Présentation Du Package MelangeGaussien, cours MBL, semestre 1 MALIA, 2024, 1–7
doi: objet numérique sans identifiant (préparation pédagogique)
Date de soumission : décembre 2024
Supervisé par : Dr Julien JACQUES

Package R Pour Clustering de Mélanges Gaussiens: Fondements Mathématiques et Applications

Mehdi Mansour[1,*]

[1]Departement Informatique, ICOM, 5 avenue Pierre Mendès France, 69500 Bron, France

[*]Corresponding author. mehdi.mansour@univ-lyon2.fr

Abstract

Ce rapport présente un nouveau package R implémentant une approche de clustering par mélanges gaussiens avec estimation par l'algorithme EM (Expectation-Maximization). Le package propose une implémentation flexible supportant différentes structures de covariance et une sélection automatique du nombre de clusters via les critères AIC/BIC. L'utilisation de techniques numériques robustes et une architecture orientée objet en font un outil fiable pour l'analyse de données quantitatives multivariées.

Key words: Model Based Learning, Clustering, Gaussien Mixture, Expectation Maximisation (EM), MNIST

1 Introduction

Le clustering par mélanges gaussiens s'appuie sur la théorie des modèles de mélange probabilistes pour aborder le problème de la classification non supervisée. Cette approche repose sur l'hypothèse que les données observées sont générées par un mélange de distributions gaussiennes multivariées, chacune caractérisant une sous-population homogène. La formalisation probabiliste qui en découle permet non seulement d'estimer la structure de partitionnement sous-jacente, mais aussi de quantifier l'incertitude associée à cette estimation via les probabilités a posteriori d'appartenance aux clusters.

Dans ce cadre théorique, l'estimation des paramètres du modèle est réalisée par maximum de vraisemblance via l'algorithme EM (Expectation-Maximization), garantissant une convergence monotone vers un maximum local de la fonction de vraisemblance

Dans ce contexte, nous présentons un nouveau package R implémentant l'estimation de mélanges gaussiens par l'algorithme EM. Notre implémentation se distingue par trois aspects principaux : une architecture orientée objet moderne, une attention particulière à la stabilité numérique, et une flexibilité dans la spécification des modèles.

L'architecture orientée objet, basée sur le système de classes S4 de R, répond aux standards actuels du machine learning en offrant une interface familière (`fit`, `predict`) tout en permettant une gestion fine de l'état d'apprentissage. Cette approche facilite notamment la reprise d'entraînement, le diagnostic de convergence, et l'extensibilité du code. Les états d'apprentissage (paramètres, historiques de convergence) sont encapsulés de manière cohérente, permettant une interaction transparente avec le modèle à toute étape de son cycle de vie.

La stabilité numérique, critique pour l'estimation robuste des mélanges gaussiens, est assurée par plusieurs mécanismes. Les calculs de vraisemblance sont effectués en log-échelle via une implémentation optimisée de `logsumexp`. L'inversion des matrices de covariance utilise la décomposition SVD, offrant une meilleure gestion des matrices mal conditionnées. Une régularisation paramétrable des matrices de covariance prévient les situations dégénérées.

Le package propose quatre structures de covariance (`full`, `tied`, `diagonal`, `spherical`), permettant un compromis flexible entre expressivité du modèle et parcimonie. La sélection du nombre optimal de composantes est facilitée par l'implémentation des critères AIC et BIC, accompagnée d'outils de visualisation pour l'analyse des résultats.

Cette implémentation répond au cahier des charges du projet avec plusieurs parties d'approfondissement personnel et d'explorartion. Elle a été validée sur plusieurs jeu de données minimalistes et sur MNIST.

2 Fondement Théorique

2.1 Modèle Probabiliste du Mélange

Le modèle de mélange gaussien suppose que les données $X = (x_1, \ldots, x_n)$ sont générées selon la densité :

$$f(x) = \sum_{k=1}^{K} \pi_k \phi(x|\mu_k, \Sigma_k) \tag{1}$$

où :

- K est le nombre de composantes
- π_k sont les proportions du mélange ($\sum \pi_k = 1$)
- $\phi(x|\mu_k, \Sigma_k)$ est la densité gaussienne de moyenne μ_k et matrice de covariance Σ_k

2.2 Structures de Covariance

La matrice de covariance Σ_k dans un mélange gaussien détermine la forme géométrique, l'orientation et le volume de chaque cluster. Le choix de sa structure impacte directement le nombre de paramètres à estimer et, par conséquent, la complexité du modèle. Nous présentons ici les quatre structures implémentées, ordonnées de la plus libre à la plus contrainte.

2.2.1 Structure Complète (Full)

Dans sa forme la plus générale, chaque composante k possède sa propre matrice de covariance Σ_k sans contrainte particulière :

$$\Sigma_k = \begin{pmatrix} \sigma_{k11} & \sigma_{k12} & \cdots & \sigma_{k1p} \\ \sigma_{k21} & \sigma_{k22} & \cdots & \sigma_{k2p} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{kp1} & \sigma_{kp2} & \cdots & \sigma_{kpp} \end{pmatrix} \tag{2}$$

Caractéristiques :

- Nombre de paramètres : $K \times \frac{p(p+1)}{2}$ (matrices symétriques)
- Permet des clusters de formes elliptiques arbitraires
- Estimation ML : $\hat{\Sigma}_k = \frac{\sum_{i=1}^{n} t_{ik}(x_i-\mu_k)(x_i-\mu_k)^T}{\sum_{i=1}^{n} t_{ik}}$

2.2.2 Structure Homoscédastique (Tied)

Cette structure impose l'homoscédasticité : tous les clusters partagent la même matrice de covariance Σ :

$$\Sigma_k = \Sigma \quad \forall k \in \{1, \ldots, K\} \tag{3}$$

Caractéristiques :

- Nombre de paramètres : $\frac{p(p+1)}{2}$ (une seule matrice)
- Clusters de même forme et orientation, mais positions différentes
- Estimation ML : $\hat{\Sigma} = \frac{\sum_{k=1}^{K}\sum_{i=1}^{n} t_{ik}(x_i-\mu_k)(x_i-\mu_k)^T}{\sum_{k=1}^{K}\sum_{i=1}^{n} t_{ik}}$

2.2.3 Structure Diagonale (Diagonal)

Dans ce modèle, les matrices sont contraintes à être diagonales, supposant l'indépendance conditionnelle des variables, soit l'orthogonalité des variables dans un même cluster :

$$\Sigma_k = \begin{pmatrix} \sigma_{k1}^2 & 0 & \cdots & 0 \\ 0 & \sigma_{k2}^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_{kp}^2 \end{pmatrix} \tag{4}$$

Caractéristiques :

- Nombre de paramètres : $K \times p$
- Axes principaux alignés avec les axes des coordonnées
- Estimation ML : $\hat{\sigma}_{kj}^2 = \frac{\sum_{i=1}^{n} t_{ik}(x_{ij}-\mu_{kj})^2}{\sum_{i=1}^{n} t_{ik}}$

2.2.4 Structure Sphérique (Spherical)

La forme la plus contrainte suppose des clusters sphériques, où chaque matrice est proportionnelle à l'identité :

$$\Sigma_k = \sigma_k^2 I_p = \begin{pmatrix} \sigma_k^2 & 0 & \cdots & 0 \\ 0 & \sigma_k^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_k^2 \end{pmatrix} \tag{5}$$

Caractéristiques :

- Nombre de paramètres : K (un scalaire par cluster)
- Clusters parfaitement sphériques
- Estimation ML : $\hat{\sigma}_k^2 = \frac{\sum_{i=1}^{n} t_{ik}\|x_i-\mu_k\|^2}{p\sum_{i=1}^{n} t_{ik}}$

2.2.5 Implications sur la Classification

Le choix de la structure de covariance influence directement :

1. **Frontières de décision** :
   - Full : Quadratiques générales
   - Tied : Linéaires
   - Diagonal/Spherical : Quadratiques avec axes alignés
2. **Critères de sélection** : Le BIC pour chaque structure est :

$$\text{BIC} = -2\ell(\hat{\theta}) + \nu_m \log(n) \tag{6}$$

où ν_m est le nombre de paramètres selon la structure m :

$$\nu_{\text{full}} = K(1 + p + \frac{p(p+1)}{2}) - 1$$

$$\nu_{\text{tied}} = K(1 + p) + \frac{p(p+1)}{2} - 1$$

$$\nu_{\text{diag}} = K(1 + 2p) - 1$$

$$\nu_{\text{spher}} = K(1 + p + 1) - 1$$

3. **Stabilité numérique** :
   - Full : Nécessite régularisation forte
   - Tied : Plus stable, partage d'information
   - Diagonal/Spherical : Naturellement plus stables

3 Architecture et Implémentation

3.1 Architecture Orientée Objet

Notre implémentation du modèle de mélange gaussien repose sur une classe S4 `MelangeGaussien` qui encapsule de manière cohérente l'état du modèle et ses comportements. Cette architecture permet une meilleure traduction, du moins sur le plan de la lisibilité, de la théorie statistique en une implémentation numérique.

3.2 Structure des Attributs

Les attributs de la classe se répartissent en quatre catégories fonctionnelles :

3.2.1 Hyperparamètres du Modèle

Ces attributs définissent la configuration initiale du modèle :

- `nb_clusters` : *numeric* $\in \mathbb{N}^*$ (nombre de composantes)
- `type_covariance` : *character* $\in$ {full, tied, diagonal, spherical}
- `nb_initialisations` : *numeric* $\in \mathbb{N}^*$ (nombre de départs aléatoires)
- `epsilon_convergence` : $\epsilon > 0$ (seuil de convergence)
- `regul_inversibilite` : $\lambda > 0$ (paramètre de régularisation)

3.2.2 Paramètres Estimés

Ces attributs stockent les paramètres $\theta = (\pi_k, \mu_k, \Sigma_k)_{k=1}^{K}$ estimés par EM :

- `proportions_` : $\pi \in \Delta^{K-1}$ (simplexe de dimension K-1)
- `moyennes_` : $\mu \in \mathbb{R}^{K \times p}$

- `covariances_` : Σ (structure selon type_covariance)
- `prob_z_sachant_x_` : $t_{ik} = P(Z_i = k|X_i = x_i, \theta)$

3.2.3 Métriques de Convergence

Ces attributs permettent le suivi de l'estimation :

- `nb_iterations_realisees_` : $\phi \in \mathbb{N}$ (itérations effectuées)
- `log_vraisemblance_estimee_` :

$$\ell(\theta) = \frac{1}{n}\sum_{i=1}^{n} \log\left(\sum_{k=1}^{K} \pi_k \phi(x_i|\mu_k, \Sigma_k)\right) \tag{7}$$

- `etat_convergence_` : indicateur de convergence basé sur

$$\delta^{(\phi)} = \frac{|\ell(\theta^{(\phi+1)}) - \ell(\theta^{(\phi)})|}{|\ell(\theta^{(\phi)})|} \tag{8}$$

3.2.4 Historiques

Ces attributs permettent le diagnostic et la visualisation :

- `evolution_logvraisemblance_estimee` : $(\ell(\theta^{(\phi)}))_{\phi=1}^{\Phi}$
- `evolution_relative_lv` : $(\delta^{(\phi)})_{\phi=1}^{\Phi-1}$

3.3 Méthodes Principales

3.3.1 Méthodes d'Estimation

- **Initialisation** (`initialize()`) :

$$\theta^{(0)} \sim random(\theta) \text{ ou } \theta^{(0)} = \text{kmeans}(X, K) \tag{9}$$

- **Ajustement** (`fit(X)`) :

  1. Pour chaque initialisation $r = 1, \ldots, R$:
     - Initialiser $\theta_r^{(0)}$
     - Répéter jusqu'à $\delta^{(\phi)} < \epsilon$:
       - E-step : $t_{ik} = P(Z_i = k|x_i, \theta^{(\phi)})$
       - M-step : $\theta^{(\phi+1)} = \text{argmax}_\theta Q(\theta|\theta^{(\phi)})$
  2. Retourner le meilleur θ selon $\ell(\theta)$

- **Prédiction** (`predict(X)`) :

$$k^*(x) = \text{argmax}_k P(Z = k|X = x, \theta) =_k \pi_k \phi(x|\mu_k, \Sigma_k) \tag{10}$$

3.3.2 Méthodes de Diagnostic

- **Score** (`score(X)`) : calcul de la log-vraisemblance
- **Critères d'information** (`aic(X)`, `bic(X)`) :

$$\text{AIC} = -2\ell(\theta) + 2\nu \tag{11}$$

$$\text{BIC} = -2\ell(\theta) + \nu \log(n) \tag{12}$$

- **Visualisations** (`plot_convergence()`, `plot_clusters()`)

3.3.3 Méthodes Utilitaires

- **Calcul des densités** (`estimer_log_densite()`) :

$$\log \phi(x|\mu, \Sigma) = -\frac{p}{2}\log(2\pi) - \frac{1}{2}\log|\Sigma| - \frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu) \tag{13}$$

- **Stabilité numérique** (`logsumexp()`) :

$$\log\sum_k \exp(a_k) = m + \log\sum_k \exp(a_k - m) \text{ où } m = \max_k a_k \tag{14}$$

3.4 Stabilité Numérique

La stabilité numérique a été cruciale dans notre implémentation qui se sert de ses propres algorithmes et ne fait pas appel à des librairies optimisées. Particulièrement lors du calcul des responsabilités t_{ik} dans l'étape E. Plusieurs stratégies sont mises en œuvre pour garantir cette stabilité :

3.4.1 Calculs en Log-échelle

Le calcul direct des responsabilités :

$$t_{ik} = \frac{\pi_k \phi(x_i|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \phi(x_i|\mu_j, \Sigma_j)} \tag{15}$$

présente deux risques numériques majeurs :

1. **Underflow** : La densité gaussienne peut produire des valeurs extrêmement petites :

$$\phi(x|\mu, \Sigma) = \frac{1}{(2\pi)^{p/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right) \tag{16}$$

2. **Overflow** : La somme au dénominateur peut dépasser les limites de représentation numérique.

Solution : Passage en log-échelle

$$\log(t_{ik}) = \log(\pi_k) + \log(\phi(x_i|\mu_k, \Sigma_k)) - \log\left(\sum_{j=1}^{K} \pi_j \phi(x_i|\mu_j, \Sigma_j)\right) \tag{17}$$

$$= \log(\pi_k) + \ell_{ik} - \text{logsumexp}(\{\log(\pi_j) + \ell_{ij}\}_{j=1}^{K}) \tag{18}$$

où :

$$\ell_{ik} = -\frac{p}{2}\log(2\pi) - \frac{1}{2}\log|\Sigma_k| - \frac{1}{2}(x_i - \mu_k)^T\Sigma_k^{-1}(x_i - \mu_k) \tag{19}$$

3.4.2 Optimisation du LogSumExp

Le calcul de $\log\sum_j \exp(a_j)$ est stabilisé par la technique du "log-sum-exp trick" :

$$\log\sum_{j=1}^{K} \exp(a_j) = m + \log\sum_{j=1}^{K} \exp(a_j - m) \tag{20}$$

où $m = \max_j a_j$. Implémentation optimisée :

```
logsumexp <- function(x) {
    # Étape 1 : Extraction du maximum
    x_max <- max(x)
    # Étape 2 : Soustraction et exp stable
    # Étape 3 : Addition et log final
    x_max + log(sum(exp(x - x_max)))
}
```

Avantages :

- Évite les overflow en soustrayant le maximum
- Garantit que $\exp(a_j - m) \leq 1$
- Préserve la précision numérique

3.4.3 Décomposition SVD pour l'Inversion

L'inversion des matrices de covariance utilise la SVD:

$$\Sigma = UDU^T \implies \Sigma^{-1} = UD^{-1}U^T \tag{21}$$

Avantages :

- Plus stable que l'inversion directe
- Gestion des matrices mal conditionnées via pseudo-inverse
- Détection des singularités via valeurs propres

Implémentation :

```
svd_cov <- svd(cov_k)
inv_cov_k <- svd_cov$u %*% diag(1/svd_cov$d) %*% t(svd_cov$u)
```

3.4.4 Régularisation

Pour garantir l'inversibilité, une régularisation est ajoutée :

$$\Sigma_k \leftarrow \Sigma_k + \lambda I_p \tag{22}$$

où λ est le paramètre `regul_inversibilite`. Cette régularisation:

- Assure que Σ_k est définie positive
- Améliore le conditionnement
- Limite l'impact des situations dégénérées

3.5 Initialisations Multiples

L'utilisation de R initialisations permet :

- D'éviter les optima locaux sous-optimaux
- De réduire la dépendance aux conditions initiales
- D'explorer l'espace des paramètres

Le meilleur modèle est sélectionné selon :

$$\theta^* = \text{argmax}_{r=1,\ldots,R} \ell(\theta_r) \tag{23}$$

4 Algorithme EM utilisé

4.1 Fondement Théorique

L'algorithme EM (Expectation-Maximization) est une méthode itérative pour l'estimation du maximum de vraisemblance en présence de données manquantes. Le modèle génératif s'écrit:

Pour chaque observation $i = 1, \ldots, n$:

$$Z_i \sim \mathcal{M}(\pi_1, \ldots, \pi_K) \tag{24}$$

$$X_i | Z_i = k \sim \mathcal{N}(\mu_k, \Sigma_k) \tag{25}$$

La log-vraisemblance complète s'écrit alors:

$$\log L_c(\theta; X, Z) = \sum_{i=1}^{n} \sum_{k=1}^{K} Ind(Z_i = k)[\log \pi_k + \log \phi(x_i | \mu_k, \Sigma_k)] \tag{26}$$

4.2 Principe de l'Algorithme

L'algorithme maximise itérativement l'espérance de la log-vraisemblance complète conditionnellement aux observations:

$$Q(\theta | \theta^{(\phi)}) = \mathbb{E}_{Z|X,\theta^{(\phi)}}[\log L_c(\theta; X, Z)] \tag{27}$$

4.3 Implémentation

4.3.1 Étape E

Calcul des probabilités a posteriori via une implémentation stable en log-échelle:

$$t_{ik} = P(Z_i = k | x_i, \theta) = \frac{\pi_k \phi(x_i | \mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \phi(x_i | \mu_j, \Sigma_j)} \tag{28}$$

4.3.2 Étape M

Mise à jour des paramètres par maximisation de Q:

1. Proportions : $\pi_k^{new} = \frac{1}{n} \sum_{i=1}^{n} t_{ik}$
2. Moyennes : $\mu_k^{new} = \frac{\sum_{i=1}^{n} t_{ik} x_i}{\sum_{i=1}^{n} t_{ik}}$
3. Covariances selon la structure choisie:

- Full : $\Sigma_k^{new} = \frac{\sum_{i=1}^{n} t_{ik}(x_i - \mu_k^{new})(x_i - \mu_k^{new})^T}{\sum_{i=1}^{n} t_{ik}}$
- Tied : $\Sigma^{new} = \frac{\sum_{k=1}^{K}\sum_{i=1}^{n} t_{ik}(x_i - \mu_k^{new})(x_i - \mu_k^{new})^T}{\sum_{k=1}^{K}\sum_{i=1}^{n} t_{ik}}$
- Diagonal : $\sigma_{kj}^{2\ new} = \frac{\sum_{i=1}^{n} t_{ik}(x_{ij} - \mu_{kj}^{new})^2}{\sum_{i=1}^{n} t_{ik}}$
- Spherical : $\sigma_k^{2\ new} = \frac{\sum_{i=1}^{n} t_{ik} \| x_i - \mu_k^{new} \|^2}{p \sum_{i=1}^{n} t_{ik}}$

4.4 Convergence

La convergence dans notre implémentation est évaluée via:

$$\delta^{(\phi)} = \frac{|\ell(\theta^{(\phi+1)}) - \ell(\theta^{(\phi)})|}{|\ell(\theta^{(\phi)})|} \tag{29}$$

L'algorithme converge monotonement vers un maximum local de la log-vraisemblance, justifiant l'utilisation de départs multiples.

5 Application Expérimentale

5.1 Données Synthétiques

Nous avons d'abord testé notre implémentation sur un jeu de données synthétique en 2D composé de trois clusters gaussiens:

- Cluster 1: 100 observations ($\mu = 2$, $\sigma = 0.5$)
- Cluster 2: 100 observations ($\mu = 5$, $\sigma = 0.7$)
- Cluster 3: 200 observations ($\mu = 8$, $\sigma = 1.0$)

5.2 Résultats

Les quatre modèles implémentés (Full, Tied, Diagonal, Spherical) ont été évalués selon plusieurs critères résumés dans le tableau suivant:

Table 1. Comparaison des performances des différents modèles

Modèle	LogLik	AIC	BIC	Paramètres	Itérations
Full	-3.273	2652.062	2719.917	17	35
Tied	-3.412	2751.598	2795.504	11	18
Diagonal	-3.273	2646.191	2702.071	14	15
Spherical	-3.281	2646.590	2690.496	11	14

5.3 Discussion

L'analyse des résultats révèle plusieurs points intéressants :

1. Le modèle sphérique obtient le meilleur BIC (2690.496), tandis que le modèle diagonal présente le meilleur AIC (2646.191), suggérant un bon compromis entre ajustement et parcimonie.
2. Le modèle "Tied" montre des performances significativement inférieures (BIC=2795.504), indiquant que l'hypothèse d'homoscédasticité n'est pas adaptée à ces données.
3. Les temps de convergence varient significativement : le modèle Full nécessite 35 itérations contre 14-15 pour les modèles plus contraints.

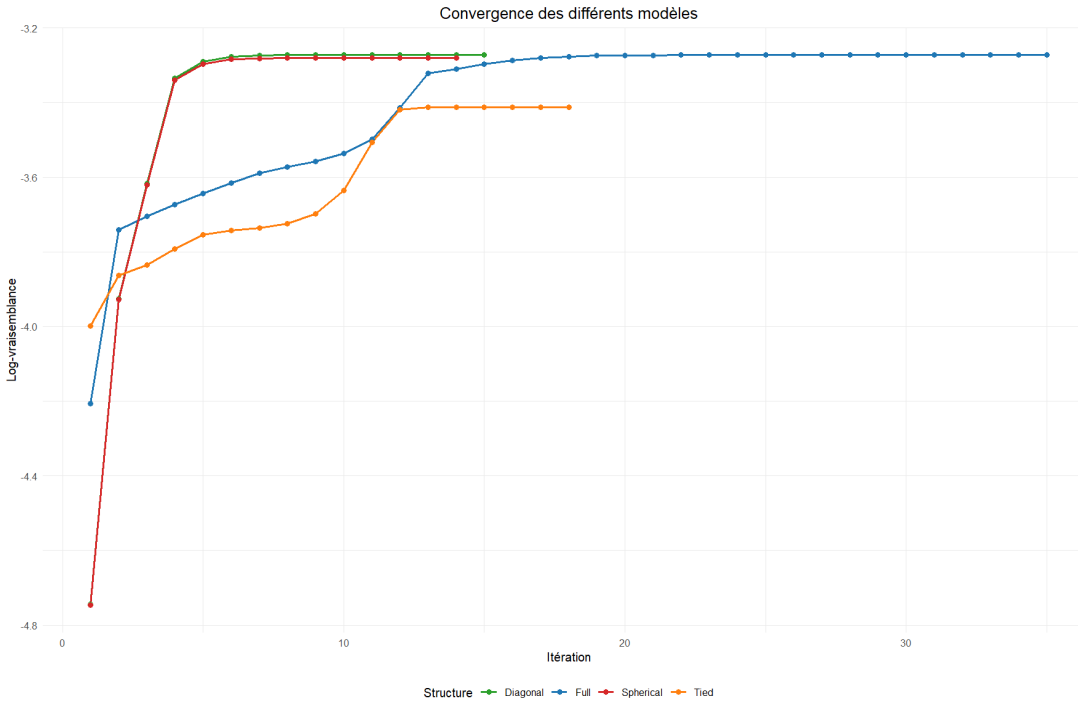

Fig. 1: Convergence de la log-vraisemblance pour les différents modèles

4. Tous les modèles parviennent à une classification quasi-parfaite des données, avec des matrices de confusion très proches du réel.

Full			
	1	2	3
1	100	0	0
2	0	1	199
3	0	99	1

Tied			
	1	2	3
1	0	100	6
2	100	0	0
3	0	0	194

Diagonal			
	1	2	3
1	100	0	0
2	0	1	199
3	0	99	1

Spherical			
	1	2	3
1	100	0	0
2	0	0	197
3	0	100	3

Fig. 2: Matrices de confusion pour chaque structure de covariance

5. Proportions estimées :

   - Full : (0.250, 0.498, 0.252)
   - Tied : (0.267, 0.250, 0.483)
   - Diagonal : (0.250, 0.498, 0.252)
   - Spherical : (0.250, 0.496, 0.254)

   Les proportions réelles étant (0.25, 0.25, 0.50), nous observons une excellente estimation pour tous les modèles sauf "tied".

6 Application à MNIST

6.1 Protocole Expérimental

Nous avons évalué notre implémentation sur le train set MNIST, référence standard en apprentissage automatique comprenant 60 000 images d'entraînement de chiffres manuscrits en 28×28 pixels.

6.1.1 Configuration des Modèles

L'exploration de l'impact du nombre de clusters sur le BIC s'est faite selon les paramètres suivants :

- Structure de covariance : matrices libres (type "full")
- Initialisation : aléatoire
- Nombre d'initialisations : 10
- Critère de convergence : $\epsilon = 10^{-4}$
- Nombre de clusters testés : $K \in \{4, 7, 10, 13, 16\}$

6.2 Résultats

6.2.1 Complexité et BIC

Le Tableau 2 résume la complexité paramétrique de chaque modèle testé, le nombre d'itérations qu'il a fallu pour qu'il converge et le BIC obtenu après convergence.

Table 2. Comparaison des performances des modèles

K	Paramètres	Itérations	BIC
4	1 234 019	42	−259 439 843
7	2 159 534	27	−259 608 270
10	3 085 049	23	−253 468 781
13	4 010 564	28	−244 470 889
16	4 936 079	19	−232 087 871

L'analyse de ce tableau pour les différents modèles révèle plusieurs aspects:

La complexité paramétrique du modèle augmente de manière Linéaire avec le nombre de clusters, évoluant de 1.2M paramètres pour K=4 jusqu'à 4.9M paramètres pour K=16. Cette augmentation importante est principalement due à l'utilisation de matrices de covariance complètes en dimension 784. Cette complexité était prévisible par l'aspect théorique présenté au début de l'article.

Le critère BIC, qui est à minimiser avec notre implémentation, augmente avec l'augmentation du nombre de clusters, passant de -259M pour K=4 à -232M pour K=16. Cette progression était également prévisible par la nature même du critère du BIC qui inclut une pénalisation de la complexité du modèle.

Ces résultats ne laissent pas la possiblité de faire un choix pertinent avec ce critère, qui nous orienterait dans cette configuration vers un nombre de clusters de 7.

6.2.2 Visualisation des Prototypes

Les images des prototype par nombre de composantes est disponible dans la section appendice.

Une analyse visuelle des prototypes (moyennes des clusters) pour différentes valeurs de K révèle l'évolution de la capacité de représentation du modèle :

Pour K=4, les prototypes montrent une représentation très grossière des chiffres, où chaque cluster capture des caractéristiques communes à plusieurs chiffres. Cette configuration minimale ne permet pas de distinguer efficacement les dix classes de chiffres.

Avec K=7, on observe l'émergence de motifs plus spécifiques. Les prototypes commencent à capturer des caractéristiques distinctives des chiffres manuscrits, comme les boucles des "6" et "9", ou la verticalité de certains traits. Cependant, le nombre de clusters reste insuffisant pour représenter l'ensemble des variations des dix chiffres.

Pour K=10, correspondant au nombre réel de classes, les prototypes montrent une meilleure spécialisation. Chaque prototype tend à correspondre à un chiffre spécifique, bien que certaines ambiguïtés persistent, notamment entre les chiffres visuellement proches comme "5" et "3", ou "7" et "9". On constate la très bonne reconnaissance du chiffre "6" avec même une sous classification en variantes stylistiques plus ou moins inclinés et avec une boucle plus ou moins grande.

L'augmentation à K=13 permet l'émergence du "5" et de variantes du "2".

Enfin, avec K=16, il y a une granularité plus fine et l'apparition du "0". L'augmentation du nombre de classe a peut être permis de diminuer le mélange entre chiffres mal reconnus et a peut être améliore ainsi la finesse des images. Cependant des chiffres comme le "1" et "7" restent encore éparpillés dans le reste.

6.2.3 Proportions

Les proportions estimées pour chaque configuration sont présentées dans le Tableau 3.

L'analyse de ces proportions révèle une forte asymétrie dans la distribution des clusters. Pour toutes les valeurs de K, on observe:

- Une concentration forte dans quelques clusters majeurs
- L'apparition systématique de clusters très minoritaires

Table 3. Proportions estimées pour chaque cluster selon K

K=4	{0.227, 0.051, 0.626, 0.096}
K=7	{0.339, 0.314, 0.003, 0.003, 0.067, 0.045, 0.228}
K=10	{0.239, 0.337, 0.002, 0.003, 0.042, 0.005, 0.237, 0.004, 0.0005, 0.130}
K=13	{0.237, 0.326, 0.002, 0.003, 0.039, 0.005, 0.240, 0.004, 0.0005, 0.136, 0.001, 0.0003, 0.007}
K=16	{0.006, 0.235, 0.001, 0.002, 0.193, 0.005, 0.320, 0.003, 0.0003, 0.189, 0.001, 0.0003, 0.005, 0.0001, 0.040, 0.0002}

- Une structure hiérarchique marquée avec une grande amplitude dans les proportions (de 30% à 0.01%)

Cette distribution déséquilibrée suggère que le modèle capture certains patterns dominants dans les données MNIST, tout en isolant des variantes plus rares dans des clusters minoritaires. Cette observation, combinée à l'analyse visuelle des prototypes, indique une tendance du modèle à sur-segmenter certains chiffres tout en ayant du mal à distinguer d'autres.

7 Conclusion

Ce projet a permis de développer une implémentation complète de l'algorithme EM pour l'estimation de mélanges gaussiens. Bien que fonctionnelle et validée sur des données synthétiques, cette implémentation a révélé plusieurs limitations significatives lors de son application à des données réelles complexes comme MNIST.

L'implémentation "from scratch" de tous les algorithmes, bien qu'instructive, entraîne des temps de calcul importants, particulièrement pour les matrices de grande dimension. L'absence d'optimisations natives comme celles présentes dans les bibliothèques spécialisées (BLAS, LAPACK) se fait particulièrement sentir lors des opérations matricielles intensives.

L'application sur MNIST a mis en évidence également les limites du modèle de mélange gaussien pour des données complexes:

- La dimensionnalité élevée (784) conduit à un nombre très important de paramètres à estimer
- L'hypothèse gaussienne s'avère trop restrictive pour modéliser efficacement la variabilité des chiffres manuscrits
- Les critères de sélection classiques (BIC) ne fournissent pas d'indications pertinentes sur le nombre optimal de clusters
- La visualisation des prototypes révèle une difficulté à distinguer certains chiffres (comme 1 et 7)

Plusieurs améliorations pourraient être envisagées :

- L'intégration de bibliothèques optimisées pour les calculs matriciels
- L'implémentation d'une version parallélisée pour les initialisations multiples
- L'ajout d'une étape de réduction de dimensionnalité préalable
- L'exploration de structures de covariance plus parcimonieuses pour la grande dimension

Malgré ces limitations, ce package constitue une base fonctionnelle pour l'analyse de données par modèles de mélange, particulièrement adaptée aux jeux de données de taille modérée.

8 APPENDIX

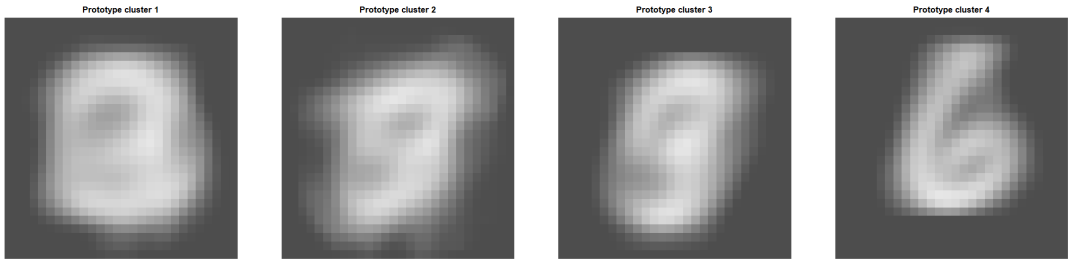

Fig. 3: Prototypes obtenus pour K = 4 clusters

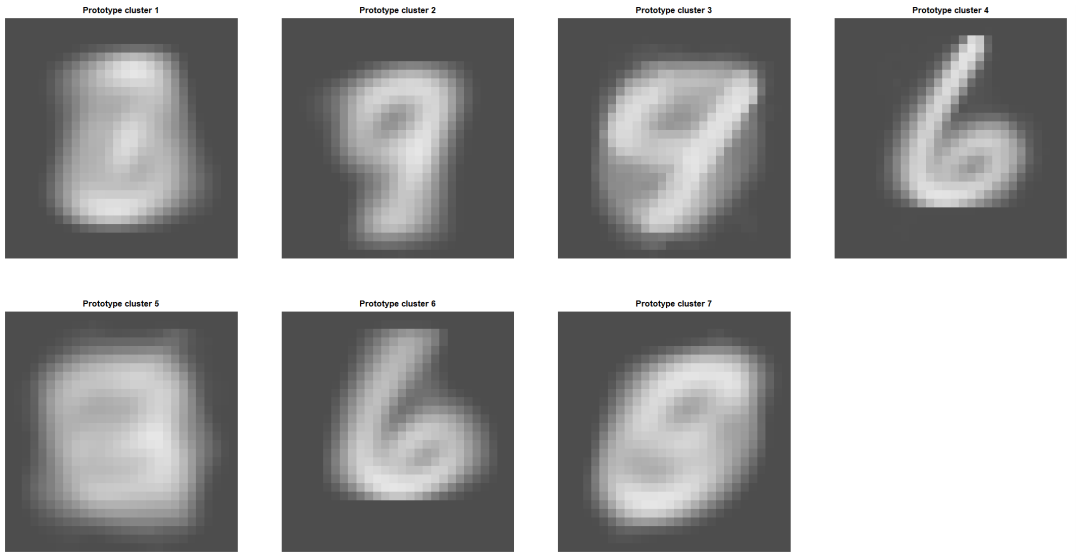

Fig. 4: Prototypes obtenus pour K = 7 clusters

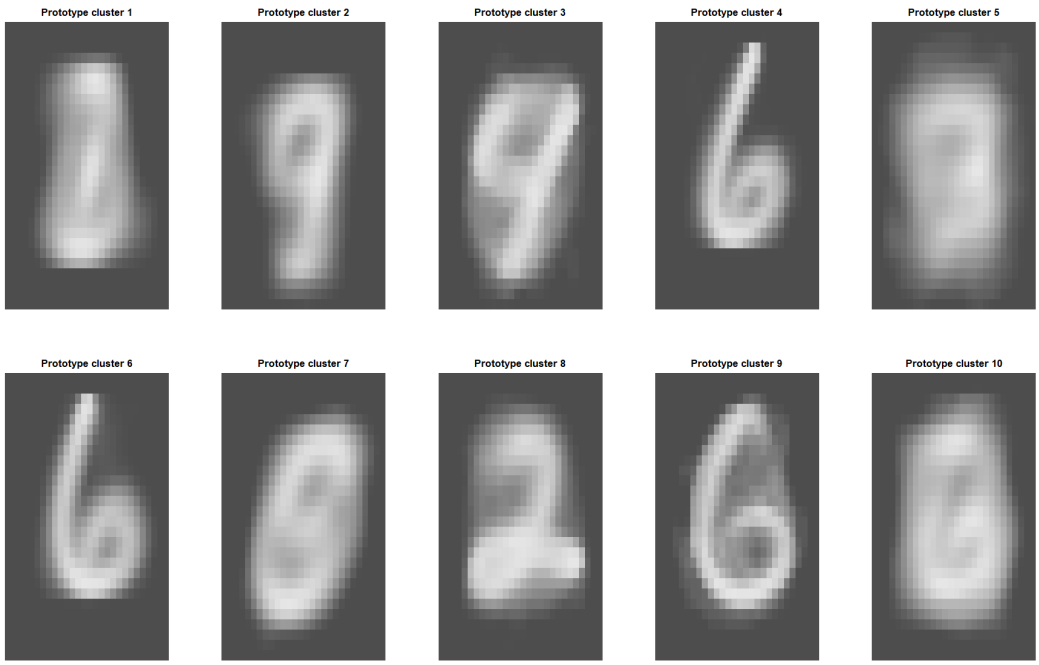

Fig. 5: Prototypes obtenus pour K = 10 clusters

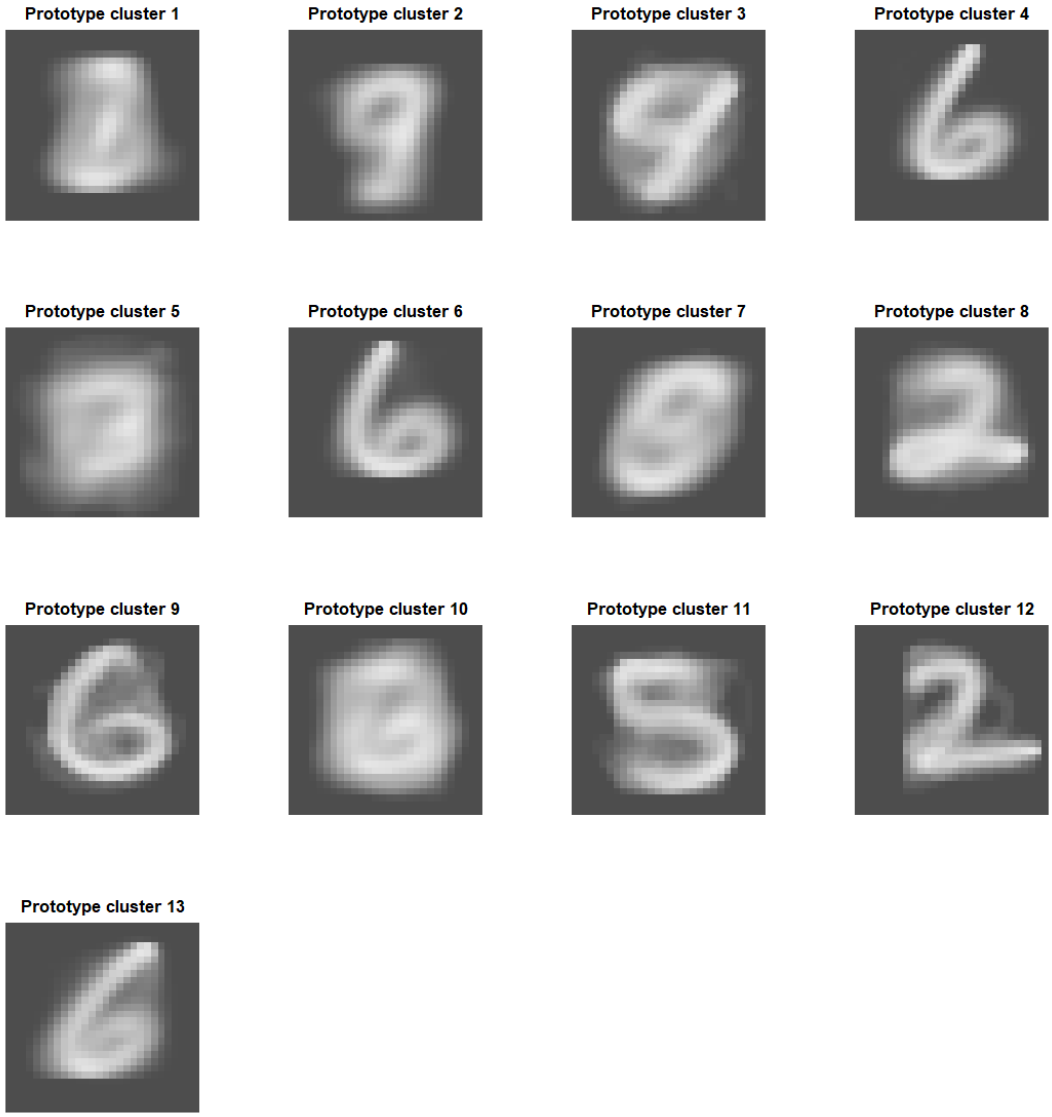

Fig. 6: Prototypes obtenus pour K = 13 clusters

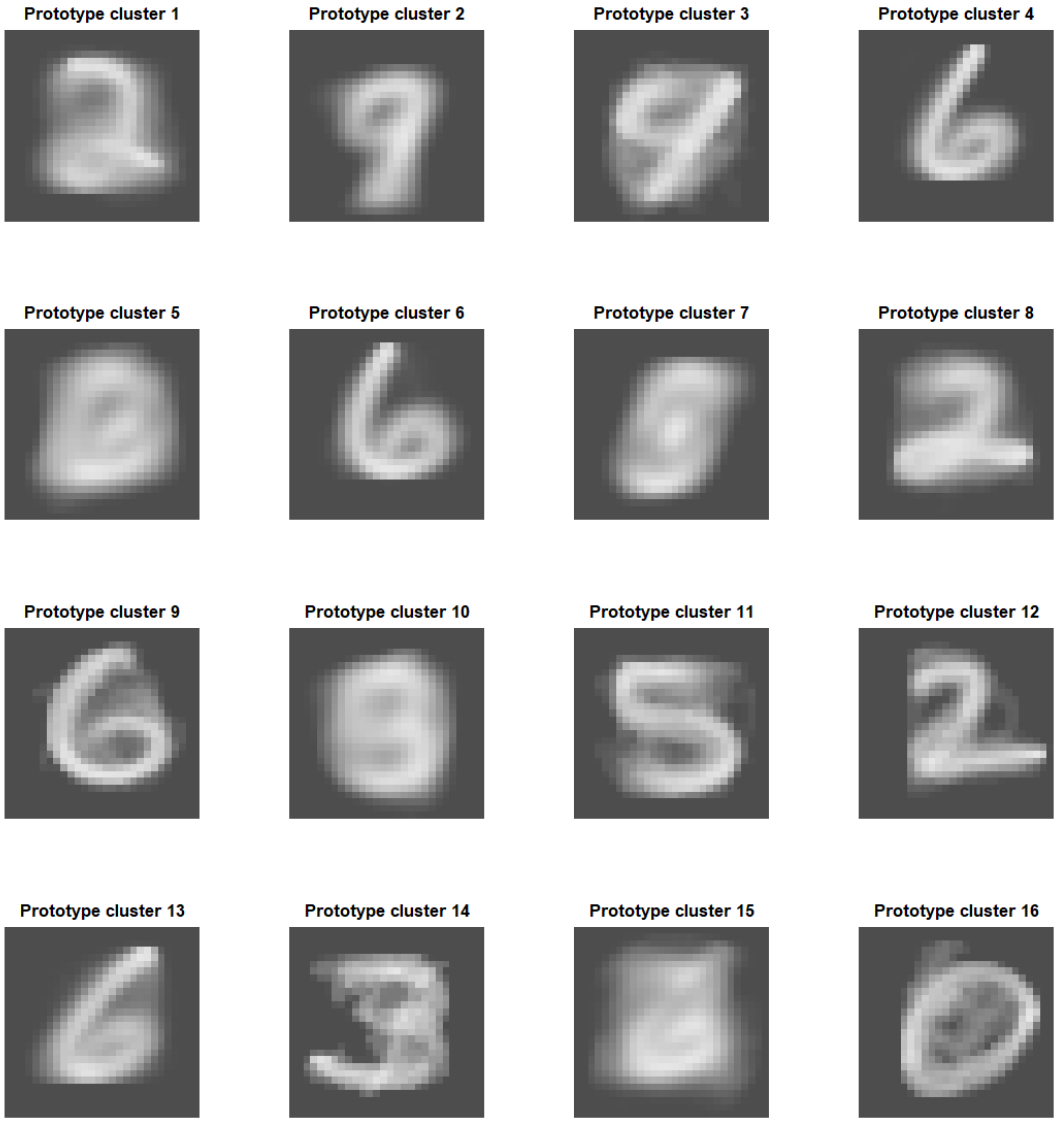

Fig. 7: Prototypes obtenus pour K = 16 clusters