Quelques méthodes de fouille de données

Ivan Kojadinovic

IUT STatistique et Informatique Décisionnelle
Laboratoire de Mathématiques et de leurs Applications
UMR CNRS 5142
Avenue de l'Université - BP 1155
64013 Pau Cedex - France

Chapitre 1

Deux méthodes de classification automatique

Ce chapitre est en partie fondé sur un cours de Philippe Besse, dispensé à l'INSA Toulouse.

1.1 Les données

Les données concernent n individus statistiques sur lesquels p variables ont été observées.

Notation : Dans la suite, le tableau "individus x variables" est notée X et ses éléments X_{ij}, avec $i \in \{1, ..., n\}$ l'indice des lignes et $j \in \{1, ..., p\}$ l'indice des colonnes.

Lorsque nécessaire, nous effectuerons une lecture en colonne de X et nous noterons les p variables observées $V_1, ..., V_p$.

$$X = \begin{array}{c|cccc} & V_1 & V_2 & \cdots & V_p \\ \hline 1 & \ddots & & & \\ 2 & & \ddots & & \\ \vdots & & & \ddots & \\ n & & & & \ddots \end{array}$$

Par exemple :
- individus : "voitures" ;
- V_1 : couleur ; V_2 : consommation ; V_3 : puissance fiscale.

Dans la suite, **l'ensemble des individus est noté** Ω.

Dans l'exemple précédent on voit que les variables ne sont pas toutes de même type :
- "couleur" est qualitative (nominale) ;
- "consommation" est quantitative (continue) ;

— "puissance fiscale" est quantitative (discrète) ;

Dans le cas de la classification automatique, on peut considérer trois situations relativement à la nature des variables :

— les variables sont toutes qualitatives ;
— les variables sont toutes quantitatives ;
— un mélange de variables qualitatives et quantitatives (le plus difficile en pratique).

Nous allons uniquement traiter le cas où **toutes les variables sont quantitatives**.

1.2 Objectifs de la classification automatique

Nous allons énoncer ces objectifs dans le cas classique d'une classification automatique de Ω.

Définition 1. *On appelle partition de Ω en c classes la donnée de c sous-ensembles (groupes, classes) de Ω, notés $S_1, ..., S_c$ tels que :*

— $S_i \neq \varnothing$ *pour tout* $i \in \{1, ..., c\}$;
— $S_i \cap S_j = \varnothing$ *pour* $i \neq j$;
— $S_1 \cup S_2 \cup \cdots \cup S_c = \Omega$.

L'objectif de la classification automatique de Ω est d'obtenir une partition de Ω notée $\{S_1, ..., S_c\}$ telle que les sous-ensembles/groupes/classes de la partition soient aussi "homogènes" que possible et que les sous-ensembles (groupes, classes) soient aussi "séparés" les uns des autres que possible.

Remarques :

— Nous allons préciser les notions de séparation et d'homogénéité mathématiquement dans la suite.
— Le nombre de classes c peut être soit donné par l'utilisateur soit déterminé automatiquement.

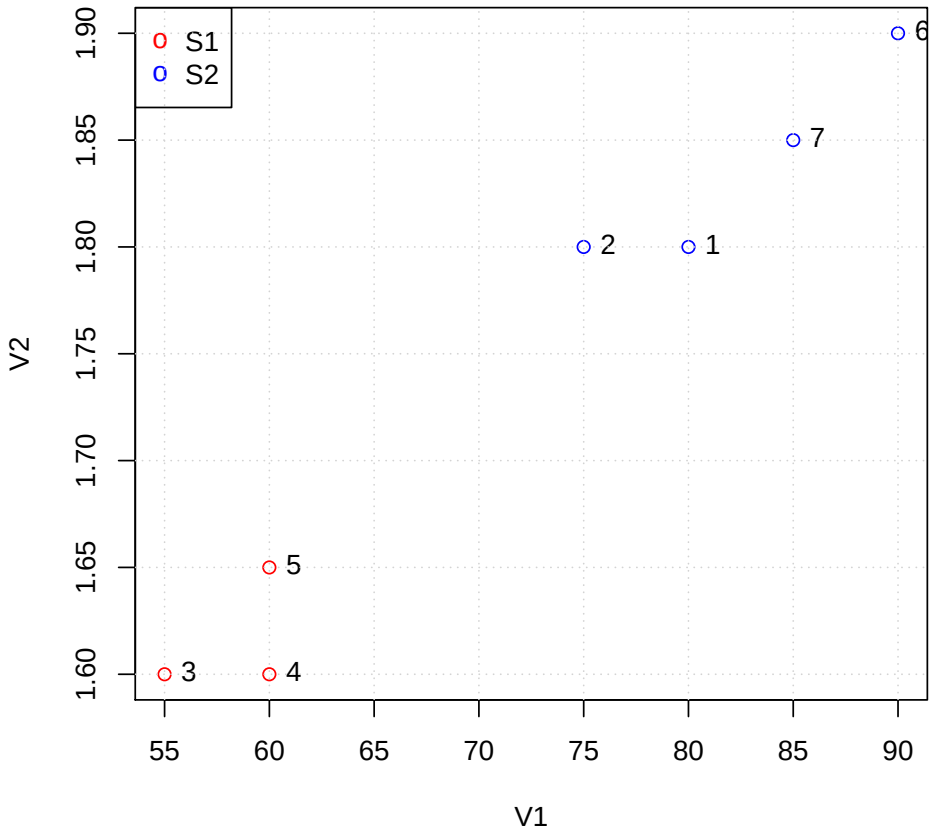
Vocabulaire : Attention aux différences français/anglais suivantes :

FRANÇAIS	ANGLAIS
Classification automatique	Clustering
Discrimination	Classification (supervised classification)

1.3 Exemple introductif

— $n = 7$, $\Omega = \{1, 2, ..., 7\}$, ensemble de patients ;
— $p = 2$, V_1 : poids (en kg) ; V_2 : taille (en m).

	V_1	V_2
1	80	1.8
2	75	1.8
3	55	1.6
4	60	1.6
5	60	1.65
6	90	1.9
7	85	1.85



Partition "idéale" : $\{S_1, S_2\}$ avec $S_1 = \{3, 4, 5\}$ et $S_2 = \{1, 2, 6, 7\}$.

1.4 Mesures de distance usuelles

On rappelle qu'une distance sur $\mathbb{R}^p$ est une fonction de $\mathbb{R}^p \times \mathbb{R}^p$ dans $\mathbb{R}^+$ vérifiant :

(i) $d(\vec{x},\vec{y}) = d(\vec{y},\vec{x}) \ \forall \ (\vec{x},\vec{y}) \in \mathbb{R}^p \times \mathbb{R}^p$;

(ii) $d(\vec{x},\vec{x}) = 0 \ \forall \ \vec{x} \in \mathbb{R}^p$;

(iii) $d(\vec{x},\vec{y}) \leq d(\vec{x},\vec{z}) + d(\vec{z},\vec{y}) \ \forall \ (\vec{x},\vec{y},\vec{z}) \in \mathbb{R}^p \times \mathbb{R}^p \times \mathbb{R}^p$.

En classification automatique, on confond un individu de Ω avec sa description qui est un vecteur de $\mathbb{R}^p$. Pour cette raison, les distances entre individus que l'on va considérer doivent être définies comme des fonctions de $\mathbb{R}^p \times \mathbb{R}^p$ dans $\mathbb{R}^+$.

La distance la plus connue est la distance Euclidienne et s'obtient à partir de la norme L_2. Soient $\vec{x},\vec{y} \in \mathbb{R}^p$. La distance Euclidienne entre $\vec{x}$ et $\vec{y}$ est définie par :

$$d_2(\vec{x},\vec{y}) = \sqrt{\sum_{i=1}^{p}(x_i - y_i)^2}.$$

La distance, entre $\vec{x} \in \mathbb{R}^p$ et $\vec{y} \in \mathbb{R}^p$, correspondant à la norme L_1, dite *de Manhattan*, est définie par :

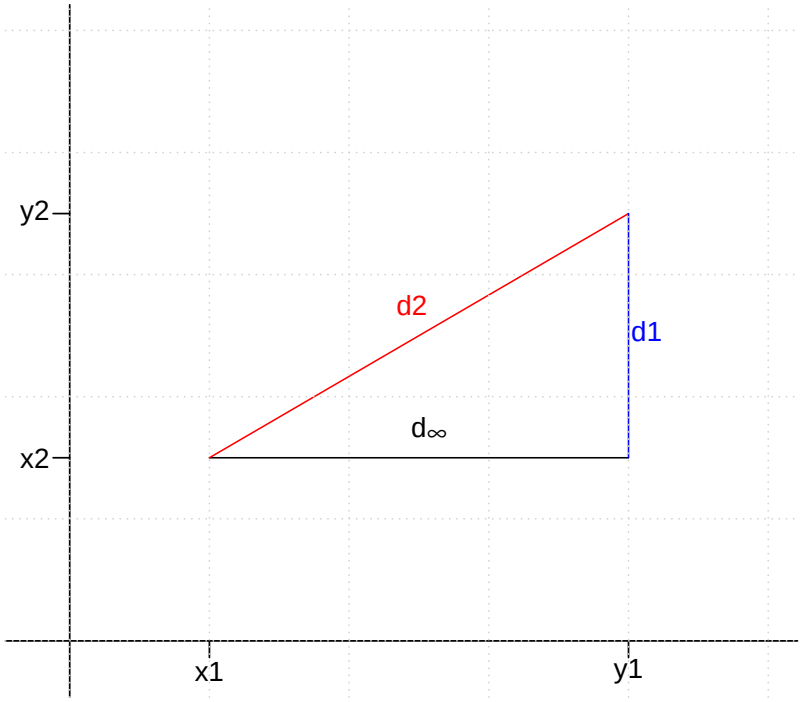
$$d_1(\vec{x},\vec{y}) = \sum_{i=1}^{p}|x_i - y_i|.$$

La distance, entre $\vec{x} \in \mathbb{R}^p$ et $\vec{y} \in \mathbb{R}^p$, correspondant à la norme L_∞, dite *du maximum*, est définie par :

$$d_\infty(\vec{x},\vec{y}) = \max_{i\in\{1,\ldots,p\}}|x_i - y_i|.$$

Exercice 1. *Vérifier que d_1, d_2 et d_∞ sont des distances sur $\mathbb{R}^p$.*
Indication : *pour d_2, utiliser l'inégalité de Cauchy-Schwarz.*

1.5 Lien entre les 3 distances



Exercice 2. *Vérifier que* $d_\infty(\vec{x},\vec{y}) \leq d_2(\vec{x},\vec{y}) \leq d_1(\vec{x},\vec{y})$ *pour tout* $\vec{x},\vec{y} \in \mathbb{R}^p$.

1.6 Retour sur l'exemple introductif

On peut vérifier que :

- $d_1(3,2) = |55-75| + |1.6-1.8|$;
- $d_2(3,2) = \sqrt{(55-75)^2 + (1.6-1.8)^2}$;
- $d_\infty(3,2) = \max\{|55-75|, |1.6-1.8|\}$.

On voit que les échelles de mesure sur lesquelles sont définies les variables ont une très forte influence sur les distances. La taille, exprimée en m, contribue très peu relativement au poids, exprimé en kg.

1.7 Importance du centrage-réduction

Pour résoudre les problèmes liés à des différences d'échelles des variables, on peut utiliser la technique du centrage-réduction. Par exemple, considérons les données suivantes extraites d'un tableau de données plus volumineux :

Individu	P (poids en kg)	T (taille en m)
1	70	1.9
2	60	1.6
3	90	1.8
4	60	1.7
5	80	1.8
$\vdots$	$\vdots$	$\vdots$
Moyenne	60	1.7
Ecart-type	10	0.1

Les données centrées-réduites correspondantes sont alors obtenues en considérant les variables suivantes :

$$P' = \frac{P - 60}{10} \qquad \text{and} \qquad T' = \frac{T - 1.7}{0.1}.$$

Les données centrées réduites sont ainsi :

Individu	P'	T'
1	1	2
2	0	-1
3	3	1
4	0	0
5	2	1
$\vdots$	$\vdots$	$\vdots$

Les nouvelles variables P' et T' ont même moyenne (égale à 0) et même écart-type (égal à 1), et peuvent ainsi être vues comme étant définies sur une même échelle de mesure (abstraite).

Pour de nombreux jeux de données, on peut considérer que les variables centrées-réduites prennent leurs valeurs dans $[-5, 5]$ et les valeurs extrêmes par rapport à la moyenne sont celles < -3 ou > 3. Cette remarque est très liée à la distribution normale mais elle fournit généralement de bonnes intuitions.

1.8 Influence sur le calcul des distances

Sur les données centrées-réduites, on obtient :

— $d_1(3,2) = |0-3| + |-1-1|$;
— $d_2(3,2) = \sqrt{(0-3)^2 + (-1-1)^2}$;
— $d_\infty(3,2) = \max\{|0-3|, |-1-1|\}$.

Soient deux individus i et j de Ω. Soient $\vec{x}$ et $\vec{y}$ leurs descriptions par rapport aux données initiales et soient $\vec{x'}$ et $\vec{y'}$ leurs descriptions par rapport aux données centrées-réduites.

On a alors :

$$\vec{x'} = \left(\frac{x_1 - 60}{10}, \frac{x_2 - 1.7}{0.1}\right) \qquad \text{et} \qquad \vec{y'} = \left(\frac{y_1 - 60}{10}, \frac{y_2 - 1.7}{0.1}\right).$$

Ainsi, relativement aux données initiales, on peut écrire :

$$d_1(i,j) = d_1(\vec{x}, \vec{y}) = |x_1 - y_1| + |x_2 - y_2|,$$

alors que, relativement aux données centrées-réduites, on trouve :

$$\begin{aligned} d_1(i,j) = d_1(\vec{x'}, \vec{y'}) &= \left|\frac{x_1 - 60}{10} - \frac{y_1 - 60}{10}\right| + \left|\frac{x_1 - 1.7}{0.1} - \frac{y_1 - 1.7}{0.1}\right| \\ &= \frac{1}{10}|x_1 - y_1| + \frac{1}{0.1}|x_2 - y_2|. \end{aligned}$$

En d'autres termes, calculer les distances avec d_1 sur les données centrées-réduites revient à pondérer les écarts aux niveaux des cordonnées par l'inverse des écarts-types des variables correspondantes. Un effet similaire se produit avec d_2 et d_∞.

1.9 Mesures de qualité d'une partition

Pour mesurer la qualité d'une partition, nous allons utiliser la notion d'inertie, très appropriée dans le cas considéré où toutes les variables sont quantitatives.

Définition 2. *Soit $S = \{\vec{x_1}, \vec{x_2}, ..., \vec{x_k}\}$ un ensemble de k vecteurs de $\mathbb{R}^p$. L'inertie de S est definie par :*

$$I_S = \frac{1}{k}\sum_{i=1}^{k} d_2^2(\vec{x_i}, \vec{g}),$$

où $\vec{g}$ centre de gravité de S.

Remarque : Pour $p = 1$ et $\vec{x}, \vec{y} \in \mathbb{R}^1$,

$$d_2(\vec{x}, \vec{y}) = \sqrt{(x_1 - y_1)^2} = |x_1 - y_1|.$$

Dans ce cas,

$$I_S = \frac{1}{k}\sum_{i=1}^{k}(x_{i1} - \bar{x}_1)^2 \qquad \text{ou} \qquad \bar{x}_1 = \frac{1}{k}\sum_{i=1}^{k} x_{i1}.$$

En d'autres termes, lorsque $p = 1$, l'inertie est la **variance empirique** des données.

Exercice 3. *Soient $p = 2$ et $k = 4$ avec $\vec{x_1} = (2, 3)$, $\vec{x_2} = (2.5, 3.5)$, $\vec{x_3} = (2, 3.5)$, $\vec{x_4} = (2.5, 3)$, $\vec{y_1} = (2, 4)$, $\vec{y_2} = (3, 5)$, $\vec{y_3} = (2, 5)$ et $\vec{y_4} = (3, 4)$. Calculer les inerties de $S = \{\vec{x_1}, \vec{x_2}, \vec{x_3}, \vec{x_4}\}$ et $S' = \{\vec{y_1}, \vec{y_2}, \vec{y_3}, \vec{y_4}\}$. Lequel des deux groupes est le plus homogène (resp. séparé) ? Faire une représentation graphique des deux groupes.*

1.9.1 Une mesure d'homogénéité : l'inertie intraclasse

Soit $P = \{S_1, ..., S_c\}$ une partition de Ω. Pour mesurer l'homogénéité de P, on va calculer la moyenne (pondérée) des inerties des classes qui la composent. C'est l'**inertie intraclasse**. Elle est définie par :

$$I_{intra}(P) = \sum_{i=1}^{c} \frac{|S_i|}{|\Omega|} I_{S_i},$$

où $|S_i|$ est le cardinal de S_i et $|\Omega|$ est le cardinal de Ω.

La quantité $I_{intra}(P)$ s'interprète donc comme l'**homogénéité moyenne** de la partition P.

1.9.2 Une mesure de séparation : l'inertie interclasse

Soit $P = \{S_1, ..., S_c\}$ une partition de Ω. L'inertie interclasse est (presque) l'inertie du nuage des c centres de gravité de $S_1, ..., S_c$. Plus précisément, l'inertie interclasse est définie par :

$$I_{inter}(P) = \sum_{i=1}^{c} \frac{|S_i|}{|\Omega|} d_2^2(\vec{g}, \vec{g_{S_i}}),$$

où $\vec{g}$ est le centre de Ω et $\vec{g_{S_i}}$ est le centre de gravité de la classe S_i, pour tout $i \in \{1, \ldots, c\}$.

1.9.3 Théorème de Koening-Huyghens

Théorème 1 (Théorème de Koening-Huyghens)**.** *Soit $P = \{S_1, ..., S_c\}$ une partition de Ω en c classes. Alors,*

$$I_{inter}(P) + I_{intra}(P) = I_\Omega,$$

où I_Ω, l'inertie de Ω, est appelée **inertie totale**.

Exercice 4. *Démontrer le résultat précédent.*

1.10 Conséquences pour la classification automatique

On considère deux partitions P_1 et P_2 de Ω en k classes. Un critère naturel pour comparer P_1 et P_2 serait de choisir la partition la plus *homogène*. Pour cela, on peut calculer $I_{intra}(P_1)$ et $I_{intra}(P_2)$, et renvoyer la partition avec la *plus petite inertie intraclasse*.

Un autre critère naturel serait de choisir la partition la plus *séparée*. Pour cela, on peut calculer $I_{inter}(P_1)$ et $I_{inter}(P_2)$, et renvoyer la partition avec la *plus grande inertie interclasse*.

Supposons que l'on ait trouvé que $I_{intra}(P_1) < I_{intra}(P_2)$. Cela est équivalent à $I_\Omega - I_{intra}(P_1) > I_\Omega - I_{intra}(P_2)$, puis à $I_{inter}(P_1) > I_{inter}(P_2)$ (d'après le théorème de Koening-Huyghens).

Ainsi, choisir la partition la plus homogène (en termes d'inertie intraclasse) revient automatiquement à choisir la partition la plus séparée (en terme d'inertie interclasse).

1.11 L'algorithme des k moyennes mobiles

Supposons k, le nombre de classes, fixé par l'utilisateur et que nous disposons d'une puissance de calcul infinie. Nous pourrions alors considérer l'algorithme suivant :

> Pour chaque partition P de Ω en k classes :
> Calculer $I_{intra}(P)$
> Fin pour
> Renvoyer une partition qui réalise le minimum

Un tel algorithme est **irréalisable** en pratique car le nombre de partitions en k classes d'un ensemble de n éléments (le nombre de Bell) croît à une vitesse exponentielle avec k et n. Même si l'on arrive à exécuter l'algorithme pour un n donné, il suffira que le nombre d'individus augmente de quelques unités pour que la puissance de calcul soit insuffisante. C'est pour cette raison que la classification automatique est fondée sur des algorithmes heuristiques.

L'un des algorithmes les plus connus est l'algorithme des k moyennes mobiles :

Initialisation : On génère k centres de classes fictifs.

Itération :

(i) Allouer chaque individu au centre le plus proche au sens de d_2. On obtient ainsi une partition de Ω en k classes (ou moins si l'une des classes devient vide).

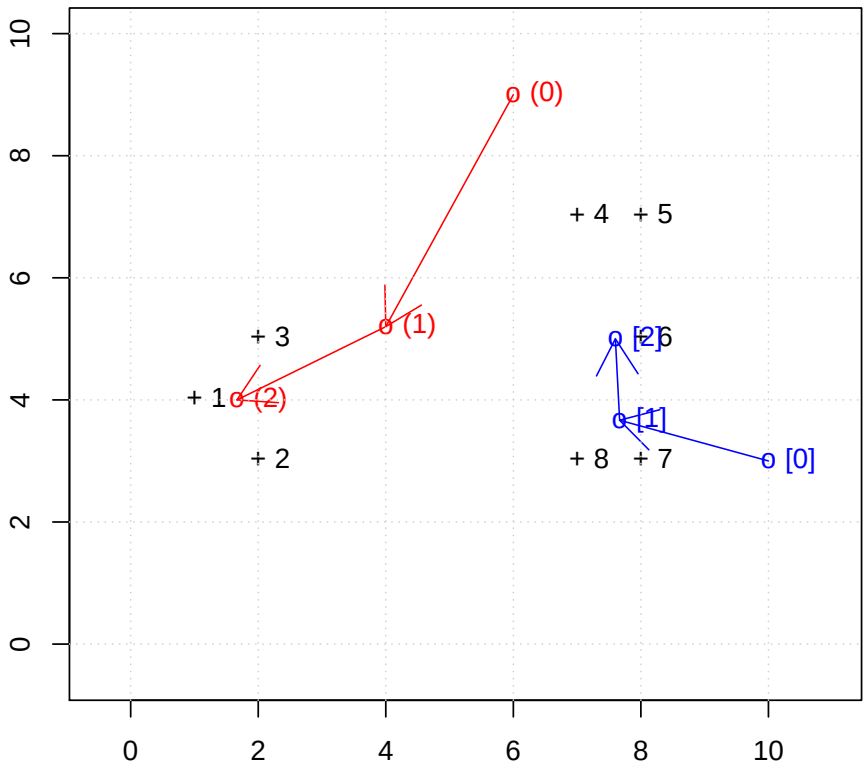
(ii) Calculer le centre de gravité de chaque classe, qui devient le nouveau centre de la classe.

Répéter (i) et (ii) jusqu'à ce que l'inertie interclasse ne croisse plus (resp. l'intraclasse ne décroisse plus), c'est-à-dire, jusqu'à stabilisation des centres.

Exemple : $p = 2$, $n = |\Omega| = 8$ et $k = 2$.

```
> ## Les données
> x <- c(1, 2, 2, 7, 8, 8, 8, 7)
> y <- c(4, 3, 5, 7, 7, 5, 3, 3)
> data.frame(V1 = x, V2 = y)
```

	V1	V2
1	1	4
2	2	3
3	2	5
4	7	7
5	8	7
6	8	5
7	8	3
8	7	3



Itération 1

— Classe bleue $[\cdot]$: $\{6, 7, 8\}$

— Classe rouge $(\cdot)$: $\{1, 2, 3, 4, 5\}$

Itération 2

— Classe bleue $[\cdot]$: $\{4, 5, 6, 7, 8\}$

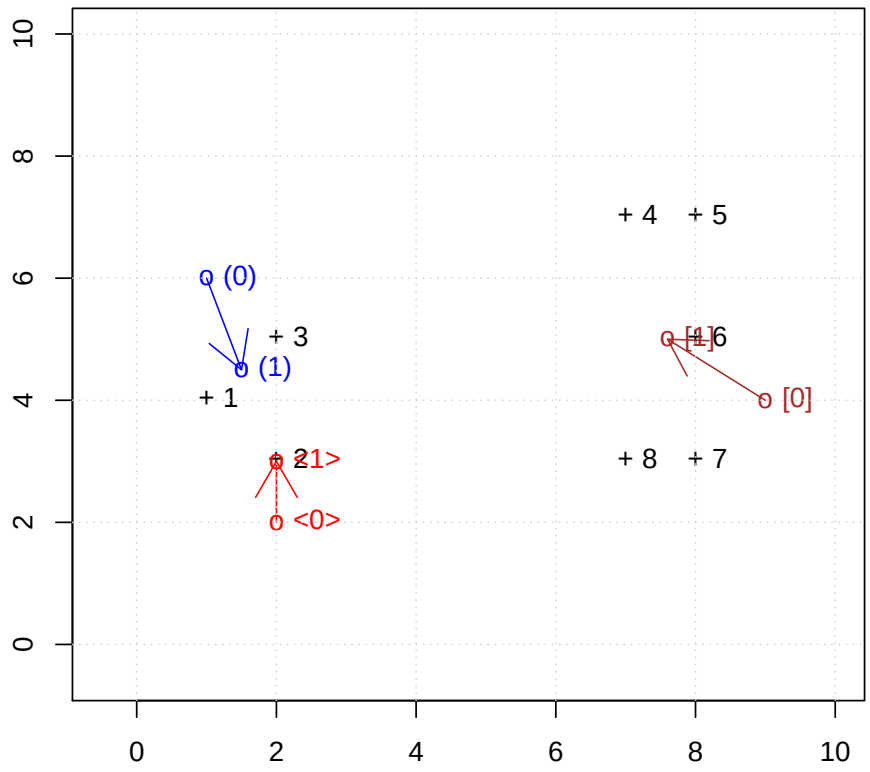
— Classe rouge $(\cdot)$: $\{1, 2, 3\}$

Itération 3

— Classe bleue $[\cdot]$: $\{4, 5, 6, 7, 8\}$
— Classe rouge $(\cdot)$: $\{1, 2, 3\}$

Dans R, la fonction permettant de réaliser l'algorithme s'appelle `kmeans()`.

L'étape d'initialisation peut conduire à une "mauvaise" partition même si cela arrive très rarement en pratique. Par exemple, avec $k = 3$:



Itération 1

— Classe marron $[\cdot]$: $\{4, 5, 6, 7, 8\}$
— Classe rouge $< \cdot >$: $\{2\}$
— Classe bleue $(\cdot)$: $\{1, 3\}$

Itération 2

— Classe marron $[\cdot]$: $\{4, 5, 6, 7, 8\}$
— Classe rouge $< \cdot >$: $\{2\}$
— Classe bleue $(\cdot)$: $\{1, 3\}$

On pourrait considérer qu'une partition plus "naturelle" en trois classes est $\{\{1, 2, 3\}, \{4, 5\}, \{6, 7, 8\}\}$

plutôt que celle trouvée. Une deuxième exécution de l'algorithme pourrait d'ailleurs conduire à ce résultat.

En pratique, pour éviter ce genre de résultats "instables", l'algorithme est exécuté plusieurs fois et on garde les partitions avec la plus petite inertie intraclasse (c'est-à-dire, avec la plus grande inertie interclasse).

1.12 Classification ascendante hiérarchique

Nous nous limitons, encore une fois, au cas où toutes les variables sont quantitatives. Cela dit, les principes restent les mêmes si les variables sont qualitatives ou consistent en un mélange quantitatif / qualitatif.

Les principes de la classification ascendante hiérarchique sont les suivants :

1. On part de la partition de Ω dans laquelle chaque individu forme une classe ;
2. À chaque itération, on regroupe 2 classes parmi celles qui sont les plus proches, jusqu'à obtention d'une seule classe, c'est-à-dire, de la partition $\{\Omega\}$.

L'algorithme de la CAH produit ainsi une suite de partitions emboîtées. Il existe ensuite des stratégies pour choisir une de ces partitions. Les deux ingrédients de la CAH sont :

(i) Une distance entre individus, c'est-à-dire, une distance sur Ω / $\mathbb{R}^p$.

(ii) Une méthode pour calculer la "distance" entre 2 classes (groupes, sous-ensembles). Cela s'appelle un **lien** entre classes.

Concernant (ii), nous allons définir les cinq liens les plus couramment utilisés.

Soient $A, B \subset \Omega$, $A \cap B = \varnothing$:

Lien simple :

$$d_S(A,B) = \min_{i\in A, j\in B} d(i,j).$$

Lien complet :

$$d_C(A,B) = \max_{i\in A, j\in B} d(i,j).$$

Lien moyen :

$$d_M(A,B) = \frac{1}{|A||B|} \sum_{i\in A, j\in B} d(i,j).$$

Deux autres liens sont particulièrement adaptés aux données quantitatives :

Distance entre barycentres/centroïdes :

$$d_B(A,B) = d(\vec{g_A}, \vec{g_B}),$$

où $\vec{g_A}$ est le centre de gravité de A et $\vec{g_B}$ est celui de B.

Saut de Ward :

$$d_W(A,B) = \frac{|A||B|}{|A|+|B|} d_2^2(\vec{g_A}, \vec{g_B}).$$

Algorithme de la CAH :

Choisir une distance entre individus et un lien entre classes.

Initialisation : Chaque individu forme une classe. Calculer les distances entre classes.

Itération : Itérer les 2 étapes suivantes jusqu'à l'obtention de la partition en une seule classe, c'est-à-dire, $\{\Omega\}$.

(i) Regrouper 2 classes parmi celles qui sont les plus proches (au sens du lien choisi).

(ii) Mettre à jour le tableau des distances entre classes (on remplace les 2 classes regroupées par une seule et on recalcule sa distance aux autres classes).

Exercice 5. *On considère* $\Omega = \{1, 2, 3, 4, 5\}$, $p = 2$ *et les données suivantes :*

Individus	V_1	V_2
1	*1*	*1*
2	*1.5*	*1.5*
3	*3*	*3*
4	*3.5*	*3.5*
5	*3*	*3.5*

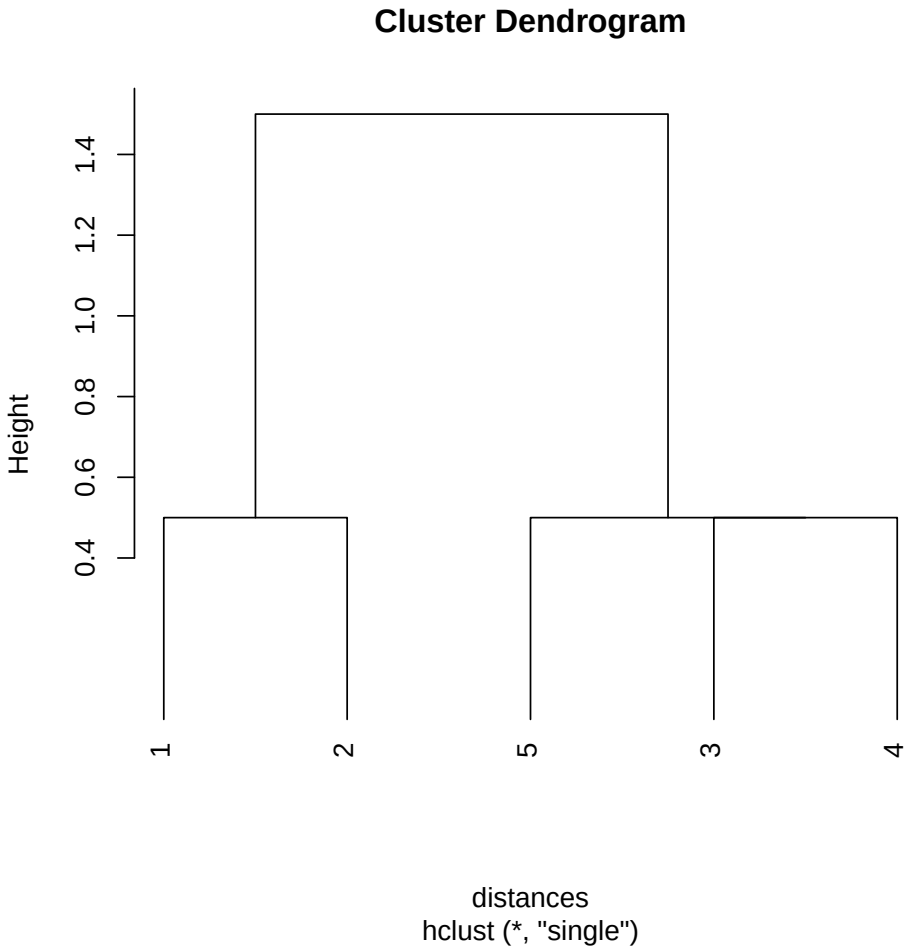
Effectuer la CAH des individus de Ω *en utilisant* d_∞ *comme distance et* d_S *comme lien.*

Le résultat de la CAH est une suite de partitions emboîtées. Par exemple, pour l'exercice précédent :

1. $\{\{1\}, \{2\}, \{3\}, \{4\}, \{5\}\}$ (l'inertie intraclasse vaut 0 ; partition "la plus fine") ;
2. $\{\{1, 2\}, \{3\}, \{4\}, \{5\}\}$;
3. $\{\{1, 2\}, \{3, 4\}, \{5\}\}$,
4. $\{\{1, 2\}, \{3, 4, 5\}\}$;
5. $\{\{1, 2, 3, 4, 5\}\}$ (partition "triviale").

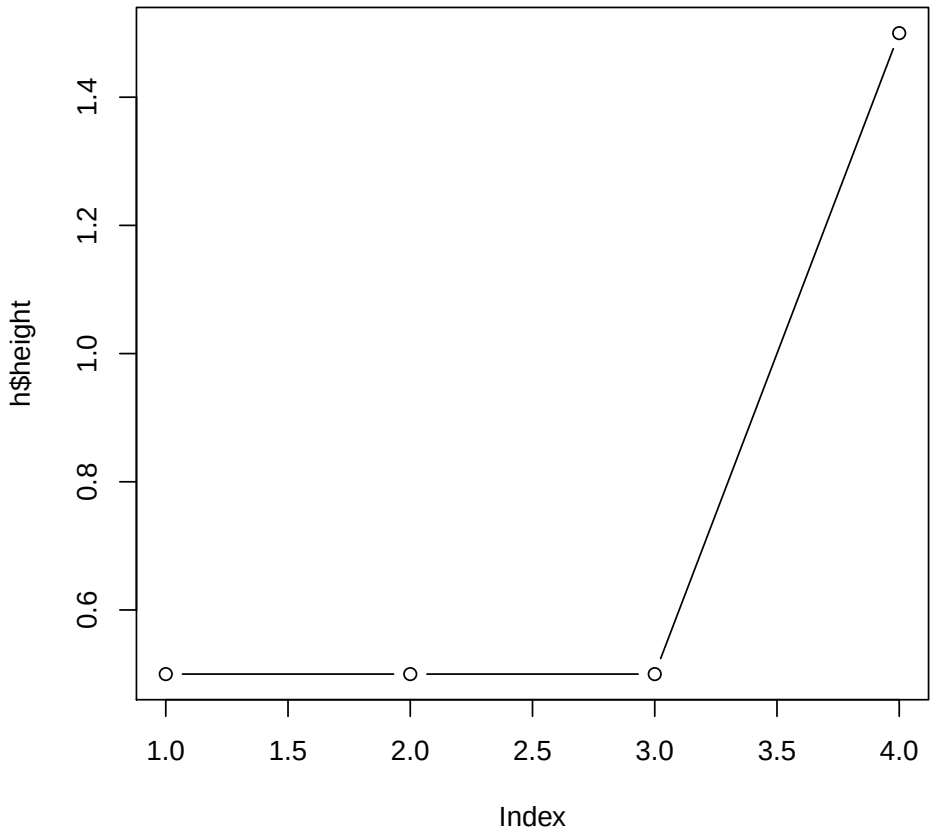
Le résultat de la CAH est représenté à l'aide d'un diagramme de classes appelé *dendrogramme*.

```
> ## Données
> V1 <- c(1, 1.5, 3, 3.5, 3)
> V2 <- c(1, 1.5, 3, 3.5, 3.5)
> d <- data.frame(V1, V2)
> ## Distances
> distances <- dist(d, method = "maximum")
> ## CAH
> h <- hclust(distances, method = "single")
> ## Dendrogramme
> plot(h, hang = -1)
```



Remarque : Si on coupe le dendrogramme en dessous de la "hauteur" 0.5, on obtient la partition de l'itération 1, c'est-à-dire, $\{\{1\},\{2\},\{3\},\{4\},\{5\}\}$. Si on coupe entre 0.5 et 1.5, on trouve la partition de l'itération 3, c'est-à-dire, $\{\{1,2\},\{3,4,5\}\}$.

Pour choisir l'une des partitions formées durant la CAH, on peut utiliser l'heuristique suivante : on cherche un "coude" dans le "graphe des hauteurs".



Exercice 6. *On considère $n = |\Omega| = 5$, $p = 2$ et les données suivantes :*

	V1	*V2*
1	*1*	*2*
2	*10*	*11*
3	*2*	*3*
4	*11*	*12*
5	*12*	*13*

a) Effectuer une CAH avec d_2 et le saut de Ward.

b) Dessiner le dendrogramme et le graphe des hauteurs correspondant.

c) Choisir une des partitions compatibles avec le dendrogramme à partir du graphe des hauteurs.

Une propriété importante du saut de Ward : Parmi toutes les partitions qui pourraient être formées à l'issue de l'itération courante, le saut de Ward forme celle qui correspond à

la plus faible augmentation de l'inertie intraclasse. Par exemple, pour l'exercice précédent, après l'itération 2, on a la partition $P_2 = \{\{1,3\},\{2\},\{4,5\}\}$ et on pourrait former $P_3' = \{\{1,2,3\},\{4,5\}\}$, $P_3'' = \{\{1,3\},\{2,4,5\}\}$ ou $P_3''' = \{\{2\},\{1,3,4,5\}\}$.

L'utilisation du saut de Ward garantit que :

$$I_{intra}(P_3'') - I_{intra}(P_2) \leq I_{intra}(P_3') - I_{intra}(P_2)$$

$$I_{intra}(P_3'') - I_{intra}(P_2) \leq I_{intra}(P_3''') - I_{intra}(P_2)$$

1.13 Combinaison de la CAH et des k moyennes mobiles

Les deux méthodes ont leurs avantages et inconvénients :

- la CAH ne nécessite pas la donnée du nombre de classes ; en revanche, c'est une méthode coûteuse en temps de calcul qui ne peut donc pas être exécutée sur de gros volumes de données ;
- les k moyennes mobiles sont beaucoup plus rapides mais nécessitent la donnée du nombre de classes k.

En combinant les deux méthodes, on obtient un bon compromis :

1. on exécute les k moyennes mobiles en demandant un grand nombre de classes (par exemple $k = 5\%$ ou 10% de n) ;
2. sur les centres de gravités des classes obtenues, on effectue une CAH (avec saut de Ward, par exemple) ce qui permet de déterminer un nombre de classes à garder ;
3. on exécute finalement les k moyennes mobiles sur toutes les données en utilisant le nombre de classes déterminé en 2.

Chapitre 2

Introduction aux arbres de décision

Ce chapitre est fondé sur le cours d'Intelligence Artificielle de François Denis et Rémi Gilleron, Université Lille 3.

2.1 Introduction

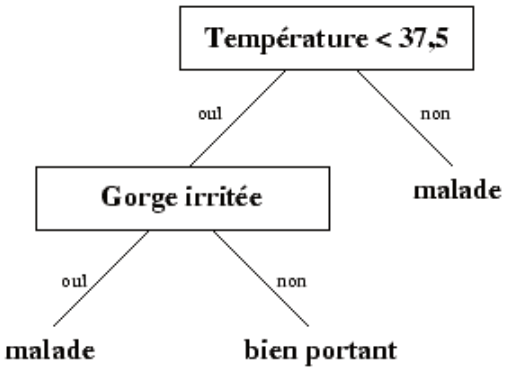
Les **arbres de décision** font partie des méthodes d'**apprentissage supervisé**, i.e., ils permettent de modéliser la "relation" qui existe entre un ensemble de variables **explicatives** et une variable **à expliquer**. Bien que la variable à expliquer puisse être de tout type, nous nous limiterons dans cette introduction au cas où elle est qualitative.

Les arbres de décision représentent **graphiquement un ensemble de règles** et sont **aisément interprétables**. Pour les arbres de grande taille, la procédure globale d'apprentissage peut être difficile à appréhender. Cependant, l'arbre reste **toujours compréhensible**.

Les algorithmes d'apprentissage par arbres de décision sont **efficaces**, **disponibles** dans la plupart des environnements de fouille de données (SAS, R, Rattle, Weka, etc).

2.2 Exemple introductif

— Les données correspondent à un ensemble de patients.
— Il y a deux **classes :** "malade" et "bien portant".
— Les patients sont décrits par deux **variables** : **température** (variable continue) et **gorge irritée** (variable binaire ou booléenne).



2.3 Généralités

Un arbre de décision est un arbre au sens informatique du terme. Les noeuds d'un arbre seront **repérés** par des positions qui sont des mots sur $\{1, ..., p\}^*$, où p est l'arité maximale des noeuds.

Pour l'exemple :

- e est étiqueté par le test **Température < 37,5**,
- 1 est étiqueté par le test **Gorge irritée**,
- 2 est étiqueté par la feuille **malade**,
- 11 est étiqueté par la feuille **malade**,
- 12 est étiqueté par la feuille **bien portant**.

Les noeuds internes sont appelés **noeuds de décision**. Un tel noeud est étiqueté par un test qui peut être appliqué à toute description des objets étudiés. En général, chaque test examine la valeur d'une unique variable. Les **réponses possibles** au test correspondent aux labels des arcs issus de ce noeud.

Les feuilles sont étiquetées par une modalité (classe) de la variable à expliquer appelée **classe par défaut**.

Un arbre de décision est la **représentation graphique** d'une **procédure de discrimination** : à toute description complète est associée une seule feuille de l'arbre de décision. Cette association est définie en commençant à la **racine** de l'arbre et **en descendant** dans l'arbre selon les réponses aux tests qui étiquettent les noeuds internes. La classe associée est alors la classe par défaut associée à la feuille qui correspond à la description.

La procédure de classification obtenue a une **traduction** immédiate en terme de **règles de production**.

Attention : les systèmes de production obtenus sont particuliers car l'ordre dans lequel on examine les variables est fixé et les règles de décision sont **mutuellement exclusives**.

Pour l'exemple : un patient ayant une température de 39 et ayant la gorge non irritée sera classé comme malade par cet arbre.

La traduction de cet arbre en règles de production est :
- SI Température < 37,5 ET gorge irritée ALORS malade
- SI Température < 37,5 ET NON(gorge irritée) ALORS bien portant
- SI NON(Température < 37,5) ALORS malade

2.4 Notations

Étant donnés un **échantillon** S (les données ou exemples d'apprentissage), un **ensemble de classes** $1, ..., c$ (les modalités de la variable à expliquer) et un arbre de décision t, à chaque position p de t correspond un **sous-ensemble de l'échantillon** qui est l'ensemble des exemples qui satisfont les tests de la racine jusqu'à cette position.

Définitions : Pour toute position p de t :

- $N(p)$ est le cardinal de l'ensemble des exemples associé à p,
- $N(k/p)$ est le cardinal de l'ensemble des exemples associé à p qui sont de classe k,
- $P(k/p) = N(k/p)/N(p)$ la proportion d'éléments de classe k à la position p.

Exercice 7. *On dispose d'un échantillon de 200 patients. On sait que 100 sont malades et 100 sont bien portants. La répartition entre les deux classes M (pour "malade") et S (pour "bien portant") est donnée par :*

	gorge irritée	*gorge non irritée*
température < 37,5	*(6 S, 37 M)*	*(91 S, 1 M)*
température >= 37,5	*(2 S, 21 M)*	*(1 S, 41 M)*

Vérifier que l'on a alors : N(11)=43 ; N(S/11)=6 ; N(M/11)=37 ; P(S/11)=6/43 et P(M/11)=37/43.

2.5 Algorithmes d'apprentissage : exemple introductif

Exemple simple : Une banque dispose des informations suivantes sur un ensemble de clients :

client	M	A	R	E	I
1	moyen	moyen	village	oui	oui
2	élevé	moyen	bourg	non	non
3	faible	âgé	bourg	non	non
4	faible	moyen	bourg	oui	oui
5	moyen	jeune	ville	oui	oui
6	élevé	âgé	ville	oui	non
7	moyen	âgé	ville	oui	non
8	faible	moyen	village	non	non

— La variable ternaire M décrit la **moyenne des montants** sur le compte client.
— La seconde variable ternaire A donne la **tranche d'âge** du client.
— La troisième variable ternaire R décrit la **localité de résidence** du client.
— La dernière variable binaire E a la valeur **oui** si le client a un **niveau d'études supérieures**.
— La classe associée à chacun de ces clients correspond au contenu de la colonne I. La classe **oui** correspond à un client qui effectue une **consultation de ses comptes bancaires en utilisant Internet**.

Objectif : obtenir un arbre de décision qui soit capable de dire, **pour un nouveau client** dont on connaît les valeurs des variables M (montant), A (âge), R (résidence) et E (études), si ce client va effectuer des consultations de ses comptes par Internet.

Objectif reformulé : à partir des données historiques, construire un arbre de décision qui **classifie** les clients.

Principe : construction descendante, i.e., lorsqu'un test est choisi, on divise l'ensemble d'apprentissage pour chacune des branches et on réapplique récursivement l'algorithme.

Sur l'exemple, initialisation avec l'arbre vide. L'échantillon contient 8 éléments : 3 de classe oui, 5 de classe non. À la racine de l'arbre qui n'est étiqueté par aucun test, l'échantillon peut être caractérisé par le couple (3,5).

Question : Ce noeud est-il **terminal**, i.e., est-il nécessaire de rechercher un test qui **discrimine** de façon plus intéressante l'échantillon ?

On attribuerait une feuille si nous étions dans le cas (0,8), c'est-à-dire, si **aucun client n'utilise Internet**. Ce n'est pas le cas : il faut **choisir un test**. Quatre choix possibles :
— (3,5) → M (1,2) (2,1) (0,2)
— (3,5) → A (1,0) (2,2) (0,3)
— (3,5) → R (1,1) (1,2) (1,2)
— (3,5) → E (3,2) (0,3)

2.6 Algorithme d'apprentissage : mesures d'entropie

Il est nécessaire d'avoir des quantités qui permettent de **comparer les différents choix possibles**, i.e., fonctions qui permettent de **mesurer le degré de mélange** des exemples entre les différentes classes.

Une telle fonction doit vérifier la propriété suivante : elle doit prendre son **minimum lorsque tous les exemples sont dans une même classe** (le noeud est **pur**) et son **maximum** lorsque les **exemples sont équirépartis**.

Exemple : 8 éléments et 2 classes : une telle fonction devra prendre son minimum pour les couples (0,8) et (8,0), et son maximum pour le couple (4,4).

Il existe de nombreuses fonctions qui satisfont ces propriétés : ce sont les **mesures d'entropie**.

Les plus connues sont la fonction de **Gini** et la fonction de **Shannon**.

Soit p une position. Nous avons alors :

$$\begin{aligned} Shannon(p) &= -\sum_{k=1}^{c} P(k/p)\log_2 P(k/p), \\ Gini(p) &= 1 - \sum_{k=1}^{c} P(k/p)^2 \\ &= 2\sum_{k<k'} P(k/p)P(k'/p). \end{aligned}$$

Exercice 8. *Considérons le cas de deux classes et appelons x la proportion d'éléments de classe 1 en position p. Montrer que :*

- $Shannon(p) = -x\log_2 x - (1-x)\log_2(1-x)$ *prend ses valeurs dans l'intervalle* $[0,1]$*, a son minimum pour* $x = 0$ *et* $x = 1$ *qui vaut 0 et a son maximum pour* $x = 1/2$ *qui vaut 1 ;*
- $Gini(p) = 2x(1-x)$ *prend ses valeurs dans l'intervalle* $[0, 1/2]$*, a son minimum pour* $x = 0$ *et* $x = 1$ *qui vaut 0 et a son maximum pour* $x = 1/2$ *qui vaut 1/2 ;*
- *ces deux fonctions sont* **symétriques** *par rapport à* $x = 1/2$.

Pour notre exemple, considérons, par exemple, l'arbre construit à l'aide de la variable E :

- $Shannon(e) = -3/8\log_2 3/8 - 5/8\log_2 5/8 \approx 0.954$
- $Shannon(1) = -3/5\log_2 3/5 - 2/5\log_2 2/5 \approx 0.970$
- $Shannon(2) = -0/3\log_2 0/3 - 3/3\log_2 3/3 = 0$
- $Gini(e) = 2 \times 3/8 \times 5/8 \approx 0.469$
- $Gini(1) = 2 \times 3/5 \times 2/5 = 0.480$
- $Gini(2) = 2 \times 0/3 \times 3/3 = 0$

2.7 Algorithme d'apprentissage : fonction gain

Soit i la mesure d'entropie choisie. On définit une **fonction gain** par :

$$Gain(p,t) = i(p) - \sum_{j=1}^{n} P_j i(p_j),$$

où p désigne une position, t un test d'arité n et P_j est la proportion d'éléments de S à la position p qui vont en position p_j (qui satisfont la jème branche du test).

Si on considère pour i l'entropie de Shannon, le terme $i(p)$ représente l'**entropie actuelle du noeud p**, le deuxième terme de la différence représente l'**entropie espérée en introduisant le test** t qui est égale à la somme pondérée des entropies des nouveaux noeuds créés.

On souhaite obtenir les **entropies les plus faibles** possibles car, d'après les propriétés des mesures d'entropie, si l'entropie est faible, la plupart des éléments se trouvent dans une même classe. On cherche donc à obtenir le **gain maximum**.

Pour notre exemple :

- $Gain(e, M) = Shannon(e) - (3/8 Shannon(1) + 3/8 Shannon(2) + 2/8 Shannon(3)) = Shannon(e) - 0.688$
- $Gain(e, A) = Shannon(e) - (1/8 Shannon(1) + 4/8 Shannon(2) + 3/8 Shannon(3)) = Shannon(e) - 0.500$
- $Gain(e, R) = Shannon(e) - (2/8 Shannon(1) + 3/8 Shannon(2) + 3/8 Shannon(3)) = Shannon(e) - 0.939$
- $Gain(e, E) = Shannon(e) - (5/8 Shannon(1) + 3/8 Shannon(2)) = Shannon(e) - 0.607$

Le **gain maximal** ou encore l'**entropie espérée minimale** est obtenue pour le choix du test A (le choix du test R est très mauvais, ce qui correspond bien à l'intuition).

2.8 Algorithme d'apprentissage : principes

Idée centrale : Diviser récursivement et le plus efficacement possible les **exemples de l'ensemble d'apprentissage** par des tests définis à l'aide des variables jusqu'à ce que l'on obtienne des sous-ensembles d'exemples ne contenant (presque) que des exemples appartenant tous à une même classe.

Dans toutes les méthodes, on trouve les **trois opérateurs** suivants :

- **Décider si un noeud est terminal**, c'est-à-dire, décider si un noeud doit être étiqueté comme une feuille. Par exemple : tous les exemples sont dans la même classe, il y a moins d'un certain nombre d'erreurs, ...
- **Sélectionner un test à associer à un noeud.** Par exemple : aléatoirement, utiliser des critères statistiques, ...
- **Affecter une classe à une feuille.** On attribue la classe majoritaire sauf dans le cas où l'on utilise des fonctions coût ou risque.

2.9 Algorithme d'apprentissage générique

Algorithme d'apprentissage générique
entrée : langage de description ; échantillon S
début
    Initialiser à l'arbre vide ; la racine est le noeud courant
    répéter
        Décider si le noeud courant est terminal
        Si le noeud est terminal alors

Affecter une classe
sinon
Sélectionner un test et créer le sous-arbre
FinSi
Passer au noeud suivant non exploré s'il en existe
Jusqu'à obtenir un arbre de décision
fin

2.10 Algorithme d'apprentissage : performance

Avec un tel algorithme, on peut calculer un arbre de décision dont l'**erreur apparente** est faible, voire nulle. Un arbre de décision **parfait** est un arbre de décision tel que tous les exemples de l'ensemble d'apprentissage soient **correctement classifiés**.

Attention : Un tel arbre n'existe pas toujours, e.g., s'il existe deux exemples tels qu'à **deux descriptions identiques** correspondent **deux classes différentes**.

L'objectif est de construire un arbre d'erreur de classification la plus petite possible. Mais :
- l'**erreur apparente** est une vision **très optimiste** de l'**erreur réelle**,
- trouver un arbre de décision d'erreur apparente minimale est, en général, un **problème NP-complet**.

L'algorithme précédent recherche un "bon" arbre d'**erreur apparente faible** : l'algorithme procède de façon descendante sans jamais remettre en question les choix effectués. **MAIS** on ne peut jamais exclure qu'un autre choix de test conduise en fait à un meilleur arbre.

L'arbre construit est d'erreur apparente faible car les feuilles sont étiquetées de telle manière qu'il y ait peu d'erreurs. **MAIS**, il se peut que l'erreur réelle soit importante, c'est-à-dire que l'arbre construit soit bien adapté à l'échantillon mais ait un pouvoir de prédiction faible, i.e., l'arbre **"apprend bien"** mais **"généralise mal"**.

Il faut donc trouver un compromis entre **qualité de l'apprentissage** et **capacité de généralisation**. Idéalement, il faudrait trouver un critère qui permette d'arrêter la croissance de l'arbre au bon moment. Malheureusement, dans l'état actuel des recherches, un tel critère n'a pu être trouvé. De plus, le risque d'arrêter trop tôt la croissance de l'arbre est plus important que de l'arrêter trop tard. Par conséquent, les méthodes procèdent souvent en deux phases : **construction** puis **élagage**, i.e., dans une seconde phase, on élague l'arbre obtenu pour essayer de faire **diminuer l'erreur réelle** (élaguer un arbre consiste à en supprimer certains sous-arbres).

Les méthodes se distinguent donc les unes des autres par les choix des opérateurs, mais aussi par les méthodes d'élagage utilisées.

2.11 Méthodes existantes

Les deux méthodes à base d'arbres de décision les plus importantes sont :

- **CART** (Breiman et al. 1984) :
  - méthode développée par des statisticiens,
  - construction d'**arbres de décision binaires**,
  - peut être étendue pour traiter le cas de variables continues,
  - la **fonction de Gini** est utilisée pour associer un test à un noeud,
  - l'élagage de l'arbre se fait par estimation de l'erreur réelle en utilisant un **ensemble test**.
- **ID3** (Quinlan 1983),
  - méthode la plus connue,
  - améliorée en 93 par une nouvelle version **C4.5**,
  - pas de restriction à des variables binaires.
  - le choix du test associé à un noeud se fait à l'aide de la **fonction de Shannon**,
  - la méthode peut prendre en compte le cas où les valeurs de certaines variables sont non spécifiées (**valeurs manquantes**),
  - gère les variables continues,
  - choix entre **arbres et règles**,
  - l'élagage se fait sur l'arbre ou sur le système de règles et se base sur une estimation de l'erreur réelle à partir de l'ensemble d'apprentissage.

2.12 Conclusion

- Les arbres de décision donnent souvent de **bons résultats dans la pratique** (e.g. NASA).
- Les arbres de décision possèdent l'avantage d'être compréhensibles par tout utilisateur (si la taille de l'arbre produit est raisonnable), et d'avoir une **traduction** immédiate en terme de **règles de production**.
- Pour le système à base de règles induit, les règles sont mutuellement exclusives et l'ordre dans lequel sont examinées les variables est figé.
- Les **méthodes sont non optimales** : les arbres produits ne sont pas les meilleurs.
- En effet, les choix dans la construction des arbres, basés sur de nombreuses **heuristiques**, ne sont jamais remis en question (pas de retour en arrière (ou backtracking)).

Chapitre 3

Introduction à l'extraction de règles d'association

Cette brève introduction est fondée sur le cours de Ricco Rakotomalala, dispensé à l'Université de Lyon 2.

L'extraction de *règles d'association* est une méthode exploratoire de *données de transactions* dont le but est de découvrir des relations entre deux ou plusieurs variables présence/absence (binaires donc).

3.1 Données de transactions

Exemple introductif :

N° de transaction (caddie)	Contenu
1	bonbons, chips, soda, pâtes
2	pain, beurre
3	chips, bière
4	yaourts, viande
⋮	⋮

Remarques :

- Dans cette introduction sur les règles d'association, on ne tiendra pas compte de la quantité d'un produit dans un caddie mais uniquement de la présence/absence de produits.
- La liste des produits est immense.
- Le nombre de produits différents est très variable d'un caddie à l'autre.

Objectif des règles d'association :

1. Mettre en évidence les produits achetés ensemble.
2. Transcrire cette connaissance sous forme de règles.

Règle d'association : SI antécédent ALORS conséquent.

Une autre représentation des données est obtenue en utilisant le *codage disjonctif complet*. Par exemple, avec 4 produits $p_1, \ldots, p_4$:

N° caddie	p_1	p_2	p_3	p_4
1	1	1	1	0
2	1	0	1	0
3	0	1	1	0
4	1	1	1	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Remarque : Dès que l'on peut se ramener à un tableau "individus x variables" avec des variables binaires (présence/absence), on peut utiliser les méthodes d'extraction de règles d'association. Il s'agira typiquement alors de détecter des co-occurrences de certaines associations "variable = modalité".

3.2 Critères d'évaluation de règles d'association

Il y a deux critères principaux : le *support* et la *confiance*.

Exemple :

Caddie	p_1	p_2	p_3	p_4
1	1	1	1	0
2	1	0	1	0
3	1	1	1	0
4	1	0	1	0
5	0	1	1	0
6	0	0	0	1

Soit la règle R1 : SI p_1 ALORS p_2.

Le *support* est un indicateur de fiabilité de la règle :

$$supp(R1) = \#\text{ caddies contenant } p_1 \text{ et } p_2$$

(version *absolue*) ou

$$supp(R1) = \frac{\#\text{ caddies contenant } p_1 \text{ et } p_2}{\#\text{ total de caddies}}$$

(version *relative*).

Pour l'exemple :

$$supp(R1) = 2 \text{ (version absolue)}$$

ou

$$supp(R1) = \frac{2}{6} \text{ (version relative).}$$

La *confiance* est une estimation de la précision de la règle :

$$conf(R1) = \frac{supp(R1)}{supp(\text{antécédent de R1})}.$$

Pour l'exemple :

$$conf(R1) = \frac{supp(R1)}{supp(p_1)} = \frac{2}{4} = \frac{\frac{2}{6}}{\frac{4}{6}} = \frac{1}{2}.$$

Informellement, le support relatif est l'estimation d'une probabilité :

$$supp(R1) = \widehat{\Pr}\big(\text{(avoir un caddie qui contient) } p_1 \text{ et } p_2\big).$$

La confiance s'apparente alors à une probabilité conditionnelle estimée :

$$conf(R1) = \frac{\widehat{\Pr}(p_1 \text{ et } p_2)}{\widehat{\Pr}(p_1)} = \widehat{\Pr}(p_2 \mid p_1).$$

Une "bonne" règle d'association est une règle qui a au moins un support et une confiance élevés.

3.3 Algorithme d'extraction de règles d'association

Paramètres de l'algorithme :

- le support minimum (par exemple, 2 transactions) ;
- la confiance minimum (par exemple, 0.75).

L'idée des deux seuils ci-dessus est de limiter le nombres de règles produites.

Vocabulaire utilisé :

- *item* : produit ;
- *itemset* : ensemble de produits ;
- $supp(itemset)$: # de transactions qui contiennent tous les "items" de l' "itemset" ;
- $card(itemset)$: # de produits dans l' "itemset".

S'il y a n "items", il y a 2^n "itemsets" possibles en comptant l'ensemble vide $\varnothing$. Nous allons présenter l'algorithme d'extraction de règles sur l'exemple ci-dessus.

Étape 1 : Recherche des "itemsets" fréquents

On part de l'exemple avec 4 produits $p_1, \ldots, p_4$ et on fixe les deux seuils à 2 et 0.75, respectivement. La recherche d' "itemsets" fréquents est représentée ci-dessous par un *treillis*. Les "itemsets" satisfaisant la contrainte $supp(itemset) \geq 2$ sont soulignés.

Étape 2 : Recherche de règles d'association à partir des "itemsets" fréquents de cardinal $\geq$ à 2

- $\{p_1, p_2\}$: on peut former 2 règles :
  - $p_1 \rightarrow p_2$:$conf(p_1 \rightarrow p_2) = 0.5$ ✗
  - $p_2 \rightarrow p_1$: $conf(p_2 \rightarrow p_1) = 0.66$ ✗
- $\{p_1, p_3\}$: on peut former 2 règles :

- $p_1 \to p_3 : conf(p_1 \to p_3) = 1$ ✓
- $p_3 \to p_1 : conf(p_3 \to p_1) = 0.8$ ✓

— $\{p_2, p_3\}$: on peut former 2 règles :
- $p_2 \to p_3 : conf(p_2 \to p_3) = 1$ ✓
- $p_3 \to p_2 : conf(p_3 \to p_2) = 0.6$ ✗

— $\{p_1, p_2, p_3\}$: on peut former 6 règles :
- $p_1, p_2 \to p_3 : conf(p_1, p_2 \to p_3) = 1$ ✓
- $p_2, p_3 \to p_1 : conf(p_2, p_3 \to p_1) = 0.66$ ✗
- $p_1, p_3 \to p_2 : conf(p_1, p_3 \to p_2) = 0.5$ ✗
- $p_1 \to p_2, p_3$ ✗
- $p_2 \to p_1, p_3$ ✗
- $p_3 \to p_1, p_2$ ✗

L'algorithme présenté sur l'exemple ci-dessus est connu sous le nom d'*apriori*.

3.4 Un indicateur de pertinence des règles

Même avec des seuils élevés de support et confiance dans l'algorithme précédent, le nombre de règles générées est souvent très grand. Un des indicateurs permettant de filtrer les règles générées souvent utilisé après extraction est le *lift*.

Exemple : Que faut-il penser de la règle "SI cheveux=bruns ALORS cerveau=présent" ? Le support de cette règle (calculé à partir de données pour lesquelles les transactions correspondent, par exemple, à des patients) ne sera pas négligeable. La confiance de cette règle sera égale à 1.

Définition du *lift* :

$$lift(A \to C) = \frac{\widehat{\Pr}(C|A)}{\widehat{\Pr}(C)} = \frac{conf(A \to C)}{supp(C)}.$$

En combinant les formules précédentes, on obtient :

$$lift(A \to C) = \frac{\widehat{\Pr}(A \text{ et } C)}{\widehat{\Pr}(C)\widehat{\Pr}(A)}.$$

Ainsi, on voit que le calcul du lift revient à comparer la probabilité estimée de A et C avec la probabilité estimée de A et C sous hypothèse d'indépendance entre A et C.

En pratique, si le lift d'une règle est inférieur à 1, la règle n'a aucun intérêt.