EISTI	ING1-GI : GÉNIE INFORMATIQUE DATA EXPLORATION : CLUSTERING
Rédigé par : Astrid Jourdan	Ref :
A l'intention de : Elèves d'ING1-GI	Créé le : 22/12/2016

Exercice 1

Dans $\mathbb{R}$, on considère 5 points $x_1=1$, $x_2=2$, $x_3=9$, $x_4=12$ et $x_5=20$.

1) Appliquer l'algorithme des k-means avec les valeurs de k et les points de départ suivants. Calculer le pourcentage d'inertie expliquée par la partition obtenue.

   a) k=2, $g_1=1$ et $g_2=20$
   b) k=2, $g_1=2$ et $g_2=9$
   c) k=3, $g_1=1$, $g_2=9$ et $g_3=12$
   d) Peut-on utiliser l'inertie inter-classes pour déterminer le meilleur regroupement des deux?

2) Appliquer une méthode de classification hiérarchique ascendante en utilisant la distance minimale comme critère de dissimilarité entre classes. Tracer le dendrogramme (avec en ordonnée la distance minimale). Quel regroupement vous parait correct ?

Correction

1.a)

1ère itération

		X1	X2	X3	X4	X5
		1	2	9	12	20
g1	1	0	1	8	11	19
g2	20	19	18	11	8	0

Tableau des distances entre Xi et gk

1ère classe C1={ X1, X2, X3} → g1=(1+2+9)/3=4
2ème classe C2={ X4,X5} → g2=(12+20)/2=16

2ème itération

		X1	X2	X3	X4	X5
		1	2	9	12	20
g1	4	3	2	5	8	16
g2	16	15	14	7	4	4

Tableau des distances entre Xi et gk

1ère classe C1={ X1, X2, X3}
2ème classe C2={ X4,X5}
Les classes n'ont pas changé donc l'algorithme s'arrête

- Centre de gravité : g=(1+2+9+12+20)/5=8,8
- Inertie totale : $I_{tot}=[(1-8,8)^2+(2-8,8)^2+(9-8,8)^2+(12-8,8)^2+(20-8,8)^2]/5=48,56$
- Inertie inter classes : $I_{inter}=[3\times(4-8,8)^2+2\times(16-8,8)^2]/5=34,56$
- Pourcentage d'inertie expliquée : 34,56/48,56=0,71

Ce partitionnement explique 71% de l'inertie du nuage de points

1.b)

1ère itération

		X1	X2	X3	X4	X5
		1	2	9	12	20
g1	2	1	0	7	10	18
g2	9	8	7	0	3	11

Tableau des distances entre Xi et gk

1ère classe C1={ X1, X2} → g1=(1+2)/2=1,5
2ème classe C2={ X3,X4,X5} → g2=(9+12+20)/3=10,33

2ème itération

		X1	X2	X3	X4	X5
		1	2	9	12	20
g1	1,5	0,5	0,5	7,5	10,5	18,5
g2	10,3	9,3	8,3	1,3	1,7	9,7

Tableau des distances entre Xi et gk

1ère classe C1={ X1, X2}
2ème classe C2={ X3,X4,X5}
Les classes n'ont pas changé donc l'algorithme s'arrête

- Inertie inter classes : $I_{inter}=[2\times(1,5-8,8)^2+3\times(10,33-8,8)^2]/5=22,7$
- Pourcentage d'inertie expliquée : 22,7/48,56=0,47

Ce partitionnement explique 47% de l'inertie du nuage de points

1.c)

1ère itération

		X1	X2	X3	X4	X5
		1	**2**	**9**	**12**	**20**
g1	**1**	0	1	8	11	19
g2	**9**	8	7	0	3	11
g3	**12**	11	10	3	0	8

Tableau des distances entre Xi et gk

1ère classe C1={ X1, X2} → g1=(1+2)/2=1,5
2ème classe C2={ X3} → g2=X3=9
3ème classe C3={ X4,X5} → g3=(12+20)/2=12
Les classes n'ont pas changé donc l'algorithme s'arrête.

- Inertie inter classes : $I_{inter}=[2\times(1,5-8,8)^2+1\times(9-8,8)^2+2\times(12-8,8)^2]/5=23,37$
- Pourcentage d'inertie expliquée : 23,37/48,56=0,48

Ce partitionnement explique 48% de l'inertie du nuage de points

1.d) Le critère permet de départager les deux partitions à 2 classes, la meilleure étant la première. En revanche, il ne peut pas être utilisé pour comparer des partitions n'ayant pas le même nombre de classes. En effet, plus le nombre de classes augmente, plus le pourcentage d'inertie expliquée augmente. Le cas extrême est un nombre de classes égal au nombre d'individus et dans ce cas le pourcentage d'inertie expliquée est 100% (cf. formule de l'inertie intra = 0). Et pourtant, ce partitionnement n'est pas souhaitable. Il faut donc trouver un compromis entre un pourcentage élevé et un nombre de classes faible.

2)

1ère itération

C1={X1}, C2={X2}, C3={X3}, C4={X4}, C5={X5}

		C1	C2	C3	C4	C5
		{1}	**{2}**	**{9}**	**{12}**	**{20}**
C1	**{1}**	0	1	8	11	19
C2	**{2}**	1	0	7	10	18
C3	**{9}**	8	7	0	3	11
C4	**{12}**	11	10	3	0	8
C5	**{20}**	19	18	11	8	0

Tableau des distances entre Xi et les classes

Les deux classes les plus proches sont C1 et C2 donc on les regroupe.
C1={X1,X2}, C2={X3}, C3={X4}, C4={X5}

2ème itération

		C1	C2	C3	C4
		{1,2}	**{9}**	**{12}**	**{20}**
C1	**{1,2}**	0	7	10	18
C2	**{9}**	7	0	3	11
C3	**{12}**	10	3	0	8
C4	**{20}**	18	11	8	0

Tableau des distances entre Xi et les classes

Les deux classes les plus proches sont C3 et C2 donc on les regroupe.
C1={X1,X2}, C2={X4,X3}, C3={X5}

3ème itération

		C1 {1,2}	C2 {9,12}	C3 {20}
C1	**{1,2}**	0	7	18
C2	**{9,12}**	7	0	8
C3	**{20}**	10	8	0

Tableau des distances entre Xi et les classes

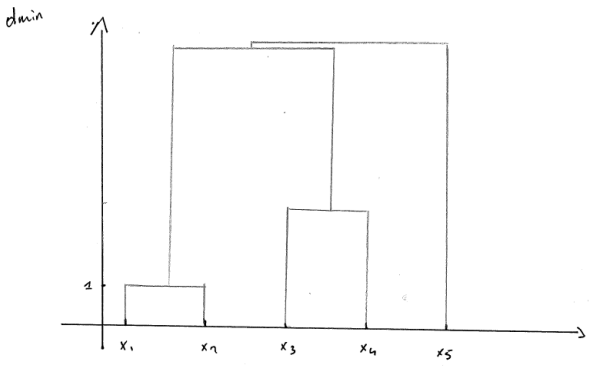
18

4ème itération

		C1 {1,2,9,12}	C2 {20}
C1	**{1,2,9,12}**	0	7
C2	**{20}**	7	0

Tableau des distances entre Xi et les classes

Les deux classes les plus proches sont C1 et C2 donc on les regroupe.
C1={X1,X2,X4,X3}, C3={X5}



Exercice 2

L'objectif de cet exercice est de tester les algorithmes k-means et CAH sur des jeux de données simulés,

Test_Clusters_Distincts.txt
Test_Clusters_Random.txt
Test_Clusters_Melanges.txt
Test_Clusters_Atypiques.txt

Pour cela, on utilisera le langage R avec ses fonctions `kmeans` et `hclust`.

Fonction kmeans

`kmeans(x,centers,nstart,…)`

Entrées :

- x = données de type matrice
- centers = soit le nombre de classes, soit une matrice contenant les coordonnées des points initiaux
- nstart=nombre d'initialisations

Sorties :

- $cluster = vecteur d'entiers de indiquant le numéro de la classe de chaque individu
- $centers = matrice des distances entre les individus et les centres de chaque classe
- $size = vecteur indiquant la taille de chaque classe
- $iter = nombre d'itérations

```
# a 2-dimensional example

x=rbind(matrix(rnorm(100, sd = 0.3), ncol = 2),      # rnorm génère une matrice de
  matrix(rnorm(100, mean = 1, sd = 0.3), ncol = 2))  # réalisations d'une loi normale
                                                     # rbind concatène des matrices

x=scale(x) #centre et réduit

cl= kmeans(x,center=2,nstart=5)  #clustering à 2 classes

print(cl) # affiche les résultats

plot(x, col = cl$cluster) # affiche les points avec une couleur différente par classe
                          # indexée par le numéro de la classe
points(cl$centers, col = 1:2, pch = 8, cex = 2) # ajoute les centres des classes
```

col = 0:2 pour afficher les 3 centres

Fonction hclust pour la construction de l'arbre

```
hclust(d, method = "ward.D2",…)
```

Entrées :

- d=structure de dissimilarité entre les individus générée par la fonction `dist`
- method = mesure de dissimilarité entre les classes

Sorties : plusieurs attributs décrivant l'arbre. On retient

- $height = vecteur indiquant la valeur du critère à chaque branche

Fonction cutree pour la construction des classes

```
cutree(tree, k = 2,…)
```

Entrées :

- tree=arbre résultant de hclust
- k = entier indiquant le nombre de classes

Sorties :

- un vecteur indiquant le numéro des classes

Fonctions pour représenter le dendrogramme

`plot(tree)` affiche le dendrogramme

`rect.hclust(tree,k=nclusters)` ajoute les classes sur le dendrogramme

```
# a 2-dimensional example

x=rbind(matrix(rnorm(100, sd = 0.3), ncol = 2),      # rnorm génère une matrice de
  matrix(rnorm(100, mean = 1, sd = 0.3), ncol = 2))  # réalisations d'une loi normale
                                                     # rbind concatène des matrices
x=scale(x) #centre et réduit
distance=dist(x,"euclidean")  #crée une structure de distance entre les individus

h=hclust(distance, "ward.D2") # crée l'arbre
plot(h$height) # affiche l'évolution du critère de dissimilarité entre classes
plot(h) # affiche le dendrogramme
rect.hclust(h,k=2) # ajoute les classes

c=cutree(h,k=2) # crée les classes
plot(x, col = c) # affiche les points avec une couleur différente par classe indexée
                 # par le numéro de la classe
```

1) Algorithme des Kmeans
   a) Tester l'algorithme des kmeans sur les données *Test_Clusters_Distincts.txt*. Essayer plusieurs nombres de classes et choisir le meilleur.
   b) Tester l'algorithme des kmeans sur les données *Test_Clusters_Distincts.txt*, *Test_Clusters_Melanges.txt* et *Test_Clusters_Random.txt*. Constater l'évolution de l'inertie expliquée.
   c) Tester l'algorithme des kmeans sur les données *Test_Clusters_Corr.txt*. Que pourrait-on faire pour améliorer le résultat.
   d) Tester l'algorithme des kmeans sur les données *Test_Clusters_Atypique.txt* avec les individus n°1 et n°1499 pour initialisation

Correction

a)
```
don=read.table("Test_Clusters_Distincts.txt",header=F) # attention à etre dans le bon répertoire
don= as.matrix(don)
k=kmeans(don,2)
```

```
    print(k)
    plot(don,col=k$cluster)
# Insister sur le caratère stochastique de l'algo
# On fait de même avec 3, 4 classes et on constate que 3 donne le plus grand pourcentage d'inertie expliquée
b) don=as.matrix(read.table("Test_Clusters_Corr.txt",header=F) )
    k=kmeans(don,2)
    print(k)
    plot(don,col=k$cluster)
   cor(don)
# Les données sont corrélées. Il faudrait utiliser la distance de Mahalanobis. On note que la fonction kmeans de R
            ne permet pas de changer la distance.
c)  don=as.matrix(read.table("Test_Clusters_Atypiques.txt",header=F) )
   centre=matrix(0,2,ncol(don))
   centre[1,]=don[1,]
   centre[2,]=don[1499,]
    k=kmeans(don,centre)
    print(k)
    plot(don,col=k$cluster)
```

2) Classification ascendante hiérarchique
   a) Tester l'algorithme CAH sur les données *Test_Clusters_Distincts.txt*, *Test_Clusters_Melanges.txt* et *Test_Clusters_Random.txt*. Constater l'évolution du dendrogramme.
   b) Tester l'algorithme CAH sur les données *Test_Clusters_Atypique.txt* avec la méthode « ward.D2 » et la méthode « average ».
   c) Comparer les résultats obtenus entre CAH et Kmeans sur les données *Test_Clusters_Distincts.txt*, *Test_Clusters_Melanges.txt* et *Test_Clusters_Random.txt*.

Correction

```
a)  don=read.table("Test_Clusters_Distincts.txt",header=F)
d=dist(don,"euclidean")
h=hclust(d,"ward.D2")
plot(h$height)
X11()
plot(h)
rect.hclust(h,3)
c=cutree(h,3)
X11()
plot(don,col=c)
```

b) La moyenne étant attirée par les valeurs extrêmes, l'algorithme crée deux classes distinctes avec les individus atypiques

```
c)  don=read.table("Test_Clusters_Distincts.txt",header=F)
c.kmeans=kmeans(as.matrix(don),3)$cluster
d=dist(don,"euclidean")
h=hclust(d,"ward.D2")
c.CAH=cutree(h,3)
table(c.kmeans,c.CAH)
```

les resultats sont identiques pour les clusters distincts et ensuite de moins en moins semblables.

Quelques rappels sur R

R *est un langage et un environnement logiciel open source pour les calculs statistiques et graphiques. R devient incontournable dans le traitement exploratoire et statistique des données.*

Site Internet : Statistiques avec R
http://zoonek2.free.fr/UNIX/48_R_2004/all.html

- `# pour écrire un commentaire`
- `help(`*`fonction`*`)  # pour obtenir l'aide sur une fonction`
- `ls()  # permet d'afficher tous les objets de la session de travail`
- `rm()  # efface les objets de la session de travail`
- `q()  # permet de quitter R`

La session de travail avec tous les objets qui auront été définis dedans (matrices, dataframes, fonctions, …) peut être sauvegardée puis rechargée à chaque utilisation :

- `getwd()  # affiche le répertoire courant`
- `setwd("K:/ING1/GI/Statistiques Descriptives")  # permet de fixer le chemin d'accès au répertoire. attention d'utiliser / et non \`
- `save.image(file="nom.Rdata") # sauvegarde la session de travail dans le fichier nom.RData du répertoire courant`
- `load("nom.Rdata") # recharge la session de travail`

Lecture d'un fichier

- `tab=read.table("Nom Fichier",header=TRUE ou FALSE si noms des colonnes indiqués dans le fichier,sep=" séparateur de champs",dec="symbole pour la décimale")`

Quelques instructions

- `as.matrix(tab) # transforme tab en matrice`
- `plot(matrice de points) # graphique des points`
- `points(matrice de points) # pour ajouter des points à un graphique existant`
- `X11() pour ouvrir une nouvelle fenêtre graphique`
- `tab[i,j] # élément (i,j) du tableau`
- `table(V1,V2) # crée le tableau de contingence des variables V1 et V2`