SY09 – P18
Projet

1 Cuisine

L'objet de cette première partie est l'étude d'un jeu de données portant sur un ensemble de recettes de cuisine de diverses origines. Pour commencer, nous allons étudier une version agrégée par origine de ce jeu de données présent dans le fichier recettes-pays.data.

1. Faire une analyse exploratoire de ce jeu de données.

2. Faire une analyse en composantes principales de ce jeu de données. Interprétez les résultats obtenus.

On souhaite à présent grouper les origines des recettes en fonction de leur ressemblance.

3. Faire une analyse ascendante hiérarchique avec la distance de Manhattan et un critère d'aggrégation de votre choix.

4. Grouper les origines des recettes à l'aide de l'algorithme des K-means.

5. Comparer les classifications obtenues avec une classification géographique des origines.

Le jeu de données précédent provient en fait d'un jeu de données plus gros qui décrit plusieurs recettes en spécifiant leur origine et la composition en ingrédients. Un extrait de ce jeu de données est présent dans le fichier recettes-echant.data.

6. Faites une analyse descriptive rapide de ce jeu de données.

7. On cherche à grouper les différents ingrédients. Transformez les données en tableau individu–variable portant sur les ingrédients et proposez une méthode pour calculer une matrice de similarité ou de dissimilarité entre les ingrédients. Justifiez vos choix.

8. Effectuer une classification ascendante hiérarchique à l'aide de la dissimilarité obtenue précédemment. Combien de classes d'ingrédients semblent se distinguer ?

9. En utilisant la fonction `pam` de la bibliothèque `cluster`, appliquez l'algorithme des K-médoïdes et proposer des ingrédients représentant chaque classe.

2 Classification par K-means avec distance adaptative

L'objet de cette seconde partie est d'implémenter et d'étudier une variante de l'algorithme des K-means permettant d'adapter la métrique à la distribution conditionnelle des données, et plus particulièrement à la dispersion des données dans chacune des classes.

Ce travail comporte trois aspects. Dans un premier temps, il s'agira d'implémenter cette version adaptative de l'algorithme des K-means ; elle sera ensuite appliquée à divers jeux de données afin

d'appréhender les difficultés qui peuvent résulter de la prise en compte d'une métrique adaptée à la distribution des données dans les classes recherchées.

Enfin, il s'agira de justifier la méthode d'un point de vue théorique, en montrant :

1. que l'emploi d'une métrique spécifique à chaque classe ne change pas la stratégie de mise à jour des centres des classes par rapport à l'algorithme classique des K-means ;
2. que si l'on associe une métrique spécifique à chaque classe, la métrique optimale [1] pour chaque classe est définie par (l'inverse de) la matrice de covariance empirique de la classe.

2.1 Programmation

On demande ici de programmer la méthode des K-means avec distance adaptative (voir l'algorithme 1). On prendra en compte les éléments suivants.

Distance de Mahalanobis La distance entre deux points $\boldsymbol{x}$ et $\boldsymbol{y}$, au sens d'une métrique induite par une matrice M, est définie par

$$d_M^2(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x} - \boldsymbol{y})^\top M\ (\boldsymbol{x} - \boldsymbol{y})\,. \tag{1}$$

La fonction distXY fournie calcule les distances entre les points d'un tableau de données X et ceux d'un tableau de données Y, au sens d'une matrice M : elle retourne une matrice D de distances (au carré), de dimensions $n_x \times n_y$, où n_x et n_y désignent respectivement les nombres d'individus dans X et dans Y : l'élément d_{ij} de cette matrice correspond à la distance entre $\boldsymbol{x}_i$ et $\boldsymbol{y}_j$ au sens de M.

La distance de Mahalanobis entre un point $\boldsymbol{x}$ et le centre μ_k de sa classe est la distance entre ces points, au sens de l'inverse V_k^{-1} de la matrice de covariance empirique de la classe :

$$d_{V_k^{-1}} = (\boldsymbol{x} - \mu_k)^\top V_k^{-1}\ (\boldsymbol{x} - \mu_k)\,.$$

L'algorithme des K-means avec distance adaptative consiste à utiliser l'algorithme des K-means en remplaçant à chaque itération la distance euclidienne par la distance de Mahalanobis. L'objectif est de tenir compte de la dispersion des points qui peut varier selon la classe considérée.

Nous verrons par la suite que ce surcroît de flexibilité a un prix : cela nécessite d'estimer davantage de paramètres (à chaque itération, les matrices de covariance empirique des classes en plus des centres de gravité). Cette variante est donc plus sensible au manque de données que l'algorithme des K-means classique.

Arguments d'entrée La fonction à développer accepte évidemment comme arguments d'entrée le tableau de données que l'on cherche à regrouper en classes, et le nombre de classes à rechercher (on pourrait alternativement donner directement à la fonction des coordonnées des centres des classes à partir desquels itérer).

La fonction prend de même en arguments un nombre d'itérations maximal n_{iter}, un nombre d'essais n_{ess}, et une précision ε (par défaut, $n_{\text{iter}} = 100$, $n_{\text{ess}} = 1$ et $\varepsilon = 10^{-5}$). Le premier argument a pour but d'empêcher l'algorithme d'itérer indéfiniment à la recherche d'un optimum. Le second représente le nombre de fois où l'algorithme sera appliqué au jeu de données X à partir de différentes initialisations, de manière à augmenter les chances de converger vers un « bon » optimum local du critère optimisé. Le dernier permet de déterminer quand on peut considérer que l'algorithme a convergé.

Initialisation Les centres des classes μ_k peuvent être initialisés par K points tirés au hasard dans le jeu de données X. On pourra pour cela s'appuyer sur la fonction sample, qui permet de tirer au hasard un certain nombre d'éléments dans un vecteur.

Chaque matrice de covariance normalisée $\tilde{V}_k$ peut être initialisée par $(\rho_k)^{-1/p}\, I_p$, où I_p est la matrice identité de dimension p, et ρ_k est le volume souhaité pour la matrice $\tilde{V}_k^{-1}$ (voir ci-dessous). On pourra prendre par défaut $\rho_k = 1$, pour tout $k = 1, \ldots, K$.

1. au sens du critère de la somme des distances aux centres

Mise à jour des paramètres L'algorithme des K-means avec distance adaptative répète les opérations suivantes, jusqu'à convergence :

1. une nouvelle partition des données $P = (P_1, \ldots, P_K)$ en K groupes est déterminée, en affectant chaque point à la classe dont le centre μ_k est le plus proche au sens de la distance de Mahalanobis (calculée avec la matrice de covariance normalisée $\tilde{V}_k$ — voir ci-dessous) : on pourra pour cela s'appuyer sur la fonction which.min, combinée avec la fonction apply ;
2. à partir de cette nouvelle partition, les centres des classes et les matrices de covariance (normalisées) sont mis à jour ; plus particulièrement :

$$\begin{aligned} \mu_k &\longleftarrow \frac{1}{n_k} \sum_{i:\boldsymbol{x}_i \in P_k} \boldsymbol{x}_i ; \\ V_k &\longleftarrow \frac{1}{n_k} \sum_{i:\boldsymbol{x}_i \in P_k} (\boldsymbol{x}_i - \mu_k)(\boldsymbol{x}_i - \mu_k)^\top , \\ \tilde{V}_k &\longleftarrow (\rho_k \det V_k)^{-1/p} V_k , \end{aligned}$$

où $n_k = \operatorname{card}\{i : \boldsymbol{x}_i \in P_k\}$.

Pour diverses raisons (qui seront en partie élucidées par la suite), il est nécessaire de normaliser les matrices de covariance empiriques V_k utilisées dans le calcul de la distance de Mahalanobis. À chaque étape de l'algorithme, la matrice $\tilde{V}_k$ normalisée est ainsi calculée à partir de V_k de telle sorte que $\det \tilde{V}_k^{-1} = \rho_k$.

Convergence À chaque itération de l'algorithme, la convergence peut être testée en calculant la distance δ entre les centres μ_k des classes calculés à l'itération présente et ceux $\mu_{k,\text{prec}}$ calculés à l'itération précédente :

$$\delta = \sum_{k=1}^{K} \|\mu_k - \mu_{k,\text{prec}}\|^2 .$$

On supposera que la convergence est atteinte dès lors que $\delta \leq \varepsilon$, où ε est une valeur de précision fixée par l'utilisateur.

Essais À chaque essai, une fois que l'algorithme a convergé vers un optimum local — c'est-à-dire une partition définie par K centres $\mu_k^\star$ et matrices $\tilde{V}_k^\star$, on dispose également de la valeur du critère correspondante, c'est-à-dire la somme des distances des points au centre de gravité de leur classe :

$$J^\star = J(\mu_k^\star, \tilde{V}_k^\star) = \sum_{k=1}^{K} \sum_{i:\boldsymbol{x}_i \in P_k} d^2_{(\tilde{V}_k^\star)^{-1}}(\boldsymbol{x}_i, \mu_k).$$

Si la valeur de ce critère est plus faible que celle $J(\mu_k^{\text{prec}}, \tilde{V}_k^{\text{prec}})$ obtenue après une exécution précédente de l'algorithme (pour les mêmes valeurs de ρ_k), alors l'optimum local considéré est meilleur que l'optimum local précédent.

À chaque nouvelle convergence de l'algorithme, il convient alors de conserver en mémoire les paramètres correspondant à l'optimum local courant, s'ils sont meilleurs que ceux correspondant au meilleur optimum local obtenu précédemment.

Conseils de développement L'algorithme à développer est simple et peut être réalisé avec un certain nombre de fonctions usuelles de R : apply, sample, which.min, ainsi que la fonction distXY fournie.

On pourra stocker les coordonnées des centres des classes dans une matrice $K \times p$ et les matrices de covariance dans un tableau $p \times p \times K$. Pour retourner tous les arguments de sortie, on pourra utiliser une liste.

On évitera autant que possible les boucles (il pourra être nécessaire de boucler sur les classes, mais le code ne devrait pas comporter de boucles sur les individus).

2.2 Application

Données synthétiques

On cherche dans un premier temps à tester l'algorithme développé sur diverses données synthétiques, de manière à appréhender son fonctionnement et ses limitations. On considère pour cela trois jeux de données Synth1, Synth2 et Synth3. On pourra charger les données au moyen du code suivant :

```
X <- read.csv("SynthN.csv", header=T, row.names=1)
z <- X[,3]
X <- X[,-3]
```

Pour chacun des jeux de données, on effectuera une classification au moyen de l'algorithme des K-means classique, puis au moyen de l'algorithme des K-means avec distance adaptative. On comparera les résultats obtenus et on interprétera en fonction des données. On peut utiliser la fonction adjustedRandIndex (bibliothèque mclust) pour calculer l'adéquation de la partition trouvée à la partition réelle.

Données réelles

Iris On souhaite effectuer une classification des données Iris que l'on pourra charger au moyen du code suivant :

```
data(iris)
X <- iris[,1:4]
z <- iris[,5]
```

Déterminer une classification des données avec la méthode des K-means classique, puis avec les K-means adaptatifs, pour $K = 2, 3, \ldots, 5$. Calculer la valeur du critère optimisé (inertie dans le premier cas, distance totale dans le second) pour $K = 1$. Afficher les valeurs de critère en fonction de K. Quel semble être le meilleur nombre de classes dans chacun des cas ?

Effectuer une partition des données pour $K = 2$, puis pour $K = 3$. Comparer les résultats obtenus par les deux méthodes, et interpréter.

Spam On souhaite effectuer une classification des données Spam que l'on pourra charger au moyen du code suivant :

```
Spam <- read.csv("data/spam.csv", header=T, row.names=1)
X <- Spam[,-58]
z <- Spam[,58]
```

Le traitement de ces données peut poser un certain nombre de problèmes. Vous avez le choix des armes, en gardant à l'esprit que toute stratégie de pré-traitement, modification de l'algorithme, etc doivent être rendues explicites et si possible justifiées.

Les chargés de TD sont à votre disposition, dans une certaine mesure, pour vous aiguiller sur la piste de solutions.

2.3 Justification

L'objectif de cette partie est de justifier l'algorithme d'un point de vue théorique.

Plus particulièrement, supposons que l'on dispose d'une partition des données $P = (P_1, \ldots, P_K)$, où la classe de chaque individu est codée par une variable binaire

$$z_{ik} = \begin{cases} 1 & \text{si } \boldsymbol{x}_i \in P_k, \\ 0 & \text{sinon.} \end{cases}$$

On cherche à caractériser chaque classe par un prototype $\boldsymbol{v}_k$ et une métrique définie par une matrice M_k. On souhaite montrer que la minimisation du critère de distance

$$J\left(\{v_k, M_k\}_{k=1,\dots,K}\right) = \sum_{k=1}^{K}\sum_{i=1}^{n} z_{ik} d_{ik}^2 \tag{2}$$

sous la contrainte $\det M_k = \rho_k$ pour tout $k = 1, \dots, K$, où la distance $d_{ik}^2 = d_{M_k}^2(\boldsymbol{x}_i, \boldsymbol{v}_k)$ est définie par l'équation (1), donne

$$\boldsymbol{v}_k = \overline{\boldsymbol{x}}_k = \frac{1}{n_k}\sum_{i=1}^{n} z_{ik}\boldsymbol{x}_i, \tag{3}$$

$$M_k^{-1} = \left(\rho_k \det V_k\right)^{-1/p} V_k \text{ , avec } V_k = \frac{1}{n_k}\sum_{i=1}^{n} z_{ik}\left(\boldsymbol{x}_i - \boldsymbol{v}_k\right)\left(\boldsymbol{x}_i - \boldsymbol{v}_k\right)^{\top}. \tag{4}$$

À chaque itération, les prototypes des classes sont donc obtenus en calculant les centres de gravité, et les métriques sont définies par les (inverses des) matrices de covariance normalisées.

Remarquons que fixer $\det M_k = \rho_k$ pour tout $k = 1, \dots, K$ revient à fixer le volume de chaque classe : si l'on n'impose pas cette contrainte, il peut arriver qu'une ou plusieurs classes « absorbent » tous les points au détriment d'une ou plusieurs autres, ce qui cause des problèmes numériques, la matrice de covariance d'une classe d'effectif faible pouvant être de rang incomplet (et donc non inversible).

On admettra que pour minimiser le critère défini par (2)-(1) par rapport à $\boldsymbol{v}_k$ et M_k sous la contrainte $\det M_k = \rho_k$, on peut calculer le lagrangien

$$\mathcal{L}\left(\{v_k, M_k, \lambda_k\}_{k=1,\dots,K}\right) = J\left(\{v_k, M_k\}_{k=1,\dots,K}\right) - \sum_{k=1}^{K}\lambda_k(\det M_k - \rho_k), \tag{5}$$

où λ_k est le multiplicateur de Lagrange de la k^{e} contrainte, c'est-à-dire $\det M_k = \rho_k$. Il convient de minimiser ce lagrangien par rapport à λ_k, $\boldsymbol{v}_k$ et M_k : pour cela, il faut vérifier conjointement

$$\frac{\partial\mathcal{L}\left(\{v_k, M_k, \lambda_k\}\right)}{\partial\lambda_k} = 0, \quad \frac{\partial\mathcal{L}\left(\{v_k, M_k, \lambda_k\}\right)}{\partial M_k} = \mathbb{0}, \quad \frac{\partial\mathcal{L}\left(\{v_k, M_k, \lambda_k\}\right)}{\partial\boldsymbol{v}_k} = \mathbb{0}.$$

On définira la dérivée $\partial f(M)/\partial M$ d'une fonction $f(M)$ par rapport à une matrice M de terme général m_{ij} comme la matrice B de terme général $b_{ij} = \partial f(M)/\partial m_{ij}$.

Il faudrait de surcroît montrer que la matrice des dérivées secondes (matrice hessienne) est définie positive pour les valeurs de paramètres qui annulent le vecteur des dérivées premières (vecteur gradient). On pourra admettre ce résultat.

Questions Pour le calcul des dérivées, on pourra s'aider du « Matrix Cookbook », disponible à l'URL : `http://www2.imm.dtu.dk/pubdb/views/edoc_download.php/3274/pdf/imm3274.pdf`. Les calculs devront être justifiés rigoureusement.

10. Calculer $\partial\mathcal{L}\left(\{v_k, M_k, \lambda_k\}\right)/\partial\boldsymbol{v}_k$; montrer que la mise à jour des prototypes revient à calculer les centres de gravité des classes définis par l'équation (3).

11. Calculer $\partial\mathcal{L}\left(\{v_k, M_k, \lambda_k\}\right)/\partial M_k$, puis $\partial\mathcal{L}\left(\{v_k, M_k, \lambda_k\}\right)/\partial\lambda_k$. Montrer que la mise à jour des matrices M_k définissant la métrique spécifique à chaque classe revient à calculer les inverses des matrices de covariance empiriques normalisées définies par l'équation (4).

Données : Tableau de données X (matrice $n \times p$), nb. de classes K (entier), volumes des classes ρ_k (vecteur de longueur K), nb. max. d'itérations n_{iter} (entier), nb. d'essais n_{ess} (entier), tolérance ε pour vérifier la convergence (réel positif)

Résultat : valeur du critère, nb. d'itérations, partition des données, centres des classes, matrices de covariance empiriques normalisées des classes

pour $i = 1, 2, \ldots, n_{\text{ess}}$ **faire**

- initialiser μ_k et $\tilde{V}_k$, pour $k = 1, \ldots, K$
- **tant que** la convergence n'est pas atteinte et n_{iter} n'est pas dépassé **faire**
  - mettre à jour la partition des données : affecter chacun des points de X au centre μ_k le plus proche, au sens de la distance de Mahalanobis
  - mémoriser les anciens centres des classes : $\mu_{k,\text{prec}} \longleftarrow \mu_k$
  - mettre à jour les centres des classes μ_k et les matrices de covariance empiriques $\tilde{V}_k$ normalisées au moyen de la partition calculée précédemment
  - mettre à jour le critère de distance d_{tot} (somme des distances aux centres des classes)
  - mettre à jour le critère de convergence
- **fin**
- **si** $J^\star < J^{\text{opt}}$ (avec J^{opt} le meilleur optimum local donné par les essais précédents) **alors**
  - mettre à jour le meilleur optimum local : mémoriser la valeur du critère $J^\star$, le nb. d'itérations, la partition des données, les centres et les matrices de covariance
- **fin**

fin

Algorithme 1 : Algorithme des K-means avec distance adaptative