+1)
  for(i in 0:N+1){
    for (j in 0:N+1){
      A[i,(N+1)-j] = appartenance(c(ylim[1]+(i-1)*(ylim[2]-ylim[1])/N,xlim[2]+(j-1)*(xlim[1]-
xlim[2])/N),g,THETA)
    }
  }
  # -------
  # --- Limite de notre plan et pas de discretisation ---
  dx <- (xlim[2]-xlim[1])/N
  dy <- (ylim[2]-ylim[1])/N
  X <- seq(xlim[1],xlim[2],dx)
  Y <- seq(ylim[1],ylim[2],dy)
  # -------
  # --- Representation du graphique colore ---
  filled.contour(X,Y,A)
  # -------
}
```