

## Fiche de Travaux Pratiques

---

### Analyse et traitement des données Classification (non supervisée) n° 2


---

#### Master 1 Informatique Parcours « Données et Systèmes Connectés » (DSC) Saint-Étienne, France

Fabrice Muhlenbach

Laboratoire Hubert Curien, UMR CNRS 5516  
Université Jean Monnet de Saint-Étienne  
18 rue du Professeur Benoît Lauras  
42000 SAINT-ÉTIENNE, FRANCE  
<https://perso.univ-st-etienne.fr/muhlfabr/>

### Résultats attendus

L'objectif de cette séance de TP est de se familiariser avec quelques méthodes de classification (en anglais : *clustering*, à ne pas confondre avec le terme anglais *classification* qui signifie « classement » et qui désigne une méthode d'apprentissage supervisé) existant dans , ainsi que de voir concrètement des problèmes liés à la réalisation d'une « bonne » classification avec quelques indices de qualité d'une classification.

## 1 Classification avec le jeu de données Iris


### 1.1 Chargement du jeu de données

Le jeu de données « Iris plants » est un exemple couramment utilisé en analyse de données. Les données peuvent être chargées en mémoire par la fonction `data`. Les quatre premières variables peuvent être utilisées pour la classification mais la variable « class » (la cinquième) peut être utilisée pour vérifier la validité de la classification.

```
data("iris")
mydata <- iris[1:4]
class <- as.matrix(iris[5])
```

## 1.2 Classification par la méthode des $k$ -Means

### 1.2.1 Introduction

La méthode des  $k$ -means (ou des  $k$  moyennes) est une méthode de classification (donc qui est non supervisée) qui peut être utilisée avec la fonction  `kmeans` en passant (au moins) 2 paramètres : le nom du jeu de données et la valeur de  $k$  (représentant le nombre de groupes, ou « clusters », à construire). Par exemple, avec notre jeu de données, en prenant  $k = 3$  clusters :

```
kmeans.result <- kmeans(mydata,3)
```

La fonction `summary` fournit les résultats obtenus par l'algorithme  $k$ -means :

```
summary(kmeans.result)
```

- `cluster` : nombre (allant de 1 à  $k$ ) indiquant le numéro du cluster correspondant à l'observation ;
- `centers` : coordonnées des centroïdes (les moyennes trouvées) ;
- `totss` : variance totale par somme des carrés des distances (*total sum of squares*) ;
- `withinss` : variance intra-cluster d'un cluster donné, c'est-à-dire au sein de chaque cluster (*within-cluster sum of squares*) ;
- `tot.withinss` : variance intra-cluster totale (*total within-cluster sum of squares*) ;
- `betweenss` : variance inter-cluster, c'est-à-dire entre les différents clusters (*between-cluster sum of squares*) ;
- `size` : nombre de points dans chaque clusters.

En affichant `kmeans.result`, nous obtenons la valeur de tous ces résultats.

### 1.2.2 Stabilité d'une classification

Pour voir la stabilité de la classification, on peut afficher le tableau de contingence avec les classes réelles des iris (ce que l'on appelle la « vérité terrain » obtenue par un expert) et les clusters obtenus par la méthode des  $k$ -means.

```
print(table(class, kmeans.result$cluster))
```

Lancer l'algorithme des  $k$ -means une dizaine de fois avec le jeu de données iris. Afficher à chaque fois le tableau de contingence ainsi que la somme des carrés intra-clusters (within-cluster sum of squares, WSS) et la somme des carrés inter-clusters (between-cluster sum of squares, BSS). On affichera les proportions de chaque variance par rapport à la variance totale en pourcentage, avec deux décimales.

```
w <- (kmeans.result$tot.withinss/kmeans.result$totss)*100
b <- (kmeans.result$betweenss/kmeans.result$totss)*100
print(paste("WSS=",round(w,2),"%"))
print(paste("BSS=",round(b,2),"%"))
print(table(class, kmeans.result$cluster))
```

Obtient-on à chaque fois les mêmes résultats ? Avez-vous remarqué le lien entre les résultats du clustering obtenus par la méthode des  $k$ -means et les valeurs des variances intra- (WSS) et inter- (BSS) clusters ?

### 1.2.3 Validation d'une classification

Les problèmes qui se posent lors de la réalisation d'une classification sont les suivants :

- Qu'est-ce qu'une « bonne » classification ?
- Combien de clusters faut-il choisir ?
- Comment trouver ces clusters ?
- Une bonne méthode de classification doit produire des clusters de qualité ayant :
  - une forte similarité au sein de chaque cluster (*high intra-class similarity*), ou une faible variance au sein de chaque cluster ;
  - une faible similarité entre les différents clusters (*low inter-class similarity*), ou une forte variance entre les différents clusters.
- La qualité du résultat d'une méthode de classification dépend à la fois de la mesure de similarité utilisée par la méthode et de son implémentation.
- La qualité d'une méthode de classification est aussi mesurée en fonction de sa capacité à découvrir des motifs cachés au sein des données.

### 1.2.4 Critère de validation interne : cohésion et séparation

La validation d'une bonne classification peut être considérée de 3 manières différentes :

- la classification est évaluée en terme de structure tirée indépendamment, imposée sur les données a priori, les critères suivant cette approche sont appelés « critères externes » ;
- la classification est évaluée en terme de quantités tenant compte des vecteurs d'attributs eux-mêmes (par exemple, la matrice de proximité), ces critères sont appelés « critères internes » ;
- la classification est évaluée par comparaison à d'autres structures de classification, résultant de l'application du même algorithme de classification mais avec d'autres paramètres (par exemple, un nombre différent de clusters à découvrir) ou de l'application de différentes méthodes de classification sur le même jeu de données, ces critères sont appelés « critères relatifs ».

Quand nous connaissons la « vérité terrain », il est possible d'appliquer des critères externes tels que la statistique de Rand, de Jaccard ou l'indice de Fowlkes-Mallows, etc. Cependant, la plupart du temps dans les apprentissages non supervisés, nous ne connaissons pas cette information qu'est la vérité terrain (à la différence du jeu de données des iris) et une réelle classe n'existe pas. C'est pourquoi l'objectif de la classification est de « construire » des classes, et c'est aussi pourquoi nous avons besoin de critères de validation internes.

La **cohésion de classification** est la somme des poids de tous les liens reliant les différents points d'un même cluster (Figure 1). En pratique, cette valeur est calculée comme la somme des carrés des distances des points au sein d'un même cluster (*WSS : within-cluster sum of squares*).

La **séparation de classification** est la somme des poids entre les points d'un cluster et des points en dehors de ce cluster (Figure 1). En pratique, cette valeur est calculée comme la somme des carrés des distances entre les différents clusters (*BSS : between-cluster sum of squares*).

Le critère de validation interne utilise ces valeurs (WSS et BSS) et une classification sera considérée comme meilleure qu'une autre si elle parvient à mieux minimiser la variance intra-cluster (WSS) et à mieux maximiser la variance inter-cluster (BSS), comme présenté sur la Figure 2.

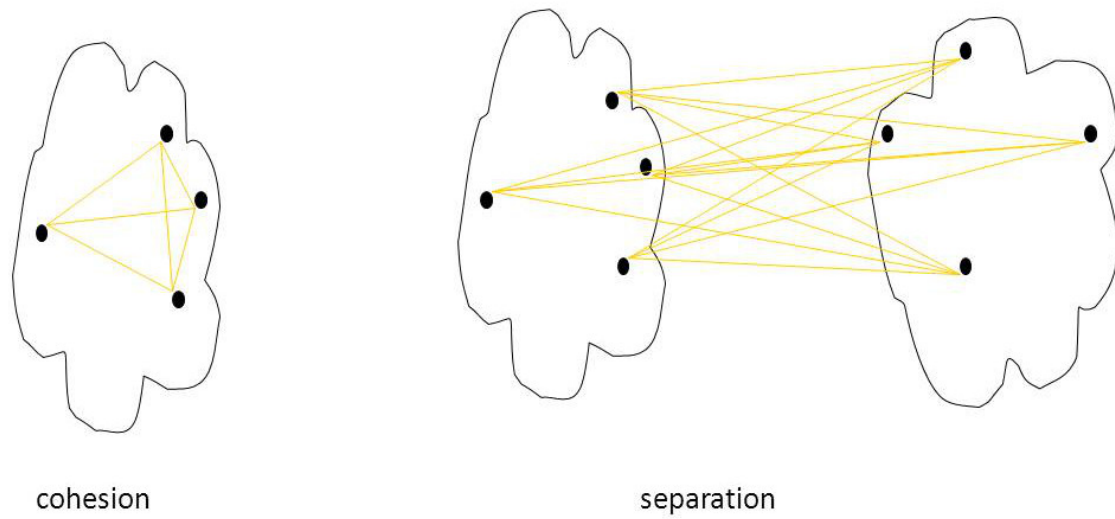


FIG. 1 – *Principes de cohésion et de séparation.*

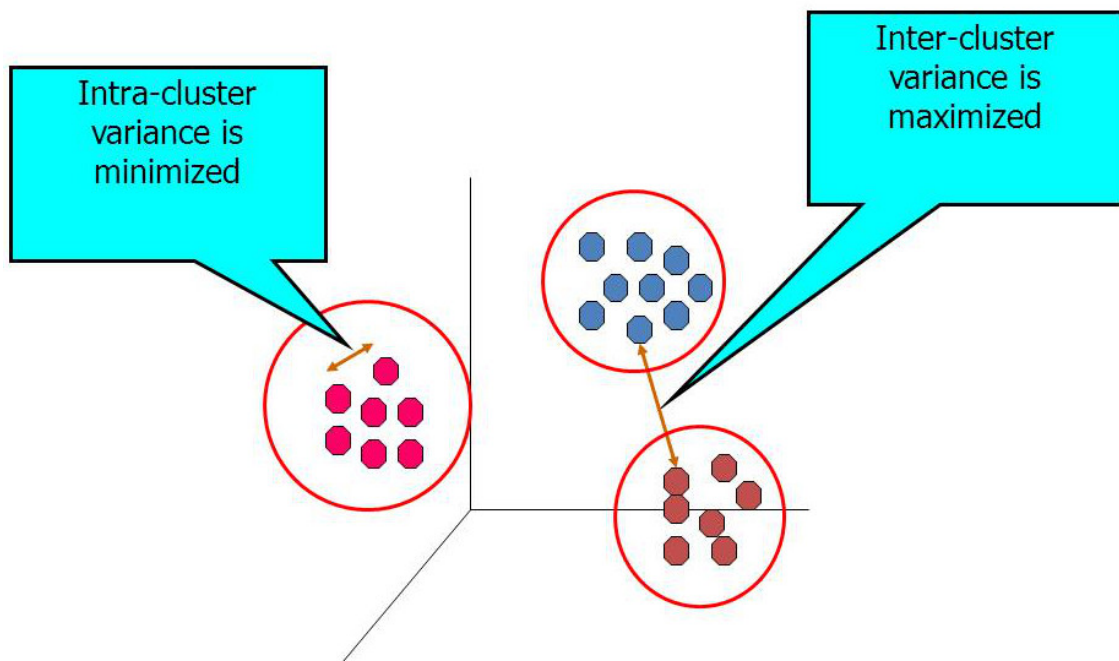


FIG. 2 – *Variance inter-cluster et variance intra-cluster.*

### 1.3 Classification hiérarchique

Avec **R**, une classification hiérarchique peut être obtenue avec la fonction `hclust` (*hierarchical clustering*). Cette fonction nécessite aussi (au moins) 2 paramètres : une matrice de distance et le nom d'une méthode d'agglomération à utiliser (par exemple, "ward", "single", "complete", "average", "mcquitty", "median" ou "centroid").

Ainsi, avec notre jeu de données et une méthode d'agrégation moyenne (average), les instructions deviennent :

```
hc <- hclust(dist(mydata), "ave")
```

Le résumé des résultats de cette classification hiérarchique est obtenu par la fonction `summary`.

```
summary(hc)
```

L'affichage de `hc` n'est pas très intéressant. Par contre, ce résultat peut être graphiquement représenté sous la forme d'un dendrogramme.

```
win.graph(800, 600, 10)
plot(hc, hang = -1, labels=class)
```

À noter que quand il y a trop d'individus dans un jeu de données, le résultat d'une classification hiérarchique est difficile à lire. Il est possible de ne considérer qu'un échantillon des données (fonction `sample`) et de lancer la classification hiérarchique sur cet échantillon.

```
idx <- sample(1:dim(iris)[1], 40)
irisSample <- iris[idx,]
irisSample$Species <- NULL
hc <- hclust(dist(irisSample), method="ave")
win.graph(800, 600, 10)
plot(hc, hang = -1, labels=iris$Species[idx])
```

Sur ce dendrogramme, on peut réaliser une coupure pour obtenir 3 clusters de la manière suivante :

```
rect.hclust(hc, k=3)
groups <- cutree(hc, k=3)
```

## 2 Classification avec le jeu de données Ruspini

### 2.1 Chargement du jeu de données

Le jeu de données Ruspini est un fichier texte qui peut être chargé en mémoire à partir de la fonction `read.table`. Au préalable, nous pouvons enlever de la mémoire les variables utilisées.

```
rm(list=ls())  
mydata <- read.table("~/R/data/ruspini.txt", quote="\")
```

### 2.2 Classification avec $k$ -Means et critères de validation interne

À la différence du jeu de données sur les iris, avec le jeu de données Ruspini nous n'avons pas de vérité terrain et nous ne savons pas combien de clusters nous devons retrouver dans les données.

Lancer l'algorithme des  $k$ -means avec un nombre de clusters  $k$  variant de 2 à 10 clusters. Pour chaque essai, afficher les valeurs des variances intra- et inter-clusters (respectivement WSS et BSS) ainsi que l'indice de qualité de classification de Ball et Hall (1965) et l'indice de qualité de classification de Calinski et Harabasz (1974).

Ces indices sont calculés de la manière suivante. L'indice de Ball et Hall est le rapport de la variance intra-classe sur le nombre de clusters :

$$B\_H\_index = WSS/k$$

et l'indice de Calinski et Harabasz calculé de la manière suivante :

$$C\_H\_index = \frac{BSS/(k-1)}{WSS/(n-k)}$$

Ces deux indices ne s'interprètent pas de la même manière. Plus petite sera la valeur donnée par l'indice de Ball et Hall, meilleure sera considérée la classification, alors que plus grande sera la valeur de l'indice de Calinski et Harabasz, meilleure la classification sera.

À l'aide de ces deux indices de qualité de la classification (et plus particulièrement du dernier), quel est le nombre de clusters  $k$  le plus pertinent de créer dans ce jeu de données ?