Analyse de données

M2 Ingénierie Statistique et Numérique

Université de Lille

Année 2019-2020

Charlotte Baey

Table des matières

Introduction

L'analyse des données est un terme regroupant plusieurs méthodes permettant d'extraire ou de synthétiser de l'information contenue dans un jeu de données. En effet, face à la capacité toujours plus grande de stockage des données, il est de plus en plus fréquent pour le statisticien de se retrouver confronté à des jeux de données de grande dimension, dont il faudra extraire des informations pertinentes.

En fonction de l'objectif de l'étude, différentes méthodes pourront être utilisées, mais elle présentent en général des caractéristiques communes : il s'agit essentiellement de méthodes descriptives (même si certaines peuvent être utilisées comme outils prédictifs), nécessitant un traitement informatique qui peut être lourd et coûteux en temps de calcul, et qui constituent, comme techniques exploratoires, une étape préliminaire à un processus de décision.

Dans ce cours, nous nous intéresserons à plusieurs classes de méthodes. Tout d'abord, les méthodes d'analyse factorielle, qui permettent notamment de réduire la dimension d'un jeu de données, et parmi lesquelles on retrouve l'analyse en composantes principales (ACP). L'objectif de ce type d'analyse est de proposer une représentation du nuage de points initial dans un sous-espace de dimension plus petite, contenant le maximum d'information. Les différentes méthodes d'analyse factorielle que nous aborderons se distinguent par le type de données qu'elles permettent de traiter : variables quantitatives pour l'ACP, variables qualitatives pour l'analyse factorielle des correspondances et l'analyse des correspondances multiples.

Puis, nous présenterons des méthodes de classification, qui permettent quant à elles d'identifier des sous-groupes dans les données. On distingue alors deux grands types de méthodes de classification : la classification dite *supervisée* et la classification dite *non supervisée*. Dans les deux cas, on dispose d'un jeu de données pouvant être représenté sous la forme d'un tableau à n lignes (les individus) et à p colonnes (les variables mesurées sur chaque individu). Cependant, dans le cas de la classification supervisée on dispose en plus d'une variable de sortie pour chaque individu de la base de données, et on cherche à prédire la valeur de cette variable de sortie en fonction des variables d'entrées. Au contraire, dans le cas de la classification non supervisée toutes les variables jouent le même rôle, et on cherche donc à regrouper les individus en groupes uniquement en fonction des variables d'entrée.

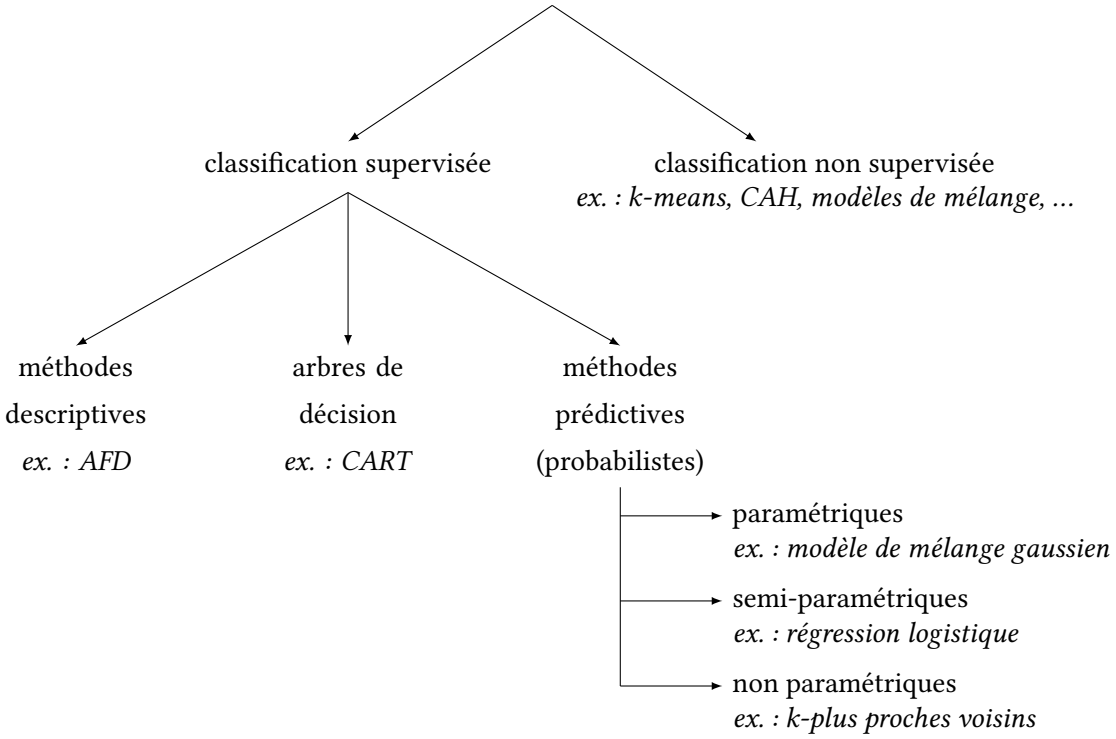


Figure 1 – Panorama des méthodes de classification abordées dans ce cours.

Exemple 1 : classification supervisée. On cherche à prédire la probabilité pour un patient de développer une maladie donnée. Pour cela, on dispose d'une base de données contenant un ensemble de variables socio-démographiques et cliniques mesurées ou observées chez des patients sains et malades, et on cherche à prédire la probabilité d'appartenir à chacun des groupes "sains" et "malades" en fonction des caractéristiques du patient.

Exemple 2 : classification non supervisée Une entreprise fournit à un cabinet de conseil en marketing une base de données contenant les informations personnelles ainsi que les préférences d'achat de ses clients. Elle demande alors au cabinet de conseil d'identifier des sous-groupes de consommateurs ayant des profils de consommation type, afin de mieux cibler ses campagnes de publicité et de fidélisation. Il s'agit ici de classification *non supervisée* car on va construire les groupes à partir des caractéristiques des clients. Ils ne sont donc pas connus *a priori.*

La figure 1 donne un panorama des différentes méthodes de classification qui seront abordées dans ce cours.

Ce polycopié est inspiré de plusieurs sources : les notes de cours de Christophe Biernacki, et les ouvrages de Celeux et Nakache (1994), Saporta (2006) et Hastie et al. *(2001).*

Première partie

Méthodes d'analyse factorielle

Introduction

Le point de départ d'une analyse factorielle est un jeu de données concernant n individus pour lesquels on détient des observations pour p variables. Ce jeu de données peut alors être représenté sous la forme d'un nuage de points de $\mathbb{R}^p$. Lorsque p est grand, il devient alors difficile de visualiser correctement ce nuage de points, et les méthodes d'analyses factorielles que nous allons détailler ici permettent alors de représenter le nuage de points dans un sous-espace de dimension plus petite, mais qui contient le maximum d'information.

En fonction de la nature des variables, on distingue plusieurs méthodes : l'analyse en composantes principales (ACP, voir chapitre 1) pour des variables quantitatives, l'analyse factorielle des correspondances (AFC, voir chapitre 2) et l'analyse des correspondances multiples (ACM, voir chapitre 3) pour des variables qualitatives.

Chapitre 1

Analyse en composantes principales (ACP)

Lorsque le nombre de variables observées est élevé, il devient difficile de proposer une représentation graphique des données et des relations existant entre les variables. L'objectif d'une analyse en composantes principales (ACP) est d'obtenir une représentation approchée d'un nuage de points dans un sous-espace de dimension plus faible tout en perdant le moins d'information possible. Pour cela, on va projeter le nuage de points sur un sous-espace affine, en veillant à obtenir une projection qui soit la plus proche possible du nuage initial, c'est-à-dire qui déforme le moins possible le nuage de points.

1.1 Notations

1.1.1 Matrice des observations

Le point de départ de l'ACP est un nuage de points de $\mathbb{R}^p$, chaque point correspondant à un individu pour lequel on a observé p variables quantitatives. On note x_{ij} la valeur de la j-ème variable mesurée sur le i-ème individu, avec $1 \leq i \leq n$ et $1 \leq j \leq p$, $X_j = (x_{1j}, \ldots, x_{nj})^t \in \mathbb{R}^n$ le vecteur contenant les observations de la variable j pour tous les individus, et $\mathbf{x}_i = (x_{i1}, \ldots, x_{ip})^t \in \mathbb{R}^p$ le vecteur des observations de l'individu i. Ces observations sont rassemblées dans une matrice notée $\mathbf{X}$, dont les colonnes correspondent aux vecteurs $X_i, 1 \leq i \leq n$:

$$\mathbf{X} = \begin{pmatrix} x_{11} & \ldots & x_{1p} \\ \vdots & \ddots & \vdots \\ x_{n1} & \ldots & x_{np} \end{pmatrix}. \tag{1.1}$$

On obtient donc en ligne, les observations de chaque individu, et en colonne, les observations de chaque variable.

1.1.2 Choix de l'origine et réduction des variables

Afin de faciliter les représentations graphiques du nuage de points, on choisit en général comme origine le centre de gravité du nuage. Le nuage se retrouve alors centré autour de cette nouvelle origine.

Pour cela, on définit le centre de gravité du nuage des individus par :

$$\mathbf{g} = \begin{pmatrix} \frac{1}{n}\sum_i x_{i1} \\ \vdots \\ \frac{1}{n}\sum_i x_{ij} \\ \vdots \\ \frac{1}{n}\sum_i x_{ip} \end{pmatrix}.$$

Autrement dit, le centre de gravité $\mathbf{g}$ est le vecteur dont la j-ème coordonnée g_j correspond à la valeur moyenne de la variable j sur les n individus. On se retrouve alors avec une version *centrée* de la matrice $\mathbf{X}$, que l'on note $\mathbf{X}_c$:

$$\mathbf{X}_c = \begin{pmatrix} x_{11} - g_1 & \dots & x_{1p} - g_p \\ \vdots & \vdots & \vdots \\ x_{n1} - g_1 & \dots & x_{np} - g_p \end{pmatrix}. \tag{1.2}$$

De même, il est souvent plus commode de travailler avec des variables réduites (c'est-à-dire dont l'écart-type vaut 1), afin de les rendre comparables entre elles et de gommer les effets d'échelle. Plus précisément, en notant $s_j^2 = \frac{1}{n}\sum_i (x_{ij} - g_j)^2$ la variance de la variable j, on divise les coordonnées de chaque colonne par l'écart-type correspondant. On se retrouve alors avec une version centrée et réduite de la matrice des observations, que l'on note $\tilde{\mathbf{X}}$:

$$\tilde{\mathbf{X}} = \begin{pmatrix} \frac{x_{11}-g_1}{s_1} & \dots & \frac{x_{1p}-g_p}{s_p} \\ \vdots & \vdots & \vdots \\ \frac{x_{n1}-g_1}{s_1} & \dots & \frac{x_{np}-g_p}{s_p} \end{pmatrix}. \tag{1.3}$$

N.B | Dans la suite, on supposera la matrice $\mathbf{X}$ centrée réduite.

1.1.3 Espace des individus et espace des variables

Deux points de vue peuvent être adoptés pour l'étude du nuage de points :

— on peut associer à chaque individu i le vecteur $\mathbf{x}_i$ contenant ses observations sur les p variables (i.e. la i-ème ligne de $\mathbf{X}$). Ce vecteur appartient à un espace vectoriel de dimension p : c'est l'*espace des individus*.

— on peut associer à chaque variable j le vecteur X_j contenant les observations de la variable j (i.e. la j-ème colonne de $\mathbf{X}$). Ce vecteur appartient à un espace vectoriel de dimension n : c'est l'*espace des variables*.

1.2 Choix d'une distance

L'étape suivante consiste à introduire une distance entre les points de chacun des deux espaces définis ci-dessus. En ACP, on utilise par convention la distance euclidienne classique. Autrement dit, la

distance entre deux individus représentés par les vecteurs $\mathbf{x}_i$ et $\mathbf{x}_{i'}$ est donnée par :

$$d^2(\mathbf{x}_i, \mathbf{x}_{i'}) = \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2. \tag{1.4}$$

Le choix de la distance euclidienne permet de donner le même poids à chacune des variables. Cependant, il est possible de considérer une métrique définie par une matrice symétrique définie positive M de la façon suivante : $d^2_M(\mathbf{x}_i, \mathbf{x}_{i'}) = (\mathbf{x}_i - \mathbf{x}_{i'})^t M(\mathbf{x}_i - \mathbf{x}_{i'})$. La distance euclidienne classique correspond donc au cas $M = Id_p$.

N.B. Choisir la métrique $M = Id_p$ après réduction des variables (i.e. après avoir divisé chacune des variables par leur écart-type), revient à choisir la métrique $M = \text{diag}(s_1^2, \ldots, s_p^2)$ sans réduction des variables, où s_j est l'écart-type de la variable j.

1.3 Inertie

On définit ensuite la notion *d'inertie* :

$$\begin{aligned} I_T &= \frac{1}{n} \sum_{i=1}^{n} d^2(\mathbf{x}_i, \mathbf{g}) \\ &= \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{p} (x_{ij} - g_j)^2. \end{aligned}$$

L'inertie mesure la dispersion des points du nuage par rapport à son centre de gravité : plus l'inertie est grande, plus le nuage est dispersé, et à l'inverse plus l'inertie est petite, plus le nuage est concentré autour de son centre de gravité.

Interprétation statistique

L'inertie du nuage de points I_T s'interprète facilement en terme de variance. En effet, on a :

$$I_T = \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{p} (x_{ij} - g_j)^2 \tag{1.5}$$

$$= \sum_{j=1}^{p} \left[\frac{1}{n} \sum_{i=1}^{n} (x_{ij} - g_j)^2 \right] \tag{1.6}$$

$$= \sum_{j=1}^{p} \left[\frac{1}{n} \sum_{i=1}^{n} \left(x_{ij} - \frac{1}{n} \sum_{i=1}^{n} x_{ij} \right)^2 \right] \tag{1.7}$$

$$= \sum_{j=1}^{p} s_j^2. \tag{1.8}$$

L'inertie totale du nuage de points est donc égale à la somme des variances des variables. En notant V la matrice de variance-covariance des variables, on a alors :

$$I_T = \text{Trace}\, V.$$

Notons que si les variables sont réduites, i.e. si leur écart-type vaut 1, alors la trace de la matrice V vaut exactement p. Autrement dit, ce n'est plus la matrice de variance-covariance qui joue un rôle, mais la matrice de corrélation. Rappelons que si le tableau de données est centré, $V = \frac{1}{n} X^t X$.

Inertie par rapport à un sous-espace vectoriel

Notons E un sous-espace vectoriel passant par le centre de gravité $\mathbf{g}$ du nuage (passant par l'origine 0 dans le cas où l'on a centré la matrice $\mathbf{X}$). L'inertie du nuage de points par rapport à ce sous-espace vectoriel E est définie par :

$$I_E = \frac{1}{n} \sum_{i=1}^{n} d^2(\mathbf{x}_i, p_{E,i}),$$

où $p_{E,i}$ est la projection orthogonale de $\mathbf{x}_i$ sur E.

Décomposition de l'inertie

Considérons une décomposition de $\mathbb{R}^p$ en deux sous-espaces vectoriels supplémentaires, M-orthogonaux, et passant par $\mathbf{g}$, i.e. $\mathbb{R}^p = E \oplus E^\perp$. On peut montrer facilement que :

$$I_T = I_E + I_{E^\perp}. \tag{1.9}$$

En effet, par le théorème de Pythagore, on a $d^2(\mathbf{x}_i, p_{E,i}) + d^2(\mathbf{x}_i, p_{E^\perp,i}) = d^2(\mathbf{x}_i, \mathbf{g})$. En projetant le nuage de points sur un sous-espace E, on perd l'inertie mesurée par I_E, et on ne conserve que celle mesurée par $I_{E^\perp}$.

Dans le cas particulier où E est un axe (i.e. un sous-espace vectoriel de dimension 1), $I_{E^\perp}$ mesure l'étirement du nuage selon cet axe. On parle également de *l'inertie portée par l'axe E*. Si on note u le vecteur directeur unitaire de l'axe E, on a l'expression matricielle suivante pour l'inertie expliquée par cet axe :

$$I_{E^\perp} = \frac{1}{n} u^t X^t X u = \langle u, \frac{1}{n} X^t X u \rangle. \tag{1.10}$$

En effet,

$$\begin{aligned}
I_{E^\perp} &= \frac{1}{n} \sum_{i=1}^{n} d^2(\mathbf{x}_i, p_{E^\perp,i}) \\
&= \frac{1}{n} \sum_{i=1}^{n} \langle \mathbf{x}_i, u \rangle^2 \\
&= \frac{1}{n} \sum_{i=1}^{n} (\mathbf{x}_i^t u)^2 \\
&= \frac{1}{n} \sum_{i=1}^{n} (\mathbf{x}_i^t u)(u^t \mathbf{x}_i) \\
&= \frac{1}{n} \sum_{i=1}^{n} (u^t \mathbf{x}_i)(\mathbf{x}_i^t u) \\
&= u^t \left(\frac{1}{n} \sum_{i=1}^{n} \mathbf{x}_i \mathbf{x}_i^t \right) u
\end{aligned}$$

$$= u^t \left(\frac{1}{n} X^t X \right) u.$$

La décomposition de l'inertie obtenue en 1.9 est connue sous le nom de Théorème de Huygens. On peut le généraliser au cas d'une décomposition de $\mathbb{R}^p$ comme somme de p sous-espaces de dimension 1 et orthogonaux entre eux :

$$\mathbb{R}^p = E_1 \oplus E_2 \oplus \cdots \oplus E_p.$$

On a alors :

$$I_T = I_{E_1^\perp} + I_{E_2^\perp} + \cdots + I_{E_p^\perp}.$$

1.4 Principes généraux

L'objectif d'une analyse en composantes principales est de projeter le nuage de points sur des sous-espaces affines de dimension plus petite, tels que la projection selon chacun de ces sous-espaces soit la plus fidèle possible. Autrement dit, on souhaite minimiser la distance entre les points du nuage initial et leurs projections selon le sous-espace considéré. En reprenant les notations de la section précédente, on cherche alors un sous-espace E tel que l'inertie I_E soit minimale, ou de façon équivalente, tel que l'inertie expliqué par E, $I_{E^\perp}$, soit maximale.

Plus précisément, on cherche en ACP une décomposition de $\mathbb{R}^p$ en sous-espaces orthogonaux de dimension 1, notés $E_1, \ldots, E_p$, passant par le centre de gravité du nuage, et tels que l'inertie expliquée par chacun de ces axes soit maximale.Pour cela, on procède de façon itérative en cherchant d'abord le premier axe E_1 tel que $I_{E_1^\perp}$ soit maximale, puis l'axe E_2 orthogonal à E_1 et tel que $I_{E_2^\perp}$ soit maximale, ...

Autrement dit, on effectue un *changement de repère* dans $\mathbb{R}^p$ tel que, dans ce nouveau système de représentation, le premier axe explique le plus possible l'inertie totale du nuage, le deuxième axe le plus possible l'inertie non prise en compte par le premier axe, et ainsi de suite ...

1.4.1 Recherche du premier axe principal

On cherche donc un axe E_1, passant par $\mathbf{g}$, et tel que I_{E_1} soit minimum, ou de façon équivalente tel que $I_{E_1^\perp}$ soit maximum. Notons u_1 le vecteur directeur unitaire de l'axe E_1. On cherche donc u_1 tel que $I_{E_1^\perp}$ soit maximum, sous la contrainte $\|u_1\|^2 = 1$.

En notant $V = \frac{1}{n} X^t X$, on obtient alors le problème d'optimisation sous contrainte suivant :

$$\begin{cases} \max_{u_1} u_1^t V u_1, \\ \|u_1\|^2 = u_1^t u_1 = 1. \end{cases}$$

Pour résoudre ce problème de maximisation sous contraintes, on utilise la méthode des multiplicateurs de Lagrange. On doit alors trouver u_1 vérifiant :

$$\frac{\partial}{\partial u_1} \left(u_1^t V u_1 - \lambda_1 (u_1^t u_1 - 1) \right) = 0.$$

c'est-à-dire, en utilisant la dérivée matricielle, vérifiant :

$$2Vu_1 - 2\lambda_1 u_1 = 0,$$

ou encore :

$$Vu_1 = \lambda_1 u_1.$$

Autrement dit, u_1 est un vecteur propre de V, la matrice de variance-covariance des données. Or, d'après l'équation 1.10, on a :

$$\begin{aligned} I_{E_1^\perp} &= u_1^t V u_1 \\ &= u_1^t(\lambda_1 u_1) \\ &= \lambda_1 \qquad\qquad \text{car } u_1 \text{ est de norme 1.} \end{aligned}$$

Comme on cherche à maximiser $I_{E_1^\perp}$, on va également chercher à maximiser λ_1, et on choisira donc la plus grande valeur propre de la matrice V.

Finalement, le vecteur directeur unitaire de l'axe E_1 est le **vecteur propre** associé à **la plus grande valeur propre** de la matrice V.

1.4.2 Recherche des axes suivants

Une fois que le premier axe a été identifié, on cherche l'axe E_2, **orthogonal à** E_1 et tel que l'inertie I_{E_2} soit minimale, ou de façon équivalent tel que $I_{E_2^\perp}$ soit maximale. En notant u_2 le vecteur directeur unitaire de E_2, on doit alors résoudre le système suivant :

$$\begin{cases} \max_{u_2} u_2^t V u_2, \\ \|u_2\|^2 = u_2^t u_2 = 1 \\ \langle u_1, u_2 \rangle = u_2^t u_1 = 0. \end{cases}$$

Par rapport au cas précédent, on a donc une contrainte de plus car le nouvel axe doit être orthogonal au premier. On utilise là encore la méthode des multiplicateurs de Lagrange. On cherche donc u_2 qui vérifie :

$$\frac{\partial}{\partial u_2}\left(u_2^t v u_2 - \lambda_2(u_2^t u_2 - 1) - \mu(u_2^t u_1)\right) = 0,$$

c'est-à-dire :

$$2Vu_2 - 2\lambda_2 u_2 - \mu u_1 = 0 \tag{1.11}$$

En multipliant à gauche par u_1^t, et en utilisant le fait que u_1 et u_2 sont orthogonaux, et que u_1 est de norme 1, on obtient :

$$\begin{aligned} 2u_1^t V u_2 - 2\lambda_2 u_1^t u_2 - \mu u_1^t u_1 = 0 &\Rightarrow 2u_1^t V u_2 - \mu = 0 \\ &\Rightarrow 2\lambda_1 u_1^t u_2 - \mu = 0 \end{aligned}$$

$$\Rightarrow \mu = 0$$

Finalement, en reprenant l'équation (1.11), on obtient :

$$Vu_2 = \lambda_2 u_2.$$

Le vecteur directeur unitaire de E_2 est donc également un vecteur propre de V. Comme on cherche à maximiser l'inertie $I_{E_2^\perp}$, on va raisonner de même que pour le premier axe et maximiser λ_2. On choisira donc pour u_2, le vecteur propre associé à la **deuxième plus grande valeur propre** de V.

On raisonne de même pour trouver les axes suivants, dont les vecteurs directeurs unitaires sont tous des vecteurs propres de la matrice V, associés aux valeurs propres ordonnées par ordre décroissant. La matrice V étant symétrique réelle, elle possède bien p vecteurs propres réels qui forment une base orthogonale de $\mathbb{R}^p$.

Les axes $E_1, E_2, \ldots, E_p$ sont appelées **axes factoriels**, ou **axes principaux d'inertie**.

1.4.3 Contribution des axes à l'inertie totale

Par le théorème de Huygens, on a :

$$I_T = I_{E_1^\perp} + I_{E_2^\perp} + \cdots + I_{E_p^\perp} = \lambda_1 + \lambda_2 + \cdots + \lambda_p.$$

L'inertie expliquée par l'axe E_j est :

$$I_{E_j^\perp} = \lambda_j,$$

et le pourcentage d'inertie expliquée par cet axe, aussi appelé **contribution relative**, est :

$$\frac{\lambda_j}{\lambda_1 + \cdots + \lambda_p}.$$

De la même façon, on peut définir le pourcentage d'inertie expliquée par le sous-espace $F_h = E_1 \oplus \cdots \oplus E_h$ par :

$$\frac{\lambda_1 + \cdots + \lambda_h}{\lambda_1 + \cdots + \lambda_p}.$$

Ces pourcentages d'inertie expliquée permettent de rendre compte de la part de variabilité du nuage expliquée par le sous-espace considéré. En particulier, si les dernières valeurs propres sont faibles, les axes associés expliqueront une faible part de l'inertie totale et pourront être négligés. Dans la pratique, on représentera alors le nuage de points dans un sous-espace de dimension $d < p$, où d est choisi tel que l'inertie expliquée par les d premiers axes soit proche de 1.

1.5 Représentation des individus dans les nouveaux axes

Notons y_{ij} la coordonnée de l'individu i selon l'axe E_j. Par définition, on a :

$$y_{ij} = \langle \mathbf{x}_i, u_j \rangle = u_j^t \mathbf{x}_i.$$

En notant $\mathbf{y}_i$ le vecteur des coordonnées de l'individu i dans la nouvelle base, et U la matrice des vecteurs propres, on obtient alors :

$$\mathbf{y}_i = U^t \mathbf{x}_i.$$

En notant Y la matrice contenant en ligne les coordonnées des individus dans la nouvelle base, on a :

$$Y^t = U^t X^t \Leftrightarrow Y = XU. \tag{1.12}$$

1.5.1 Qualité de représentation des individus

Comme on travaille avec des projections du nuage initial dans un sous-espace, il faut faire attention à l'interprétation géométrique du nouveau nuage de points obtenu. En effet, si le fait que deux points projetés soient éloignés l'un de l'autre assure que les deux points initiaux (i.e. non projetés) sont également éloignés l'un de l'autre, le fait que deux points projetés soient proches l'un de l'autre n'implique pas forcément que les points initiaux le soient aussi.

Pour interpréter correctement la proximité entre deux points projetés, il faut s'assurer au préalable que ces points sont bien représentés dans le sous-espace sur lequel on les a projetés. Pour cela, on s'intéresse à l'angle entre le point $\mathbf{x}_i$ et le sous-espace sur lequel on projette, et plus précisément au cosinus carré de cet angle. Ainsi, si ce cosinus carré est proche de 1, cela veut dire que l'individu est bien représenté par sa projection sur le sous-espace. Par suite, si deux individus bien représentés par leurs projections sur un sous-espace et ont des projections qui sont proches, alors on peut en déduire qu'ils sont également proches dans l'espace entier. Le cosinus carré de l'angle α_{ij} entre $\mathbf{x}_i$ et u_j est donné par :

$$\cos^2(\alpha_{ij}) = \frac{\langle \mathbf{x}_i, u_j \rangle^2}{\|\mathbf{x}_i\|^2} = \frac{(u_j^t \mathbf{x}_i)^2}{\|\mathbf{x}_i\|^2} = \frac{y_{ij}^2}{\|\mathbf{x}_i\|^2}. \tag{1.13}$$

De même, on peut calculer le cosinus carré de l'angle $\alpha_{ijj'}$ entre $\mathbf{x}_i$ et le sous-espace engendré par les axes E_j et $E_{j'}$ par :

$$\cos^2(\alpha_{ijj'}) = \cos^2(\alpha_{ij}) + \cos^2(\alpha_{ij'}).$$

Remarque. *Comme $\|\mathbf{y}_i\| = \|\mathbf{x}_i\|$ (il s'agit simplement d'un changement de base, la norme du vecteur restant inchangée), on a :*

$$\sum_{j=1}^{p} \cos^2(\alpha_{ij}) = 1.$$

Remarque. *Si un individu est proche du centre de gravité du nuage, alors $\|\mathbf{x}_i\|$ est faible et la représentation de ce point sur chaque axe ou sous-espace sera de bonne qualité.*

1.5.2 Interprétation des nouveaux axes en fonction des individus

On rappelle que l'inertie expliquée par un axe E_j est :

$$I_{E_j^\perp} = \frac{1}{n} \sum_{i=1}^{n} d^2(\mathbf{x}_i, p_{E_j^\perp, i}) = \frac{1}{n} \sum_{i=1}^{n} \langle \mathbf{x}_i, u_j \rangle^2.$$

Comme tous les individus ont le même poids, ils contribuent chacun à cette inertie à hauteur de :

$$\mathrm{ca}(i,j) = \frac{1}{n}\langle \mathbf{x}_i, u_j\rangle^2 = \frac{1}{n} y_{ij}^2.$$

On parle de *contribution absolue*. Autrement dit, plus la projection d'un individu sur un axe donné est grande, plus il contribuera à expliquer l'axe en question. Inversement, un individu dont la projection sur un axe est faible, et donc proche du centre de gravité, contribuera peu à l'inertie portée par cet axe.

On peut aussi s'intéresser à la *contribution relative* d'un individu à un axe E_j. L'inertie portée par l'axe étant λ_j, la contribution relative est donnée par :

$$\mathrm{cr}(i,j) = \frac{\mathrm{ca}_i}{\lambda_j} = \frac{y_{ij}^2}{n\lambda_j}.$$

1.5.3 Reconstruction de la matrice de données

L'ACP permet de reconstruire la matrice de données X, et de construire des approximations de X de rang inférieur, c'est-à-dire de rang $k, k < p$. Ce problème est connu sous le nom d'approximation de rang faible. La meilleure approximation de rang k de la matrice X est la matrice A_k vérifiant :

$$A_k = \arg\min_{A_k} \|X - A_k\|^2, \quad \text{avec} \quad \mathrm{rang}(A_k) = k, \tag{1.14}$$

et elle s'obtient à l'aide des k premiers vecteurs propres de la matrice X^tX, i.e. les vecteurs propres associés aux k plus grandes valeurs propres de X^tX. Plus précisément, on a :

$$A_k = XU_kU_k^t,$$

où U_k est la matrice de taille $p \times k$ contenant en colonnes les k premiers vecteurs propres, i.e. $U_k = [u_1u_2 \dots u_k]$. Ce résultat est connu sous le nom de *théorème de Eckart-Young*.

L'erreur d'approximation est donnée par la somme des valeurs propres restantes :

$$\|X - A_k\|^2 = \sum_{j=k+1}^{p} \lambda_j$$

1.6 Représentation des variables

1.6.1 Composantes principales

La recherche des axes factoriels nous a permis d'identifier une nouvelle base orthonormée de l'espace $\mathbb{R}^p$. Les nouveaux axes obtenus sont des combinaisons linéaires des axes initiaux, et on peut donc associer aux variables initiales les combinaisons linéaires correspondantes afin de former de "nouvelles variables", appelées *composantes principales*. On peut alors s'intéresser à l'interprétation de ces "nouvelles variables", en cherchant à les relier aux "anciennes variables".

Définition 1. *On note $c_1, \dots, c_p$ les composantes principales, avec c_k la composante principale correspondant à l'axe E_j. On a :*

$$c_k = \sum_{j=1}^{p} u_{kj} X_j = Xu_k$$

Proposition 1. *Les composantes principales vérifient :*

- $\mathbb{E}(c_j) = 0$
- $\mathrm{Var}(c_j) = \lambda_j$
- $\mathrm{Cov}(c_j, c_k) = 0$

Démonstration. Les composantes principales sont centrées comme combinaisons linéaires de variables centrées (rappelons que l'on a supposé que le nuage de points était centré).

On se place maintenant dans $\mathbb{R}^n$ pour calculer la covariance entre deux composantes principales.

$$\mathrm{Cov}(c_j, c_k) = \frac{1}{n} u_k^t X^t X u_j = u_k^t V u_j = \lambda_j u_k^t u_j = \begin{cases} 0 & \text{si } k \neq j \\ \lambda_j & \text{si } k = j \end{cases}$$

□

1.6.2 Qualité de la réprésentation et cercle des corrélations

Afin d'étudier le lien entre les composantes principales et les variables initiales, on s'intéresse aux corrélations entre les nouvelles et les anciennes variables. On définit d'abord la covariance entre une variable initiale X_j et une composante principale c_k :

$$\mathrm{Cov}(X_j, c_k) = \frac{1}{n} u_k^t X^t X_j = \frac{1}{n} u_k^t X^t X \begin{pmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{pmatrix} = u_k^t V \begin{pmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{pmatrix} = \lambda_k u_k^t \begin{pmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{pmatrix} = \lambda_k u_{kj}. \tag{1.15}$$

On en déduit la corrélation entre X_j et c_k :

$$\mathrm{COR}(j,k) = \frac{\mathrm{Cov}(X_j, c_k)}{\sqrt{\mathrm{Var}(X_j)\,\mathrm{Var}(c_k)}} = \frac{\lambda_k u_{kj}}{\sqrt{s_j^2 \lambda_k}} = \sqrt{\lambda_k}\,\frac{u_{kj}}{s_j}.$$

Si les variables sont réduites, alors $\mathrm{COR}(j,k) = \sqrt{\lambda_k} u_{kj}$. Cette quantité donne la qualité de représentation de la variable j sur l'axe E_k. Plus elle est proche de 1 en valeur absolue, plus la variable est bien représentée par l'axe k. Le signe de la corrélation permet de savoir si la variable contribue positivement ou négativement à la définition de l'axe k.

On représente souvent graphiquement ces corrélations en dimension 2, à l'aide de ce que l'on appelle un *cercle des corrélations*. Plus précisément, on choisit deux axes E_j et E_k, et l'on trace les points dont les coordonnées sont les corrélations de chacune des variables avec les j-ème et k-ème composantes principales. Un exemple est fourni dans la figure 1.1.

Plus une variable est proche du cercle de corrélation, mieux elle est représentée par le plan considéré. Si deux variables **proches du cercle de corrélation** (et donc bien représentées dans le plan considéré) sont proches l'une de l'autre, alors elles sont elle-mêmes fortement corrélées positivement. Inversement, deux variables **proches du cercle de corrélation** mais symétriquement opposées par rapport à l'origine seront fortement corrélées négativement.

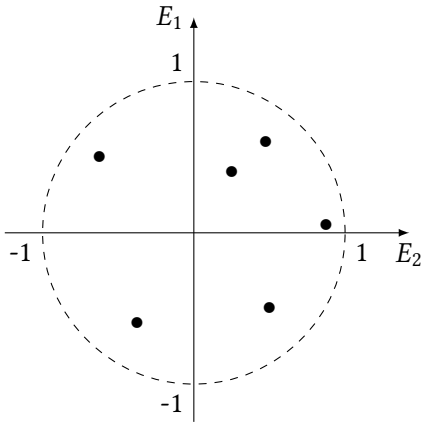


FIGURE 1.1 – Exemple de cercle des corrélations sur le premier plan factoriel. L'abscisse correspond à la corrélation entre chaque variable initiale et la composante principale c_1, et l'ordonnée à la corrélation entre chaque variable et c_2.

1.7 En pratique

Dans la pratique, on retient un nombre $q < p$ d'axes principaux, sur lesquels on va projeter notre nuage de points. On doit alors proposer une interprétation des nouveaux axes obtenus, ou de façon équivalente des composantes principales. On peut s'aider pour cela des outils définis dans les sections précédentes (contribution des individus et des variables dans la définition des axes). Plus précisément, on réalisera les étapes successives suivantes :

1. centrage de la matrice de données
2. réduction de la matrice de données si nécessaire
3. calcul des valeurs propres de V, et choix du nombre d'axes à retenir en fonction du pourcentage d'inertie que l'on souhaite conserver. On peut également utiliser des règles "empiriques" comme la règle de Kaiser dans le cas d'une ACP sur variables réduites, qui consiste à ne garder que les axes pour lesquels la valeur propre λ_j est plus grande que $1/p$, ou la règle du "coude" qui consiste à repérer un "coude" dans le graphe des valeurs propres (voir un exemple sur la figure 1.2).
4. interprétation des nouvelles variables, à l'aide des cercles de corrélations : **attention, les variables doivent être proches du bord du cercle pour être bien représentées** dans le plan factoriel considéré.
5. complément pour l'interprétation des nouveaux axes à l'aide des individus et de leurs contributions à la fabrication des axes. Par exemple, on peut retenir les individus dont la contribution est supérieure à la contribution moyenne $1/n$. Attention, si on souhaite comparer des individus entre eux, il faut d'abord s'assurer qu'ils sont bien représentés dans le plan factoriel considéré.

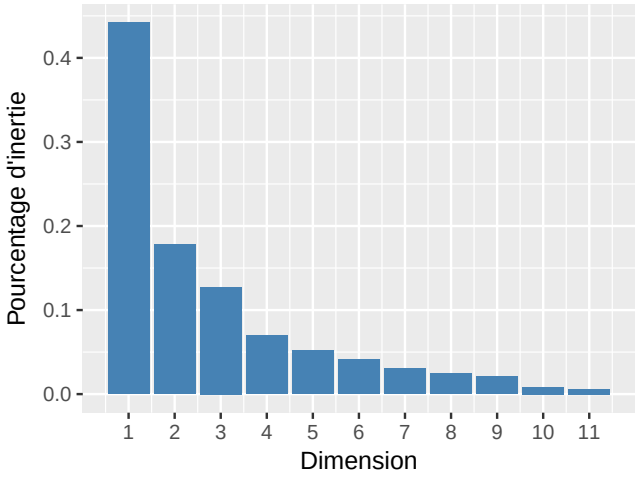


FIGURE 1.2 – Éboulis des valeurs propres, exprimé en pourcentage de l'inertie totale. On peut observer un "coude" entre les dimensions 3 et 4, et on gardera ainsi 3 axes.

1.8 Effet taille

Lorsque toutes les variables X_j sont corrélées **positivement** entre elles, la première composante principale définit ce que l'on appelle un *effet taille*. En effet, on sait qu'une matrice symétrique dont tous les termes sont positifs admet un premier vecteur propre dont toutes les composantes sont de même signe (c'est le théorème de Frobenius). La première composante principale est alors corrélée positivement avec toutes les variables. Si un tel effet se produit, on veillera à ne pas considérer cet axe dans l'interprétation des résultats, que l'on débutera avec le deuxième axe factoriel.

1.9 Notations générales

Dans ce chapitre, on a allégé les notations en considérant la distance euclidienne classique entre les individus vus comme points de $\mathbb{R}^p$, et en affectant à chacun des individus un poids $1/n$. Cependant il est possible de placer l'ACP dans un cadre un peu plus général que l'on présente ici.

De façon générale, on peut munir $\mathbb{R}^p$ de la métrique définie par la matrice M de dimension $p \times p$. On définit alors le produit scalaire par :

$$\langle \mathbf{x}_i, \mathbf{x}_j \rangle_M = \mathbf{x}_j^t M \mathbf{x}_i.$$

De même, on peut affecter à chaque individu des poids différents. On choisit alors une matrice de poids D_p de dimension $n \times n$. La matrice de variance-covariance s'écrit alors :

$$V = \mathbf{X}^t D_p \mathbf{X},$$

et on a :

$$I_T = \text{Trace}(VM).$$

L'ACP consiste alors à chercher les valeurs propres de la matrice VM.

Chapitre 2

Analyse factorielle des correspondances (AFC)

L'analyse factorielle des correspondances (AFC) a été proposée par Jean-Pierre Benzécri dans les années 1980. Elle permet d'analyser le lien, encore appelé *correspondance*, entre deux variables qualitatives. Elle peut être vue comme une ACP particulière, basée sur la métrique du chi-deux.

2.1 Tableau de contingence et nuages de points associés

Notons X_1 et X_2 les deux variables qualitatives que l'on souhaite étudier, p le nombre de modalités de la variable X_1 et q le nombre de modalités de la variable X_2. Le point de départ d'une AFC est le tableau de contingence N obtenu en croisant les deux variables X_1 et X_2. Plus précisément, on a :

$$N = \begin{pmatrix} n_{11} & \cdots & n_{1q} \\ \vdots & & \vdots \\ n_{p1} & \cdots & n_{pq} \end{pmatrix}, \tag{2.1}$$

où n_{ij} est le nombre d'observations pour lesquelles $X_1 = i$ et $X_2 = j$.

On note D_r et D_c les matrices diagonales contenant respectivement les effectifs marginaux de X_1 et X_2 :

$$D_r = \begin{pmatrix} n_{1\bullet} & & 0 \\ & \ddots & \\ 0 & & n_{p\bullet} \end{pmatrix} \quad \text{et} \quad D_c = \begin{pmatrix} n_{\bullet 1} & & 0 \\ & \ddots & \\ 0 & & n_{\bullet q} \end{pmatrix}, \tag{2.2}$$

où $n_{i\bullet} = \sum_{j=1}^{q} n_{ij}$ et $n_{\bullet j} = \sum_{i=1}^{p} n_{ij}$.

Définition 2. *On appelle profils des lignes le tableau $X_r = D_r^{-1}N$ et profils des colonnes le tableau $X_c = D_c^{-1}N^t$.*

Les profils-lignes forment un nuage de p points dans $\mathbb{R}^q$. On affecte alors à chacun de ces p points un poids porportionnel à sa fréquence marginale (plus une modalité est rare, moins elle a de poids), et

on obtient la matrice de poids $\frac{D_r}{n}$. On peut alors définir le centre de gravité de ce nuage de points :

$$g_r = X_r^t \frac{D_r}{n} \mathbf{1}_p = \frac{1}{n} (D_r N)^t D_r \mathbf{1}_p = \frac{1}{n} N^t \mathbf{1}_p = \begin{pmatrix} \frac{n_{\bullet 1}}{n} \\ \vdots \\ \frac{n_{\bullet q}}{n} \end{pmatrix}. \tag{2.3}$$

On retrouve le profil marginal. De la même façon, les profils-colonnes forment un nuage de q points dans $\mathbb{R}^p$, de poids donnés par la matrice $\frac{D_c}{n}$. Leur centre de gravité est :

$$g_c = X_c^t \frac{D_c}{n} \mathbf{1}_q = \begin{pmatrix} \frac{n_{1\bullet}}{n} \\ \vdots \\ \frac{n_{p\bullet}}{n} \end{pmatrix}. \tag{2.4}$$

N.B. Comme la somme par ligne des matrices X_r et X_c vaut 1, les nuages de points formés par les profils-lignes ou par les profils-colonnes vivent en fait dans des espaces de dimension respectives $p-1$ et $q-1$, c'est-à-dire dans un hyperplan de $\mathbb{R}^p$ ou de $\mathbb{R}^q$.

Exemple : Si $N = \begin{pmatrix} 3 & 4 & 1 \\ 5 & 2 & 3 \end{pmatrix}$, alors on a $D_r = \begin{pmatrix} 8 & 0 \\ 0 & 10 \end{pmatrix}$ et $D_c = \begin{pmatrix} 8 & 0 & 0 \\ 0 & 6 & 0 \\ 0 & 0 & 4 \end{pmatrix}$.

On obtient alors :

$$X_r = \begin{pmatrix} 3/8 & 4/8 & 1/8 \\ 5/10 & 2/10 & 3/10 \end{pmatrix} = \begin{pmatrix} 0.375 & 0.5 & 0.125 \\ 0.5 & 0.2 & 0.3 \end{pmatrix}, \qquad g_r = \begin{pmatrix} 8/18 \\ 6/18 \\ 4/18 \end{pmatrix} = \begin{pmatrix} 0.44 \\ 0.33 \\ 0.22 \end{pmatrix}$$

$$X_c = \begin{pmatrix} 3/8 & 5/8 \\ 4/6 & 2/6 \\ 1/4 & 3/4 \end{pmatrix} = \begin{pmatrix} 0.375 & 0.625 \\ 0.67 & 0.33 \\ 0.25 & 0.75 \end{pmatrix}, \qquad g_c = \begin{pmatrix} 8/18 \\ 10/18 \end{pmatrix} = \begin{pmatrix} 0.44 \\ 0.66 \end{pmatrix}$$

Si on représente le nuage de points formés par les profils colonnes, en notant x_i le point dont les coordonnées sont données par la i-ème ligne de la matrice X_c, on obtient la figure 2.1. Bien qu'il s'agisse d'un nuage de points de $\mathbb{R}^2$, les points sont bien alignés sur une droite.

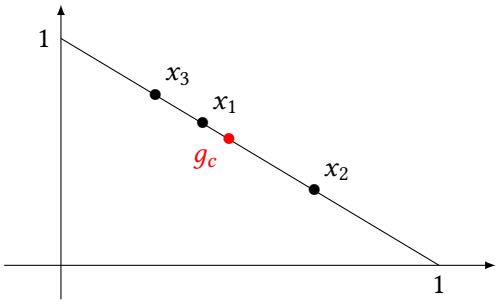


FIGURE 2.1 – Représentation du nuage de points des profils-colonnes.

Écart à l'indépendance

Si les deux variables X_1 et X_2 sont indépendantes, alors la répartition des observations selon les différentes modalités de l'une des variables et la même quelque soit la modalité de l'autre. Autrement dit, on devrait avoir des profils-lignes identiques et des profils-colonnes identiques.

Exemple d'indépendance : Supposons que l'on s'intéresse à la répartition des résultats d'un groupe de $n = 36$ étudiants à un examen selon trois catégories (échec, rattrapage, succès) et en fonction du sexe. Les deux variables qui nous intéressent ici sont donc X_1 le résultat à l'examen, et X_2 le sexe. Si ces deux variables sont indépendantes, alors il y a, en proportion, autant de garçons que de filles qui ont échoué, qui passent le rattrapage, ou qui ont réussi l'examen. Par exemple, on observerait le tableau de contingence suivant :

$$N = \begin{pmatrix} 2 & 4 & 6 \\ 4 & 8 & 12 \end{pmatrix},$$

où la première ligne correspond à la répartition des résultats filles selon les trois catégories possibles, et la deuxième ligne à la répartition des résultats des garçons.

On obtient les profils lignes et colonnes suivants :

$$X_r = \begin{pmatrix} 0.17 & 0.33 & 0.5 \\ 0.17 & 0.33 & 0.5 \end{pmatrix} \qquad X_c = \begin{pmatrix} 0.33 & 0.67 \\ 0.33 & 0.67 \\ 0.33 & 0.67 \end{pmatrix}.$$

Autrement dit, en cas d'indépendance, la fréquence est la même dans chaque ligne (et dans chaque colonne), et tous les points sont alors confondus avec le centre de gravité du nuage. En effet, on a dans ce cas :

$$\forall j, \quad \frac{n_{ij}}{n_{i\bullet}} = \sum_{i=1}^{p} \frac{n_{ij}}{n_{i\bullet}} \underbrace{\frac{n_{i\bullet}}{n}}_{\substack{\text{poids de} \\ \text{la ligne } i}} = \frac{1}{n} \sum_{i=1}^{p} n_{ij} = \frac{n_{\bullet j}}{n}.$$

Donc, si les deux variables X_1 et X_2 sont indépendantes, on a $n_{ij} = \frac{n_{i\bullet} n_{\bullet j}}{n}$.

Étudier la forme du nuage de points permet d'évaluer à quel point on s'écarte de la situation correspondant à l'indépendance.

2.2 Métrique du chi-deux

L'objectif de l'AFC est donc d'étudier la dispersion des points autour du centre de gravité du nuage. Pour cela, on va devoir choisir une métrique appropriée afin de définir la distance entre deux points.

Dans le cas de l'AFC, la distance utilisée est la *distance du chi-deux*, qui permet de mettre plus de poids sur les modalités de petits effectifs, et la métrique associée à cette distance est appelée *métrique*

du chi-deux. Elle est donnée par la matrice nD_c^{-1} pour les profils-lignes X_r, et par la matrice nD_r^{-1} pour les profils-colonnes X_c.

À titre d'exemple dans le cas de deux profils-lignes i et i', on a :

$$d^2_{\chi^2}(i,i') = \sum_{i=1}^{q} \frac{n}{n_{\bullet j}} \left(\frac{n_{ij}}{n_{i\bullet}} - \frac{n_{i'j}}{n_{i'\bullet}} \right)^2 . \tag{2.5}$$

On peut définir de la même manière la distance du chi-deux entre deux profils-colonnes.

Avec cette distance, si on observe de grands écarts sur des modalités peu représentées, ceux-ci ont plus de poids dans le calcul de la distance. Et inversement, on donne moins de poids à des écarts importants qui pourraient être dus au fait que l'on a seulement observé plus de points sur cette modalité.

Exemple :
Si on reprend l'exemple présenté au début du chapitre, avec

$$X_r = \begin{pmatrix} 0.375 & 0.5 & 0.125 \\ 0.5 & 0.2 & 0.3 \end{pmatrix},$$

on a $d^2_{\chi^2}(1,2) = \frac{18}{8}(0.375-0.5)^2 + \frac{18}{6}(0.5-0.2)^2 + \frac{18}{4}(0.125-0.2)^2 = 0.384$. La troisième modalité, qui ne représente que 4 individus, a plus de poids dans le calcul de la distance. Si on ne pondérait pas par $n/n_{\bullet j}$, on aurait $d^2(1,2) = (0.375-0.5)^2 + (0.5-0.2)^2 + (0.125-0.2)^2 = 0.136$.

Écart à l'indépendance
Avec la métrique du chi-deux, l'inertie totale du nuage de points correspond alors à la statistique de test utilisée dans le test d'indépendance du chi-deux :

$$\varphi^2 = \sum_{i=1}^{p} \frac{n_{i\bullet}}{n} d^2_{\chi^2}(i, g_r) \tag{2.6}$$

$$= \sum_{i=1}^{p} \sum_{i=1}^{q} \frac{n_{i\bullet}}{n} \frac{n}{n_{\bullet j}} \left(\frac{n_{ij}}{n_{i\bullet}} - \frac{n_{\bullet j}}{n} \right)^2 \tag{2.7}$$

$$= \sum_{i=1}^{p} \sum_{i=1}^{q} \frac{\left(n_{ij} - \frac{n_{i\bullet} n_{\bullet j}}{n} \right)^2}{n_{i\bullet} n_{\bullet j}} \tag{2.8}$$

$$\varphi^2 = \sum_{i=1}^{p} \sum_{j=1}^{q} \frac{n_{ij}}{n_{i\bullet} n_{\bullet j}} - 1 \tag{2.9}$$

On a vu en effet qu'en cas d'indépendance, tous les points étaient confondus avec le centre de gravité du nuage. Dans ce cas, l'inertie totale du nuage est nulle, et la statistique de test du chi-deux est nulle : quelque soit le niveau du test, on ne rejettera pas l'hypothèse nulle d'indépendance, i.e. la p-valeur est égale à 1. À l'inverse, plus on s'écarte de l'indépendance, plus l'inertie augmente, et donc plus la statistique de test du chi-deux augmente. En fonction du niveau que l'on s'est fixé pour le test d'indépendance, on pourra donc être amené à rejetter l'hypothèse d'indépendance.

N.B. Les deux nuages de points (i.e. celui des profils-lignes et celui des profiles-colonnes) ont la même inertie totale.

Équivalence distributionnelle

Un autre argument en faveur de la distance du chi-deux est celui du principe *d'équivalence distributionnelle.* Supposons que deux colonnes (ou deux lignes) aient le même profil. On pourrait alors avoir envie de regrouper les deux colonnes correspondantes, dont l'effectif serait la somme des effectifs des deux colonnes. On aimerait cependant que ce regroupement ne modifie pas la distance entre les points du nuage.

La distance du chi-deux permet justement de respecter cette propriété. Notons j' et j'' les deux colonnes dont les profils sont identiques, et $\tilde{j}$ la colonne résultant de la fusion de j' et j''. Calculons maintenant la distance entre deux lignes i et i', avant la fusion des deux colonnes.

$$\begin{aligned} d^2_{\chi^2}(i,i') &= \sum_{j=1}^{q} \frac{n}{n_{\bullet j}} \left(\frac{n_{ij}}{n_{i\bullet}} - \frac{n_{i'j}}{n_{i'\bullet}} \right)^2 \\ &= \sum_{\substack{j=1 \\ j\neq j',j''}}^{q} \frac{n}{n_{\bullet j}} \left(\frac{n_{ij}}{n_{i\bullet}} - \frac{n_{i'j}}{n_{i'\bullet}} \right)^2 + \frac{n}{n_{\bullet j'}} \left(\frac{n_{ij'}}{n_{i\bullet}} - \frac{n_{i'j'}}{n_{i'\bullet}} \right)^2 + \frac{n}{n_{\bullet j''}} \left(\frac{n_{ij''}}{n_{i\bullet}} - \frac{n_{i'j''}}{n_{i'\bullet}} \right)^2 \end{aligned}$$

Après la fusion des deux colonnes, la colonne $\tilde{j}$ vérifie :

$$n_{\bullet\tilde{j}} = n_{\bullet j'} + n_{\bullet j''} \quad \text{et} \quad n_{i\tilde{j}} = n_{ij'} + n_{ij''} \quad \text{avec } \frac{n_{ij'}}{n_{\bullet j'}} = \frac{n_{ij''}}{n_{\bullet j''}}.$$

La distance entre i et i' après fusion est alors donnée par :

$$\begin{aligned} \tilde{d}^2_{\chi^2}(i,i') &= \sum_{\substack{j=1 \\ j\neq\tilde{j}}}^{q} \frac{n}{n_{\bullet j}} \left(\frac{n_{ij}}{n_{i\bullet}} - \frac{n_{i'j}}{n_{i'\bullet}} \right)^2 + \frac{n}{n_{\bullet\tilde{j}}} \left(\frac{n_{i\tilde{j}}}{n_{i\bullet}} - \frac{n_{i'\tilde{j}}}{n_{i'\bullet}} \right)^2 \\ &= \sum_{\substack{j=1 \\ j\neq\tilde{j}}}^{q} \frac{n}{n_{\bullet j}} \left(\frac{n_{ij}}{n_{i\bullet}} - \frac{n_{i'j}}{n_{i'\bullet}} \right)^2 + \frac{n}{n_{\bullet j'} + n_{\bullet j''}} \left(\frac{n_{ij'} + n_{ij''}}{n_{i\bullet}} - \frac{n_{i'j'} + n_{i'j''}}{n_{i'\bullet}} \right)^2. \end{aligned}$$

On montre alors très facilement que $d^2_{\chi^2}(i,i')$ et $\tilde{d}^2_{\chi^2}(i,i')$ sont égales.

Démonstration. Notons

$$A = \frac{n}{n_{\bullet j'} + n_{\bullet j''}} \left(\frac{n_{ij'} + n_{ij''}}{n_{i\bullet}} - \frac{n_{i'j'} + n_{i'j''}}{n_{i'\bullet}} \right)^2 \quad \text{et} \quad B = \frac{n}{n_{\bullet j'}} \left(\frac{n_{ij'}}{n_{i\bullet}} - \frac{n_{i'j'}}{n_{i'\bullet}} \right)^2 + \frac{n}{n_{\bullet j''}} \left(\frac{n_{ij''}}{n_{i\bullet}} - \frac{n_{i'j''}}{n_{i'\bullet}} \right)^2,$$

et montrons que $A = B$. On a :

$$\begin{aligned} A &= \frac{n}{n_{\bullet j'} + n_{\bullet j''}} \left(\frac{n_{ij'} + n_{ij''}}{n_{i\bullet}} - \frac{n_{i'j'} + n_{i'j''}}{n_{i'\bullet}} \right)^2 \\ &= \frac{n}{n_{\bullet j'} + n_{\bullet j''}} \left(\frac{n^2_{ij'} + n^2_{ij''} + 2n_{ij'}n_{ij''}}{n^2_{i\bullet}} + \frac{n^2_{i'j'} + n^2_{i'j''} + 2n_{i'j'}n_{i'j''}}{n^2_{i'\bullet}} - 2\,\frac{(n_{ij'} + n_{ij''})(n_{i'j'} + n_{i'j''})}{n_{i\bullet}n_{i'\bullet}} \right) \end{aligned}$$

$$
\begin{aligned}
=&\frac{n}{n_{\bullet j'}+n_{\bullet j''}}\left(\frac{n_{ij'}^2+n_{ij''}^2+n_{ij'}^2\frac{n_{\bullet j''}}{n_{\bullet j'}}+n_{ij''}^2\frac{n_{\bullet j'}}{n_{\bullet j''}}}{n_{i\bullet}^2}+\frac{n_{i'j'}^2+n_{i'j''}^2+n_{i'j'}^2\frac{n_{\bullet j''}}{n_{\bullet j'}}+n_{i'j''}^2\frac{n_{\bullet j'}}{n_{\bullet j''}}}{n_{i'\bullet}^2}\right.\\
&\left.-2\,\frac{\left(n_{ij'}+n_{ij'}\frac{n_{\bullet j''}}{n_{\bullet j'}}\right)(n_{i'j'}+n_{i'j''})}{n_{i\bullet}n_{i'\bullet}}\right)\\
=&\frac{n}{n_{\bullet j'}+n_{\bullet j''}}\left(\frac{n_{ij'}^2\left(1+\frac{n_{\bullet j''}}{n_{\bullet j'}}\right)+n_{ij''}^2\left(1+\frac{n_{\bullet j'}}{n_{\bullet j''}}\right)}{n_{i\bullet}^2}+\frac{n_{i'j'}^2\left(1+\frac{n_{\bullet j''}}{n_{\bullet j'}}\right)+n_{i'j''}^2\left(1+\frac{n_{\bullet j'}}{n_{\bullet j''}}\right)}{n_{i'\bullet}^2}\right.\\
&\left.-2\,\frac{n_{ij'}\left(1+\frac{n_{\bullet j''}}{n_{\bullet j'}}\right)(n_{i'j'}+n_{i'j''})}{n_{i\bullet}n_{i'\bullet}}\right)\\
=&\,n\left(\frac{n_{ij'}^2}{n_{i\bullet}^2n_{\bullet j'}}+\frac{n_{ij''}^2}{n_{i\bullet}^2n_{\bullet j''}}+\frac{n_{i'j'}^2}{n_{i'\bullet}^2n_{\bullet j'}}+\frac{n_{i'j''}^2}{n_{i'\bullet}^2n_{\bullet j''}}-2\,\frac{n_{ij'}(n_{i'j'}+n_{i'j''})}{n_{i\bullet}n_{i'\bullet}n_{\bullet j'}}\right)\\
=&\frac{n}{n_{\bullet j'}}\left(\frac{n_{ij'}^2}{n_{i'\bullet}^2}+\frac{n_{i'j'}^2}{n_{i'\bullet}^2}\right)+\frac{n}{n_{\bullet j''}}\left(\frac{n_{ij''}^2}{n_{i\bullet}^2}+\frac{n_{i'j''}^2}{n_{i'\bullet}^2}\right)-2\,\frac{n}{n_{\bullet j'}}\frac{n_{ij'}n_{i'j'}}{n_{i\bullet}n_{i'\bullet}}-2\,\frac{n}{n_{\bullet j'}}\frac{n_{ij'}n_{i'j''}}{n_{i\bullet}n_{i'\bullet}}\\
=&\frac{n}{n_{\bullet j'}}\left(\frac{n_{ij'}^2}{n_{i\bullet}^2}+\frac{n_{i'j'}^2}{n_{i'\bullet}^2}-2\,\frac{n_{ij'}n_{i'j'}}{n_{i\bullet}n_{i'\bullet}}\right)+\frac{n}{n_{\bullet j''}}\left(\frac{n_{ij''}^2}{n_{i\bullet}^2}+\frac{n_{i'j''}^2}{n_{i'\bullet}^2}-2\,\frac{n_{ij''}n_{i'j''}}{n_{i\bullet}n_{i'\bullet}}\right)\\
=&\,B
\end{aligned}
$$

□

2.3 ACP des deux nuages de profils

Une fois que l'on a défini la matrice des données X_r (respectivement X_c), la matrice de poids D_r/n (resp. D_c/n) et la métrique M_r (resp. M_c) on dispose de tous les éléments pour faire une ACP. Sur les profils-lignes, on cherche les valeurs propres de la matrice V_rM_r, avec $V_r = (X_r - \mathbf{1}_p g_r)^t \frac{D_r}{n}(X_r - \mathbf{1}_p g_r)$ et $M_r = nD_c^{-1}$. On cherche donc les valeurs propres de $(X_r - \mathbf{1}_p g_r)^t D_r (X_r - \mathbf{1}_p g_r) D_c$. De plus, on a :

$$V_r = N^t \frac{D_r^{-1}}{n} N - g_r g_r^t,$$

d'où

$$V_rM_r = N^t D_r^{-1} N D_c^{-1} - n g_r g_r^t D_c^{-1}.$$

De même, sur les profils-colonnes, on cherche les valeurs propres de la matrice $V_cM_c = ND_c^{-1}N^tD_r^{-1} - ng_cg_c^tD_r^{-1}$.

Or, le centre de gravité du nuage des profils lignes, g_r, est M_r-orthogonal au nuage des profils-lignes (de même, g_c est M_c-orthogonal au nuage des profils-colonnes). En effet, soit r_i une ligne de la matrice X_r, c'est-à-dire un des points du nuage des profils-lignes, avec $r_i = (n_{i1}/n_{i\bullet}, \ldots, n_{iq}/n_{i\bullet})^t$. Alors on a :

$$\langle r_i - g_r, g_r \rangle_{M_r} = (r_i - g_r)^t M_r g_r$$

$$
\begin{aligned}
&= (r_i - g_r)^t n D_c^{-1} g_r \\
&= (r_i - g_r)^t n D_c^{-1} \frac{N^t}{n} \mathbf{1}_p \\
&= (r_i - g_r)^t \mathbf{1}_q \\
&= \sum_{j=1}^{q} \left(\frac{n_{ij}}{n_{i\bullet}} - \frac{n_{\bullet j}}{n} \right) \\
&= 0
\end{aligned}
$$

Dire que g_r est M_r-orthogonal au nuage de points signifie que c'est un vecteur propre de $V_r M_r$ associé à la valeur propre 0. En effet, V_r étant constitué de combinaisons linéaires des lignes de X_r, on a $V_r M_r g_r = 0$. On a donc ainsi identifié un axe principal, et ce sont les autres axes principaux, orthogonaux à g_r, qui vont nous intéresser.

On remarque alors que $V_r M_r$ a les mêmes vecteurs propres que la matrice $N^t D_r^{-1} N D_c^{-1} = X_r^t D_r X_r M_r$, associés aux mêmes valeurs propres, sauf g_r qui est alors associé à la valeur propre 1. En effet, soit v un vecteur propre de $V_r M_r$, M_r-orthogonal à g_r, et de valeur propre λ. Alors :

$$
\begin{aligned}
X_r^t D_r X_r M_r v &= V_r M_r v + n g_r g_r^t D_c^{-1} v \\
&= \lambda v + g_r g_r^t M_r v \\
&= \lambda v + g_r \langle g_r, v \rangle_{M_r} \\
&= \lambda v
\end{aligned}
$$

Puis :

$$
\begin{aligned}
X_r^t D_r X_r M_r g_r &= V_r M_r g_r + n g_r g_r^t D_c^{-1} \\
&= g_r \| g_r \|_{M_r}^2 \\
&= 1,
\end{aligned}
$$

car g_r est M_r-orthonormé. Par conséquent, il n'est pas nécessaire de centrer le nuage de points des profils-lignes avant de réaliser l'ACP, et on peut travailler directement avec la matrice $N^t D_r^{-1} N D_c^{-1}$. Dans le cas des profils-colonnes, on s'intéressera à la matrice $N D_c^{-1} N^t D_r^{-1}$.

Lien entre les deux ACP

Il existe un lien entre les vecteurs et valeurs propres des deux matrices $N^t D_r^{-1} N D_c^{-1}$ et $N D_c^{-1} N^t D_r^{-1}$. Notons v un vecteur propre de $N^t D_r^{-1} N D_c^{-1}$, de norme 1 pour la métrique M_r, et associé à la valeur propre λ. On a :

$$
\begin{aligned}
N^t D_r^{-1} N D_c^{-1} v = \lambda v &\Rightarrow (N D_c^{-1}) N^t D_r^{-1} N D_c^{-1} v = \lambda (N D_c^{-1}) v \\
&\Rightarrow (N D_c^{-1} N^t D_r^{-1}) N D_c^{-1} v = \lambda (N D_c^{-1} v),
\end{aligned}
$$

donc $k N D_c^{-1} v$ est un vecteur propre pour $N D_c^{-1} N^t D_r^{-1}$, associé à la valeur propre λ. On cherche ensuite la valeur de k telle que les vecteurs propres soient de norme 1 pour la métrique M_c. Autrement dit, on

cherche k tel quue :

$$\begin{aligned}(kND_c^{-1}v)^t M_c(kND_c^{-1}v) = 1 &\Leftrightarrow k^2 v^t D_c^{-1} N^t M_c N D_c^{-1} v = 1\\ &\Leftrightarrow k^2 n\, v^t D_c^{-1} N^t D_r^{-1} N D_c^{-1} v = 1\\ &\Leftrightarrow k^2 n\, v^t D_c^{-1} \lambda v = 1\\ &\Leftrightarrow k = \frac{1}{\sqrt{\lambda}}.\end{aligned}$$

En résumé, si v est un vecteur propre de $N^t D_r^{-1} N D_c^{-1}$, pour la valeur propre λ, alors $\frac{1}{\sqrt{\lambda}} N D_c^{-1} v$ est un vecteur propre pour $N D_c^{-1} N^t D_r^{-1}$, pour la même valeur propre. De même, si w est un vecteur propre pour $N D_c^{-1} N^t D_r^{-1}$, pour la valeur propre λ, alors $\frac{1}{\sqrt{\lambda}} N^t D_r^{-1} w$ est un vecteur propre pour $N^t D_r^{-1} N D_c^{-1}$, pour la même valeur propre.

Les matrices $V_r M_r$ et $V_c M_c$ ont donc le même rang $r \leq \min(p-1, q-1)$, et dans la pratique on effectue l'ACP sur la plus petite matrice. En notant λ_k la k-ème valeur propre, on a alors :

$$\varphi^2 = \sum_{k=1}^{r} \lambda_k. \tag{2.10}$$

Chaque direction ou axe principal explique donc une partie de l'écart à l'indépendance entre les deux variables.

Facteurs principaux et composantes principales

Une fois les vecteurs propres identifiés, on peut en déduire les facteurs principaux et les composantes principales, en reprenant les notations utilisées dans le chapitre sur l'ACP. On remarque alors que les facteurs principaux de l'ACP des profils-lignes sont, à une constante près, les composantes principales de l'ACP des profils-colonnes, et vice versa.

En effet, soit v un vecteur propre de $V_r M_r$, associé à la valeur propre λ, et w le vecteur propre correspondant de $V_c M_c$. Alors $M_r v = n D_c^{-1} v$ est un facteur principal pour le nuage des profils-lignes, et $X_r M_r v = D_r^{-1} N n D_c^{-1} v = n\sqrt{\lambda} D_r^{-1} w$ est la composante principale correspondante. Inversement, $M_c w = n D_r^{-1} w$ est un facteur principal pour le nuage des profils-colonnes, et $X_c M_c w = D_c^{-1} N^t n D_r^{-1} w = n\sqrt{\lambda} D_c^{-1} v$.

Contribution des profils

Définition 3. *Soit c une composante principale du nuage des profils-lignes. On définit la contribution d'une ligne i par :*

$$CTR(i) = \frac{n_{i\bullet}}{n} \frac{c_i^2}{\lambda},$$

avec c_i la i-ème coordonnée de c.
On définit de même la contribution des profils-colonnes.

Chapitre 3

Analyse des correspondances multiples (ACM)

Dans le chapitre précédent, on s'intéressait au lien pouvant exister entre deux variables qualitatives. Dans ce chapitre, on cherche à décrire un nuage de points de n individus décrits par p variables qualitatives. Cette fois-ci, les lignes et les colonnes de la matrice des observations jouent donc des rôles différents. On peut voir l'analyse des correspondances multiples (ACM) comme une version de l'ACP adaptée aux variables discrètes.

3.1 Données

En général, on dispose d'un tableau de taille $n \times p$, dont l'élèment correspond à l'observation de la variable j pour l'individu i. A titre d'illustration, considérons le cas suivant, où l'on observe deux variables sur $n = 4$ individus, X_1 codée 0 ou 1 selon le sexe de l'individu, et X_2 codée 1, 2 ou 3 selon la classe d'âge de l'individu :

$$T = \begin{pmatrix} 1 & 3 \\ 2 & 2 \\ 1 & 1 \\ 1 & 1 \end{pmatrix}.$$

Si à première vue, ce tableau est proche du type de tableau que l'on obtient en ACP, en réalité le codage des variables qualitatives est arbitraire, et cela n'a donc pas de sens de faire des opérations algébriques directement sur T. Dans l'exemple précédent, on aurait en effet très bien pu coder le sexe 1 ou 2 au lieu de 0 ou 1. Par ailleurs, toutes les variables qualitatives ne sont pas ordinales, et l'ordre induit par le codage peut ne pas correspondre à une réalité physique.

Tableau disjonctif complet (TDC)

C'est pourquoi on travaille en ACM avec le *tableau disjonctif complet*, (TDC), qui contient autant de colonnes qu'il y a de catégories totales. Chaque variable initiale est découpée en autant de sous-variables

qu'elle ne peut prendre de valeurs. Par exemple, une variable initiale correspondant au sexe de l'individu et prenant 2 valeurs possibles sera découpée en deux sous-variables. Le TDC ne contient alors que des 0 et des 1, selon que l'individu i appartient à la sous-catégorie considérée ou non. On le note $\mathbf{X}$. En reprenant l'exemple précédent, on obtient le TDC suivant associé à T :

$$\mathbf{X} = \begin{array}{c} \begin{matrix} X_{1,1} & X_{1,2} & X_{2,1} & X_{2,2} & X_{2,3} \end{matrix} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \end{pmatrix} \end{array}$$

Le tableau disjonctif complet est de taille $n \times m$, avec $m = \sum_{j=1}^{p} m_j$ le nombre total de catégories, et m_j le nombre de catégories de la variable X_j. En notant x_{ik} l'élèment situé sur la ligne i et la colonne k (i.e. $x_{ik} = 1$ si l'individu i possède la modalité k et 0 sinon), on a :

$$n_{i\bullet} := \sum_{k=1}^{m} x_{ik} = p \quad \text{et} \quad n_{\bullet k} := \sum_{i=1}^{n} x_{ik}.$$

$n_{\bullet k}$ correspond donc à l'effectif marginal de la modalité k. On note D_c la matrice contenant sur sa diagonale les effectifs marginaux des m catégories (i.e. la somme de chaque colonne du TDC) :

$$D_c = \begin{pmatrix} D_1 & 0 & \dots & 0 \\ 0 & D_2 & \dots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & D_p \end{pmatrix},$$

où chaque bloc $D_j, 1 \leq j \leq p$ correspond aux modalités de la variable j. En reprenant l'exemple précédent, on a :

$$D_c = \left(\begin{array}{cc|ccc} 3 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ \hline 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{array}\right), \qquad D_1 = \begin{pmatrix} 3 & 0 \\ 0 & 1 \end{pmatrix} \quad \text{et} \quad D_2 = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Remarque. *Pour chaque variable X_j, la somme des m_j colonnes du TDC correspondant à ses modalités vaut 1. Comme il y a p variables, le rang de la matrice* $\mathbf{X}$ *est au plus* $\max(n, m-p)$. *En général, le nombre d'individus n est supérieur au nombre de catégories : dans ce cas le rang de* $\mathbf{X}$ *est au plus égal à $m-p$.*

Tableau de Burt

Une autre représentation possible du tableau T est sous la forme de ce que l'on appelle le *tableau de Burt*. Ce tableau, noté B, s'obtient facilement à partir du TDC : $B = X^t X$. Il est de taille $m \times m$, et contient

les croisements deux à deux de toutes les variables, c'est-à-dire tous les tableaux de contingence deux à deux des variables. Toujours en reprenant notre exemple, on trouve :

$$B = \left(\begin{array}{cc|ccc} 3 & 0 & 2 & 0 & 1 \\ 0 & 1 & 0 & 2 & 0 \\ \hline 2 & 0 & 2 & 0 & 0 \\ 0 & 2 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{array}\right).$$

Le tableau de Burt contient moins d'information que le tableau disjonctif complet, car on ne conserve pas l'information sur les individus : tout est agrégé au niveau de la catégorie.

3.2 Principe général

L'ACM consiste en une extension de l'AFC, en considérant $\mathbf{X}$ comme un tableau de contingence. Si on s'intéresse à la variabilité des individus et de leurs profils, on travaillera avec les profils-lignes, et si on s'intéresse aux liens entre les modalités, on s'intéressera aux profils-colonnes. Comme on l'a vu au chapitre 2, il y a un lien entre ces deux analyses, et dans la pratique on s'intéressera souvent à la fois aux individus et aux variables.

En pratique, on réalise une ACP du nuage des profils-lignes, en utilisant la métrique du chi-deux. En reprenant les notations du chapitre 2, on a donc :

- matrice des profils-lignes $X_r = D_r^{-1}X$, où D_r est la matrice contenant sur sa diagonale les sommes de chaque ligne de la matrice $\mathbf{X}$. On a donc $D_r = p\ Id_n$.
- matrice de poids D_r/n
- métrique $M_r = nD_c^{-1}$

L'inertie totale du nuage de points s'écrit donc :

$$I_T = \text{Trace}(V_r M_r), \quad \text{avec } V_r = X_r^t \frac{D_r}{n} X_r - g_r g_r^t,$$

et g_r centre de gravité du nuage des profils-lignes. Finalement, on cherche donc les valeurs propres de la matrice :

$$V_r M_r = X_r^t D_r^{-1} X_r D_c^{-1} - n g_r g_r^t D_c^{-1}.$$

3.2.1 Rappels sur la distance du chi-deux

Distance entre deux individus

Pour calculer la distance entre deux individus, on se place dans le nuage des profils lignes et on utilise la métrique du chi-deux nD_c^{-1}. On a :

$$d^2_{\chi^2}(i,i') = \sum_{k=1}^{m} \frac{np}{n_{\bullet k}} \left(\frac{x_{ik}}{n_{i\bullet}} - \frac{x_{i'k}}{n_{i'\bullet}}\right)^2 = \frac{n}{p} \sum_{j=1}^{m} \frac{1}{n_{\bullet k}} (x_{ik} - x_{i'k})^2.$$

Autrement dit, deux individus seront proches s'ils possèdent les mêmes modalités, en particulier s'ils ont en commun des modalités rares ($n_{\bullet k}$ petit).

Distance entre deux modaliés

pour calculer la distance entre deux modalités, on se place dans le nuage des profils-colonnes et on utilise la métrique du chi-deux nD_r^{-1}. On a :

$$d^2_{\chi^2}(k,k') = \sum_{i=1}^{n} \frac{np}{n_{i\bullet}} \left(\frac{x_{ik}}{n_{\bullet k}} - \frac{x_{ik'}}{n_{\bullet k'}} \right)^2 = n \sum_{i=1}^{n} \left(\frac{x_{ik}}{n_{\bullet k}} - \frac{x_{ik'}}{n_{\bullet k'}} \right)^2 .$$

Deux modalités sont proches si elles sont possédées par les mêmes individus.

3.2.2 Inertie totale

En reprenant les notations de l'AFC, appliquées au TDC $\mathbf{X}$, et en remarquant que la somme par ligne $n_{i\bullet}$ vaut p pour tous les individus, on a :

$$\begin{aligned}
I_T &= \sum_{i=1}^{n} \frac{n_{i\bullet}}{np} d^2_{\chi^2}(i, g_r) \\
&= \sum_{i=1}^{n} \sum_{k=1}^{m} \frac{n_{i\bullet}}{np} \frac{np}{n_{\bullet k}} \left(\frac{x_{ik}}{n_{i\bullet}} - \frac{n_{\bullet k}}{np} \right)^2 \\
&= \sum_{i=1}^{n} \sum_{k=1}^{m} \frac{p}{n_{\bullet k}} \frac{1}{p^2} \left(x_{ik} - \frac{n_{\bullet k}}{n} \right)^2 \\
&= \frac{1}{p} \sum_{i=1}^{n} \sum_{k=1}^{m} \frac{1}{n_{\bullet k}} \left(x_{ik}^2 - 2x_{ik} \frac{n_{\bullet k}}{n} + \frac{n_{\bullet k}^2}{n^2} \right) \\
&= \frac{1}{p} \sum_{i=1}^{n} \sum_{k=1}^{m} \frac{x_{ik}}{n_{\bullet k}} - \frac{2}{np} \sum_{i=1}^{n} \sum_{k=1}^{m} x_{ik} + \frac{1}{pn^2} \sum_{i=1}^{n} \sum_{k=1}^{m} n_{\bullet k} \\
I_T &= \frac{m}{p} - 1.
\end{aligned}$$

On remarque alors que l'inertie totale ne dépend que du nombre de classes et du nombre de variables. Elle n'a pas d'interprétation statistique comme en ACP ou en AFC.

Remarque. *La distance entre une modalité k et le centre de gravité du nuage des profils-colonnes est :*

$$d^2_{\chi^2}(k, g_c) = \sum_{i=1}^{n} \frac{np}{n_{i\bullet}} \left(\frac{x_{ik}}{n_{\bullet k}} - \frac{1}{n} \right)^2 = \frac{n}{n_{\bullet k}} - 1.$$

Plus une modalité est rare, plus elle est loin du centre de gravité du nuage. De même, si on exprime l'inertie totale en utilisant les points du nuage des profils-colonnes, on a

$$I_T = \sum_{k=1}^{m} \frac{n_{\bullet k}}{np} d^2_{\chi^2}(k, g_c).$$

La part d'inertie due à la modalité k est $\frac{n_{\bullet k}}{np} d^2_{\chi^2}(k, g_c) = \frac{1}{p}(1 - \frac{n_{\bullet k}}{n})$: plus une modalité est rare, plus la part d'inertie due à cette modalité est importante.

Pour ces raisons, il peut être intéressant, dans certains cas, de regrouper entre elles les catégories d'une même variable.

Remarque. *La part d'inertie due à une variable j est la somme des inerties dues à chacune de ses m_j modalités. Cette part d'inertie est égale à $\frac{1}{p}(m_j - 1)$: plus une variable a de modalités, plus la part d'inertie liée à cette variable est importante.*
Dans la pratique, on évitera d'avoir des déséquilibres trop importants dans le nombre de modalités des variables (par exemple, des variables avec peu de modalités et d'autres avec un grand nombre de modalités).

3.3 Propriétés des valeurs propres

3.3.1 Premier critère pour le choix du nombre d'axes

De par la nature des données du TDC (variables binaires, colonnées démultipliées par le nombre de modalités, redondacne de l'information, ...), il est difficile de concentrer l'inertie sur les premiers facteurs de l'ACM. Cependant, on peut proposer un premier seuil pour éliminer certaines valeurs propres. En effet, en supposant que la matrice $\mathbf{X}$ est de plein rang on a $q = m - p$ valeurs propres non nulles. Dans ce cas :

$$\sum_{k=1}^{q} \lambda_k = I_T = \frac{m}{p} - 1 \Leftrightarrow \frac{1}{q}\sum_{k=1}^{q} \lambda_k = \frac{1}{p}.$$

Autrement dit, une valeur propre vaut *en moyenne* $1/p$. On peut donc sélectionner dans un premier temps les valeurs propres supérieures à ce seuil, c'est-à-dire celles qui contribuent plus que la moyenne. En revanche, comme les pourcentage d'inertie expliqués par chaque axes sont petits par construction de l'ACM, on n'utilise pas en pratique le critère du pourcentage d'inertie pour choisir le nombre d'axes à retenir.

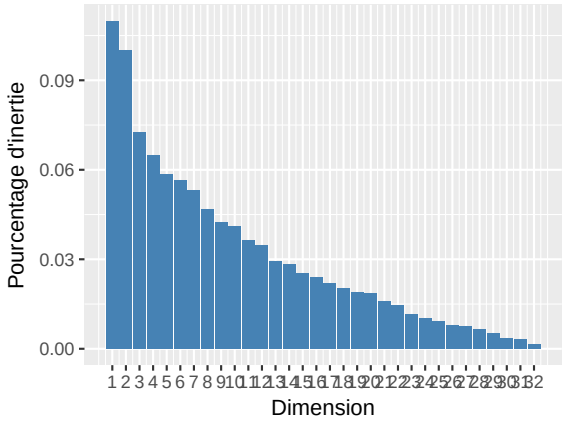
3.3.2 Correction des valeurs propres

On a vu dans la section précédente qu'il n'est pas possible en ACM d'interpréter l'inertie totale de façon aussi intéressante qu'en ACP ou AFC. Par contre, on peut relier la somme des *carrés* des valeurs propres à des indices statistiques. En effet, si on note $\lambda_k, 1 \leq k \leq q$ les valeurs propres de $V_r M_r$, avec q le rang de la matrice $V_r M_r$, alors λ_k^2 est valeur propre de $(V_r M_r)^t (v_r M_r)$. On peut alors montrer que :

$$\sum_{k=1}^{q} \lambda_k^2 = \frac{1}{p^2}\sum_{j=1}^{p}(m_j - 1) + \frac{1}{p^2}\sum_{j=1}^{p}\sum_{\substack{j'=1 \\ j' \neq j}}^{p} \varphi_{jj'}^2, \tag{3.1}$$

où $\varphi_{jj'}^2$ est le coefficient de Pearson pour le croisement des variables X_j et $X_{j'}$ (voir equation (2.6) au chapitre 2).

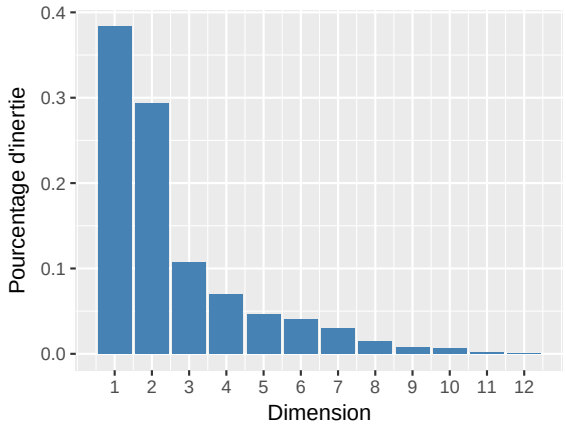
Or, on a vu au début de ce chapitre que la tableau de Burt pouvait s'obtenir à partir du TDC : $B = X^t X$. En faisant une AFC du tableau de Burt, on va alors récupérer les valeurs propres $\lambda_k^2, 1 \leq k \leq q$. Cette pratique est courante dans le monde anglo-saxon, et présente l'avantage de fournir une meilleure



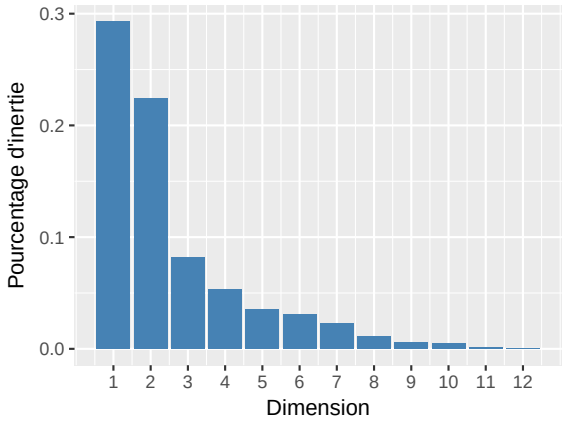
(a) *À partir des valeurs propres initiales*

Axe	Sans correction	Benzécri	Greenacre
1	10.98	38.37	29.32
2	9.99	29.30	22.39
3	7.27	10.66	8.15
4	6.48	6.99	5.34
5	5.85	4.60	3.52
6	5.66	4.00	3.06

(b) *Pourcentage d'inertie cumulée*



(c) *Après correction de Benzécri*



(d) *Après correction de Greenacre*

FIGURE 3.1 – Pourcentages d'inertie obtenus à partir des valeurs propres initiales obtenues par l'ACP des profils-lignes du TDC, ou après correction de Benzécri et de Greenacre.

interprétation des valeurs propres, mais rend l'interprétation en terme d'individus plus difficile car non immédiate.

Deux corrections ont été proposées pour améliorer les pourcentages d'inertie expliqués par chaque axe, en partant des résultats obtenus dans le cas d'une AFC du tableau de Burt. La figure 3.1 présente un exemple d'application de ces corrections.

Correction de Benzécri

En utilisant les propriétés du tableau de Burt et l'interprétation des valeurs propres de l'AFC de ce tableau, Benzecri (1979) a proposé la correction suivante :

1. sélectionner les ℓ valeurs propres supérieures au seuil $1/p$
2. calculer les valeurs propres *corrigées* :

$$\tilde{\lambda}_k = \left[\left(\frac{p}{p-1}\right)\left(\lambda_k - \frac{1}{p}\right)\right]^2$$

3. calculer la somme des valeurs propres corrigées
4. tracer l'éboulis des valeurs propres corrigées en traçant le pourcentage d'inertie cumulée corrigée

Correction de Greenacre

La correction de Benzécri permet d'augmenter la part d'inertie expliquée par les premiers axes, mais elle a tendance à légèrement sur-estimer cette part. C'est pourquoi, Greenacre (1993) a proposé une correction supplémentaire pour le calcul de l'inertie expliquée par chaque axe.

Quand la correction de Benzécri propose de calculer le pourcentage d'inertie d'un axe k par $\frac{\tilde{\lambda}_k}{\sum_{k=1}^{\ell}\tilde{\lambda}_k}$, Greenacre propose d'utiliser :

$$\frac{\tilde{\lambda}_k}{I_G}, \quad \text{où} \quad I_G = \left[\left(\frac{p}{p-1}\right)\left(\sum_{k=1}^{\ell}\lambda_k^2 - \frac{m-p}{p^2}\right)\right]^2.$$

3.4 Interprétation des résultats

3.4.1 Contributions des individus et des modalités

Définition 4. *La contribution d'un individu i à l'axe factoriel l est donnée par :*

$$ctr_l(i) = \frac{n_{i\bullet}}{np}\frac{a_i^2}{\lambda_l} = \frac{a_{il}^2}{n\lambda_l},$$

où a_{il} est la coordonnée de l'individu i sur l'axe factoriel l. La contribution d'une modalité k à l'axe factoriel l est données par :

$$ctr_l(k) = \frac{n_{\bullet k}}{np}\frac{a_{kl}^2}{\lambda_l},$$

où a_{kl} est la coordonnées de la modalité k sur l'axe factoriel l.

La contribution d'une variable est égale à la somme des contributions de chacune de ses modalités. On remarque que la contribution d'une modalité à un axe dépend de sa fréquence : on peut alors retenir les modalités qui contribuent plus que leurs poids, i.e. celles pour lesquelles$|a_{kl}| > \sqrt{\lambda_k}$.

On retient que ce sont les modalités et les individus les plus excentrés sur les axes qui contribuent le plus.

3.4.2 Rapport de corrélation

En ACP, on mesure la corrélation entre les variables initiales et les composantes principales, toutes les variables étant quantitatives. En ACM, les variables initiales étant qualitatives et les composantes principales quantitatives, on a besoin d'un autre critère statistique pour étudier la liaison entre les variables et les composantes principales.

Le rapport de corrélation est un critère reposant sur la décomposition de l'inertie totale d'une variable quantitative y selon les modalités d'une variable qualitative X, en une somme de l'inertie inter-classes et des inerties intra-classes, (décomposition due au théorème de Huygens). L'inertie inter-classes

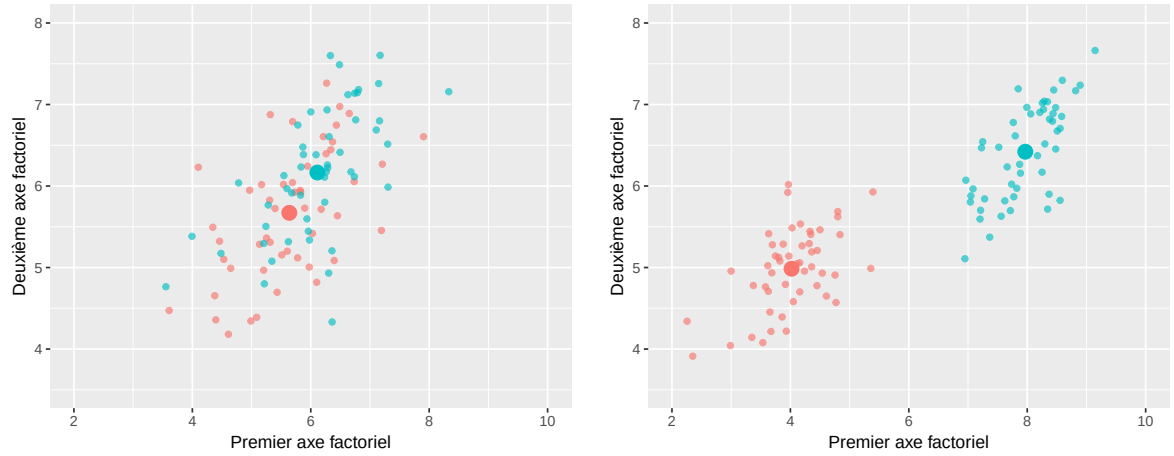


FIGURE 3.2 – Illustration de deux situations correspondant à un rapport de corrélation proche de 0 (à gauche) et proche de 1 (à droite).

correspond à l'inertie des centres de gravité des classes définies par les modalités de X, et les inerties intra-classes correspondent à l'inertie des individus d'une classe par rapport au centre de gravité de la classe à laquelle ils appartiennent. Le rapport de corrélation est défini comme le ratio entre l'inertie inter-classes et l'inertie totale de la variable$Y$, et varie entre 0 et 1.

Lorsque ce ratio est proche de 1, cela signifie que l'inertie inter-classes est élevée et que les inerties intra-classes sont faibles : autrement dit, les individus d'une même classe sont regroupés autour du centre de gravité de la classe, et les classes sont bien séparées les unes des autres. Dans ce cas, il y a une liaison forte entre la variable qualitative X et la variable quantitative Y. En revanche, lorsque ce ratio est proche de 0, cela signifie que l'inertie inter-classes et faible et qu'au moins une des inerties intra-classes est élevée : les classes sont alors proches les unes des autres, et les individus dispersés au sein des classes. C'est une situation où les variables X et Y ne sont pas liées. On trouvera une illustration de ces deux situations sur la figure 3.2.

Le rapport de corrélation entre la variable j et l'axe factoriel l est donné par :

$$R_l(j) = \frac{1}{\lambda_l} \sum_{k=1}^{m_j} \frac{n_{\bullet k}}{n}.$$

3.4.3 Relations barycentriques

Il est possible de représenter les individus et les modalités sur le même graphe. On a alors l'interprétation suivante :

— les individus sont au barycentre des modalités qu'ils possèdent

— les modalités sont au barycentre des individus qui les possèdent

Deuxième partie

Classification supervisée
Analyse discriminante

Introduction

En classification supervisée, on connaît la répartition des individus dans des classes, et l'objectif consiste à expliquer l'appartenance à une classe en fonction d'un certain nombre d'autres variables. Plus précisément, on cherche à expliquer une variable qualitative Y à K modalités (la variable représentant le numéro de classe) à l'aide d'un ensemble de p prédicteurs notés $X_1, \ldots, X_p$.

Les objectifs d'une analyse discriminante peuvent être de deux natures, descriptive ou prédictive. Dans le premier cas, on cherche à décrire chacune des classes de la population, dans l'autre, on cherche à prédire l'appartenance à chacune des classes. On parle également *d'analyse discriminante*, ou de *scoring*. Ce type de méthodes est très utilisé par exemple en marketing, lorsqu'il s'agit de décrire les clients d'une base de données pour savoir s'il ont acheté ou non un produit donné, ou dans le domaine bancaire, pour savoir si un client est à risque au moment de lui octroyer un crédit.

On distingue deux grands types de méthodes, d'une part les méthodes géométriques, plutôt adaptées à un objectif descriptif et dont l'analyse factorielle discriminante est l'exemple phare (chapitre 1), et d'autre part les méthodes probabilistes où l'on cherche à modéliser la probabilité d'appartenance à chaque classe, plutôt adaptées à un objectif prédictif (chapitre 2). Enfin, parmi les méthodes probabilistes, on distingue les méthodes paramétriques, semi-paramétriques ou non paramétriques. Notons toutefois que la frontière entre méthodes descriptives et prédictives n'est pas toujours très nette. Ainsi, il est possible de définir des règles d'affectation géométriques à partir des résultats d'une analyse factorielle discriminante (voir section 1.4).

Chapitre 1

Analyse factorielle discriminante

L'analyse factorielle discriminante (AFD) est une méthode géométrique dont l'objectif est d'identifier un axe permettant de discriminer au mieux les différentes classes. En cela, elle ressemble aux méthodes factorielles de type ACP, et à la classification automatique.

1.1 Notations

Rappelons que nous cherchons à modéliser une variable **qualitative** à K modalités à l'aide d'un ensemble de p variables explicatives **quantitatives**. On suppose donc que l'on dispose d'un échantillon de n observations de la variable à expliquer Y et des variables explicatives $X_1, \ldots, X_p$. On note $\mathbf{y} = (y_1, \ldots, y_n)^t$ le vecteur des valeurs de la variable Y mesurées sur tous les individus, et $\mathbf{X}$ la matrice contenant les observations des variables explicatives, avec $(\mathbf{X})_{i,j} = x_{ij}$ la valeur de la j-ème variable mesurée sur le i-ème individu, et $\mathbf{x}_i = (x_{i1}, \ldots, x_{ip})^t \in \mathbb{R}^p$ le vecteur des observations de l'individu i (i.e., la i-ème ligne de la matrice $\mathbf{X}$). On suppose également (ce sera souvent le cas en pratique) que $p < n$.

1.1.1 Variances globale, inter et intra-classe

On va commencer par introduire quelques notations. On note $\mathbf{g} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i$ le centre de gravité du nuage de points, et $\mathbf{g}_k = \frac{1}{n_k}\sum_{\mathbf{x}_i \in C_k} \mathbf{x}_i$ le centre de gravité de la classe C_k, $k = 1, \ldots, K$.

— La matrice $p \times p$ de covariance globale V_T, est définie par :

$$V_T = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i \mathbf{x}_i^t - \mathbf{g}\mathbf{g}^t. \tag{1.1}$$

— La matrice de covariance de la classe C_k est définie par :

$$V_k = \frac{1}{n_k}\sum_{\mathbf{x}_i \in C_k} \mathbf{x}_i \mathbf{x}_i^t - \mathbf{g}_k \mathbf{g}_k^t. \tag{1.2}$$

— Les matrices de covariance intra- et inter classes V_W et V_B sont données respectivement par :

$$V_W = \frac{1}{n}\sum_{k=1}^{K} n_k V_k, \qquad V_B = \frac{1}{n}\sum_{k=1}^{K} n_k \mathbf{g}_k \mathbf{g}_k^t - \mathbf{g}\mathbf{g}^t. \tag{1.3}$$

On a également la relation suivante reliant les variances totale, intra et inter-classes :

$$V_T = V_B + V_W. \tag{1.4}$$

Remarque. — *Il est classique de supposer, comme en analyse en composantes principales, que les données sont centrées, c'est-à-dire que le centre de gravité du nuage de points* $\mathbf{g}$ *est situé à l'origine. Si ce n'est pas le cas, il suffit de translater le nuage de points, ce qui ne modifie évidemment pas sa structure mais simplifie grandement les calculs. En particulier, les matrices de covariance globale et inter classes s'écrivent dans ce cas :*

$$V_T = \frac{1}{n}\sum_{k=1}^{n} \mathbf{x}_i \mathbf{x}_i, \qquad V_B = \frac{1}{n}\sum_{k=1}^{K} n_k \mathbf{g}_k \mathbf{g}_k^t. \tag{1.5}$$

— *On a considéré jusqu'à présent que tous les individus avaient le même poids* $1/n$*. Il est possible de généraliser les résultats précédents en supposant que l'individu* i *est affecté du poids* p_i*. Les poids de chaque classe sont alors donnés par* $q_k = \sum_{x_i \in C_k} pi$*, et les quantités définies précédemment s'écrivent :*

$$\mathbf{g}_k = \frac{1}{q_k}\sum_{x_i \in C_k} p_i \mathbf{x}_i, \qquad \mathbf{g} = \sum_{i=1}^{n} p_i \mathbf{x}_i, \qquad V_k = \frac{1}{q_k}\sum_{x_i \in C_k} p_i \mathbf{x}_i \mathbf{x}_i^t - \mathbf{g}_k \mathbf{g}_k^t,$$

$$V_B = \sum_{k=1}^{K} q_k \mathbf{g}_k \mathbf{g}_k^t - \mathbf{g}\mathbf{g}^t, \qquad V_W = \sum_{k=1}^{K} q_k V_k.$$

Dans ce cas, on ne parle plus de variance, mais de variance pondérée ou de matrice d'inerties.

1.1.2 Choix de la distance

On a maintenant besoin de définir une distance entre les individus de notre échantillon. Pour cela, on introduit la forme quadratique suivante, définie pour une matrice M symétrique, définie positive de taille $p \times p$:

$$d^2(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i - \mathbf{x}_j)^t M (\mathbf{x}_i - \mathbf{x}_j). \tag{1.6}$$

Le choix de la matrice M est laissé à l'utilisateur, et peut permettre de pondérer différemment les différentes coordonnées, en particulier lorsque les variables n'ont pas les mêmes unités.

1.1.3 Inertie

L'inertie totale I_T du nuage de points est définie comme la somme des distances au carré entre chaque point du nuage et son centre de gravité. Autrement dit,

$$I_T = \frac{1}{n}\sum_{i=1}^{n} (\mathbf{x}_i - \mathbf{g})^t M (\mathbf{x}_i - \mathbf{g}). \tag{1.7}$$

On montre alors facilement que $I_T = \text{Trace}\, V_T M$.

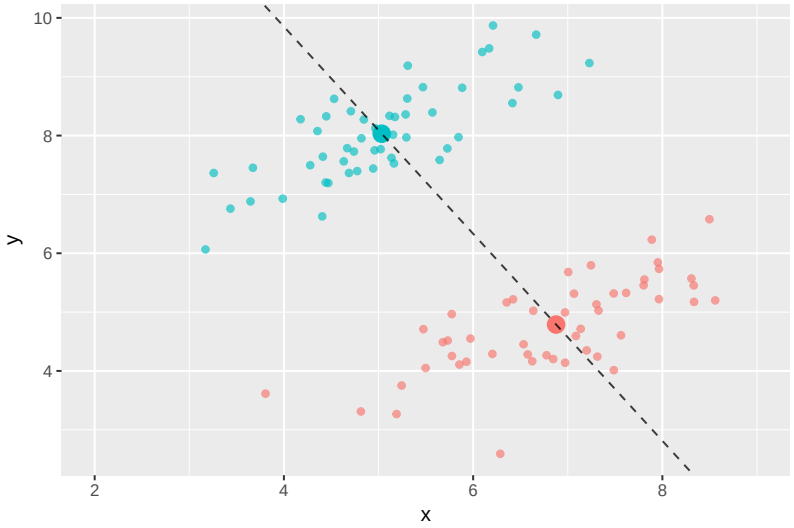


FIGURE 1.1 – Discrimination des deux groupes : la meilleure séparation est obtenue après projection sur l'axe en pointillés.

1.2 Principes généraux

L'objectif de l'AFD est de trouver de nouvelles variables, aussi appelées *variables discriminantes*, correspondant à des directions qui permettent de séparer le mieux possible les K groupes après projection sur les axes ainsi définis. La figure 1.1 permet d'illustrer la problématique. La projection selon l'axe en pointillés permet d'obtenir la meilleure séparation entre les groupes.

On note Δu l'axe de vecteur directeur u. On peut alors décomposer l'inertie projetée sur Δu avec la métrique M :

$$\underbrace{u^t M V_T M u}_{\text{variance totale projetée sur } u} = \underbrace{u^t M V_B M u}_{\text{variance inter-classes projetée sur } u} + \underbrace{u^t M V_W M u.}_{\text{variance intra-classes projetée sur } u} \tag{1.8}$$

Un axe aura un fort pouvoir discriminant si la projection du nuage de points sur cet axe produit des groupes bien séparés et homogènes. Autrement dit, on cherche à maximiser le terme d'inertie inter-classes $u^t M V_B M u$ et à minimiser le terme d'inertie intra-classes $u^t M V_W M u$. En pratique, cela peut se traduire par la maximisation du critère suivant :

$$g(u) = \frac{u^t M V_B M u}{u^t M V_T M u}. \tag{1.9}$$

Théorème 1. *Si la matrice V_T est inversible, alors g est maximale pour v vecteur propre de $V_T^{-1} V_B$ associé à sa plus grande valeur propre λ_1, avec $\lambda_1 = \max_u g(u)$.*

Démonstration.

$$\frac{\partial g(u)}{\partial u} = 0 \Leftrightarrow \frac{(u^t M V_T M u)(2 M V_B M u) - (u^t M V_B M u)(2 M V_T M u)}{(u^t M V_T M u)^2} = 0$$

$$\Leftrightarrow (u^t M V_T M u) M V_B M u = (u^t M V_B M u) M V_T M u$$

$$
\begin{aligned}
&\Leftrightarrow MV_BMu = g(u)MV_TMu \\
&\Leftrightarrow (MV_TM)^{-1}MV_BMu = g(u)u \\
&\Leftrightarrow M^{-1}V_T^{-1}V_BMu = g(u)u \\
&\Leftrightarrow V_T^{-1}V_B(Mu) = g(u)(Mu) \\
&\Leftrightarrow V_T^{-1}V_Bv = \lambda_1 v,
\end{aligned}
$$

avec $v = Mu$ et $\lambda_1 = \max_u g(u)$. Maximiser $g(u)$ revient donc à prendre le vecteur propre de $V_T^{-1}V_B$ associé à la plus grande valeur propre λ_1. □

Remarque. *Le choix de la métrique M n'intervient pas dans le choix de l'axe le plus discriminant. On prendra donc, pour simplifier, $M = I$ dans la suite du document.*

Définition 5. *Le premier facteur discriminant est u_1, le vecteur propre associé à la la plus grande valeur propre de $V_T^{-1}V_B$. La première variable discriminante est $v_1 = \mathbf{X}u_1$, et le pouvoir discriminant de l'axe Δu_1 est λ_1.*

Nombre d'axes discriminants

Après avoir identifié le premier axe discriminant, il est possible de déterminer les axes discriminants suivants en cherchant les autres valeurs et vecteurs propres de la matrice $V_T^{-1}V_B$. Par ailleurs, on a $\text{rang}(V_T^{-1}V_B) \leq \min(\text{rang}\, V_T, \text{rang}\, V_B)$, c'est-à-dire $\text{rang}(V_T^{-1}V_B) \leq \min(p, k-1)$, si on suppose que la matrice des données $\mathbf{X}$ est de plein rang p. Si le nombre de classes est plus petit que le nombre de paramètres, plus spécifiquement si $k - 1 < p$, on aura alors **au plus** $k - 1$ axes discriminants.

Propriétés

- Comme on a $0 \leq u^tV_Bu \leq u^tV_Tu$, i.e. $0 \leq g(u) \leq 1$, on a $\lambda_1 \in [0, 1]$.
- Si $\lambda_1 = 0$, alors $u_1^tV_Bu_1 = 0$, ce qui signifie que l'inertie inter-classes après projection sur Δu_1 est nulle. En supposant que les $\mathbf{x}_i$ sont tous distincts, cela signifie que le meilleur axe discriminant ne permet pas de séparer les centres de gravité des classes qui sont confondus. Il peut cependant exister une séparation non linéaire, par exemple si les classes sont des centres concentriques.
- Si $\lambda_1 = 1$, alors $u_1^tV_Bu_1 = u_1^tV_Tu_1$, ce qui signifie que l'inertie intra-classes après projection sur Δu_1 est nulle, et donc que les projections des $\mathbf{x}_i$ sur l'axe discriminant Δu_1 sont confondues avec les centres de gravité des classes $\mathbf{g}_k$. La discrimination est alors parfaite, si les projections des centres de gravité sont elle-mêmes distinctes.
- La valeur propre λ_1 est une mesure pessimiste du pouvoir discriminant, car on peut avoir $\lambda < 1$ même s'il est possible de discriminer parfaitement les groupes.

Remarque. *La matrice V_T doit être inversible, et donc la matrice des données $\mathbf{X}$ doit être de plein rang. Il faut donc faire attention aux éventuels problèmes de multicolinéarité des colonnes de $\mathbf{X}$.*

Lien avec l'analyse en composantes principales

L'objectif d'une analyse en composantes principales est d'identifier le meilleur sous-espace de faible dimension permettant de représenter la variabilité observée dans notre jeu de données multivariées. En présence de p variables, on cherche donc un ensemble de q combinaisons linéaires des p variables initiales, avec $q < p$, permettant de capturer l'essentiel de la variabilité de notre nuage de points.

L'objectif d'une analyse factorielle discriminante est de trouver des combinaisons linéaires des p variables initiales qui permettent de mieux séparer les groupes présents au sein de notre nuage de points. S'il y a K groupes, avec $K < p$, on pourra alors trouver au maximum $K - 1$ variables discriminantes.

Cependant, l'AFD peut être vue comme une analyse en composantes principales de la matrice des centres de gravités des classes avec la métrique V_T^{-1}. Les variables discriminantes obtenues sont donc **non corrélées** entre elles. De plus, il est possible d'interpréter les variables discriminantes à l'aide du cercle des corrélations.

Présence de variables qualitatives

On a supposé jusqu'à présent que les p variables étaient continues. En présence de variables qualitatives, plusieurs stratégies sont possibles :

— si on dispose uniquement de variables qualitatives, on peut utiliser par exemple la méthode Disqual, introduite par Gilbert Saporta. Cette méthode consiste à effectuer tout d'abord une analyse des correspondances du tableau des indicatrices obtenues à partir des différentes modalités de chaque variable qualitative, puis à réaliser une AFD sur les variables définies par les coordonnées des variables qualitatives initiales sur les axes factoriels obtenus dans l'analyse des correspondances. D'autres méthodes sont également disponibles, comme l'analyse discriminante barycentrique. On pourra se reporter par exemple à l'ouvrage de Gilles Celeux et Jean-Pierre Nakache, *Analyse discriminante sur variables qualitatives*.

— si on dispose à la fois de variables qualitatives et quantitatives, on peut soit transformer les variables quantitatives en variables qualitatives et appliquer la méthode Disqual, soit se reporter sur des méthodes valables dans le cas mixte, comme par exemple la régression logistique.

1.3 Conventions pratiques

Il existe une autre convention, plutôt répandue dans le monde anglo-saxon, et également dans certains logiciels de traitement statistique, qui consiste à maximiser le critère

$$\tilde{g}(u) = \frac{u^t V_B u}{u^t V_W u}.$$

Comme on a $V_T = V_W + V_B$, maximiser ce critère est équivalent à maximiser (1.9). En effet, cela revient à chercher la valeur propre maximale μ_1 de $V_W^{-1} V_B$, et l'on peut montrer facilement que les vecteurs propres de $V_W^{-1} V_B$ sont les mêmes que ceux de $V_T^{-1} V_B$, avec $\mu_1 = \frac{\lambda_1}{1-\lambda_1}$. L'avantage de l'approche décrite dans les sections précédentes est que les valeurs propres de $V_T^{-1} V_B$ sont comprises entre 0 et 1, alors

que celles de $V_W^{-1}V_B$ sont simplement positives ou nulles. L'interprétation des valeurs propres de $V_T^{-1}V_B$ est donc plus facile.

Par exemple, la fonction `lda` du package `MASS` de `R` calcule la décomposition de $V_W^{-1}V_B$ en *valeurs singulières*, et la fonction `discrimin` du package `ade4` calcule la décomposition de $V_T^{-1}V_B$ en valeurs propres.

Une autre convention utilisée dans certains logiciels statistiques consiste à définir des variables discriminantes telles que la variance intra-classes vaut 1. C'est le cas notamment du logiciel SAS et de la procédure `CANDISC`. Dans ce cas, la variance inter-classes et la variance totale après projection sur l'axe Δu_1 valent respectivement $u_1^t V_B u_1 = \frac{\lambda_1}{1-\lambda_1}$ et $u_1^t V_T u_1 = \frac{1}{1-\lambda_1}$.

1.4 Règles géométriques d'affectation

Même si l'AFD est une méthode dite descriptive, il est toutefois possible, une fois les axes discriminants construits, d'utiliser des règles géométriques permettant d'affecter un nouvel individu à l'un des k classes de notre échantillon. Un exemple simple d'une telle règle consiste à affecter l'individu à la classe dont le centre de gravité est le plus proche. Il faut alors choisir une métrique M pour calculer la distance entre ce nouveau point et les centres de gravité des classes. La règle de Mahalanobis consiste à choisir $M = V_T^{-1}$.

L'un des inconvénients principal de la règle d'affectation définie ci-dessus est qu'elle ne tient pas compte de l'importance de chaque groupe, mais uniquement de la distance entre un nouveau point et les centres de gravité des classes. Par exemple, lorsque les dispersions intra-classes sont très différentes, l'utilisation de la métrique de Mahalanobis n'est pas toujours la plus adaptée, car on aimerait pouvoir "pondérer" chaque classe en fonction de sa dispersion. Dans ce cas, on peut proposer par exemple d'utiliser des métriques "locales" $M_i = V_k^{-1}$, pour calculer la distance entre un point et le centre de gravité de la classe C_i.

Cependant, aucune notion de risque d'erreur n'est associée à ce type de règle géométrique. Pour cela, il est en effet nécessaire de se placer dans un cadre probabiliste.

Chapitre 2

Méthodes discriminantes probabilistes

Dans ce chapitre, on s'intéresse à la discrimination probabiliste, dont l'objectif est d'estimer une règle de classement permettant d'affecter tout nouvel individu à l'une des classes de l'échantillon. Dans un premier temps, on introduit le formalisme probabiliste ainsi que les notations qui seront utilisées dans cette section. Puis, on présente trois approches probabilistes : une méthode paramétrique avec le modèle de mélange Gaussien (section 2.3), une méthode semi-paramétrique avec la régression logistique (section 2.4) et une méthode non paramétrique avec la méthode des k plus proches voisins (section 2.5).

2.1 Formalisation probabiliste du problème

2.1.1 Notations

On va d'abord introduire un certain nombre de notations qui seront utilisées dans cette section. On note $X \in \mathbb{R}^p$ le vecteur aléatoire d'observations d'un individu, et $Y \in \{1, \dots, K\}$ la variable aléatoire indiquant la classe à laquelle appartient l'individu. On note f_k la densité conditionnelle de X sachant qu'il appartient à la classe k, c'est-à-dire :

$$X \mid Y = k \sim f_k,$$

On note p_k la probabilité qu'a *a priori* un individu i de se trouver dans la classe k, i.e. $\mathbb{P}(Y = k) = p_k$, avec $\sum_k p_k = 1$ (on suppose qu'un individu ne peut appartenir qu'à une seule classe).
La loi jointe du couple (X, Y) est $p_k f_k$, et la loi marginale de X, notée f_X, est :

$$f_X(\mathbf{x}) = \sum_{k=1}^{K} p_k f_k(\mathbf{x}).$$

Finalement, on définit la probabilité *conditionnelle* que X soit dans la classe C_k, sachant que l'on a observé $X = \mathbf{x}$, par :

$$t_k(\mathbf{x}) = \mathbb{P}(Y = k \mid X = \mathbf{x}) = \frac{p_k f_k(\mathbf{x})}{f_X(\mathbf{x})}.$$

Il s'agit de la loi conditionnelle de la variable Y, sachant X.

2.1.2 Eléments de la théorie de la décision

Définition 6. *On appelle* règle d'allocation *une fonction* $r : \mathbb{R}^p \mapsto \{1, \dots, K\}$ *qui à chaque individu* $\mathbf{x}$ *associe un numéro de groupe* $r(\mathbf{x})$. *On parle aussi de règle de décision, ou de règle d'affectation.*

Définition 7. *On appelle* fonction de perte *une fonction* $L : \{1, \dots, K\}^2 \to \mathbb{R}_+$, *où* $L(k, \ell)$ *mesure la perte que l'on subit en classant un individu provenant de la classe* C_ℓ *dans la classe* C_k.

On impose $L(k, k) = 0$, $k = 1 \dots, K$, c'est-à-dire qu'il n'y a pas de perte en cas de bon classement. Parmi les fonctions de perte les plus couramment utilisées, on citera notamment la fonction de perte 0-1, qui attribue une perte de 1 à chaque mauvais classement, et de 0 à chaque bon classement, autrement dit $L_{0-1}(k, \ell) = \mathbf{1}_{k \neq \ell}$. Pour une règle de classement r, la perte associée est alors $L(r(X), Y)$.

Définition 8. *Pour une règle de décision* r, *on définit la* fonction de risque *associée comme étant l'espérance de la fonction de perte :*

$$R(r) = \mathbb{E}_{(X,Y)} \left[L(r(X), Y) \right].$$

Dans le cas de la fonction de perte 0-1, on obtient $R(r) = \mathbb{E}_{(X,Y)}(\mathbf{1}_{r(X) \neq Y}) = \mathbb{P}(r(X) \neq Y)$.

Définition 9. *La* règle optimale de Bayes *est la règle* r^* *qui minimise la fonction de risque :*

$$r^* \in \arg\min_{r \in \Delta} R(r),$$

où Δ *est l'ensemble des règles de décision.*

Proposition 2. *Si on peut minimiser la fonction de risque conditionnel* $r \mapsto \mathbb{E}_{Y|X=x}(L(r(X), Y) \mid X = \mathbf{x})$ *pour tout* $\mathbf{x} \in \mathbb{R}^p$, *alors la règle définie par :*

$$r^*(x) = \arg\min_r \mathbb{E}_{Y|X=x}(L(r(X), Y) \mid X = \mathbf{x})$$

est une règle optimale de Bayes.

Démonstration. En effet, on remarque tout d'abord que

$$R(r) = \mathbb{E}_{(X,Y)} \left[L(r(X), Y) \right] = \mathbb{E}_X \left[\mathbb{E}_{Y|X=x}(L(r(X), Y) \mid X = \mathbf{x}) \right]. \tag{2.1}$$

Puis, comme $\mathbb{E}_{Y|X=x}(L(r^*(X), Y) \mid X = \mathbf{x}) \leq \mathbb{E}_{Y|X=x}(L(r(X), Y) \mid X = \mathbf{x})$ pour toute règle r, par définition de r^*, on a :

$$\forall r \in \Delta, R(r^*) \leq R(r). \tag{2.2}$$

Cela équivaut bien à dire que $r^* \in \arg\min_r R(r)$.

□

Dans la pratique, cela signifie qu'il suffit d'optimiser la règle de décision individu par individu. Avec la fonction de perte 0-1, minimiser le risque conditionnel correspond à trouver la règle r telle que

$$r(\mathbf{x}) = \arg\min_r \mathbb{P}(r(X) \neq Y \mid X = \mathbf{x})$$

Comme r est à valeurs dans $\{1, \ldots, K\}$, cela revient à trouver, pour un $\mathbf{x}$ donné, la valeur k telle que l'expression précédente soit minimale :

$$\begin{aligned} r(\mathbf{x}) &= \arg\min_k \mathbb{P}(k \neq Y \mid X = \mathbf{x}) \\ &= \arg\min_k [1 - \mathbb{P}(Y = k \mid X = \mathbf{x})] \\ &= \arg\max_k t_k(\mathbf{x}) \end{aligned}$$

Cela revient donc à affecter chaque individu à la classe C_k dont la probabilité conditionnelle $t_k(\mathbf{x})$ est maximale.

Remarque. *Si les probabilités a priori p_k sont égales ($p_1 = \cdots = p_K = \frac{1}{K}$), alors on a :*

$$r^*(\mathbf{x}) = k \quad si \quad f_k(\mathbf{x}) > f_{k'}(\mathbf{x}) \quad \forall k \neq k', \tag{2.3}$$

autrement dit, on affecte l'individu à la classe dont la vraisemblance conditionnelle est maximale.

Résumé des méthodes abordées

Nous allons voir dans les sections suivantes trois exemples de méthodes, qui abordent le problème de façon différente :

- en section 2.3, le **modèle de mélange Gaussien** (que l'on peut généraliser aux modèles de mélange paramétriques) qui repose sur l'hypothèse que les densités conditionnelles f_k appartiennent à une famille paramétrique de densités (dans le cas du modèle de mélange Gaussien, on suppose que ces densités sont Gaussiennes). Les paramètres du modèle sont alors les paramètres de ces densités conditionnelles, ainsi que les probabilités *a priori* d'appartenir à chaque classe, c'est-à-dire les p_k. On les estime en maximisant la vraisemblance *jointe* des couples d'observation $(X_i, Y_i), i = 1, \ldots, n$, c'est-à-dire en maximisant le produit des $p_k f_k(\mathbf{x}_i)$.
- en section 2.4, le **modèle de régression logistique** qui est une méthode *semi-paramétrique* ne faisant aucune hypothèse sur les densités conditionnelles f_k, mais uniquement sur la forme des ratios $\ln \frac{f_k}{f_K}$, pour $k = 1, \ldots, K-1$, qui sont supposés linéaires. On en déduit la linéarité des ratios $\ln \frac{t_k}{t_K}$, et on estime les coefficients de cette expression linéaire en maximisant la vraisemblance *conditionnelle* des $Y_i \mid X_i$, c'est-à-dire en maximisant le produit des $t_k(\mathbf{x}_i)$.
- en section 2.5, la **méthode des k-plus proches voisins**, qui est une méthode non paramétrique qui cherche à estimer directement les densités conditionnelles $t_1(\mathbf{x}), \ldots, t_K(\mathbf{x})$ pour une observation $\mathbf{x}$ donnée. Il n'y a pas de paramètres à estimer, car il s'agit d'une méthode non paramétrique.

2.2 Evaluation et validation d'un modèle

Quand on construit un modèle prédictif, il est essentiel d'évaluer sa capacité à se généraliser au-delà de l'échantillon d'observations sur lequel il a été construit. En effet, par construction la plupart des modèles auront tendance à sur-ajuster les données sur lesquelles ils ont été élaborés, mais s'avèreront moins performant sur de nouveaux jeux de données. Il s'agit en fait du fameux compromis biais-variance, omniprésent en statistique, et qui devra fuider, ici encore, le choix du modèle final retenu.

Afin d'évaluer les performances d'un modèle, on pourra procéder de deux manières différentes, selon la taille de l'échantillon initial.

Construction d'un échantillon d'apprentissage et d'un échantillon test

Lorsque l'on dispose d'un échantillon $\mathcal{D}_n$ de taille suffisante, on peut diviser le diviser en deux sous échantillons disjoints notés $\mathcal{D}_a$ et $\mathcal{D}_t$ et appelés respectivement ensemble d'apprentissage et ensemble de test. Notons n_a la taille de l'échantillon d'apprentissage et n_t la taille de l'échantillon test.

On construit dans un premier temps un estimateur de la règle de classement, noté $\hat{r}_a$. Puis, on estime le risque associé sur l'échantillon test par :

$$\hat{R}_t(\hat{r}_a) = \frac{1}{n_t} \sum_{x_i \in \mathcal{D}_t} L(\hat{r}_a(x_i^t), y_i^t), \tag{2.4}$$

où (x_i^t, y_i^t), $i = 1, \ldots, t$ sont les observations appartenant à l'échantillon test. Dans le cas de la fonction de perte 0-1, on a

$$\hat{R}_t(\hat{r}_a) = \frac{1}{n_t} \sum_{x_i \in \mathcal{D}_t} \mathbf{1}_{\hat{r}(x_i^t) \neq y_i^t},$$

ce qui correspondant à l'erreur de classement sur l'échantillon test.

Dans la pratique, on prend généralement $n_t \in [20\%n; 25\%n]$ et $n_a = n - n_t$. Cependant, pour obtenir de bonnes propriétés pour $\hat{r}$ à partir de $\mathcal{D}_a$, et de $\hat{R}_t$ à partir de $\mathcal{D}_t$, il faut que les deux échantillons soient de tailles suffisantes.

Validation croisée

Lorsque l'échantillon n'est pas suffisament grand, on peut utiliser la méthode de validation croisée aussi appelée "leave-one-out" : on construit n échantillons d'apprentissage de taille $n - 1$ on considérant à chaque fois tous les individus sauf 1, qui constitue l'échantillon test. On va donc estimer n fois la règle de classement sur ces n échantillons d'apprentissage, et estimer l'erreur de classement sur les n échantillons tests correspondant, en regardant simplement si l'individu de l'échantillon test est bien classé ou non. L'estimateur de la méthode de validation croisée est ensuite donné par :

$$\hat{e}_{cv} = \frac{1}{n} \sum_{i=1}^{n} \hat{e}_{\{i\}}, \tag{2.5}$$

où $\hat{e}_{\{i\}}$ est l'estimateur de l'erreur de classement obtenu sur l'échantillon test réduit à l'individu $\{i\}$.

On peut également généraliser cette approche en considérant n échantillons d'apprentissage de tailles $n - v$ et leurs n échantillons test correspondant de taille v. On parle alors de "v-fold cross-validation".

2.3 Modèle de mélange Gaussien

En reprenant les notations de la section précédente, on suppose dans cette section que la densité f_k est celle d'une loi Gaussienne multivariée $\mathcal{N}_p(\mu_k, \Sigma_k)$. Autrement dit,

$$f_k(\mathbf{x}) = \frac{1}{(2\pi)^{p/2} \mid \Sigma_k \mid^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \mu_k)^t \Sigma_k^{-1}(\mathbf{x} - \mu_k)\right). \tag{2.6}$$

Chaque classe est donc paramétrée par une moyenne μ_k et une matrice de covariance Σ_k.

On note $\theta = (p_1, \ldots, p_K, \mu_1, \ldots, u_K, \Sigma_1, \ldots, \Sigma_K)$ l'ensemble des paramètres à estimer.

Définition 10. *On parle d'homoscédasticité lorsque $\Sigma_1 = \cdots = \Sigma_K$, et d'hétéroscédasticité dans le cas contraire.*

2.3.1 Règle de classement

On se place dans un premier temps dans le cas où θ est connu. On définira alors une règle de classement théorique, et l'on verra dans la section suivante comment estimer cette règle en pratique, lorsque θ est inconnu.

Pour simplifier les notations, on présente le cas $K = 2$. Les résultats se généralisent à $K > 2$ classes sans difficulté particulière.

Rappelons que dans le cas de deux groupes, la règle optimale de classement de Bayes r^* est définie par :

$$r^*(\mathbf{x}) = 1 \Leftrightarrow t_1(\mathbf{x}) > t_2(\mathbf{x}) \Leftrightarrow \ln \frac{t_1(\mathbf{x})}{t_2(\mathbf{x})} > 0 \tag{2.7}$$

Posons $G_\theta(\mathbf{x}) = \ln \frac{t_1(\mathbf{x})}{t_2(\mathbf{x})}$, on peut alors définir l'équation de la surface discriminante par :

$$G_\theta(\mathbf{x}) = 0.$$

Or par définition de t_k, on a :

$$G_\theta(\mathbf{x}) = \ln \frac{f_1(\mathbf{x})}{f_2(\mathbf{x})} + \ln \frac{p_1}{p_2}.$$

En utilisant l'expression exacte de f_1 et f_2, et en remarquant que le deuxième terme ne dépend pas de x, on obtient l'équation suivante pour la surface discriminante :

$$G_\theta(\mathbf{x}) = 0 \iff \ln \frac{\mid \Sigma_2 \mid}{\mid \Sigma_1 \mid} - (\mathbf{x} - \mu_1)^t \Sigma_1^{-1}(\mathbf{x} - \mu_1) + (\mathbf{x} - \mu_2)^t \Sigma_2^{-1}(\mathbf{x} - \mu_2) + 2 \ln \frac{p_1}{p_2} = 0. \tag{2.8}$$

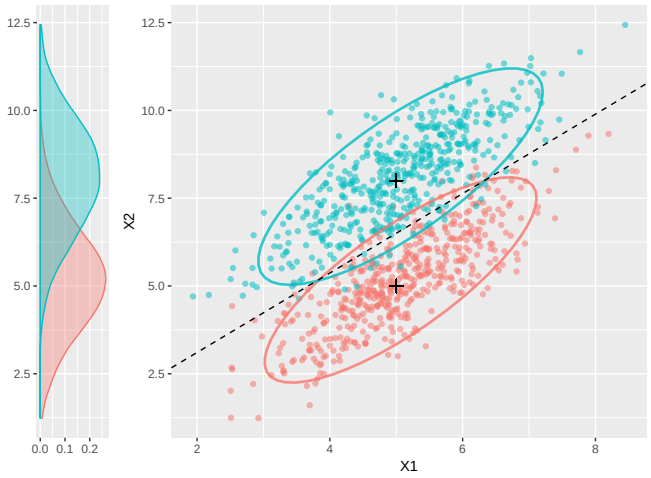


FIGURE 2.1 – Modèle de mélange Gaussien à deux groupes homoscédastiques. Si l'axe reliant les deux centres de gravité est celui permettant de minimiser la variance intra-groupe projetée et de maximiser la variance inter-groupe projetée (voir Chapitre sur l'AFD), il existe un fort recouvrement entre les distributions de chaque groupe après projection sur cet axe. La frontière de classement du modèle de mélange Gaussien (en tirets noirs) permet de minimiser le recouvrement entre les distributions après projection, minimisant ainsi le taux de mauvais classement.

Il s'agit d'une équation du second ordre en $\mathbf{x}$, la frontière de classement sera donc quadratique.

Dans le cas homoscédastique, i.e. lorsque $\Sigma_1 = \Sigma_2 = \Sigma$, l'équation (2.8) devient :

$$G_\theta(\mathbf{x}) = (\mu_1 - \mu_2)^t \Sigma^{-1}(\mathbf{x} - \frac{\mu_1 + \mu_2}{2}) + \ln \frac{p_1}{p_2}. = 0, \tag{2.9}$$

autrement dit la frontière de classement est linéaire dans ce cas (voir Figure 2.1).

Remarque. *La fonction G_θ est aussi appelée* ***fonction score****.*

Remarque. *Dans le cas où l'on a plus de deux groupes, on peut calculer de la même façon les log-ratios suivants :*

$$G_{k\ell}(\mathbf{x}) = \ln \frac{t_k(\mathbf{x})}{t_\ell(\mathbf{x})} \tag{2.10}$$

Inversement, on peut obtenir la probabilité ***a posteriori*** *à partir de la fonction score :*

$$t_1(\mathbf{x}) = \mathbb{P}(Y = 1 \mid X = \mathbf{x}) = \frac{\exp G_\theta(\mathbf{x})}{1 + \exp G_\theta(\mathbf{x})}. \tag{2.11}$$

Taux d'erreur théorique dans le cas homoscédastique

Dans le cas de deux groupes homoscédastiques, la règle de classement r_θ est déterminée par la surface discriminante définie par l'équation (2.9). Plus précisément, on a :

$$r_\theta(\mathbf{x}) = \begin{cases} 1 & \text{si } G_\theta(\mathbf{x}) > 0 \\ 2 & \text{si } G_\theta(\mathbf{x}) < 0 \end{cases}. \tag{2.12}$$

On a deux possibilités de se tromper lors du classement d'un individu : soit l'affecter à tort dans la classe C_1, soit l'affecter à tort dans la classe C_2. Calculons $e_1(r)$, la probabilité pour qu'un individu de la classe C_1 soit mal classé. On a :

$$\begin{aligned} e_1(r_\theta) &= \mathbb{P}(G_\theta(X) < 0 \mid Y = 1) \\ &= \mathbb{P}(G_\theta(X) < 0 \mid X \sim \mathcal{N}_p(\mu_1, \Sigma)) \end{aligned}$$

Comme $G(X)$ s'écrit comme une combinaison linéaire des composantes de X (dans le cas homoscédastique), conditionnellement à $X \sim \mathcal{N}_p(\mu_1, \Sigma)$, $G(X)$ suit une loi normale de paramètres :

$$\begin{aligned} \mathbb{E}(G_\theta(X)) &= (\mu_1 - \mu_2)^t \Sigma^{-1} \mathbb{E}\left(X - \frac{\mu_1 + \mu_2}{2}\right) + \ln \frac{p_1}{p_2} \\ &= (\mu_1 - \mu_2)^t \Sigma^{-1} \left(\mu_1 - \frac{\mu_1 + \mu_2}{2}\right) + \ln \frac{p_1}{p_2} \\ &= \frac{1}{2}(\mu_1 - \mu_2)^t \Sigma^{-1}(\mu_1 - \mu_2) + \ln \frac{p_1}{p_2}. \end{aligned}$$

$$\begin{aligned} \operatorname{Var}(G_\theta(X)) &= \operatorname{Var}\left((\mu_1 - \mu_2)^t \Sigma^{-1} X\right) \\ &= (\mu_1 - \mu_2)^t \Sigma^{-1} \operatorname{Var} X \left((\mu_1 - \mu_2)^t \Sigma^{-1}\right)^t \\ &= (\mu_1 - \mu_2)^t \Sigma^{-1}(\mu_1 - \mu_2). \end{aligned}$$

Dans le cas homoscédastique, Σ est la matrice de covariance totale, et on reconnaît alors dans les expressions précédentes, la distance de Mahalanobis entre μ_1 et μ_2. Notons là D^2, avec $D^2 = (\mu_1 - \mu_2)^t \Sigma^{-1}(\mu_1 - \mu_2)$. Finalement, on a donc $G_\theta(X) \mid Y = 1 \sim \mathcal{N}(\frac{1}{2}D^2 + \ln \frac{p_1}{p_2}, D^2)$. De la même façon, on peut montrer que $G_\theta(X) \mid Y = 2 \sim \mathcal{N}(-\frac{1}{2}D^2 + \ln \frac{p_1}{p_2}, D^2)$. En notant Φ la fonction de répartition de la loi normale centrée réduite, on a donc :

$$\begin{aligned} e_1(r_\theta) &= \mathbb{P}(G_\theta(X) < 0 \mid Y = 1) = \Phi\left(-\frac{1}{2}D - \frac{\ln \frac{p_1}{p_2}}{D}\right) \\ e_2(r_\theta) &= \mathbb{P}(G_\theta(X) > 0 \mid Y = 2) = 1 - \Phi\left(\frac{1}{2}D - \frac{\ln \frac{p_1}{p_2}}{D}\right). \end{aligned}$$

Finalement, la probabilité globale d'erreur est donc :

$$e(r_\theta) = p_1 e_1(r_\theta) + p_2 e_2(r_\theta) = p_1\, \Phi\left(-\frac{1}{2}D - \frac{\ln \frac{p_1}{p_2}}{D}\right) + p_2 \left(1 - \Phi\left(\frac{1}{2}D - \frac{\ln \frac{p_1}{p_2}}{D}\right)\right). \tag{2.13}$$

Estimation de la règle de classement

Jusqu'à présent on a supposé θ connu, mais dans la pratique ce n'est pas le cas. On va alors construire un estimateur $\hat{\theta}$ de θ, et utiliser la méthode du "plug-in" en estimant r_θ par $r_{\hat{\theta}}$.

On suppose que l'on dispose pour cela d'un échantillon de n individus dont on connaît la classe, c'est-à-dire que l'on dispose de n couples d'observations $(X_1, Z_1), \ldots, (X_n, Y_n)$. Les (X_i, Y_i) sont indépendants et identiquement distribués, de loi jointe $f_\theta = p_k f_k$, avec $X_i \in \mathbb{R}^p$ et $Y_i \in \{1, \ldots, K\}$. On utilise la méthode du maximum de vraisemblance.

La log-vraisemblance du modèle s'écrit :

$$\ell_n(\theta) = \sum_{i=1}^{n} \ln f_\theta(X_i, Y_i) = \sum_{k=1}^{K} \sum_{X_i \in C_k} (\ln p_k + \ln f_k(X_i)) \tag{2.14}$$

$$= \sum_{k=1}^{K} \sum_{X_i \in C_k} \left(\ln p_k - \frac{p}{2} \ln 2\pi - \frac{1}{2} \ln \mid \Sigma_k \mid - \frac{1}{2} (X_i - \mu_k)^t \Sigma_k^{-1} (X_i - \mu_k) \right) \tag{2.15}$$

$$= \sum_{k=1}^{K} n_k (\ln p_k - \frac{1}{2} \ln |\Sigma_k|) - \frac{np}{2} \ln 2\pi - \frac{1}{2} \sum_{k=1}^{K} \sum_{X_i \in C_k} (X_i - \mu_k)^t \Sigma_k^{-1} (X_i - \mu_k), \tag{2.16}$$

où n_k est le nombre d'individus dans la classe C_k.

Proposition 3. *Les estimateurs du maximum de vraisemblance des paramètres sont donnés par :*

$$\hat{p}_k = \frac{n_k}{n} \tag{2.17}$$

$$\hat{\mu}_k = \frac{1}{n_k} \sum_{X_i \in C_k} X_i \tag{2.18}$$

$$\hat{\Sigma}_k = \frac{1}{n_k} \sum_{X_i \in C_k} (X_i - \mu_k)(X_i - \mu_k)^t \tag{2.19}$$

Dans le cas homoscédastique, on a :

$$\hat{\Sigma} = \frac{1}{n} \sum_{k=1}^{K} \sum_{X_i \in C_k} (X_i - \mu_k)(X_i - \mu_k)^t. \tag{2.20}$$

Démonstration. On rappelle les formules de dérivation matricielle suivantes :

$$\frac{\partial v^t M v}{\partial v} = (M + M^t)\, v, \quad \frac{\partial \mathrm{Trace}(AM)}{\partial M} = A, \quad \frac{\partial \log |M|}{\partial M} = M^{-1}$$

On a :

$$\frac{\partial \ell_n(\theta)}{\partial \mu_k} = -\frac{1}{2} \sum_{X_i \in C_k} 2\, \Sigma_k^{-1} (X_i - \mu_k) \quad \text{car } \Sigma_k^{-1} \text{ est symétrique}$$

$$= \Sigma_k^{-1} \left(\sum_{X_i \in C_k} X_i - n_k \mu_k \right)$$

$$\text{d'où} \quad \frac{\partial \ell_n(\theta)}{\partial \mu_k} = 0 \Leftrightarrow \sum_{X_i \in C_k} X_i - n_k \mu_k = 0 \Leftrightarrow \hat{\mu}_k = \frac{1}{n_k} \sum_{X_i \in C_k} X_i$$

Pour la matrice de covariance, on calcule la dérivée de $\ell(\theta)$ par rapport à Σ_k^{-1}. On remarque tout d'abord que $(X_i - \mu_k)^t \Sigma_k^{-1} (X_i - \mu_k) \in \mathbb{R}$, et donc que cette quantité est égale à sa trace : $(X_i - \mu_k)^t \Sigma_k^{-1} (X_i - \mu_k) = \mathrm{Trace}((X_i - \mu_k)^t \Sigma_k^{-1} (X_i - \mu_k)) = \mathrm{Trace}((X_i - \mu_k)(X_i - \mu_k)^t \Sigma_k^{-1})$, en utilisant la propriété $\mathrm{Trace}(AB) = \mathrm{Trace}(BA)$ (dans le cas où les deux produits matriciels ainsi définis ont un sens). On a alors :

$$\frac{\partial \ell_n(\theta)}{\partial \Sigma_k^{-1}} = \frac{n_k}{2} \Sigma_k - \frac{1}{2} \sum_{X_i \in C_k} (X_i - \mu_k)(X_i - \mu_k)^t$$

$$\text{d'où} \quad \frac{\partial \ell_n(\theta)}{\partial \Sigma_k^{-1}} = 0 \Leftrightarrow \hat{\Sigma}_k = \frac{1}{n_k} \sum_{X_i \in C_k} (X_i - \mu_k)(X_i - \mu_k)^t$$

Pour les p_k, on ré-écrit $\sum_k n_k \ln p_k$ sous la forme $\sum_{k=1}^{K-1} n_k \ln p_k + (n - \sum_{k=1}^{K-1} n_k) \ln(1 - \sum_{k=1}^{K-1} p_k)$. En dérivant par rapport à p_k, on obtient donc deux termes :

$$\frac{\partial \ell_n(\theta)}{\partial p_k} = \frac{n_k}{p_k} - \frac{n - \sum_{k=1}^{K-1} n_k}{1 - \sum_{k=1}^{K-1} p_k} = \frac{n_k}{p_k} - \frac{n_K}{p_K}$$

$$\text{d'où} \quad \frac{\partial \ell_n(\theta)}{\partial p_k} = 0 \Leftrightarrow \hat{p}_k = \frac{n_k/n \left(1 - \sum_{k=1}^{K-1} \hat{p}_k\right)}{1 - \sum_{k=1}^{K-1} n_k/n}$$

en prenant en compte la contrainte $\sum_{k=1}^{K} \hat{p}_k = 1$.

$$\begin{aligned}
\sum_{k=1}^{K} \hat{p}_k = 1 &\Leftrightarrow \sum_{k=1}^{K} \frac{n_k}{n} \left(\frac{1 - \sum_{k=1}^{K-1} \hat{p}_k}{1 - \sum_{k=1}^{K-1} n_k/n} \right) = 1 \\
&\Leftrightarrow 1 - \sum_{k=1}^{K-1} \hat{p}_k = 1 - \sum_{k=1}^{K-1} n_k/n \\
&\Leftrightarrow \hat{p}_K = \frac{1}{n} n_K \\
&\Leftrightarrow \hat{p}_K = \frac{n_K}{n}
\end{aligned}$$

Comme on avait $\frac{\partial \ell_n(\theta)}{\partial p_k} = 0 \Leftrightarrow n_k/p_k = n_K/p_K$, on en déduit $\hat{p}_k = n_k/n$ pour tout $k = 1, \ldots, K$.

□

Remarque. *L'estimateur du maximum de vraisemblance de Σ_k n'est pas sans biais. on peut alors proposer les estimateurs sans biais suivants :*

$$\tilde{\Sigma}_k = \frac{n_k}{n_k - 1} \hat{\Sigma}, k \qquad \textit{dans le cas hétéroscédastique}$$
$$\tilde{\Sigma} = \frac{n}{n - K} \hat{\Sigma}, \qquad \textit{dans le cas homoscédastique.}$$

2.3.2 Sélection de modèles

Sélection de variables

Lorsque le nombre de variables, c'est-à-dire la dimension de X, est grand, il peut être judicieux de ne travailler qu'avec un sous-ensemble de variables : celles dont le pouvoir discriminant sera le plus élevé. Cependant, les critères "usels" de sélection de variables comme le critère AIC ou BIC ne sont pas tout à fait adaptés à l'objectif de la classification supervisée, qui n'est pas tant d'identifier la meilleure représentation de la distribution des données, mais plutôt de sélectionner une règle de classement permettant de minimiser l'erreur de classement. Dans la pratique, on compare alors les performances des différents modèles en compétition en comparant les taux d'erreur estimés, associés à leurs règles de classement.

Cependant, plus le nombre de variables dans le modèle augmente, plus le biais de l'estimateur du taux d'erreur diminue, mais plus sa variance augmente : il s'agit alors de réaliser un compromis entre

le biais et la variance de l'estimateur du taux d'erreur. Lorsque le nombre total de variables reste raisonnable, on peut envisager de comparer de façon exhaustive tous les sous-modèles possibles, afin d'identifier le meilleur sous-ensemble de variables de taille m parmi les p variables initiales, au sens du taux d'erreur. En pratique cependant, cette approche devient vite trop coûteuse en temps de calcul, et l'on préfèrera la méthode dite séquentielle :

1. on sélectionne la variable j_1 qui permet d'obtenir le plus faible taux d'erreur dans un modèle à 1 variable
2. on sélectionne la variable j_2 qui permet d'obtenir le plus faible taux d'erreur, lorsqu'elle est utilisée conjointement à j_1 dans un modèle à 2 variables
3. on sélectionne la variable j_3 qui permet d'obtenir le plus faible taux d'erreur, lorsqu'elle est utilisée conjointement à j_1 et j_2 dans un modèle à 3 variables
4. etc.

Cette méthode est sous-optimale, mais permet d'obtenir de bons résultats dans la pratique.

Choix de modèle

Dans le cas du modèle de mélange Gaussien, on peut également vouloir choisir entre le modèle homoscédastique ou le modèle hétéroscédastique. De la même façon qu'un nombre élevé de variables diminue le biais de l'estimateur du taux d'erreur tout en augmentant sa variable, un modèle trop complexe où chaque classe a sa propre covariance peut s'avérer moins robuste en prédiction qu'un modèle plus parcimonieux de type homoscédastique. Là encore, on peut comparer les modèles à l'aide de leurs taux d'erreur respectifs et retenir le modèle correspondant au taux d'erreur minimal.

2.4 Régression logistique

N.B. Nous supposons dans cette section que les variables explicatives sont soit quantitatives, soit binaires. S'il existe des variables qualitatives à $q > 2$ classes, on les décompose en $q-1$ variables indicatrices binaires, et l'on considère le modèle de régression logistique contenant ces $q-1$ variables à la place de la variable qualitative à q classes initiale.

Par exemple, supposons que l'on dispose d'une variable X indiquant le niveau d'études, et codée :
- $X = 1$ pour un niveau d'études inférieur ou égal au baccalauréat
- $X = 2$ pour un BAC+2
- $X = 3$ pour BAC+5 et plus

Alors, on pourra par exemple introduire les deux variables indicatrices suivantes :
- $X_1 = 1$ si $X = 2$ et $X_1 = 0$ sinon
- $X_2 = 1$ si $X = 3$ et $X_2 = 0$ sinon

2.4.1 Le modèle logistique

Le modèle de mélange Gaussien vu dans la section précédente repose sur l'hypothèse d'une distribution normale des données dans chaque classe. En présence de deux groupes, et en cas d'égalité des matrices de covariance, on a vu dans la section précédente que le log-ratio entre les probabilités *a posteriori* $\ln \frac{t_1(\mathbf{x})}{t_2(\mathbf{x})}$, était alors **linéaire** en $\mathbf{x}$:

$$G_\theta(\mathbf{x}) = \ln \frac{t_1(\mathbf{x})}{t_2(\mathbf{x})} = (\mu_1 - \mu_2)^t \Sigma^{-1} (\mathbf{x} - \frac{\mu_1 + \mu_2}{2}) + \ln \frac{p_1}{p_2},$$

ce qui provenait en particulier de la linéarité du log-ratio des densités conditionnelles :

$$\ln \frac{f_1(\mathbf{x})}{f_2(\mathbf{x})} = (\mu_1 - \mu_2)^t \Sigma^{-1} \mathbf{x} - (\mu_1 - \mu_2)^t \Sigma^{-1} \left(\frac{\mu_1 + \mu_2}{2} \right).$$

Le modèle de régression logistique consiste alors en une généralisation de cette approche, car elle repose sur l'hypothèse fondamentale selon laquelle le ratio $\ln \frac{f_1(\mathbf{x})}{f_2(\mathbf{x})}$ s'écrit comme une fonction *linéaire* de $\mathbf{x}$. Dans le cas où $K = 2$, aussi appelé *régression logistique dichotomique*, cela s'écrit :

$$\ln \frac{f_1(\mathbf{x})}{f_2(\mathbf{x})} = \beta_0^0 + \beta^t \mathbf{x}. \tag{2.21}$$

On peut également revenir à une expression faisant intervenir les probabilités *a posteriori* t_1 et t_2, en utilisant la relation, qui est vraie même en dehors du cas Gaussien :

$$\ln \frac{t_1(\mathbf{x})}{t_2(\mathbf{x})} = \ln \frac{f_1(\mathbf{x})}{f_2(\mathbf{x})} + \ln \frac{p_1}{p_2} = \beta_0 + \beta^t \mathbf{x},$$

avec $\beta_0 = \beta_0^0 + \ln \frac{p_1}{p_2}$.

On obtient alors, comme dans le cas Gaussien (voir équation (2.11)) :

$$\begin{aligned}
\mathbb{P}(Y = 1 \mid X = \mathbf{x}) &= \frac{p_1 f_1(\mathbf{x})}{p_1 f_1(\mathbf{x}) + p_2 f_2(\mathbf{x})} \\
&= \frac{\frac{p_1 \, f_1(\mathbf{x})}{p_2 \, f_2(\mathbf{x})}}{1 + \frac{p_1 \, f_1(\mathbf{x})}{p_2 \, f_2(\mathbf{x})}} \\
&= \frac{\exp(\beta_0 + \beta^t \mathbf{x})}{1 + \exp(\beta_0 + \beta^t \mathbf{x})}
\end{aligned}$$

On parle alors de régression *logistique* car le lien entre la fonction score linéaire $G_\theta(\mathbf{x}) = \beta_0 + \beta^t \mathbf{x}$ et la probabilité $\mathbb{P}(Y = 1 \mid X = \mathbf{x}) = t_1(\mathbf{x})$ se fait à l'aide de la fonction logit :

$$\operatorname{logit} t_1(\mathbf{x}) := \ln \frac{t_1(\mathbf{x})}{1 - t_1(\mathbf{x})} = \ln \frac{t_1(\mathbf{x})}{t_2(\mathbf{x})} = G_\theta(\mathbf{x}). \tag{2.22}$$

La Figure 2.2 propose une illustration du principe de la régression logistique dans le cas dichotomique. On considère une variable binaire codant le statut d'un patient (0 : sain, et 1 : malade), et l'on

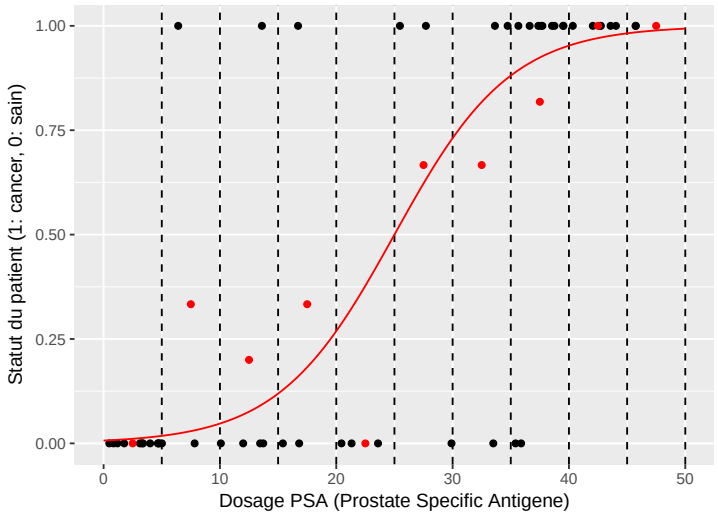


FIGURE 2.2 – Illustration de la régression logistique dichotomique (présence d'un cancer de la prostate en fonction du dosage de PSA, données fictives). Les points noirs représentent les valeurs de la variable de groupe pour chaque individu de la population (0 si le patient n'a pas de cancer, et 1 sinon), les points rouges correspondent à la proportion de patients atteints d'un cancer sur chaque intervalle délimité par les droites verticales en pointillés, et la courbe rouge représente la courbe logistique ajustée aux données.

souhaite étudier son association avec une variable explicative X. Une régression linéaire de Y en fonction de X n'est pas du tout adaptée de par la distribution des données. L'idée de la régression logistique est alors de considérer la probabilité d'être malade en fonction de la variable X. Si on divise la variable X en classes, et que l'on calcule la proportion d'individus malades ayant une valeur de X comprise dans chacun des intervalles définis par les classes, on peut chercher une relation de type logistique entre ces proportions et la variable X. Ceci est illustré sur la figure par la recherche d'une courbe logistique (en rouge) permettant d'ajuster les points rouges.

Le modèle logistique fait partie des méthodes dites *semi-paramétriques*, car s'il repose bien sur l'hypothèse selon laquelle le ratio $\ln \frac{f_1(\mathbf{x})}{f_2(\mathbf{x})}$ est linéaire, il couvre une gamme plus étendue de lois de probabilité que le modèle de mélange Gaussien. En particulier, et c'est une différence majeure avec le cas Gaussien, il est possible de traiter, dans le cadre du modèle logistique, le cas de variables explicatives binaires.

On suppose maintenant K quelconque. On choisit une catégorie de référence, à laquelle seront comparées les $K-1$ classes. Ce choix peut être soit dicté par le contexte de l'analyse, soit choisi arbitrairement, mais n'a pas de conséquence sur l'interprétation des résultats. Nous prendrons dans la suite du document la classe C_K comme catégorie de référence. La régression logistique consiste alors à calculer les log-ratios suivants, pour $k = 1, \ldots, K-1$:

$$\ln \frac{f_k(\mathbf{x})}{f_K(\mathbf{x})} = \beta_{0k}^0 + \beta_k^t \mathbf{x}, \tag{2.23}$$

où $\beta_{0k}^0 \in \mathbb{R}$ et $\beta_k \in \mathbb{R}^d$.

Remarque. *Comme indiqué au début de la section 2.4, on suppose que les variables explicatives sont soit quantitatives, soit binaires. Si c'était le cas initialement, la dimension d est alors simplement égale au nombre de variables explicatives initiales. Si par contre certaines variables qualitatives à plus de deux deux classes étaient présentes initialement, elles ont été transformées en $q-1$ variables binaires, avec q le nombre de classes de la variable qualitative. Dans ce cas, la dimension d est égale à la somme du nombre de variables quantitatives et binaires initiales et du nombre d'indicatrices créées par la décomposition de la variables qualitative. On peut donc avoir $d > p$, où p est le nombre de variables explicatives initiales.*

À partir de l'expression (2.23), on peut obtenir la probabilité *a posteriori* d'appartenir à la classe C_k, en notant $\beta_{0k} = \beta^0_{0k} + \ln \frac{p_k}{p_K}$:

$$\ln \frac{t_k(\mathbf{x})}{t_K(\mathbf{x})} = \beta_{0k} + \beta_k^t \mathbf{x}, \tag{2.24}$$

soit :

$$\begin{cases} t_k(\mathbf{x}) & = \dfrac{\exp(\beta_{0k} + \beta_k^t \mathbf{x})}{1 + \sum_{\ell=1}^{K-1} \exp(\beta_{0\ell} + \beta_\ell^t \mathbf{x})}, \quad k = 1, \ldots, K-1 \\ t_K(\mathbf{x}) & = \dfrac{1}{1 + \sum_{\ell=1}^{K-1} \exp(\beta_{0\ell} + \beta_\ell^t \mathbf{x})} \end{cases} \tag{2.25}$$

Dans la suite du document, pour mettre en évidence la dépendance des t_k en β, on écrira $t_k(\mathbf{x}; \boldsymbol{\beta})$.

Remarque. *On rencontre parfois, dans certains logiciels, d'autres fonctions de lien entre les probabilités $t_k(\mathbf{x})$ et la fonction score $G_\theta(\mathbf{x})$. On peut par exemple utiliser la fonction* probit*, définie par* $\text{probit}(x) = \Phi^{-1}(x)$*, avec Φ^{-1} la fonction de répartition de la loi normale centrée réduite. Cependant, les résultats obtenus sont similaires à ceux obtenus avec la fonction logit car* $\text{logit}(x) \simeq \sqrt{8/\pi}\,\text{probit}(x)$.

2.4.2 Estimation des paramètres

On note $\boldsymbol{p} = (p_1, \ldots, p_K)^t$ et $\boldsymbol{\beta} = (\beta_{01}, \ldots, \beta_{0K-1}, \beta_1, \ldots, \beta_{K-1})^t$. On rappelle que les $\beta_{01}, \ldots, \beta_{0K-1}$ sont des scalaires, mais que les $\beta_1, \ldots, \beta_{K-1}$ sont des vecteurs. On cherche à estimer $\theta = (\boldsymbol{p}\ \boldsymbol{\beta})$. Comme précédemment, on suppose que l'on dispose pour cela d'un échantillon de n couples de vecteurs aléatoires $(X_1, Y_1), \ldots, (X_n, Y_n)$, où $X_i \in \mathbb{R}^p$ et où $Y_i \in \{1, \ldots, K\}$. On introduit Z_i le vecteur aléatoire à valeurs dans $\{0,1\}^K$ dont toutes les coordonnées sont nulles sauf celle correspondant au numéro de la classe à laquelle appartient l'individu i. Autrement dit, $Y_i = k \leftrightarrow Z_{ik} = 1$. Z_i suit une loi multinomiale de paramètres 1 et $p = (t_1(\mathbf{x}_i, \boldsymbol{\beta}), \ldots, t_K(\mathbf{x}_i, \boldsymbol{\beta})^t$. On a :

$$Z_i \mid X_i = \mathbf{x}_i \sim \mathcal{M}_K(1, t_1(\mathbf{x}_i; \boldsymbol{\beta}), \ldots, t_K(\mathbf{x}_i; \boldsymbol{\beta})). \tag{2.26}$$

On note $\mathbf{x}_1, \ldots, \mathbf{x}_n$, et $z_1, \ldots, z_n$, les réalisations des variables aléatoires précédentes. La loi marginale des X_i ne dépend pas de θ, on peut donc exprimer la vraisemblance seulement en fonction de la

loi conditionnelle des Z_i sachant X_i. En effet, en notant $L_n(\theta)$ la vraisemblance des observations, on a :

$$L_n(\theta) = \prod_{i=1}^{n} f_\theta(X_i, Z_i) = \prod_{i=1}^{n} f_\theta(Z_i \mid X_i = \mathbf{x}_i) f(\mathbf{x}_i),$$

où on a utilisé ici l'autre décomposition possible de la loi jointe des (X_i, Z_i) que dans le cas du modèle de mélange Gaussien [1]. Comme la loi f ne dépend pas de θ, on la considère comme une constante, et maximiser $L_n(\theta)$ par rapport à θ revient à maximiser $\prod_{i=1}^{n} f_\theta(Z_i \mid X_i = \mathbf{x}_i)$ par rapport à θ.

Il nous reste à expliciter la quantité $f_\theta(Z_i \mid X_i = \mathbf{x}_i)$. Pour un individu i, on a par définition de la loi multinomiale :

$$f_\theta(Z_i \mid X_i = \mathbf{x}_i) := \mathbb{P}_\theta(Z_i = (z_{i1}, \ldots, z_{iK}) \mid X_i = \mathbf{x}_i) = \left(\prod_{k=1}^{K-1} t_k(\mathbf{x}_i; \boldsymbol{\beta})^{Z_{ik}} \right) t_K(\mathbf{x}_i; \boldsymbol{\beta})^{1 - \sum_{k=1}^{K-1} Z_{ik}}. \tag{2.27}$$

La vraisemblance de l'échantillon s'écrit alors :

$$\begin{aligned} L_n(\theta) &\propto \prod_{i=1}^{n} \mathbb{P}_\theta(Z_i = (z_{i1}, \ldots, z_{iK}) \mid X_i = \mathbf{x}_i) \\ &\propto \prod_{i=1}^{n} \left(\prod_{k=1}^{K-1} t_k(\mathbf{x}_i; \boldsymbol{\beta})^{Z_{ik}} \right) t_K(\mathbf{x}_i; \boldsymbol{\beta})^{1 - \sum_{k=1}^{K-1} Z_{ik}}. \end{aligned}$$

La log-vraisemblance est donnée par :

$$\begin{aligned} \ell_n(\boldsymbol{\beta}) &= \ln L_n(\theta) \\ &= \sum_{i=1}^{n} \left[\sum_{k=1}^{K-1} Z_{ik} \ln t_k(\mathbf{x}_i; \boldsymbol{\beta}) + \left(1 - \sum_{k=1}^{K-1} Z_{ik} \right) \ln t_K(\mathbf{x}_i; \boldsymbol{\beta}) \right] + \text{cste} \\ &= \sum_{i=1}^{n} \left[Z_{ik} \ln \frac{t_k(\mathbf{x}_i; \boldsymbol{\beta})}{t_K(\mathbf{x}_i; \boldsymbol{\beta})} + \ln t_K(\mathbf{x}_i; \boldsymbol{\beta}) \right] + \text{cste} \\ &= \sum_{i=1}^{n} \left[Z_{ik} \left(\beta_{0k} + \beta_k^t \mathbf{x}_i \right) - \ln \left(1 + \sum_{\ell=1}^{K-1} \exp\left(\beta_{0\ell} + \beta_\ell^t \mathbf{x}_i \right) \right) \right] + \text{cste}. \end{aligned} \tag{2.28}$$

On obtient ensuite le maximum de vraisemblance en résolvant l'équation $\partial \ell_n(\boldsymbol{\beta}) / \partial \boldsymbol{\beta} = 0$. Cette équation est non linéaire en $\boldsymbol{\beta}$ et n'a pas de solution analytique. Elle se résoud alors numériquement, par exemple à l'aide de l'algorithme de Newton-Raphson (voir Annexe A pour plus de détails sur cet algorithme). On note $\hat{\boldsymbol{\beta}}$ l'estimateur du maximum de vraisemblance ainsi obtenu.

Estimation des probabilités *a priori* p_k

La loi conditionnelle des Z_i sachant X_i nous permet d'estimer les paramètres composant $\boldsymbol{\beta}$. Pour estimer les p_k, on peut utiliser la loi marginale des Z_i. Il s'agit de la loi multinomiale $\mathcal{M}_K(1, (p_1, \ldots, p_K))$. Dans ce cas, l'estimateur du maximum de vraisemblance des p_k est donné, comme dans le cas Gaussien, par :

$$\hat{p}_k = \frac{n_k}{n}.$$

1. en effet, on a soit $f(X_i, Z_i) = f(X_i|Z_i) f(Z_i)$, décomposition utilisé dans la section 2.3, soit $f(X_i, Z_i) = f(Z_i|X_i) f(X_i)$, décomposition utilisée ici

En effet, si on considère la vraisemblance associée uniquement aux observations $Z_1, \dots, Z_n$, on a :

$$\begin{aligned} L_n(\boldsymbol{p}) &= \prod_{i=1}^{n} \mathbb{P}_\theta(Z_i = (z_{i1}, \dots, z_{iK})) \\ &= \prod_{i=1}^{n} \prod_{k=1}^{K} p_k^{z_{ik}} \end{aligned}$$

d'où la log-vraisemblance :

$$\ell_n(\boldsymbol{p}) = \sum_{i=1}^{n} \sum_{k=1}^{K} z_{ik} \ln p_k = \sum_{k=1}^{K} \ln p_k \sum_{i=1}^{n} z_{ik} = \sum_{k=1}^{K} n_k \ln p_k$$

On reconnait l'expression que l'on a maximisée en p_k dans la section précédente.

Estimation de la constante β_{0k}^0
Une fois que l'on a estimé $\boldsymbol{\beta}$ et $\boldsymbol{p}$, on peut en déduire une expression pour la constante β_{0k}^0 (voir équation (2.24)) :

$$\hat{\beta}_{0k}^0 = \hat{\beta}_{0k} - \ln \frac{\hat{p_k}}{\hat{p_K}}. \tag{2.29}$$

2.4.3 Interprétation des résultats

Définition 11. *On appelle* $\text{odds}_{k\ell}$ *ou "côte" le rapport entre les probabilités t_k et t_ℓ, autrement dit :*

$$\text{odds}_{k\ell}(\mathbf{x}) = \frac{\mathbb{P}(Y = k \mid X = \mathbf{x})}{\mathbb{P}(Y = \ell \mid X = \mathbf{x})} = \frac{t_k(\mathbf{x})}{t_\ell(\mathbf{x})}, \tag{2.30}$$

qui peut s'interpréter par "combien de fois plus de chances a t-on d'avoir $Y = k$ au lieu de $Y = \ell$, lorsque $X = \mathbf{x}$".

Par exemple, si l'on considère deux catégories "sain" et "malade", et que l'on a $\mathbb{P}(\text{malade} \mid \mathbf{x}) = 0.75$ et $\mathbb{P}(\text{sain} \mid \mathbf{x}) = 0.25$ alors l'odds d'être malade par rapport à être en bonne santé est $0.75/0.25 = 3$, cela signifie que l'on a trois fois plus de risque d'être malade que de ne pas l'être, lorsque $X = \mathbf{x}$. On dit aussi que le risque d'être malade est de 3 contre 1.

Définition 12. *On appelle "odds-ratio" ou rapport des côtes, le ratio :*

$$OR_{k\ell}(\mathbf{x}, \mathbf{x}') = \frac{\text{odds}_{k\ell}(\mathbf{x})}{\text{odds}_{k\ell}(\mathbf{x}')}, \tag{2.31}$$

qui peut s'interpréter par "combien de fois plus de chances a t-on d'avoir $Y = k$ au lieu de $Y = \ell$ lorsque $X = \mathbf{x}$ par rapport au cas où $X = \mathbf{x}'$".

Par exemple, si on reprend le cas des deux catégories "sain" et "malade" précédentes, et que l'on a une population constituée de 20 femmes et 20 hommes, dont respectivement 7 et 11 sont malades, on obtient un odds-ratio des hommes par rapport aux femmes de $[(11/20)/(9/20)]/[(7/20)/(13/20)] = 2.27$, ce qui signifie que les hommes ont 2.27 fois plus de risque d'être malades que les femmes.

Les coefficients du modèle s'interprètent alors facilement en terme d'odds-ratio. Prenons par exemple le cas de la régression logistique binaire et considérons une variable explicative X_j. Calculons l'odds-ratio $\mathrm{OR}_{01}(\mathbf{x}_1, \mathbf{x}_2)$, où $\mathbf{x}_1$ et $\mathbf{x}_2$ ne différent que sur la j-ème coordonnée, i.e. $\mathbf{x}_1 = (x_1, \ldots, x_j, \ldots, x_p)^t$ et $\mathbf{x}_2 = (x_1, \ldots, x'_j, \ldots, x_p)^t$. Alors on a :

$$\begin{aligned}
\ln \mathrm{OR}_{01}(\mathbf{x}_1, \mathbf{x}_2) &= \ln \mathrm{odds}_{01}(\mathbf{x}_1) - \ln \mathrm{odds}_{01}(\mathbf{x}_2) \\
&= \ln \frac{t_1(\mathbf{x}_1; \beta)}{t_0(\mathbf{x}_1; \beta)} - \ln \frac{t_1(\mathbf{x}_2; \beta)}{t_0(\mathbf{x}_2; \beta)} \\
&= \beta_0 + \sum_{i=1, i\neq j}^{p} \beta_i x_i + \beta_j x_j - \left(\beta_0 + \sum_{i=1, i\neq j}^{p} \beta_i x_i + \beta_j x'_j \right) \\
&= \beta_j (x_j - x'_j).
\end{aligned}$$

Le coefficient associé à la variable X_j correspond donc au logarithme de l'odds-ratio de la catégorie 1 par rapport à la catégorie 0, lorsque la variable X_j augmente d'une unité.
Dans le cas d'une régression logistique à plus de deux classes, on a :

$$\begin{aligned}
\ln \mathrm{OR}_{k\ell}(\mathbf{x}_1, \mathbf{x}_2) &= \ln \mathrm{odds}_{k\ell}(\mathbf{x}_1) - \ln \mathrm{odds}_{k\ell}(\mathbf{x}_2) \\
&= \ln \frac{t_k(\mathbf{x}_1; \beta)}{t_\ell(\mathbf{x}_1; \beta)} - \ln \frac{t_k(\mathbf{x}_2; \beta)}{t_\ell(\mathbf{x}_2; \beta)} \\
&= \ln \frac{t_k(\mathbf{x}_1; \beta)}{t_K(\mathbf{x}_1; \beta)} \frac{t_K(\mathbf{x}_1; \beta)}{t_\ell(\mathbf{x}_1; \beta)} - \ln \frac{t_k(\mathbf{x}_2; \beta)}{t_K(\mathbf{x}_2; \beta)} \frac{t_K(\mathbf{x}_2; \beta)}{t_\ell(\mathbf{x}_2; \beta)} \\
&= \ln \frac{t_k(\mathbf{x}_1; \beta)}{t_K(\mathbf{x}_1; \beta)} - \ln \frac{t_k(\mathbf{x}_2; \beta)}{t_K(\mathbf{x}_2; \beta)} - \left(\ln \frac{t_\ell(\mathbf{x}_1; \beta)}{t_K(\mathbf{x}_1; \beta)} - \ln \frac{t_\ell(\mathbf{x}_2; \beta)}{t_K(\mathbf{x}_2; \beta)} \right) \\
&= \beta_{kj}(x_j - x'_j) - \beta_{\ell j}(x_j - x'_j) \\
&= \left(\beta_{kj} - \beta_{\ell j}\right)(x_j - x'_j)
\end{aligned}$$

On obtient donc l'odds-ratio de la classe k par rapport à la classe ℓ lorsque la variable X_j augmente d'une unité en prenant l'exponentielle de la différence entre les coefficients associés à la variable X_j.

2.4.4 Intervalles de confiance et tests

L'estimateur du maximum de vraisemblance $\hat{\boldsymbol{\beta}}$ possède plusieurs propriétés que l'on peut exploiter pour construire des intervalles de confiance, et en particulier celle d'être asymptotiquement normal. Si on note $I_n(\boldsymbol{\beta})$ la matrice d'information de Fisher, avec

$$I_n(\boldsymbol{\beta}) = -\mathbb{E}\left(\frac{\partial^2 \ell_n(\boldsymbol{\beta})}{\partial \boldsymbol{\beta} \partial \boldsymbol{\beta}^t} \right),$$

on a :

$$\sqrt{n}(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}) \xrightarrow[n\to\infty]{\text{loi}} \mathcal{N}(0, I(\boldsymbol{\beta})^{-1}), \tag{2.32}$$

où $I(\boldsymbol{\beta})$ est la matrice d'information de Fisher associée à une observation.

Intervalle de confiance

On obtient l'intervalle de confiance asymptotique symétrique de niveau $1-\alpha$ suivant pour la j-ème composante de β_k :

$$IC_\alpha(\beta_{kj}) = \left[\hat{\beta}_{kj} - q^{\mathcal{N}(0,1)}_{1-\alpha/2}\hat{\sigma}_{kj}, \hat{\beta}_{kj} + q^{\mathcal{N}(0,1)}_{1-\alpha/2}\hat{\sigma}_{kj}\right], \tag{2.33}$$

où $q^{\mathcal{N}(0,1)}_{1-\alpha/2}$ est le quantile d'ordre $1-\alpha/2$ de la loi normale centrée réduite, et $\hat{\sigma}_{kj}$ le (kj)-ème terme de la diagonale de la matrice $I_1(\boldsymbol{\beta})^{-1}$.

Dans le cas de la régression logistique dichotomique, on en déduit l'intervalle de confiance suivant pour l'odds-ratio associé à la variable X_j, $\mathrm{OR}_{01}(j)$:

$$IC_\alpha(\mathrm{OR}_{01}(j)) = \left[\exp\left(\hat{\boldsymbol{\beta}}_j - q^{\mathcal{N}(0,1)}_{1-\alpha/2}\hat{\sigma}_j\right), \exp\left(\hat{\boldsymbol{\beta}}_j + q^{\mathcal{N}(0,1)}_{1-\alpha/2}\hat{\sigma}_j\right)\right], \tag{2.34}$$

Dans le cas de la régression logistique à plus de deux classes, il faut d'abord obtenir un intervalle de confiance pour la différence $\beta_{jk} - \beta_{j\ell}$, ce qui s'obtient facilement car le vecteur $\hat{\boldsymbol{\beta}}$ est gaussien.

Tests

Afin d'évaluer l'influence d'un ensemble de variables explicatives sur le ratio $\ln \frac{t_k(\mathbf{x})}{t_K(\mathbf{x})}$, on peut proposer le test suivant :

$$H_0 : R\beta_k = 0 \quad \text{contre} \quad H_1 : R\beta_k \neq 0,$$

où R est une matrice de plein rang de taille $r \times \dim(\beta_k)$. Ce test permet notamment de comparer deux modèles emboîtés, le modèle défini par H_0 contenant moins de variables explicatives que le modèle défini par H_1. Par exemple, si on souhaite tester l'hypothèse nulle selon laquelle les variables explicatives X_1 et X_2 n'ont pas d'influence dans le modèle, on a :

$$R = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \end{pmatrix}, \tag{2.35}$$

d'où $R\beta_k = (\beta_{k1}\ \beta_{k2})^t$, et l'hypothèse nulle revient donc à tester si β_{k1} et β_{k2} sont nuls simultanément.

On décrit ici trois procédures de test permettant de tester les hypothèses ci-dessus. Ces trois tests sont asymptotiquement équivalents. La figure 2.3 illustre la différence entre les trois approches.

Test du rapport de vraisemblance

La statistique du *test du rapport de vraisemblance* est donnée par :

$$T_n = -2\ln\frac{\sup_{R\beta_k=0} L_n(\boldsymbol{\beta})}{\sup_{R\beta_k\neq 0} L_n(\boldsymbol{\beta})} = -2\ln\frac{L_n(\hat{\boldsymbol{\beta}}_0)}{L_n(\hat{\boldsymbol{\beta}}_1)} = -2(\ell_n(\hat{\boldsymbol{\beta}}_0) - \ell_n(\hat{\boldsymbol{\beta}}_1)), \tag{2.36}$$

où $\hat{\boldsymbol{\beta}}_0$ et $\hat{\boldsymbol{\beta}}_1$ sont les estimateurs du maximum de vraisemblance obtenus respectivement sous H_0 et sous H_1, c'est-à-dire dans les modèles définis par ces deux hypothèses.

La loi asymptotique de la statistique de test T_n sous l'hypothèse nulle H_0 est :

$$T_n \xrightarrow[n\to\infty]{H_0} \chi^2_r, \tag{2.37}$$

où r est le rang de la matrice R. En pratique, si on compare deux modèles emboîtés, l'un sous H_1 avec q_1 paramètres et l'autre sous H_0 avec q_0 paramètres, on a $r = q_1 - q_0$.

Intuitivement, si l'hypothèse nulle est vraie, alors les valeurs de la log-vraisemblance obtenues sous les deux hypothèses doivent être proches : on rejette donc H_0 si la différence est trop grande. On construit alors un seuil de rejet asymptotique de niveau α, $c_\alpha = q^{\chi^2_r}_{1-\alpha}$, correspondant au quantile d'ordre $1-\alpha$ de la loi du chi-deux à r degrés de liberté, et on rejette l'hypothèse nulle si la statistique de test T_n est supérieure à ce seuil c_α.

Test de Wald

Le test de Wald exploite la normalité asymptotique de l'estimateur du maximum de vraisemblance, rappelée à l'équation (2.32). On déduit de cette équation la convergence en loi suivante :

$$\sqrt{n}\,(R\hat{\boldsymbol{\beta}} - R\boldsymbol{\beta}) \xrightarrow[n\to\infty]{\text{loi}} \mathcal{N}_r(0, R\,I(\boldsymbol{\beta})^{-1}R^t), \tag{2.38}$$

où $\mathcal{N}_r$ désigne une loi normale de dimension r.
La statistique du *test de Wald* est alors définie par :

$$W_n = n(R\hat{\boldsymbol{\beta}})^t(R\,I(\hat{\boldsymbol{\beta}})^{-1}R^t)^{-1}\,(R\hat{\boldsymbol{\beta}}). \tag{2.39}$$

Sous l'hypothèse nulle, $R\boldsymbol{\beta} = 0$, et si on pose $Z_n = \sqrt{n}(R\,I(\beta)^{-1}R^t)^{-1/2}(R\hat{\boldsymbol{\beta}})$, on a $Z_n \xrightarrow[n\to\infty]{\text{loi}} \mathcal{N}_r(0, \mathrm{Id})$ d'après (2.38). On a alors $W_n = \|Z_n\|^2$.

On en déduit la loi asymptotique de la statistique de test W_n sous l'hypothèse nulle H_0 :

$$W_n \xrightarrow[n\to\infty]{H_0} \chi^2_r. \tag{2.40}$$

La statistique W_n mesure l'écart entre l'estimateur du maximum de vraisemblance et 0, on s'attend donc à rejetter H_0 si cet écart est trop grand, c'est-à-dire si l'estimateur est trop loin de 0. Comme dans le cas du test du rapport de vraisemblance, on rejette donc l'hypothèse nulle si la statistique de test W_n est supérieure au seuil c_α défini précédemment (même seuil de rejet asymptotique que pour T_n).

Test du score

Dans le test du score, on s'intéresse à la dérivée de la log-vraisemblance, aussi appelée fonction score et notée $U_n(\boldsymbol{\beta})$, avec $U_n(\boldsymbol{\beta}) = (\partial\ell_n(\boldsymbol{\beta}))/(\partial\boldsymbol{\beta})$. On peut montrer facilement (par exemple en utilisant le théorème central limite et en remarquant que l'espérance du score est nulle) la normalité asymptotique du score :

$$n^{-1/2}\,U_n(\boldsymbol{\beta}) \xrightarrow[n\to\infty]{\text{loi}} \mathcal{N}(0, I(\boldsymbol{\beta})). \tag{2.41}$$

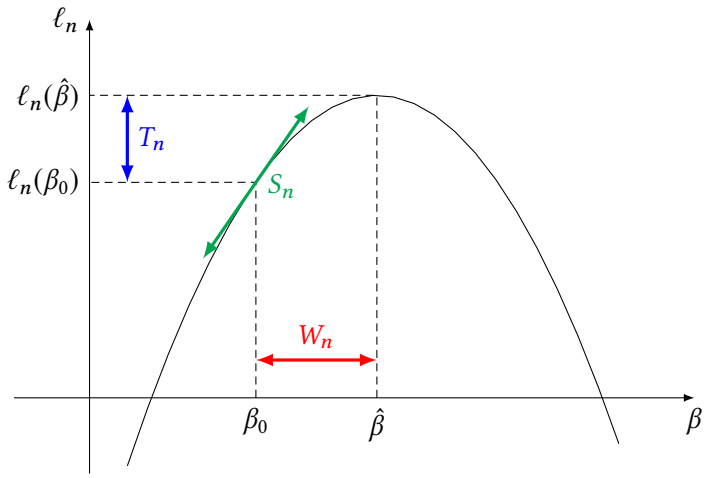


FIGURE 2.3 – Illustration de la différence entre les trois tests présentés, dans le cas d'un modèle à un seul paramètre où l'on teste $H_0 : \beta = \beta_0$ contre $H_1 : \beta \neq \beta_0$. On retrouve le cas particulier de notre exemple lorsque $\beta_0 = 0$, et en prenant $R = 1$. Le test du rapport de vraisemblance porte sur la distance entre la vraisemblance estimée sous H_0 et sous H_1, le test de Wald porte sur la distance entre β_0 et l'estimateur du maximum de vraisemblance de β sous H_1, et le test du score porte sur la pente de la dérivée de la log-vraisemblance en β_0.

La statistique du *test du score* est définie par :

$$S_n = \frac{1}{n} U_n(\hat{\boldsymbol{\beta}}_0)^t \, I(\hat{\boldsymbol{\beta}}_0)^{-1} \, U_n(\hat{\boldsymbol{\beta}}_0). \tag{2.42}$$

Sous H_0, on a la loi asymptotique suivante pour S_n :

$$S_n \xrightarrow[n\to\infty]{H_0} \chi^2_r. \tag{2.43}$$

L'intuition derrière ce test consiste à dire que si l'hypothèse nulle est vérifiée, alors la valeur de $\boldsymbol{\beta}$ sous H_0 sera proche du maximum de vraisemblance global, c'est-à-dire sans la contrainte $R\boldsymbol{\beta} = 0$. Dans ce cas, la fonction score en $\hat{\beta}_0$ sera proche de 0. Comme dans les deux tests précédents, on va donc rejetter H_0 lorsque la statistique de test S_n est trop grande. Plus précisément, on rejette H_0 si S_n est supérieure au seuil c_α défini précedemment.

2.4.5 Sélection de modèles

Les tests présentés au paragraphe précédent repose sur l'hypothèse fondamentale selon laquelle les modèles que l'on compare sous H_0 et H_1 sont **emboîtés**. Ils ne sont plus valables en dehors de ce cas-là.

Si l'on souhaite comparer des modèles non emboîtés, on peut utiliser des critères de sélection de modèles. En notant q est le nombre de paramètres du modèle, on définit les critères AIC et BIC :

$$\text{AIC} = -2\ell(\hat{\boldsymbol{\beta}}) + 2q \tag{2.44}$$

$$\text{BIC} = -2\ell(\hat{\boldsymbol{\beta}}) + q \ln n. \tag{2.45}$$

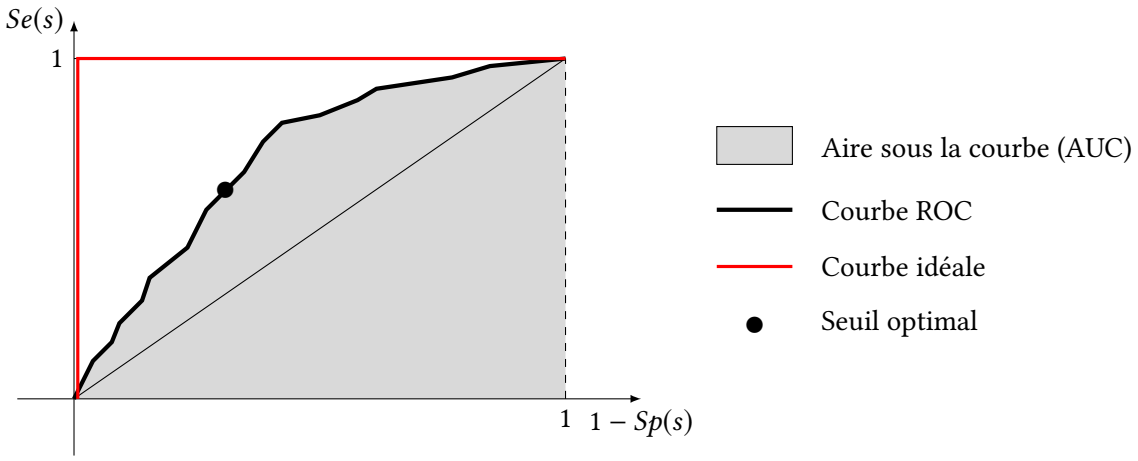


FIGURE 2.4 – Exemple de courbe ROC.

Comme on cherche le modèle associé à la plus forte valeur de la vraisemblance, on cherche à minimiser ces critères, c'est-à-dire que l'on préfèrera un modèle pour lequel la valeur des critères ci-dessus est la plus faible. Ces deux critères sont composés d'un terme de vraisemblance, qui mesure l'ajustement du modèle aux données, et d'un terme de pénalité, qui augmente avec le nombre de paramètres, c'est-à-dire avec la complexité du modèle.

2.4.6 Courbe ROC

Un outil fréquemment utilisé pour évaluer les résultats d'une régression logistique dichotomique est celui de la courbe ROC (pour *Receiver Operator Characteristic curve*). En fait, la courbe ROC peut être utilisée pour évaluer toute méthode qui fournit un score permettant de classer des individus en deux classes. Introduisons tout d'abord quelques notations et définitions.

Rappelons tout d'abord que la régression logistique fournit un score (attention à ne pas confondre ce score avec la *fonction score* U_n) défini dans le cas dichotomique par :

$$G_{\hat{\theta}}(\mathbf{x}) = \ln \frac{t_1(\mathbf{x})}{t_2(\mathbf{x})} = \hat{\beta}_0 + \hat{\beta}^t \mathbf{x}$$

On peut alors définir un seuil s et une règle de classement $r_s(\mathbf{x})$ associés à ce score :

$$r_s(\mathbf{x}) = \begin{cases} 1 & \text{si } G_{\hat{\theta}}(\mathbf{x}) \geq s \\ 0 & \text{si } G_{\hat{\theta}}(\mathbf{x}) < s \end{cases} \tag{2.46}$$

Remarque. *Dans le cas du modèle de mélange Gaussien, on avait défini une règle de classement en prenant* $s = 0$ *(voir équation* (2.12)*).*

Supposons que la variable de classe Y prenne les valeurs 0 ou 1, 0 désignant les individus "négatifs" et 1 les individus "positifs" (ex. : détection d'une maladie, identification d'un individu à risque pour un emprûnt bancaire, ...)

Définition 13. *On appelle* ***sensibilité*** *la probabilité qu'un individu positif soit effectivement déclaré positif. On parle aussi de "vrais positifs". On la note $Se(s)$ et on a :*

$$Se(s) = \mathbb{P}(G_{\hat{\theta}}(X) \geq s \mid Y = 1) = \mathbb{E}(r_s(X) \mid Y = 1). \tag{2.47}$$

Définition 14. *On appelle* ***spécificité*** *la probabilité qu'un individu négatif soit effectivement déclaré négatif. On parle aussi de "vrais négatifs". On la note $Sp(s)$ et on a :*

$$Sp(s) = \mathbb{P}(G_{\hat{\theta}}(X) < s \mid Y = 0) = 1 - \mathbb{E}(r_s(X) \mid Y = 0). \tag{2.48}$$

La quantité $1 - Sp(s)$ correspond au taux de "faux négatifs".

Définition 15. *La* ***courbe ROC*** *est la courbe définie par les points $(1 - Sp(s), Se(s))$ et paramétrée par s. C'est une courbe croissante en s.*

La courbe ROC (voir figure 2.4) évalue le pouvoir prédictif *global* d'un modèle, indépendamment de la valeur s. Plus le modèle est bon, plus la courbe est au-dessus de la première bissectrice. Dans le cas idéal, elle passe par le point de coordonnées $(0, 1)$, correspondant à un seuil permettant d'obtenir 100% de vrais positifs et 100% de vrais négatifs (soit 0% de faux négatifs). On peut utiliser la courbe ROC pour déterminer un seuil s optimal, en minimisant la quantité $(1 - Se(s))^2 + (1 - Sp(s))^2$, ce qui revient à choisir le seuil correspondant au point le plus proche de point idéal de coordonnées $(0, 1)$.

2.5 Méthodes des k-plus proches voisins

2.5.1 Principe général

La méthode des k-plus proches voisins est une méthode non paramétrique dont l'objectif est d'estimer pour chaque individu $\mathbf{x}$, la probabilité conditionnelle d'appartenir à chacune des classes $C_1, \ldots, C_K$, c'est-à-dire les $t_1(\mathbf{x}), \ldots, t_K(\mathbf{x})$.

L'idée principale de la méthode est de considérer, pour un point $\mathbf{x}$, ses k-plus proches voisins dans l'espace des variables (sous-espace de $\mathbb{R}^p$), et d'estimer la probabilité conditionnelle $t_\ell(\mathbf{x})$ en calculant la proportion de voisins appartenant à la classe C_ℓ. On rappelle que l'on dispose d'un échantillon de n couples observations (X_i, Y_i), $i = 1, \ldots, n$, où $Y_i \in \{1, \ldots, K\}$ et où $X_i \in \mathbb{R}^p$.On considère un nouveau point correspondant à une réalisation de X, et que l'on note $\mathbf{x}$. On obtient alors l'estimateur suivant pour $t_\ell(\mathbf{x})$:

$$\hat{t}_\ell(\mathbf{x}) = \frac{1}{k} \sum_{i=1}^{k} \mathbf{1}_{Y_i = \ell}, \tag{2.49}$$

où Y_i est l'observation du i-ème voisin de $\mathbf{x}$. La règle de classement optimal de Bayes nous conduit à choisir pour $\mathbf{x}$ la classe C_ℓ telle que $\hat{t}_\ell(\mathbf{x})$ soit maximale, c'est-à-dire la classe majoritaire parmi les k voisins de $\mathbf{x}$.

2.5.2 Choix des k voisins

La première question qui se pose naturellement est celle du choix des k voisins : qu'appelle t-on un voisin ? comment choisir le nombre de points k ? Pour définir la notion de voisinage, on a besoin d'une distance sur l'espace des variables, c'est-à-dire d'une distance sur $\mathbb{R}^p$. Plusieurs choix sont possibles, parmi lesquels :

— la distance euclidienne :

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{y}_1 - \mathbf{x}_1)^2 + \cdots + (\mathbf{y}_p - \mathbf{x}_p)^2} = \left(\sum_{i=1}^{p} (\mathbf{y}_i - \mathbf{x}_i)^2 \right)^{1/2} \tag{2.50}$$

— la distance de Minkowski de paramètre q :

$$d(\mathbf{x}, \mathbf{y}) = \left(\sum_{i=1}^{p} (\mathbf{y}_i - \mathbf{x}_i)^q \right)^{1/q} \tag{2.51}$$

— la distance maximale :

$$d(\mathbf{x}, \mathbf{y}) = \max_{i \in \{1, \ldots, p\}} |\mathbf{x}_i - \mathbf{y}_i| \tag{2.52}$$

— la distance de Manhattan :

$$d(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{p} |\mathbf{x}_i - \mathbf{y}_i| \tag{2.53}$$

— la distance de Canberra :

$$d(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{p} \frac{|\mathbf{x}_i - \mathbf{y}_i|}{|\mathbf{x}_i| + |\mathbf{y}_i|}. \tag{2.54}$$

Une fois que l'on a muni notre espace d'une distance, on peut définir le voisinage $\mathcal{V}_d(\mathbf{x})$ d'un point $\mathbf{x}$ comme étant l'ensemble des points situés à une distance $d_{\max}$ de $\mathbf{x}$. Si on se fixe un nombre de points k à inclure dans le voisinage, on choisira $d_{\max}$ comme la distance entre $\mathbf{x}$ et le k-ème point le plus proche. Une fois que l'on a identifié les k plus proches voisins de $\mathbf{x}$, on affecte le point à la classe majoritaire. Une illustration de l'algorithme est présenté dans les figures 2.5 et 2.6.

Le choix de la distance peut avoir une forte influence sur l'algorithme, en particulier lorsque les variables considérées ont des unités très différentes. Par exemple, supposons que l'on dispose de deux variables explicatives, l'une indiquant la fréquence cardiaque en nombre de battements par minutes, qui se situe normalement autour de 70 bat./min. pour un adulte, et le taux de cholestérol, qui se situe entre 1.5 et 2.5 g/L de sang. On voit bien que dans ce cas, c'est surtout la fréquence cardiaque qui va avoir une influence sur le calcul de la distance entre deux points. Plus précisément, une différence d'un battement par minute aura la même influence que l'augmentation d'un point du taux de cholestérol, dans le calcul de la distance. Dans ce cas là, il est recommandé de réduire les données, c'est-à-dire de diviser chaque variable par son écart-type, afin d'attribuer la même influence à chaque variable (notons

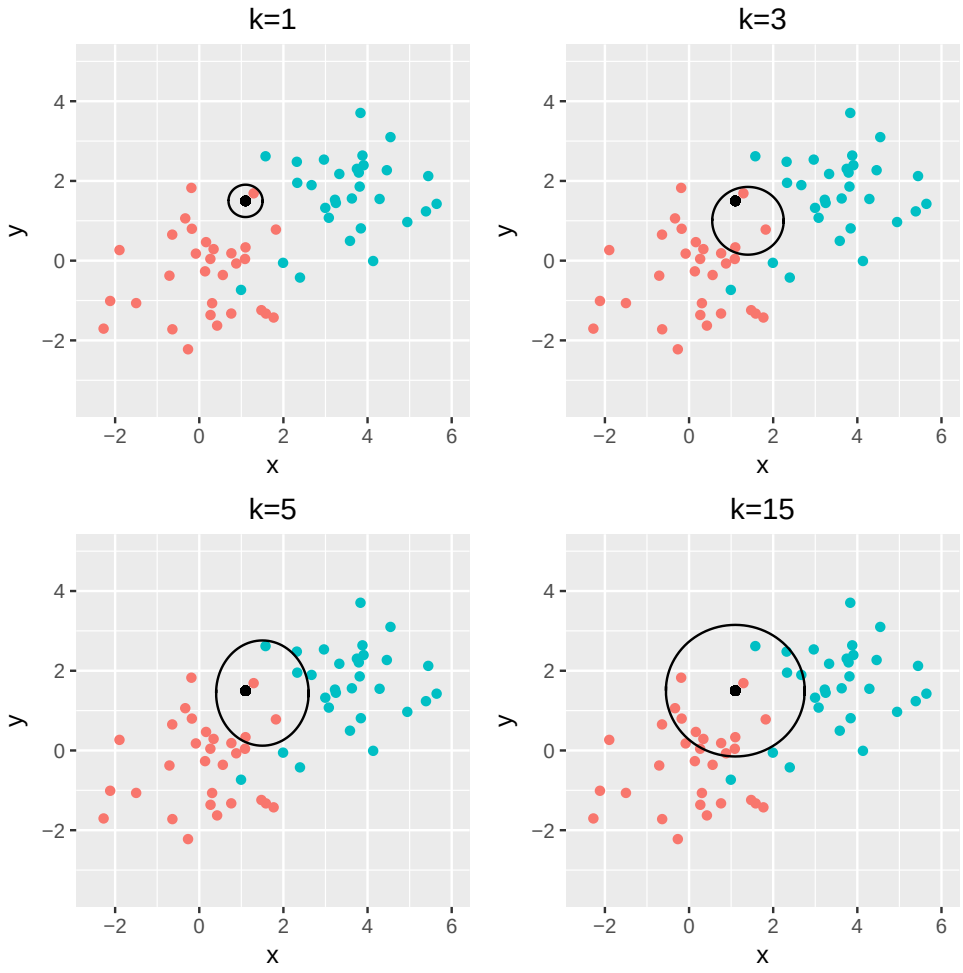


FIGURE 2.5 – Illustration de l'algorithme pour le classement du nouveau point représenté en noir, en fonction du nombre de voisins choisis.

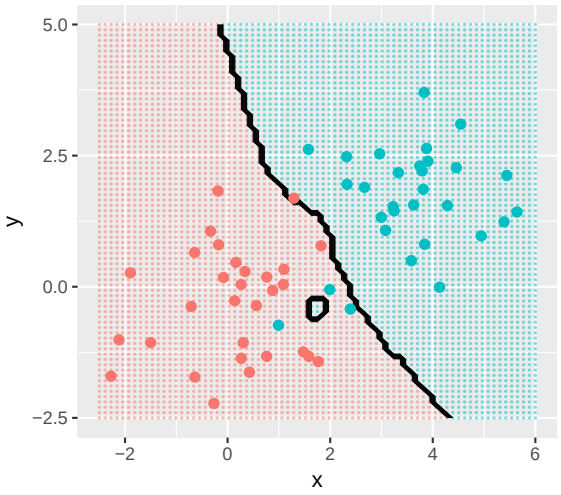


FIGURE 2.6 – Frontières de décision pour $k = 5$

que centrer les données n'a pas d'influence sur la distance).

Il est également évident que le choix du nombre de voisins k a une influence sur les résultats de l'algorithme. Si on choisit $k = 1$, on obtient une méthode locale qui s'intéresse seulement au plus proche voisin de $\mathbf{x}$. Si au contraire on choisit $k = n$, on affecte $\mathbf{x}$ à la classe majoritaire dans tout l'échantillon, et on obtient alors une règle de classement constante pour tout nouveau point. Choisir un k élevé permet de réduire le bruit en lissant les résultats, et de diminuer le risque de sur-apprentissage, mais produit également des frontières moins nettes entre les classes. En pratique, il faut alors trouver un compromis pour la valeur de k. On peut par exemple comparer différentes valeurs de k en comparant les taux de mauvais classement obtenus par validation croisée. Par ailleurs, dans le cas de deux groupes, on choisira toujours k impair.

Remarque. *Contrairement aux méthodes précédentes, on ne construit pas de règle d'affectation qui puisse être utilisée sur de nouveaux points. Avec cette méthode, on a besoin de lancer l'algorithme pour chaque nouveau point que l'on veut classer. On peut toutefois obtenir des frontières de classement en lançant l'algorithme pour des points bien choisis dans le plan (voir figure 2.6).*

2.5.3 k-plus proches voisins pondérés

Dans la méthode présentée dans la section précédente, les k voisins contribuent de façon équivalente au calcul de $\hat{t}_\ell(\mathbf{x})$. On considère dans cette section une extension de cette approche, en attribuant à chaque voisin un poids qui dépend de sa distance au point $\mathbf{x}$, et qui va en particulier décroître lorsque cette distance augmente. Ainsi, les points les plus proches de $\mathbf{x}$ sont ceux qui contribuent le plus à l'estimation de $\hat{t}_\ell(\mathbf{x})$. Pour cela, on va définir, en plus du paramètre k et de la distance d, une fonction noyau K.

Nom	Expression
rectangulaire	$K(x) = \frac{1}{2}\mathbf{1}_{\|x\|\leq 1}$
triangulaire	$K(x) = (1-\|x\|)\mathbf{1}_{\|x\|\leq 1}$
Epanechnikov	$K(x) = \frac{3}{4}(1-x^2)\mathbf{1}_{\|x\|\leq 1}$
bi-poids	$K(x) = \frac{15}{16}(1-x^2)^2\mathbf{1}_{\|x\|\leq 1}$
tri-poids	$K(x) = \frac{35}{32}(1-x^2)^3\mathbf{1}_{\|x\|\leq 1}$
cosinus	$K(x) = \frac{\pi}{4}\cos(\frac{\pi}{2}x)\mathbf{1}_{\|x\|\leq 1}$
Gaussien	$K(x) = \frac{1}{\sqrt{2\pi}}e^{-x^2/2}$
inverse	$K(x) = \frac{1}{\|x\|}$
exponentiel	$K(x) = \frac{a}{2}e^{-a\|x\|}$

TABLE 2.1 – Noyaux usuels.

Définition 16. *Une fonction K de $\mathbb{R}$ dans $\mathbb{R}$ est un **noyau** si elle vérifie :*

- $\int_{\mathbb{R}} K(x)dx = 1$
- *K est paire, c'est-à-dire $K(-x) = K(x)$*
- $K(x) \geq 0, \forall x \in \mathbb{R}$
- *K atteint son maximum en $x = 0$ (peut être relachée en "K tend vers une valeur maximale en 0")*
- *K est décroissante en $-\infty$ et en $+\infty$*

Les choix les plus classiques pour la fonction noyau sont présentés dans le tableau 2.1, et illustrés sur la figure 2.7. Le choix du noyau K n'a en pratique pas une grande influence sur les résultats, sauf pour le noyau rectangulaire.

L'idée de l'algorithme des k-plus proches voisins pondérés est d'attribuer à chaque point $\mathbf{x}_i$ de $\mathcal{V}_d$ le poids $K(d(\mathbf{x}, \mathbf{x}_i))$. Cependant, quelques précautions sont nécessaires en fonction de la combinaison (K, d) choisie. Par exemple, on remarque que plusieurs noyaux sont à support sur $[-1, 1]$. Il n'est donc pas question, en choisissant l'un de ces noyaux, d'utiliser directement la distance $d(\mathbf{x}, \mathbf{x}_i)$, qui peut tout à fait être supérieure à 1 pour tous les points de notre voisinage. Dans ce cas, on peut standardiser les distances obtenues afin qu'elles soient comprises entre 0 et 1. Plus précisément, et pour s'assurer que tous les points du voisinage ont un poids non nul, on peut normaliser la distance $d(\mathbf{x}, \mathbf{x}_i)$ entre chaque point $\mathbf{x}_i$ de $\mathcal{V}_d(\mathbf{x})$ et $\mathbf{x}$ par la distance $d(\mathbf{x}, \mathbf{x}_{k+1})$, où $\mathbf{x}_{k+1}$ est le $(k+1)$-ème plus proche voisin de $\mathbf{x}$. L'algorithme des k-plus proches voisins pondérés est décrit en détails dans l'encadré 1.

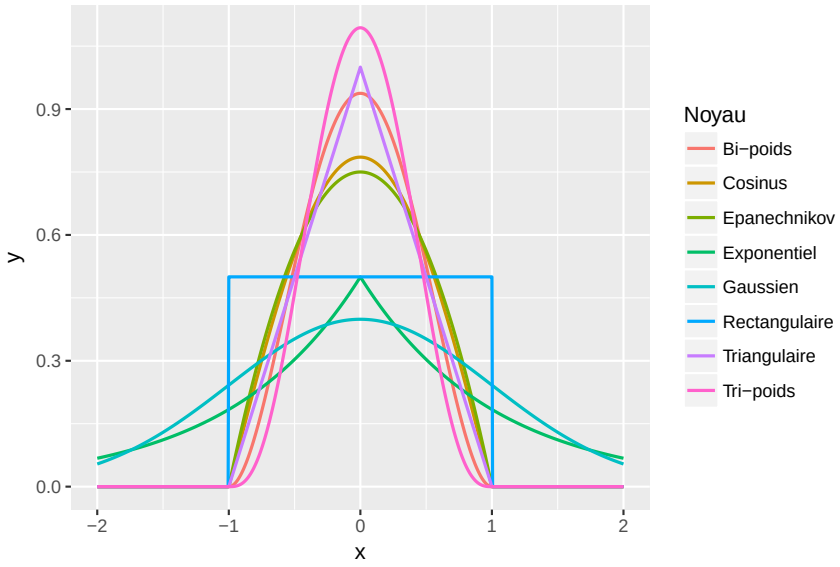


FIGURE 2.7 – Représentation des différentes fonctions noyaux présentées dans le tableau 2.1

Algorithme 1 : k-plus proches voisins pondérés pour le classement d'un point $\mathbf{x}$

---

1 Choix des paramètres de l'algorithme : nombre de voisins k, distance d et noyau K

2 Calcul de la distance $d(\mathbf{x}, \mathbf{x}_i)$ pour $i = 1, \ldots, n$

3 Sélection des $k+1$ plus proches voisins, notés $\mathbf{x}_{(1)}, \ldots, \mathbf{x}_{(k+1)}$

4 Normalisation des distances pour obtenir des distances comprises entre 0 et 1 (sauf pour les noyaux Gaussiens et exponentiel) :

$$d_i = \frac{d(\mathbf{x}, \mathbf{x}_{(i)})}{d(\mathbf{x}, \mathbf{x}_{(k+1)})}, \quad i = 1, \ldots, k$$

5 Calcul des poids associés à chaque point $w_i = K(d_i)$

6 Affectation du point $\mathbf{x}$ à la classe C_ℓ telle que la quantité suivante soit maximale :

$$\arg\max_{\ell} \frac{1}{k} \sum_{i=1}^{k} w_i \mathbf{1}_{Y_i = \ell}$$

Chapitre 3

Méthodes de discrimination par arbre (méthodes de segmentation)

Les méthodes de discrimination par arbre, aussi appelées méthodes de segmentation, sont basées sur la construction d'un arbre de décision. Un exemple d'arbre de décision est donné dans la figure 3.2. Il existe plusieurs méthodes de discrimination par arbre, et nous présentons ici la méthode CART (*Classification and Regression Tree*), introduite par Breiman *et al.* (1984), qui est l'une des plus populaires.

3.1 Principes généraux

L'idée générale des arbres de décision binaire est de procéder par divisions binaires successives de l'espace des variables, afin d'obtenir une partition de l'espace en sous-espaces sur lesquels la variable Y est constante. La figure 3.1 illustre la méthode dans le cas de deux variables explicatives X_1 et X_2, et d'une variable à expliquer Y à deux modalités.

Afin de détailler l'algorithme, on commence par introduire quelques définitions.

Définition 17.

- *Un **arbre** est un graphe orienté acyclique, constitué d'un ensemble de nœuds N et d'arrêtes E.*
- *Un **arbre enraciné** est un arbre possèdant une seule racine, et tel que chaque nœud sauf la racine e a un unique parent.*
- *Une **feuille** est un nœud n'ayant pas de successeurs. On note F l'ensemble des feuilles.*
- *Une feuille est dite **pure** si tous ses éléments appartiennent à la même classe.*

On peut construire l'arbre correspondant à un ensemble de divisions binaires, en définissant le nœud créé à l'étape k comme la condition permettant de créer les deux nouveaux sous-ensembles. En reprenant l'exemple de la figure 3.1, on obtient l'arbre de décision présenté en 3.2.

L'algorithme CART fonctionne en deux étapes. Dans un premier temps, on construit à partir des divisions successives de l'espace des variables, un arbre dont toutes les feuilles sont pures. On obtient

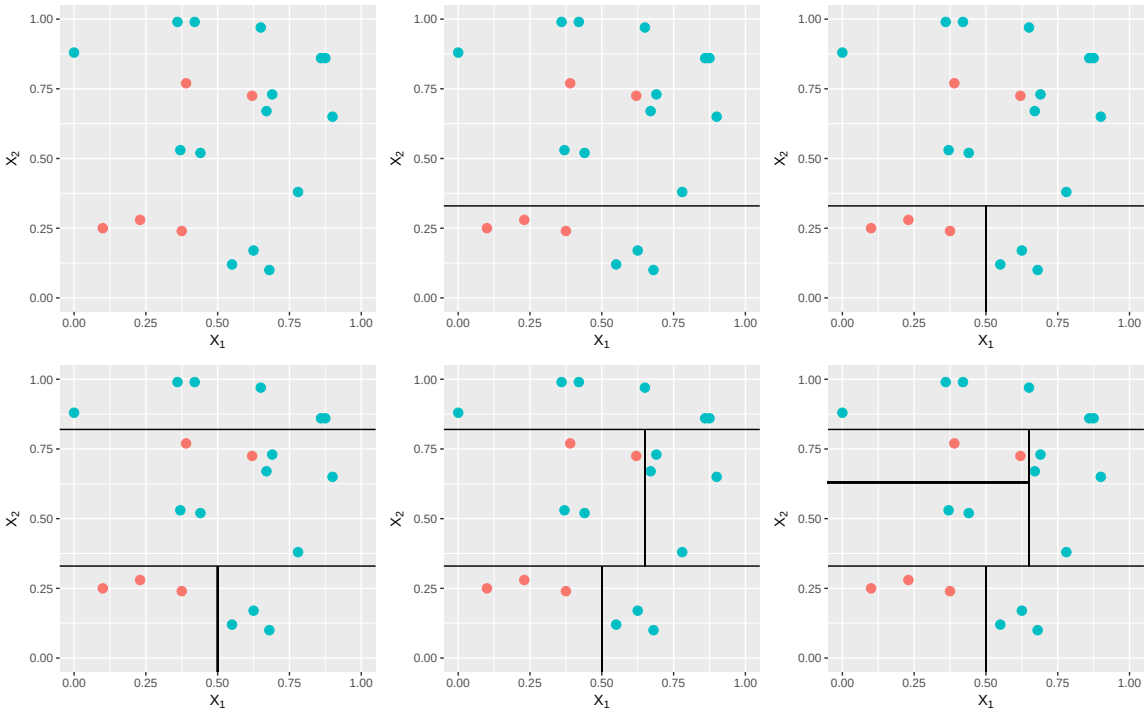


FIGURE 3.1 – Divisions binaires successives de l'espace des variables pour obtenir une partition en sous-ensembles sur lesquels les observations appartiennent à la même classe.

alors l'arbre appelé *arbre maximal* et noté $A_{\max}$. Cependant, si les données sont très fragmentées, on va typiquement obtenir des feuilles correspondant à des sous-ensemble de très petite taille, voire même réduits à un seul élément, ce qui augmente le risque de sur-apprentissage. La deuxième étape de l'algorithme consiste alors à *élaguer* l'arbre, c'est-à-dire à supprimer des branches en regroupant des feuilles (ou nœuds terminaux).

3.2 Construction de l'arbre maximal $A_{\max}$

On s'intéresse dans cette section à la construction de l'arbre maximal, qui repose sur les trois étapes suivantes :

— divisions successives des nœuds

— choix d'un critère d'arrêt de la division

— affectation des individus de chaque feuille à un groupe.

3.2.1 Division des nœuds

Comme on peut le voir déjà sur l'exemple simple présenté dans les figures 3.1 et 3.2, il y a plusieurs façons de diviser un nœud à chaque itération de l'algorithme. Comment choisir la "meilleure" division, au regard de notre objectif final ? Commençons par introduire quelques notions.

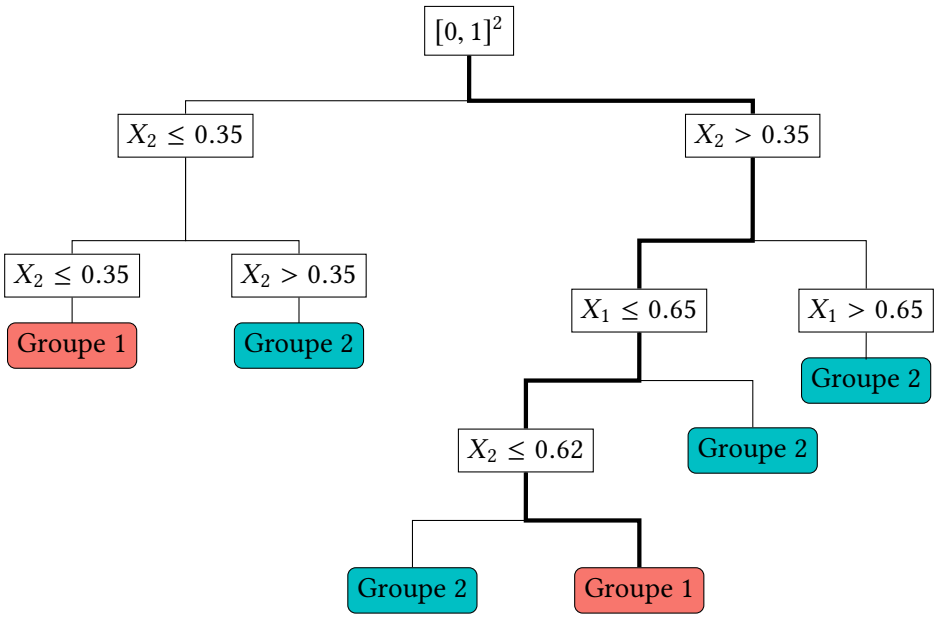


FIGURE 3.2 – Exemple d'arbre de décision associé aux divisions binaires de la figure 3.1. Pour classer un nouvel individu dont les variables explicatives valent $(x_1, x_2) = (0.50, 0.75)$, on procède de façon itérative en descendant le long des branches de l'arbre. La trajectoire suivie est représentée par un trait plus épais le long de l'arbre. Le premier nœud concerne la variable X_2, qui vaut ici 0.75 et est donc supérieure à 0.35 : on part donc dans la branche droite de l'arbre. Sur cette branche droite, le deuxième nœud concerne la variable X_1 qui vaut ici 0.50 et est donc inférieure à 0.65 : on part alors dans la branche gauche. On rencontre alors un dernier nœud qui concerne de nouveau la variable X_2 : celle-ci étant supérieure à 0.62, on part dans la branche de droite et on affecte donc le nouvel individu au groupe 1. ***Attention, il s'agit uniquement d'une illustration, les divisions ont été choisies sans respecter de critère particulier, mais uniquement pour illustrer l'approche.***

On va tout d'abord définir l'ensemble des divisions admissibles, c'est-à-dire celles parmi lesquelles il faudra choisir la "meilleure", au sens d'un critère que l'on définira plus tard.

Définition 18. *Une division est dite **admissible** si aucun des nœuds successeurs n'est vide.*

En pratique on peut compter le nombre de divisions admissibles pour une variable explicative X données, en fonction de son type :

- si X est *qualitative ordinale* à m modalités, on a $m-1$ coupures possibles
- si X est *qualitative nominale* à m modalités, on $2^{m-1}-1$ divisions possibles
- si X est *quantitative*, on a en théorie une infinité de divisions possibles, mais en pratique on observe un nombre fini de réalisations de X. On traite alors X comme s'il s'agissait d'une variable qualititative ordinale.

Dans la pratique, l'algorithme a un légère tendance à favoriser les variables avec un grand nombre de modalités, car elles offrent un plus grand nombre de divisions possibles et ont donc plus de chances d'être sélectionnées.

Dans la suite, on note $p_k(a) = \mathbb{P}(Y = k \mid a)$ la probabilité d'être dans la classe k sachant que l'on est dans le nœud a. Ces probabilités sont estimés par :

$$\hat{p}_k(a) = \frac{n_{k,a}}{n_a},$$

où $n_{k,a}$ est le nombre d'observations du nœud a qui appartiennent à la classe C_k et n_a le nombre d'observations du nœud a.

Impureté d'un nœud

On va ensuite définir l'impureté d'un nœud, qui va mesurer l'hétérogénéité d'un nœud par rapport à la variable Y.

Définition 19. *L'**impureté** d'un nœud a est définie par une fonction i, définie de l'ensemble des nœuds $\mathcal{N}$ dans $\mathbb{R}^+$ telle que :*

$$i(a) = f(p_1(a), \ldots, p_K(a)),$$

avec f vérifiant :

- *f est une fonction symétrique des variables $p_k(a)$ (i.e. invariante par permuation des variables)*
- *f est maximale si $p_k(a) = \frac{1}{K}, \forall k = 1, \ldots, K$*
- *f est minimale si $\exists k \mid p_k(a) = 1$*

Les deux fonctions d'impuretés les plus utilisées sont (voir figure 3.3) :

- l'entropie de Shannon : $i(a) = -n_a \sum_{k=1}^{K} p_k(a) \ln p_k(a)$
- l'indice de Gini : $i(a) = n_a \sum_{k=1}^{K} \sum_{\ell \neq k} p_k(a) p_\ell(a) = n_a \sum_{k=1}^{K} p_k(a)(1 - p_k(a))$

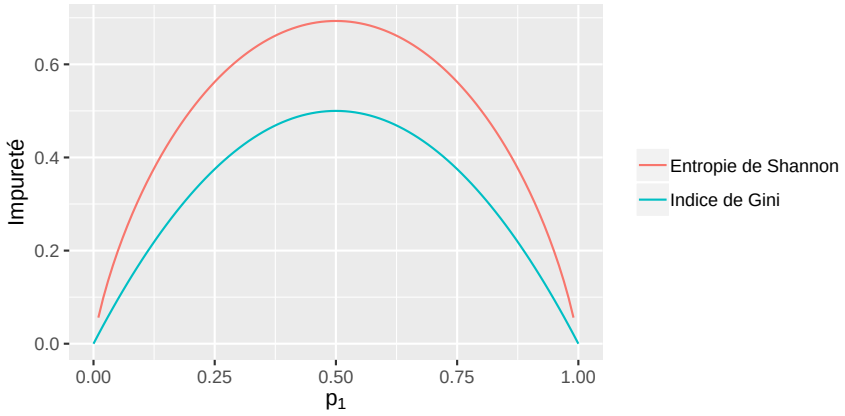


FIGURE 3.3 – Allure des fonctions d'impuretés basées sur l'indice de Gini et l'entropie de Shannon, dans le cas de deux classes, en fonction de la valeur de p_1

Critère de division d'un nœud

À partir des définitions précédentes, on peut définir le critère de division suivant : parmi toutes les divisions admissibles du nœud a, on retient celle qui maximise la réduction de l'impureté du nœud. Soit d une division du nœud a en deux nœuds descendants $a_{G,d}$ (nœud de gauche) et $a_{D,d}$ (nœud de droite). On définit $\Delta i(d, a)$ comme la réduction de l'impureté du nœud a par la division d, et on a :

$$\Delta i(d, a) = i(a) - i(a_{G,d}) - i(a_{D,d}).$$

On retient alors, parmi toutes les divisions admissibles $\mathcal{D}$ du nœud a, la division d^* qui maximise la réduction de l'impureté $\Delta i(d, a)$. Cependant dans la pratique les $p_k(a)$ sont inconnus, on va donc utiliser une estimation de l'impureté :

$$d^* = \arg\max_{d \in \mathcal{D}} \Delta\hat{i}(d, a), \qquad \text{où} \qquad \hat{i}(d, a) = f(\hat{p}_1(a), \ldots, \hat{p}_K(a)).$$

3.2.2 Critère d'arrêt

L'algorithme s'arrête lorsque tous les nœuds sont purs, ou lorsqu'il n'existe plus de division admissible pour ce nœud. Le dernier cas correspond à la situation où toutes les observations sont identiques sur les variables explicatives. Dans la pratique, on peut également arrêter la division d'un nœud si l'effectif de celui-ci est trop petit (par exemple, inférieur à 5).

3.2.3 Affectation des individus

L'arbre $A_{\max}$ permet de partitionner l'espace des variables en sous-ensembles, définis par les nœuds de l'arbre. Pour un individu dont les caractéristiques sont $\mathbf{x}$, on note $a(\mathbf{x})$ le nœud contenant $\mathbf{x}$.

Une fois que l'on a identifié le nœud auquel appartient l'individu, il reste à l'affecter à l'un des groupes $C_1, \ldots, C_K$. Si le nœud a est pur, on affecte l'individu à la classe unique représentée dans le nœud. Si le nœud a n'est pas pur, on utilise la règle optimale de Bayes présentée en section 2.1.2 (avec

la fonction de perte 0-1), c'est-à-dire que l'on affecte l'individu à la classe C_k majoritaire. La règle de décision peut s'écrire :

$$r(\mathbf{x}) = k \Leftrightarrow k = \arg\min_{\ell} p_\ell(a(\mathbf{x})) \tag{3.1}$$

Autrement dit, l'algorithme CART produit une règle de décision constante par morceaux, sur chacun des sous-ensembles de l'espace des variables définis par les nœuds terminaux de l'arbre.

N.B. L'algorithme CART peut également être utilisé dans le cadre de la régression, i.e. avec Y **quantitative**. Dans ce cas, on ne cherche pas à minimiser le taux de mauvais classement mais la variance de la variable cible. Autrement dit, on privilégiera les divisions qui conduisent à des noeuds fils au sein desquels la variance de la variable cible sera inférieure à celle dans le noeud parent. Puis, pour le sous-ensemble défini par le nœud terminal a, l'arbre renvoit $\hat{Y} = \frac{1}{n_a} \sum_{i=1}^{n_a} Y_{i,a}$, où $(Y_{i,a}), i = 1, \ldots, n_a$ est l'ensemble des observations du nœud a.

3.3 Élagage de l'arbre maximal

L'un des principaux inconvénients de l'arbre $A_{\max}$ ainsi construit est qu'il est très raffiné et particulièrement instable. Une petite modification des données peut conduire à un arbre maximal très différent. La variance de la méthode est donc élevée, et les risques de sur-ajustement aussi. Plusieurs approches peuvent être adoptées pour limiter ces risques. La démarche utilisée par l'algorithme CART consiste à élaguer l'arbre, c'est-à-dire à supprimer des branches en regroupant des nœuds.

Pour cela, on va construire une suite emboîtée de sous-arbres de $A_{\max}$ par élagages successifs, telle que le coût de mauvais classement *apparent* associé au sous-arbre obtenu à chaque étape de l'élagage soit le plus faible parmi les sous-arbres possibles ayant le même nombre de nœuds. Puis, on compare les différents sous-arbres obtenus et on retient celui qui minimise un certain critère (que l'on précisera plus loin). Comme on ne procède pas à une comparaison exhaustive de tous les sous-arbres de $A_{\max}$, mais seulement les sous-arbres de la séquence emboîtée, la stratégie est sous-optimale. Cependant, elle permet de réduire les temps de calculs et donne des résultats satisfaisants dans la pratique.

Construction de la séquence de sous-arbres

Définition 20. *Soit $\alpha > 0$. Le critère de **coût-complexité** d'un arbre A est défini par :*

$$C_\alpha(A) = C(A) + \alpha|F|, \tag{3.2}$$

où $|F|$ est le nombre de feuilles de l'arbre A, et $C(A)$ est le coût de mauvais classement de l'arbre A. En cas d'égalité des coûts (i.e. avec la fonction de perte 0-1), $C(A)$ correspond au taux de mauvais classement. On appelle α le paramètre de complexité.

Autrement dit, on pénalise le coût de l'arbre par un paramètre supplémentaire lié au nombre de nœuds terminaux. En effet, on a en général une réduction du coût de mauvais classement d'un arbre

lorsqu'on divise un ou plusieurs de ses nœuds terminaux, et on diminue donc, en général, le terme $C(A)$ en considérant un arbre A' tel que $A \subset A'$. C'est pourquoi on pénalise le coût par un paramètre rendant compte de la complexité de l'arbre [1].

Plus α est petit, plus la pénalité associée au nombre de nœuds est faible, et plus on a tendance à choisir des arbres ayant un grand nombre de nœuds. À l'inverse, si α est grand, on a tendance à choisir des arbres ayant peu de nœuds terminaux. À la limite, c'est-à-dire lorsque α est suffisamment grand, $A(\alpha)$ ne contient qu'un nœud, la racine de l'arbre $A_{\max}$.

Proposition 4. *Soit $\alpha > 0$. Il existe un unique sous-arbre de $A_{\max}$, noté $A(\alpha)$, qui minimise C_α.*

$$A(\alpha) = \arg \min_{A \subset A_{\max}} C_\alpha(A).$$

Proposition 5. *Soit $\alpha_1 \leq \alpha_2$, alors $A(\alpha_1) \subset A(\alpha_2)$.*

L'idée de l'algorithme consiste à considérer une séquence croissante $\alpha_1 = 0, \alpha_2, \ldots, \alpha_H$ de paramètres de complexité, et à construire la séquence de sous-arbres emboîtés $A(\alpha_1), A(\alpha_2), \ldots, A(\alpha_H)$.

1. on pose $\alpha_1 = 0$, et on cherche l'arbre $A_1 = A(\alpha_1)$ qui est par définition le plus petit sous-arbre de $A_{\max}$ qui minimise C_0, c'est-à-dire qui minimise le coût de mauvais classement. Comme la division d'un nœud en deux entraîne soit une décroissance du coût de mauvais classement, soit une stagnation de ce coût, on peut chercher les nœuds dont la division n'a pas pas entraîné de diminution du coût, c'est-à-dire les noeuds a tels que $C(a) = C(a_G) + C(a_D)$. On supprime alors les branches issues de ces nœuds, et c'est ainsi que l'on obtient l'arbre A_1.
2. par définition de A_1, on a pour tout nœud a, $C(a) > C(B(a))$, où $B(a)$ est la branche partant du nœud a. En effet, si ce n'était pas le cas, cela voudrait dire que le nœud a a un coût de mauvais classement plus faible, et on aurait alors élagué l'arbre en supprimant la branche issue du nœud a. Il devient intéressant d'élaguer l'arbre A_1 au niveau du nœud a lorsque :

   $$C(a) + \alpha \leq C(B(a)) + \alpha|B(a)|,$$

   c'est-à-dire lorsque

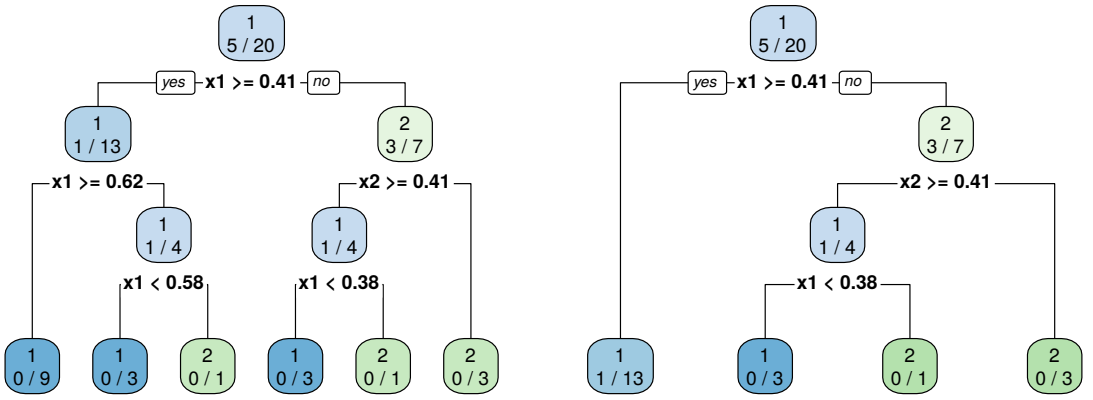
   $$\alpha \geq \frac{C(a) - C(B(a))}{|B(a)| - 1}.$$

   On note $s(a, B(a)) = \frac{C(a)-C(B(a))}{|B(a)|-1}$, et on pose

   $$\alpha_2 = \min_a s(a, B(a)).$$

   On définit alors l'arbre $A_2 = A_{\alpha_2}$ comme étant le sous-arbre obtenu en élaguant toutes les branches issues de nœuds pour lesquels $s(a, B(a))$ est minimale.
3. On réitère le processus jusqu'à obtenir un arbre à un seul nœud.

1. Cette démarche est assez classique en sélection ou comparaison de modèle, où l'on considère des critères de sélection ou de comparaison qui dépendent d'un terme d'ajustement et d'un terme de pénalité. C'est le cas par exemple des critères AIC et BIC



(a) *Arbre maximal* $A_{\max}$ *(=*$A(\alpha_1)$*)*

(b) *Arbre* $A(\alpha_2)$

FIGURE 3.4 – Elagage de l'arbre maximal obtenu par la méthode CART sur les données de la figure 3.1. Dans chaque nœud, la première ligne donne le numéro de la classe à laquelle sont affectées les observations du nœud, et la deuxième ligne donne le taux de mauvais classement dans le nœud.

Une illustration de la méthode sur les données de la figure 3.1 est présentée en figure 3.4. L'arbre maximal correspondant n'a que des feuilles pures, et le taux de mauvais classement est nul. Dans ce cas, $A(\alpha_1) = A_{\max}$. On calcule ensuite, pour chaque nœud a de l'arbre, la valeur de la fonction $s(a, B(a))$. En numérotant les nœuds de haut en bas et de gauche à droite, on obtient :$s(1, B(1)) = (0.2-0)/(6-1) = 0.05$, $s(2, B(2)) = (1/13-0)/(3-1) = 0.038$, $s(3, B(3)) = (3/7-0)/(3-1) = 0.214$, $s(4, B(4)) = (0.25-0)/1 = 0.25$ et $s(5, B(5)) = (0.25 - 0)/1 = 0.25$. On obtient alors $\alpha_2 = 0.038$, et on élague la branche 2. On recalcule $s(1, B(1)) = (0.25-1/13)/(4-1) = 0.06$: on pose donc $\alpha_3 = 0.06$ et on élague la branche partant du nœud 1. On obtient alors l'arbre contenant uniquement la racine. Il est à noter que l'illustration présentée ici repose sur le calcul du taux de mauvais classement par resubsitution, c'est-à-dire directement sur l'échantillon d'apprentissage. On obtiendrait des résultats différents si on calculait le taux de mauvais classement par validation croisée.

Une fois que l'on a construit la suite de sous-arbres emboîtés, il nous reste à en sélectionner un. Plusieurs approches sont possibles :

- *méthode du coude* : on trace l'évolution de $C_\alpha(A_\alpha)$ en fonction de α, on obtient une courbe décroissante et on peut chercher la présence d'un seuil (ou "coude") sur ce graphe
- *méthode de l'échantillon test* : on divise l'échantillon initial en un échantillon d'apprentissage, sur lequel on construit $A_{\max}$ et la séquence $A(\alpha_1), \ldots, A(\alpha_H)$, et un échantillon de test qui permet de choisir le sous-arbre qui minimise le cout de mauvais classement.
- *validation croisée* (v-fold cross validation) : on construit $\binom{n}{v}$ partitions de l'échantillon en deux parties de tailles respectives $n - v$ et v, la première servant d'échantillon d'apprentissage et la deuxième d'échantillon test. On applique alors la stratégie décrite au point précédent et on utilise la moyenne des erreurs obtenues.

Chapitre 4

Méthodes d'ensemble (boosting, bagging, forêts aléatoires)

On distingue deux grandes familles de méthodes d'ensemble : les méthodes de *boosting*, qui reposent sur une construction récursive et *adaptative* de la famille de classifieurs faibles, et les méthodes de *bagging* (pour *Bootstrap Aggregating*), qui utilisent plusieurs échantillons bootstrap [1] pour construire la famille de classifieurs. Les forêts aléatoires sont un exemple d'algorithme de type *bagging*, développé dans le cadre spécifique des arbres de décision CART.

Les méthodes d'ensemble fonctionnent bien, i.e. permettent d'obtenir de meilleurs résultats qu'avec la famille de classifieurs faibles, si ceux-ci sont particulièrement instables (ont une variance élevée). En effet, c'est dans ce cadre qu'il est alors le plus judicieux d'aggréger les résultats des différents classifieurs afin de diminuer la variance des prédictions. En principe, n'importe quelle famille de classifieurs faibles peut être utilisée, même si en pratique la plupart des méthodes reposent sur l'utilisation d'arbres de décision. Ces derniers, de part leur instabilité, sont en effet de bons candidats aux méthodes d'ensemble.

4.1 Méthodes de bagging

L'objectif des méthodes de bagging est de diminuer l'instabilité d'un algorithme, en diminuant sa variance. On a vu par exemple qu'avec les arbres de décision, une petite variation dans les données pouvait conduire à un prédicteur très différent.

Leo Breiman, à qui l'on doit des contributions majeures dans le domaine de l'apprentissage et en particulier celui des arbres CART, a notamment montré que le bagging permettait d'améliorer substantiellement les résultats d'algorithmes instables (typiquement des arbres de décision), mais pouvait au contraire légèrement dégrader les résultats de méthodes plus stables (par ex. k-plus proches voisins avec un k élevé).

1. Voir Annexe B pour une introduction au Bootstrap

4.1.1 Principes généraux

À partir de l'échantillon d'apprentissage initial $(X_1, Y_1), \ldots, (X_n, Y_n)$, on tire avec remise B échantillons bootstrap de taille n, que l'on note $\mathcal{D}_b = \{(X_1^b, Y_1^b), \ldots, (X_n^b, Y_n^b)\}$. Sur chacun des échantillons bootstrap $\mathcal{D}_b$, on construit un classifieur. Dans le cas des arbres de décision, on construit l'arbre maximal $A_{\max}^b$, associé à la règle de décision r_b telle que définie en (3.1). Enfin, on définit le classifieur final en faisant appel aux B arbres à l'aide d'un vote majoritaire.

$$r_{\text{bagg}}(\mathbf{x}) = k \Leftrightarrow k = \arg\max_{\ell} \sum_{b=1}^{B} \mathbf{1}_{r_b(\mathbf{x})=\ell} \tag{4.1}$$

De façon générale, les méthodes de bagging permettent de construire un classifieur qui aura le même biais que les classifieurs qui le composent, mais une variance plus faible. Comme il n'est pas possible d'avoir un biais et une variance faible, il faut donc faire attention au choix des classifieurs. On aura ainsi intérêt à choisir des classifieurs avec un biais faible, pour obtenir un classifieur final de faible biais, mais la variance des classifieurs ne doit pas être trop élevée, sinon il faudra aggréger un grand nombre de classifieurs pour espérer diminuer la variance de façon significative.

Remarque. *Il existe des variantes du tirage bootstrap présenté ici, où on tire avec remise un échantillon de taille $l < n$. Des résultats ont montré que si l'on choisit une taille d'échantillon l_n, avec $\lim_{n\to+\infty} l_n = +\infty$ et $\lim_{n\to+\infty} \frac{l_n}{n} = 0$, on obtient les mêmes garanties théoriques que dans le cas $l = n$.*

4.1.2 Les forêts aléatoires

L'algorithme des forêts aléatoires est une méthode de bagging appliquée aux arbres CART (pour la régression ou pour la classification). L'idée principale est la même que celle du bagging présentée dans la section précédente, mais on ajoute une part d'aléatoire dans le processus, en ne sélectionnant pour chaque échantillon bootstrap, qu'un sous-ensemble de taille m des variables explicatives. L'algorithme est décrit dans l'encadré 2.

Algorithme 2 : L'algorithme des forêts aléatoires

1. **initialisation** : choix du nombre d'échantillons bootstrap B, et du nombre de variables m
2. **pour** $b = 1, \ldots, B$ **faire**
3. tirer un échantillon boostrap de taille $\mathcal{D}_b$ à partir de l'échantillon initial
4. tirer au hasard m variables explicatives parmi les p variables initiales
5. construire un arbre CART A_b sur l'échantillon $\mathcal{D}_b$ en utilisant les m variables choisies précédemment
6. **sortie** : aggréger les arbres A_b dans la forêt $\mathcal{F}$ à l'aide d'un vote majoritaire

En pratique, comme dans le cas général du bagging, il faut faire un compromis biais-variance pour le choix de m. Lorsque m est petit, il y a moins de variables en compétition pour la division des nœuds,

et on se rapproche donc d'une procédure au hasard, basée uniquement sur le tirage des m variables parmi p. Seul le point en lequel on coupe la variable en deux sera basée sur l'échantillon. Les différents arbres construits sur les échantillons bootstrap seront donc peu corrélés, ce qui entraîne une baisse de la variance globale après aggrégation. Par contre, les classifieurs ainsi obtenus seront plus fortement biaisés, car moins bien ajustés au jeu de données. À l'inverse, si on augmente le nombre de variables m, on obtiendra des arbres bootstrap qui se ressemblent plus, une variance globale après aggrégation moins bonne mais un plus faible biais. En pratique, on construit des arbres dont les nœuds terminaux sont de petit taille, ce qui implique un faible biais mais une plus forte variance, que l'on compense par une faible valeur de m. Le choix par défaut de la fonction `randomForest`, sous `R`, est $m = \sqrt{p}$. On peut également utiliser des méthodes de validation croisée pour choisir la valeur de ce paramètre.

4.1.3 Erreur "out-of-bag"

L'un des intérêts des méthodes de bagging est qu'elles permettent d'estimer l'erreur de classement sans avoir recours à un découpage de la base de données en base d'apprentissage / base de test. En effet, par définition d'un tirage bootstrap, à chaque tirage b on ne sélectionne qu'un sous-ensemble de l'échantillon initial. Prenons l'exemple des forêts aléatoires (mais cela fonctionne de la même façon pour toutes les méthodes de type bagging). Notons $\mathcal{F}_B^i$ la forêt aléatoire constituée de l'ensemble des arbres qui ont été construits sur des échantillons bootstrap ne contenant pas l'observation i. On peut alors faire voter les arbres de cette forêt pour obtenir une prédiction $\hat{y}_{\text{OOB},i}$ par un vote majoritaire.

On définit ensuite l'*erreur out-of-bag* par :

$$\hat{e}_{\text{OOB}}(\mathcal{F}) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}_{\hat{y}_{\text{OOB},i} \neq y_i} \tag{4.2}$$

4.2 Méthodes de boosting

Le boosting a vu le jour à la fin des années 1980 et au début des années 1990, en réponse à la question de savoir s'il était possible de construire, à partir d'un ensemble de classifieurs faibles, un classifieur "fort". Le premier algorithme de boosting, *AdaBoost* (pour *Adaptive Boosting*) a été introduit quelques années plus tard par Freund et Schapire (1997), et a longtemps été considéré comme une référence. Il s'agit d'un domaine de recherche très actif, et de nombreuses variantes ou nouveaux algorithmes sont proposés régulièrement.

4.2.1 AdaBoost

Description de l'algorithme

L'algorithme AdaBoost a été initialement élaboré dans le cadre d'une classification binaire, et nous le présentons dans ce cadre afin de faciliter les notations. Nous supposons donc dans la suite que nous disposons d'un échantillon i.i.d. $(X_1, Y_1), \ldots, (X_n, Y_n)$, où $X_i \in \mathbb{R}^p$ et $Y_i \in \{-1, 1\}$.

On commence par construire une première règle de décision sur l'échantillon, où chaque observation a le même poids. On calcule l'erreur commise par cette première règle de décision, puis on met à jour les poids associés à chaque observation, en augmentant le poids de celles qui ont été mal classées. Ainsi, la règle de décision qui sera construite à l'étape d'après se concentrera plus sur ces observations mal classées. Le classifieur final est obtenu comme un vote majoritaire pondéré des différents classifieurs [2]. L'algorithme est décrit en détails dans l'encadré 3.

Algorithme 3 : AdaBoost pour la classification binaire

1 **initialisation** : on définit le poids de l'observation i, $i = 1, \ldots, n$ par $w_i^0 = 1/n$, et on note $\mathbf{w}^0 = (w_1^0, \ldots, w_n^0)$

2 **pour** $m = 1, \ldots, M$ **faire**

3 construire la règle de décision r_m sur l'échantillon pondéré par $\mathbf{w}^m$ (soit en utilisant une méthode qui puisse s'appliquer à des échantillons pondérés, soit en construisant un échantillon dans lequel les observations sont échantillonnées selon leur poids)

4 calculer le taux d'erreur apparent associé à r_m :

$$\hat{e}(r_m) = \sum_{i=1}^{n} w_i^{m-1} \mathbf{1}_{r_m(\mathbf{x}_i) \neq y_i}$$

5 calculer

$$\alpha_m = \frac{1}{2} \log \left(\frac{1 - \hat{e}(r_m)}{\hat{e}(r_m)} \right) := \frac{1}{2} \text{logit}(\hat{e}(r_m))$$

6 mettre à jour les poids :

$$w_i^m = \begin{cases} \frac{w_i^{m-1}}{z_m} \exp(-\alpha_m) & \text{si } y_i = r_m(\mathbf{x}_i) \\ \frac{w_i^{m-1}}{z_m} \exp(\alpha_m) & \text{si } y_i = r_m(\mathbf{x}_i) \end{cases}, \tag{4.3}$$

où z_m est une constante de normalisation des poids telle que la somme des poids soit égale à 1

7 **sortie** : le classifieur final est défini par la fonction R telle que $R(\mathbf{x}) = \text{signe} \left(\sum_{m=1}^{M} \alpha_m r_m(\mathbf{x}) \right)$

Dans le cas où r_m est construit à l'aide d'un arbre CART, on peut directement utiliser les poids des observations dans la construction de l'arbre. Pour cela, il suffit de calculer l'indice de Gini en $\hat{p}_1(a), \ldots, \hat{p}_K(a)$ où cette fois on définit :

$$\hat{p}_k(a) = \sum_{x_i \in a} w_i \mathbf{1}_{x_i \in C_k}.$$

2. Pour simplifier les notations, on a supposé que la variable Y prenait ses valeurs dans $\{-1, 1\}$, dans ce cas déterminer le résultat du vote majoritaire pondéré revient à trouver le signe de la somme pondérée des classifieurs

Pour que chaque classifieur ou règle r_m ait un poids positif dans le classifieur final, il faut que l'erreur $\hat{e}(r_m)$ soit strictement inférieure à 1/2. Les classifieurs peuvent donc être faibles, mais doivent néanmoins faire mieux qu'un classement au hasard. Dans la pratique, il est donc nécessaire de vérifier que cette contrainte est bien respectée à chaque itération. Si ce n'est pas le cas, l'algorithme peut être arrêté, ou les poids ré-initialisés.

Propriétés de l'algorithme

Nous avons annoncé au début de cette section que l'algorithme AdaBoost permettait de construire un classifieur avec un très faible taux d'erreur, à partir d'un ensemble de classifieurs faibles. Cette propriété a été démontrée par Freund et Schapire.

Rappelons que chaque classifieur faible est supposé faire meux qu'une règle de classement au hasard, chaque taux d'erreur est donc inférieur à 1/2. Notons $\hat{e}(r_m) = 1/2 - \gamma_m$, où γ_m mesure donc à quel point la règle r_m fait mieux que le hasard.

Théorème 2. *Soit $\hat{e}(R)$ le taux d'erreur apparent calculé sur l'échantillon, défini par :*

$$\hat{e}(R) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}_{R(\mathbf{x}_i)\neq y_i}.$$

Alors on a :

$$\hat{e}(R) \leq \exp\left(-2\sum_{m=1}^{M}\gamma_m^2\right)$$

Démonstration. Notons tout d'abord que l'équation de mise à jour des poids peut se ré-écrire de la façon suivante :

$$w_i^m = \frac{w_i^{m-1}}{z_m}\exp(-\alpha_m y_i r_m(\mathbf{x}_i)),$$

car y_i et r_m sont à valeurs dans $\{-1, 1\}$. On a alors :

$$\begin{aligned}
w_i^M &= \frac{1}{z_M} w_i^{M-1}\exp\left(-\alpha_M y_i r_M(\mathbf{x}_i)\right) && (4.4)\\
&= \frac{1}{z_M}\left(\frac{1}{z_{M-1}} w_i^{M-2}\exp\left(-\alpha_{M-1} y_i r_{M-1}(\mathbf{x}_i)\right)\right)\exp\left(-\alpha_M y_i r_M(\mathbf{x}_i)\right) && (4.5)\\
&= w_i^0 \frac{1}{\prod_{m=1}^{M} z_m}\exp\left(-\sum_{m=1}^{M}\alpha_m y_i r_m(\mathbf{x}_i)\right) && (4.6)\\
&= \frac{1}{n}\frac{1}{\prod_{m=1}^{M} z_m}\exp\left(-y_i h(\mathbf{x}_i)\right), && (4.7)
\end{aligned}$$

où $h(\mathbf{x}_i) = \sum_{m=1}^{M}\alpha_m r_m(\mathbf{x}_i)$. Montrons maintenant que le taux d'erreur est borné par $\prod_{m=1}^{M} z_m$, que nous expliciterons.

$$\begin{aligned}
\hat{e}(R) &= \frac{1}{n}\sum_{i=1}^{n}\mathbf{1}_{R(\mathbf{x}_i)\neq y_i}\\
&= \frac{1}{n}\sum_{i=1}^{n}\mathbf{1}_{y_i h(\mathbf{x}_i)\leq 0} \quad \text{car } R(\mathbf{x}) \text{ et } y_i \text{ sont à valeurs dans } \{-1, 1\}
\end{aligned}$$

$$
\begin{aligned}
&\leq \frac{1}{n}\sum_{i=1}^{n}\exp(-y_i h(\mathbf{x}_i)) \quad \text{car } 1 \leq e^{-x} \text{ pour } x \leq 0 \\
&\leq \sum_{i=1}^{n} w_i^M \prod_{m=1}^{M} z_m \\
&\leq \prod_{m=1}^{M} z_m \quad \text{car } \sum_i w_i^m = 1 \ \forall m.
\end{aligned}
$$

Il suffit maintenant d'expliciter la quantité $\prod_{m=1}^{M} z_m$. Par définition, on a :

$$
\begin{aligned}
z_m &= \sum_{i=1}^{n} w_i^m \\
&= \sum_{i=1}^{n} w_i^{m-1} e^{-\alpha_m y_i r_m(\mathbf{x}_i)} \\
&= \sum_{i|y_i=r_m(\mathbf{x}_i)} w_i^{m-1} e^{-\alpha_m y_i r_m(\mathbf{x}_i)} + \sum_{i|y_i\neq r_m(\mathbf{x}_i)} w_i^{m-1} e^{-\alpha_m y_i r_m(\mathbf{x}_i)} \\
&= e^{-\alpha_m} \sum_{i|y_i=r_m(\mathbf{x}_i)} w_i^{m-1} + e^{\alpha_m} \sum_{i|y_i\neq r_m(\mathbf{x}_i)} w_i^{m-1} \\
&= e^{-\alpha_m}\left(1 - \sum_{i=1}^{n} w_i^{m-1}\mathbf{1}_{y_i\neq r_m(\mathbf{x}_i)}\right) + e^{\alpha_m}\sum_{i=1}^{n} w_i^{m-1}\mathbf{1}_{y_i\neq r_m(\mathbf{x}_i)} \\
&= e^{-\alpha_m}(1-\hat{e}(r_m)) + e^{\alpha_m}\hat{e}(r_m) \\
&= 2\sqrt{\hat{e}(r_m)(1-\hat{e}(r_m))} \\
&= \sqrt{1-4\gamma^2}
\end{aligned}
$$

On utilise ensuite l'inégalité $1 + x \leq e^x$ pour tout x pour obtenir le résultat final.

□

AdaBoost vu comme un algorithme d'optimisation

L'algorithme AdaBoost, même s'il n'a pas été conçu initialement avec cet objectif, peut aussi être vu comme un algorithme de minimisation de la fonction de perte exponentielle, définie par :

$$
L_{\exp}(h(\mathbf{x}), y) = \exp\left(-y h(\mathbf{x})\right), \tag{4.8}
$$

où $h(\mathbf{x}) = \sum_{m=1}^{M} \alpha_m r_m(\mathbf{x})$.

Plus précisément, on peut montrer que le choix de α_m tel que précisé dans l'encadré 3 permet de minimiser l'espérance de la fonction de perte exponentielle à chaque itération m. Cette fonction de perte constitue une borne supérieure à l'erreur de classification (aussi appelée fonction de perte "0-1") [3], ce qui signifie en particulier que si on obtient une perte exponentielle nulle, l'erreur de classification est également nulle (voir Figure 4.1 pour l'allure des fonctions de perte les plus utilisées en classification).

3. voir la preuve du théorème précédent

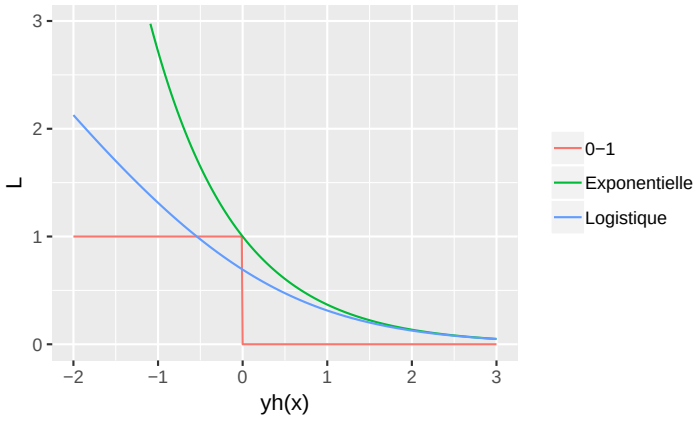


FIGURE 4.1 – Fonctions de perte usuelles en fonction de la valeur de $yh(\mathbf{x})$: fonction de perte 0-1, $L_{0-1}(h(x), y) = \mathbf{1}_{yh(x)<0}$, fonction de perte exponentielle $L_{\exp}(h(x), y) = \exp(-yh(x))$ et fonction de perte logistique $L_{logit}(h(x), y) = \log(1 + \exp(-yh(x)))$.

Pour montrer ce résultat, on se place à la fin de l'itération $m-1$, et on décompose h en deux parties : une première partie correspondant aux $m-1$ premières étapes et une deuxième partie correspondant à l'étape m, dont on souhaite montrer qu'elle permet de minimiser l'espérance de la fonction de perte exponentielle. Comme cette espérance est inconnue, en pratique on cherche à minimiser la version empirique, c'est-à-dire $\hat{L}_{\exp}(h_m(\mathbf{x}), y) = \frac{1}{n}\sum_i L_{\exp}(h(\mathbf{x}_i), y_i)$.

On a, en reprenant le calcul de la preuve précédente :

$$\begin{aligned}
\hat{L}_{\exp}(h_m(\mathbf{x}), y) &= \sum_i \exp\left(-y_i \sum_{j=1}^{m} \alpha_j r_j(\mathbf{x}_i)\right) \\
&= \sum_i \exp\left(-y_i \sum_{j=1}^{m-1} \alpha_j r_j(\mathbf{x}_i)\right) e^{-y_i \alpha_m r_m(\mathbf{x}_i)} \\
&= \sum_i n \prod_{j=1}^{m-1} z_j \; w_i^{m-1} e^{-y_i \alpha_m r_m(\mathbf{x}_i)} \\
&= C(e^{\alpha_m} - e^{-\alpha_m}) \sum_{i=1}^{n} w_i^{m-1} \mathbf{1}_{y_i \neq r_m(\mathbf{x}_i)} + Ce^{-\alpha_m} \quad \text{où } C = n \prod_{j=1}^{m-1} z_j
\end{aligned}$$

On souhaite maintenant minimiser cette fonction en fonction de r_m et de α_m. Or, minimiser $\hat{L}_{\exp}(h(\mathbf{x}), y)$ par rapport à r_m revient à minimiser $\sum_{i=1}^{n} w_i^{m-1} \mathbf{1}_{y_i \neq r_m(\mathbf{x}_i)}$, ce qui est précisément ce que l'on fait à chaque itération m, en construisant la règle de décision r_m qui minimise le taux d'erreur apparent. Pour minimiser $\hat{L}_{\exp}(h(\mathbf{x}), y)$ par rapport à α_m, on cherche à annuler la dérivée :

$$\begin{aligned}
\frac{d\hat{L}_{\exp}(h(\mathbf{x}), y)}{d\alpha_m} = 0 &\Leftrightarrow (e^{\alpha_m} + e^{-\alpha_m}) \sum_{i=1}^{n} w_i^{m-1} \mathbf{1}_{y_i \neq r_m(\mathbf{x}_i)} - e^{-\alpha_m} = 0 \\
&\Leftrightarrow (e^{\alpha_m} + e^{-\alpha_m}) \hat{e}(r_m) - e^{-\alpha_m} = 0 \\
&\Leftrightarrow e^{\alpha_m} \hat{e}(r_m) + (\hat{e}(r_m) - 1) e^{-\alpha_m} = 0
\end{aligned}$$

$$\Leftrightarrow \alpha_m = \frac{1}{2}\log\left(\frac{1-\hat{e}(r_m)}{\hat{e}(r_m)}\right)$$

L'algorithme AdaBoost fait partie de la classe des algorithmes dit "Forward Stagewise Additive", qui construisent séquentiellement ("Stagewise") un classifieur additif ("Additive"), c'est-à-dire qui ajoutent à chaque itération un nouveau classifieur, assorti d'un poids, sans modifier les poids des classifieurs obtenus aux itérations précédentes ("Forward"). En partant de $h_0(\mathbf{x}) = 0$, l'itération m s'écrit :

$$(\beta_m, b_m) \in \arg\min_{\beta, b} \sum_{i=1}^{n} L(h_{m-1}(\mathbf{x}_i) + \beta b(\mathbf{x}_i), y_i) \tag{4.9}$$

$$h_m(\mathbf{x}) = h_{m-1}(\mathbf{x}) + \beta_b b_b(\mathbf{x}). \tag{4.10}$$

L'algorithme AdaBoost vérifie $R_m(\mathbf{x}) = \text{signe}(h_m(\mathbf{x}))$, pour tout $m = 1, \ldots, M$, avec la fonction de perte exponentielle L_{exp} et avec une classe de prédicteurs b binaires.

4.2.2 Généralisations : gradient boosting et XGBoost

On a vu dans la section précédente que l'algorithme AdaBoost pouvait être vu comme un algorithme itératif de type "Forward Stagewise Additive", qui à chaque itération minimise une fonction de perte sur une classe de fonctions donnée [4]. Il est bien sûr possible de généraliser cette approche à d'autres fonctions de perte, et de considérer d'autres méthodes de boosting en les inscrivant dans un cadre plus général. On présente ici le cadre des arbres boostées (ou Boosted Trees).

En reprenant le cadre des modèles Forward Stagewise Additive, en utilisant pour la classe de prédicteurs l'ensemble des arbres à q nœuds, on obtient l'algorithme suivant :

Algorithme 4 : Boosting d'arbres de classification

1 **initialisation** : on définit $h_0(\mathbf{x}) = 0$

2 **pour** $m = 1, \ldots, M$ **faire**

3 on construit l'arbre T_m tel que

$$T_m \in \arg\min_{T \in \mathcal{T}} \sum_{i=1}^{n} L(h_{m-1}(\mathbf{x}_i) + T(\mathbf{x}_i), y_i)$$

où $\mathcal{T}$ est l'ensemble des arbres à q nœuds.

4 on définit $h_m(\mathbf{x}) = h_{m-1}(\mathbf{x}) + \lambda T_b(\mathbf{x})$

De façon générale, pour une fonction de perte quelconque, la maximisation décrite dans l'algorithme 4 n'est pas toujours explicite. Si la fonction de perte est dérivable, on peut utiliser la méthode

4. la classe de fonctions est déterminée par la méthode utilisée pour construire les classifieurs faibles : par exemple, avec un arbre de décision, il s'agit de l'ensemble des fonctions constantes par morceaux

de la plus forte pente pour minimiser itérativement la fonction de perte, en définissant h_m à partir de h_{m-1} et d'un pas η_m de la façon suivante :

$$h_m(\mathbf{x}_i) = h_{m-1}(\mathbf{x}_i) - \eta_m \left. \frac{\partial L(h(\mathbf{x}_i), y_i)}{\partial h(\mathbf{x}_i)} \right|_{h(\mathbf{x}_i)=h_{m-1}(\mathbf{x}_i)} . \tag{4.11}$$

Dès lors que l'on a choisi une fonction de perte différentiable, le calcul du gradient est immédiat, et on aimerait pouvoir définir T_b comme l'opposé du gradient de la fonction de perte. Il y a cependant deux problèmes avec cette approche. D'une part, T_b appartient, par définition, à l'ensemble des arbres à q nœuds, alors que le gradient de L n'est soumis à aucune contrainte. D'autre part, le gradient tel qu'il apparaît dans (4.11) n'est défini qu'aux points $\mathbf{x}_i$ de l'ensemble d'apprentissage. Or, l'objectif est de pouvoir généraliser les résultats à des points n'appartenant pas à cet ensemble.

La solution qui a alors été proposée consiste à construire à chaque itération T_m comme un arbre *de régression* qui va approcher le gradient de la fonction de perte. Le pas η_m sera en général choisi constant. C'est ce qu'on appelle le *learning rate*. Choisir de petites valeurs pour ce coefficient ralentit la convergence de l'algorithme, et en particulier, choisir un pas η plus petit que 1 diminue la contribution de chaque arbre à hauteur de η%. En pratique, on choisit souvent η petit (autour de 0.1 ou 0.3), car de fortes valeurs du taux d'apprentissage peuvent résulter en du sur-apprentissage. En pratique, la perte exponentielle ne se généralisant pas facilement au cas où $K > 2$, on utilisera dans la suite comme fonction de perte la déviance multinomiale, définie par :

$$L_{\text{multi}}(h(\mathbf{x}), y) = -\sum_{k=1}^{K} z_k \ln h_k(\mathbf{x}) + \ln \left(\sum_{k=1}^{K} e^{h_k(\mathbf{x})} \right) \tag{4.12}$$

où $Z = (z_1, \ldots, z_K)$ est le vecteur de taille K dont toutes les coordonnées sont nulles sauf celle correspondant au numéro de groupe, i.e. $y = k \Leftrightarrow z_k = 1, z_l = 0 \; \forall l \neq k$. En posant $t_k(\mathbf{x}) = \frac{e^{h_k(\mathbf{x})}}{\sum_{k=1}^{K} e^{h_k(\mathbf{x})}}$, comme dans le cadre de la régression logistique, on obtient $L_{\text{multi}}(t(\mathbf{x}), y) = -\sum_{k=1}^{K} z_k \ln t_k(\mathbf{x})$. On reconnaît ici l'opposé de la fonction de vraisemblance dans le cas logistique (voir (2.28)) [5].

Les algorithmes basés sur ce principe font partie de la classe des algorithmes de *gradient boosting*. Dans la lignée de ces algorithmes, Chen et Guestrin (2016) ont proposé en 2016 une librairie open-source appelée *XGBoost*, qui permet une implémentation efficace et flexible des algorithmes de gradient boosting. Le succès de cette librairie réside en partie dans sa rapidité d'exécution, XGBoost pouvant être jusqu'à 10 fois plus rapide que d'autres implémentations d'algorithmes de boosting, mais également dans sa grande flexibilité et sa capacité à s'adapter à un très grand nombre de problèmes. Les (très) bonnes performances de l'algorithme s'expliquent également grâce aux différentes améliorations et suggestions proposées par les auteurs, notamment au niveau de la construction des arbres de décision élémentaires qui sont construits à chaque itération. Une autre spécificité de l'algorithme est la prise en compte d'un terme de pénalité dans la fonction objective que l'on cherche à minimiser. Autrement dit, on cherche à chaque itération à minimiser une fonction objective qui s'écrit comme la somme

5. on notera au passage que la fonction de vraisemblance d'un modèle est un bon candidat pour la définition d'une fonction de perte, en la multipliant par exemple par -1

d'une fonction de perte (par exemple, exponentielle, logistique) et d'un terme de pénalité qui augmente lorsque la complexité des arbres de décision augmente.

Chapitre 5

Traitement des données déséquilibrées

Dans ce chapitre, on s'intéresse aux cas où il existe un déséquilibre important entre les effectifs des classes que l'on considère. On se concentrera sur le cas le plus fréquent de la classification binaire. Ce cas de figure apparaît très souvent en pratique, notamment pour la détection de fraudes, la détection de zones cancéreuses sur une IRM, la prédiction d'évènement indésirable rare, ... La plupart des méthodes de classification que nous avons vu jusqu'ici ne fonctionnent pas bien dans en cas de fort déséquilibre, car elles sont construites de façon à minimiser le taux de mauvais classement, auquel la classe minoritaire contribue très peu dans ce cas-là. Les algorithmes auront tendance à se concentrer sur la classe majoritaire. Or, dans la plupart de ces situations où les données sont déséquilibrées, c'est justement la classe minoritaire qui a le plus d'intérêt ... : on veut détecter les fraudeurs, la tumeur sur l'IRM, prédire les évènements indésirables rares, ...

Nous présentons dans ce chapitre trois approches possibles pour traiter ce type de données : 1) sur-échantillonner la classe minoritaire, 2) sous-échantillonner la classe majoritaire ou 3) introduire un coût de mauvais classement différent par classe. On note $\mathcal{D}$ le jeu de données initial, $\mathcal{D}_0$ les observations de la classe majoritaire notée C_0, et $\mathcal{D}_1$ les observations de la classe minoritaire C_1. On note également $n = \text{Card } \mathcal{D}$, $n_1 = \text{Card}\mathcal{D}_1$ et $n_0 = \text{Card}\mathcal{D}_0$, avec $n_1 < n_0$.

5.1 Critères de performance

Dans le cadre de la classification binaire, l'un des principaux indicateurs de performance utilisé est la courbe ROC (voir Figure 2.4), et l'aire sous la courbe associée. La courbe ROC s'utilise lorsque la règle de décision renvoie une probabilité d'appartenance aux classes, que l'on peut transformer en règle d'affectation en utilisant un seuil sur cette probabilité. C'est en faisant varier le seuil que l'on obtient les différents points qui constituent la courbe ROC.

Rappelons la définition de la courbe ROC, à l'aide de la matrice de confusion. Pour un seuil s donné, on obtient une certaine règle de décision qui affecte une partie des individus dans la classe C_0 et l'autre partie dans la classe C_1. Supposons que l'on souhaite en priorité prédire les individus de la classe C_1, que l'on appellera "positifs". La matrice de confusion s'écrit :

Table 5.1 – Matrice de confusion

	Observés : 1	Observés : 0
Prédits : 1	Vrais positifs (VP)	Faux positifs (FP)
Prédits : 0	Faux négatifs (FN)	Vrais négatifs (VN)

On rappelle que la sensibilité est le taux de vrais positifs, $Se = VP/(VP + FN) = VP/n_1$, et que la spécificité est le taux de vrais négatifs, $Sp = VN/(FP+VN)$. On a alors $1-Sp = FP/(FP+VN) = FP/n_0$. On définit aussi la *valeur prédictive positive*, par $VPP = VP/(VP + FP)$, et la *valeur prédictive négative*, par $VPN = VN/(VN + FN)$.

La précision est égale au taux d'observations bien classées : $Ac = (VP+VN)/(VP+VN+FP+FN) = (VP + VN)/n$. On voit déjà qu'en cas de fort déséquilibre entre les classes, i.e. lorsque $VP + FP \ll VN + FN$, il suffit d'avoir une valeur élevée de Sp pour obtenir une précision élevée. Dans ce cadre-là, la précision n'est donc pas un bon critère d'évaluation des performances d'une règle de décision.

La courbe ROC s'obtient en traçant la sensibilité en fonction de un moins la spécificité. Dans le cas de données déséquilibrées, la courbe ROC peut donner l'impression qu'une méthode a de bonnes performances globalement, alors qu'elle ne prédit pas bien la classe minoritaire. Elle ne permet pas, en effet, de répondre à la question suivante : parmi les individus que la règle de décision affecte à la classe C_1, combien sont effectivement bien prédits ? cette question est cruciale lorsqu'il s'agit de construire un outil prédictif. Comparons ainsi les deux matrices de confusion suivantes, correspondant à gauche au cas équilibré et à droite au cas déséquilibré :

	$Obs.:1$	$Obs.:0$
$Pred.:1$	500	150
$Pred.:0$	500	850

	$Obs.:1$	$Obs.:0$
$Pred.:1$	500	1500
$Pred.:0$	500	8500

Dans les deux cas (équilibré et déséquilibré), on a $Se = VP/(VP + FN) = 500/(500 + 500) = 0.5$ et $1 - Sp = FN/(FN + VN) = 0.15$. Ces deux matrices de confusion produisent donc le même point de coordonnées (0.15,0.5) sur leurs courbes ROC respectives. Or, dans le cas équilibré, $VPP = 500/(500 + 150) = 0.77$, i.e. 77% des individus prédits "1" ont été bien prédits, alors que dans le cas déséquilibré, $VPP = 0.25$ et donc, seulement 25% des individus prédits "1" ont été bien prédits.

Pour pallier à ce problème, dans le cas de données déséquilibrées il peut être utile de s'intéresser à la courbe appelée "Precision-Recall" [1] (PR) en anglais, qui s'obtient en traçant VPP en fonction de la sensibilité. En reprenant l'exemple précédent, les deux matrices de confusion produiront alors deux points distincts : (0.5,0.77) dans le cas équilibré et (0.5,0.25) dans le cas déséquilibré.

La courbe PR idéale passe par le point de coordonnées (1,1). Par ailleurs, la valeur prédictive positive minimale que l'on puisse obtenir, $VPP_{\min}$, par exemple à l'aide d'un classifieur naïf, est égale au nombre

1. Attention, "precision" en anglais est un faux ami : ce que l'on appelle *valeur prédictive positive* en français se traduit par *precision*, et ce que l'on appelle *précision* en français se traduit par *accuracy*

de positifs divisé par le nombre total d'observations (= on affecte tout le monde à la classe C_1 pour être sûr de bien classer les observations de cette classe). En pratique, si la courbe PR descend en dessous de ce point, on considère que le modèle n'est pas performant. Tout comme pour la courbe ROC, on peut également calculer l'aire sous la courbe PR. En pratique, le classifieur sera considéré comme meilleur qu'un classifieur naïf si l'aire sous la courbe est supérieure à VPP_{min}.

5.2 Sur-échantillonnage de la classe minoritaire

La première approche que l'on peut mettre en place pour réduire le déséquilibre entre les classes consiste à augmenter la taille de la classe minoritaire, en tirant uniformément avec remise des observations de cette classe, jusqu'à atteindre un effectif suffisant (pas forcément 50% des observations, mais suffisamment pour que le déséquilibre entre classes n'ait pas une grande influence sur les résultats).

Cette approche est facile à mettre en place et fonctionne bien dans certains cas, mais présente deux inconvénients : d'une part, on augmente la taille totale de la base de données, ce qui peut impliquer un surcoût computationnel, en particulier si la base de données est déjà grande, et d'autre part, ajouter des observations peut conduire à un sur-ajustement dans les zones où les mêmes observations ont été échantillonnées plusieurs fois.

On peut alors proposer des méthodes de ré-échantillonnage *informatives*, c'est-à-dire que l'on ne ré-échantillonne pas selon une loi uniforme sur les observations de la classe minoritaire. L'approche la plus utilisée dans le cadre du sur-échantillonnage de la classe minoritaire est celle de l'algorithme SMOTE (Synthetic Minority Oversampling TEchnique, Chawla *et al.* (2002)). Cette méthode propose de créer de nouvelles observations, aussi appelée *données synthétiques*, en utilisant la proximité entre les points existants de la classe minoritaire. Plus précisément, l'algorithme considère pour chaque observation de la classe minoritaire, l'ensemble de ses k-plus proches voisins, et génère de nouvelles observations le long des segments reliant chaque point à ses voisins (voir Figure 5.1). Dans la pratique, en fonction du nombre de voisins k choisis, on peut tirer au sort un nombre $k' < k$ de voisins. L'avantage de cette approche est qu'elle introduit plus de diversité dans les données de la classe minoritaire.

Dans la même lignée, l'algorithme ADASYN (ADAptive SYNthetic) a été proposé quelques années plus tard (Haibo He *et al.*, 2008). ADASYN est une variante de l'algorithme SMOTE qui génère d'autant plus d'observations synthétiques au voisinage d'un point de la classe minoritaire, que celui-ci est entouré de points de la classe majoritaire. Autrement dit, on renforce les zones de l'espace où la classe minoritaire est plus difficile à prédire.

Toutes ces techniques étant basées sur des approches d'échantillonnage, il est possible de les combiner avec des méthodes d'ensemble.

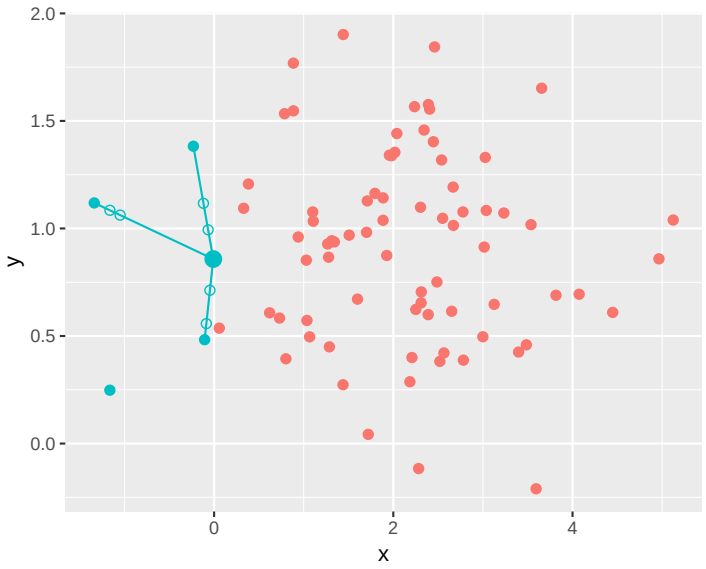


FIGURE 5.1 – Création de nouvelles observations synthétiques par l'algorithme SMOTE. Illustration basée sur les 3-plus proches voisins du point de coordonnées (-0.01,0.86). Les points générés par l'algorithme SMOTE sont représentés par des cercles.

5.3 Sous-échantillonnage de la classe majoritaire

L'alternative au sur-échantillonnage présenté dans la section précédente consiste à créer un jeu de données d'apprentissage dans lequel on sélectionne une partie des observations de la classe majoritaire. Cette sélection peut se faire de deux façons différentes : soit complètement au hasard, soit de façon informative. L'approche consistant à sélectionner au hasard un sous-ensemble de la classe majoritaire peut avoir comme conséquence de perdre une partie importante de l'information contenue dans le jeu de données. C'est pourquoi des méthodes alternatives ont été proposées.

Par exemple, l'algorithme *EasyEnsemble* propose de sous-échantillonner plusieurs sous-ensembles $\mathcal{D}_{0,i}$, $i = 1, \ldots, m$ à partir des observations de la classe majoritaire, puis de construire m règles de décision basées sur $\mathcal{D}_{0,i} \cup \mathcal{D}_1$. Puis, on agrège les résultats obtenus par chacune des règles de décision. L'aggrégation de règles obtenues sur plusieurs sous-échantillons de la classe majoritaire permet de diminuer le risque de perte d'information importante.

L'algorithme *BalanceCascade* est une méthode séquentielle qui utilise l'information contenue dans le jeu de données pour construire les sous échantillons successifs. Plus précisément, à l'itération m, on sous-échantillonne parmi $\mathcal{D}$ privé des éléments qui ont déjà été correctement classés par les classifieurs des itérations précédentes. Autrement dit, dès qu'une observation a été correctement classée à l'une des étapes de l'algorithme, elle ne sera plus échantillonnée par la suite.

D'autres approches basées sur les k-plus proches voisins ont également été proposées. Par exemple, la famille de méthodes *Near-Miss*, qui consistent à sélectionner des observations de $\mathcal{D}_0$ en fonction de leur proximité avec $\mathcal{D}_1$. Il existe trois variantes de cette méthode. La méthode *Near-Miss 1* sélectionne

les éléments de $\mathcal{D}_0$ dont la distance moyenne aux trois points de $\mathcal{D}_1$ les plus proches, est la plus faible. La méthode *Near-Miss 2* sélectionne les éléments de $\mathcal{D}_0$ dont la distance moyenne aux trois points de $\mathcal{D}_1$ les plus éloignés, est la plus faible. Enfin, la méthode *Near-Miss 3* sélectionne, pour chaque point de $\mathcal{D}_1$, un sous-ensemble de taille k parmi ses plus proches voisins appartenant à la classe majoritaire.

Une dernière approche consiste à appliquer des méthodes de classification non supervisée (voir Partie III) à l'ensemble des observations de la classe majoritaire, et de remplacer chaque cluster de points obtenu par son centre de gravité. Le nombre de clusters est choisi en fonction de la taille finale souhaitée pour le sous-échantillon de la classe majoritaire.

5.4 Coûts de mauvais classement

Si les méthodes présentées dans les sections précédentes tentent de réduire le déséquilibre entre classes, il est également possible de garder le jeu de données initial déséquilibré, mais de considérer des coûts de mauvais classement différents selon la classe. Ainsi, on pénalisera de façon plus importante les faux négatifs que les faux positifs, car ceux-ci correspondent à des individus positifs (appartenant à la classe C_1) qui ont été mal classés. On modifie ensuite la fonction de perte pour qu'elle tienne compte de ces coûts, et on peut utiliser les algorithmes "classiques" qui auront pour objectif de minimiser cette nouvelle fonction de perte.

Plus précisément, on définit la fonction de perte $L_{\text{cost}} : [0,1]^2 \mapsto \mathbb{R}^+$, telle que $L_{\text{cost}}(k,\ell)$ est le coût associé au classement d'un individu de la classe C_ℓ dans la classe C_k. On suppose alors que $L_{\text{cost}}(0,0) = L_{\text{cost}}(1,1) = 0$ (il n'y a pas de coût en cas de bon classement), et on construit L_{cost} de telle sorte que $L_{\text{cost}}(0,1) > L_{\text{cost}}(1,0)$. Autrement dit, il est plus coûteux d'affecter un individu de la classe C_1 à la classe C_0, que l'inverse.

On a vu (Proposition 2) qu'en pratique il suffisait d'optimiser la règle de décision individu par individu à l'aide du risque conditionnel. Avec cette nouvelle fonction de perte, ce risque s'écrit :

$$\mathbb{E}_{Y|X=x}(L_{\text{cost}}(r(X),Y) \mid X=\mathbf{x}) = L_{\text{cost}}(r(\mathbf{x}),0)t_0(\mathbf{x}) + L_{\text{cost}}(r(\mathbf{x}),1)t_1(\mathbf{x}).$$

Si $X = \mathbf{x}$, la règle de décision optimale r^* est définie par :

$$r^*(\mathbf{x}) = \begin{cases} 1 & \text{si } L_{\text{cost}}(1,0)t_0(\mathbf{x}) < L_{\text{cost}}(0,1)t_1(\mathbf{x}) \\ 0 & \text{si } L_{\text{cost}}(0,1)t_1(\mathbf{x}) < L_{\text{cost}}(1,0)t_0(\mathbf{x}) \end{cases} \tag{5.1}$$

N.B. Les résultats se généralisent facilement au cas multi-classes, avec

$$\mathbb{E}_{Y|X=x}(L_{\text{cost}}(r(X),Y) \mid X=\mathbf{x}) = \sum_{k=1}^{K} L_{\text{cost}}(r(\mathbf{x},k)t_k(\mathbf{x}).$$

Certains algorithmes peuvent facilement être utilisés en combinaison avec cette fonction de perte. Par exemple, les trois méthodes présentées au chapitre 5 et qui fournissent chacune une estimation de $t_k(\mathbf{x})$ que l'on peut ensuite plugger dans l'équation (5.1). Pour les arbres de décision (et toutes les méthodes basées sur des arbres de décision), il est possible d'utiliser des fonctions d'impureté *asymétriques*, qui ont pour effet de pénaliser différemment les erreurs de mauvais classement.

Conclusion

Les différentes méthodes présentées dans cette partie reposent toutes sur un certain nombre d'hypothèses. Certaines méthodes fonctionnent mieux, ou sont plus adaptées, dans certains cas. Le tableau 5.2 résume les différentes caractéristiques des méthodes de classification supervisée que nous avons abordées, et le tableau 5.3 propose différentes implémentations de ces algorithmes sous R et SAS (liste de fonctions non exhaustives, en particulier sous R).

Méthode	Type de variables	Remarques
AFD	quantitatives [2]	Pas de risque d'erreur associé au classement
Modèle de mélange Gaussien	quantitatives	Hypothèse forte sur la distribution des variables explicatives
Régression logistique	mixtes	Très utilisé dans le cas de 2 classes
k-plus proches voisins	mixtes [3]	Le temps de calcul peut devenir élevé si k est grand. La méthode fonctionne moins bien lorsque p est grand.
CART	mixtes	Besoin d'un grand nombre de données. Peu stable. Favorise les variables à grand nombre de modalités
Forêts aléatoires	mixtes	Bonnes performances Pas d'interpétation simple des résultats
AdaBoost	mixtes	Uniquement pour la classification binaire Tendance à l'over-fitting quand M augmente
XGBoost	mixtes	Très performant, dans le top 10 des algorithmes gagnants sur Kaggle [4] Difficile à interpréter, aspect boîte noire

TABLE 5.2 – Caractéristiques des différentes méthodes de classification supervisée abordées dans le cours

2. la méthode DISQUAL permet d'utiliser l'analyse factorielle discriminante à des variables qualitatives
3. dans ce cas, on transforme les variables qualitatives en variables binaires
4. plateforme de compétition en machine learning

Il existe d'autres méthodes de classification supervisée, que nous n'avons pas abordées dans le cours. Citons par exemple les réseaux de neurones, les machines à vecteurs de support (SVM), ... Nous renvoyons à l'excellent ouvrage de Hastie *et al.* (2001) (dont on trouvera une version pdf gratuite à l'adresse `https://web.stanford.edu/~hastie/Papers/ESLII.pdf`).

Méthode	R	SAS
AFD	package `MASS`, fonction `lda` package `ade4`, fonction `discrimin`	`PROC CANDISC;`
Mélange Gaussien • homoscédastique • hétéroscédastique	package `MASS` • fonction `lda` [5] • fonction `qda`	`PROC DISCRIM METHOD=NORMAL` • `POOL=YES;` • `POOL=NO;`
Régression logistique • binaire • polytomique	• fonction `glm` • package `nnet`, fonction `multinom`	`PROC LOGISTIC;`
k-plus proches voisins	package `class`, fonction `knn`	`PROC DISCRIM METHOD=NPAR K=;`
CART	package et fonction `rpart`	`PROC HPSPLIT;`
Forêts aléatoires	package et fonction `randomForest` package `ranger` (optimisé pour la grande dimension)	`PROC HPFOREST;`
AdaBoost	package `gbm`, ou `fastAdaboost`	`PROC TREEBOOST;` (seulement chez SAS Entreprise Miner)
XGBoost	package `xgboost`	

Table 5.3 – Implémentation des différentes méthodes sous R et SAS

5. dans le modèle de mélange Gaussien, l'estimation des paramètres se fait par maximum de vraisemblance, ce qui est équivalent à la méthode des moindres carrés utilisée dans le package `lda` pour l'AFD.

Troisième partie

Classification non supervisée

Chapitre 1

Introduction et définitions

La classification non supervisée a pour objectif de créer une typologie des individus d'une population, à partir d'un ensemble de variables collectées chez ces individus. On cherche à regrouper les individus en sous-groupes ou classes homogènes. On appelle également cet ensemble de méthodes "classification sans professeur", ou "clustering analysis" en anglais. Contrairement au cade de la classification supervisée de la première partie de ce document, en classification non supervisée on ne connaît pas la répartition *a priori* des individus dans les groupes. Souvent, on ne connaît pas non plus *le nombre de groupes* sous-jacents.

On distingue deux grands types de méthodes : d'une part les méthodes de partitionnement, qui ne font aucune hypothèse sur la distribution des données, et qui reposent plutôt sur la géométrie du nuage de points ; c'est le cas notamment de la méthode des k-means (chapitre 2) et de la classification ascendante hiérarchique (chapitre 3), et d'autre part les méthodes probabilistes, et en particulier les modèles de mélange (chapitre 5).

Remarque. *La page web suivante recense différents packages R de classification non supervisée :* https://cran.r-project.org/web/views/Cluster.html

Dans la suite de cette partie, on suppose que les variables explicatives sont quantitatives. Le cas où l'on dispose de variables qualitatives est abordée en section 1.2.

1.1 Définitions

Définition 21. ***Dissimilarité***. *Une fonction* $d : \Omega \times \Omega \mapsto \mathbb{R}_+$ *est une* dissimilarité *sur l'espace* Ω *si et seulement si elle vérifie :*

1. d *est symétrique :* $\forall (x, y) \in \Omega \times \Omega,\ d(x, y) = d(y, x)$
2. d *est positive :* $\forall (x, y) \in \Omega \times \Omega,\ d(x, y) \geq 0$
3. $\forall (x, y) \in \Omega \times \Omega,\ d(x, y) = 0 \Leftrightarrow x = y$ *(séparation)*

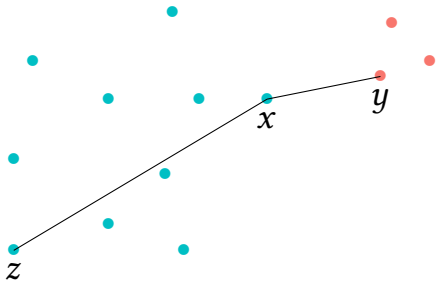


FIGURE 1.1 – Exemple de cas où la distance entre deux points appartenant à deux classes différentes peut être inférieure à la distance entre deux points appartenant à la même classe.

La notion de dissimilarité est proche de celle de distance, dont nous rappelons la définition ci-dessous. En particulier, toute distance est également une dissimilarité. La différence entre ces deux notions tient à l'existence d'une inégalité triangulaire dans le cas de la distance.

Définition 22. ***Distance.*** *Une fonction* $d : \Omega \times \Omega \mapsto \mathbb{R}_+$ *est une* distance *sur l'espace* Ω *si et seulement si elle vérifie :*

1. d *est symétrique :* $\forall (x, y) \in \Omega \times \Omega,\ d(x, y) = d(y, x)$
2. d *est positive :* $\forall (x, y) \in \Omega \times \Omega,\ d(x, y) \geq 0$
3. $\forall (x, y) \in \Omega \times \Omega,\ d(x, y) = 0 \Leftrightarrow x = y$ *(séparation)*
4. $\forall (x, y, z) \in \Omega^3, d(x, y) \leq d(x, z) + d(z, y)$ *(inégalité triangulaire)*

Une fois que l'on a muni notre espace d'une distance ou d'une dissimilarité, on va chercher à regrouper les individus de notre population en *classes* ou *groupes* homogènes. Mais qu'entend-on par *homogènes* ? Une première intuition conduit à souhaiter la propriété suivante : deux points appartenant à une même classe se ressemblent plus que deux points provenant de deux classes différentes. Autrement dit, si d est une mesure de dissimilarité cette propriété s'écrit : pour tout couple (x, y) appartenant à la même classe et pour tout couple (z, t) appartenant à deux classes différentes, $d(x, y) \leq d(z, t)$. Cependant cette propriété peut s'avérer trop forte en pratique, comme illustré sur la figure 1.1.

Les algorithmes que nous aborderons dans ce cours reposent sur la notion d'*inertie* d'un nuage de points.

Définition 23. *Soit* $x_1, \ldots, x_n$ *un ensemble de points de* $\mathbb{R}^p$. *L'*inertie *du nuage de points est définie par :*

$$I_T = \sum_{i=1}^{n} d^2(x_i, x_G), \tag{1.1}$$

où $x_G = \frac{1}{n} \sum_{i=1}^{n} x_i$ *est le centre de gravité du nuage de points et* d *la distance euclidienne.*

Supposons maintenant que le nuage de points soit composé de k classes notées $C_1, C_2, \ldots, C_k$ et formant une partition notée $\mathcal{P}$. L'inertie totale du nuage de points peut alors se décomposer de la façon

suivante :

$$I_T = \sum_{i=1}^{n} \|x_i - x_G\|^2 \tag{1.2}$$

$$= \sum_{i=1}^{k} \sum_{x_j \in C_i} \|x_i - x_G\|^2 \tag{1.3}$$

$$= \sum_{i=1}^{k} \sum_{x_j \in C_i} \|x_j - x_{G_i} + x_{G_i} - x_G\|^2 \tag{1.4}$$

$$= \sum_{i=1}^{k} \sum_{x_j \in C_i} \left(\|x_j - x_{G_i}\|^2 + \|x_{G_i} - x_G\|^2\right) \tag{1.5}$$

$$= \underbrace{\sum_{i=1}^{k} \sum_{x_j \in C_i} d^2(x_j, x_{G_i})}_{I_W(\mathcal{P})} + \underbrace{\sum_{i=1}^{k} n_i d^2(x_{G_i}, x_G)}_{I_B(\mathcal{P})}, \tag{1.6}$$

où n_i est le nombre d'éléments dans la classe C_i, et où le passage de la ligne (1.4) à la ligne (1.5) se fait grâce au théorème de Huygens, rappelé ci-dessous.

Théorème 3. *Théorème de Huygens Soient $X_1, \ldots, X_n$ des variables aléatoires i.i.d., et $\bar{X} := \frac{1}{n}\sum_{i=1}^{n} X_i$. Alors on a :*

$$\frac{1}{n}\sum_{i=1}^{n}(X_i - a)^2 = \frac{1}{n}\sum_{i=1}^{n}(X_i - \bar{X})^2 + (\bar{X} - a)^2 \tag{1.7}$$

Démonstration.

$$\begin{aligned}
\sum_{i=1}^{n}(X_i - a)^2 &= \sum_{i=1}^{n}(X_i - \bar{X} + \bar{X} - a)^2 \\
&= \sum_{i=1}^{n}(X_i - \bar{X})^2 + \sum_{i=1}^{n}(\bar{X} - a)^2 + 2\sum_{i=1}^{n}(X_i - \bar{X})(\bar{X} - a) \\
&= \sum_{i=1}^{n}(X_i - \bar{X})^2 + n(\bar{X} - a)^2 + 2(\bar{X} - a)\sum_{i=1}^{n}(X_i - \bar{X}) \\
&= \sum_{i=1}^{n}(X_i - \bar{X})^2 + n(\bar{X} - a)^2 + 2(\bar{X} - a)(n\bar{X} - n\bar{X}) \\
&= \sum_{i=1}^{n}(X_i - \bar{X})^2 + n(\bar{X} - a)^2.
\end{aligned}$$

□

Le premier terme de la décomposition, $I_W(\mathcal{P})$, est appelé *inertie intra-classes* et mesure la somme des distances au carré entre les points d'une classe et le centre de gravité de cette classe. Plus les classes sont homogènes, c'est-à-dire plus les points de chaque classes sont proches du centre de gravité de la classe, plus ce terme est petit. Le deuxième terme, $I_B(\mathcal{P})$, est appelé *inertie inter-classes* et mesure la somme des carrés des distances entre les centres de gravité de chaque classe et le centre de gravité du

nuage de points. Il mesure donc la distance entre les classes : plus les classes sont éloignées les unes des autres, plus ce terme sera élevé.

Plus les points de notre nuage sont répartis en des classes homogènes et bien distinctes les unes des autres, plus l'inertie intra-classes est faible, et donc, l'inertie totale étant constante, plus l'inertie inter-classes est élevée.

Une première idée consisterait donc à chercher la partition de notre ensemble de points correspond à une inertie intra-classes minimale, ou de façon équivalente, à une inertie inter-classes maximale. Cependant cette stratégie nous conduirait à choisir systématiquement la partition élémentaire à n classes, chacune formée d'un point, puisqu'elle correspond à une inertie intra-classes nulle. Comparer l'inertie intra- ou inter-classes de différentes partitions n'a de sens que si les partitions ont le même nombre de classes.

Une deuxième idée consiste alors, pour k fixé, à comparer toutes les partitions de taille k possibles, et à choisir la meilleure au sens de l'inertie intra- ou inter-classes. Cependant, pour ensemble de taille n, le nombre $p_{n,k}$ de partitions en k classes qu'il faudrait comparer s'avère vite prohibitif. En effet, ce nombre correspond au nombre de surjections d'un ensemble de n éléments dans un ensemble de k éléments, aussi appelé nombre de Stirling de seconde espèce, et on peut montrer qu'il vaut :

$$p_{n,k} = \frac{1}{k!}\sum_{i=0}^{k}(-1)^{i-1}\binom{n}{i}i^n. \tag{1.8}$$

Pour $n = 10$ et $k = 3$, ce nombre vaut déjà 9330 ...

En pratique, une exploration exhaustive de toutes les partitions du nuage de n points en k classes n'est donc pas faisable, et les algorithmes que nous présentons dans ce cours se contentent d'explorer un *sous-ensemble* de toutes les partitions possibles. La solution obtenue est alors approchée, puisque l'on n'explore pas toutes les partitions possibles, mais on peut espérer obtenir une bonne approximation lorsque le sous-ensemble de partitions visitées est suffisamment large. L'algorithme des k-means, décrit dans le chapitre 2, compare plusieurs partitions en k classes du nuage de points, où k est fixé préalablement. Il s'applique donc dans les cas où le nombre de classes est connu à l'avance. L'algorithme de classification ascendante hiérarchique (CAH), décrit au chapitre 3, compare quant à lui des suites de partitions emboîtées, aussi appelées *hiérarchies*, et s'applique en particulier lorsque le nombre de classes n'est pas connu à l'avance.

1.2 Variables qualitatives

Lorsque les variables explicatives sont toutes qualitatives binaires, plusieurs indices de dissimilarité ont été proposés dans la littérature. Par exemple, considèrons deux individus i et j et p variables binaires. Notons a le nombre de variables valant 1 pour i et j, b le nombre de variables valant 1 pour i et 0 pour j, c le nombre de variables valant 1 pour i et 0 pour j, et enfin, d le nombre de variables valant 0 pour i et j. On peut alors définir la dissimilarité entre i et j en partant de différents indices de similarité, par

exemple l'indice de Jaccard : $\frac{a}{a+b+c}$, l'indice de Ochiaï : $\frac{a}{\sqrt{(a+b)(a+c)}}$, ou encore l'indice de Russel et Rao : $\frac{a}{a+b+c+d}$.

Si les variables sont qualitatives à plusieurs modalités, on peut par exemple utiliser le tableau disjonctif complet, et définir la distance du chi-deux entre les lignes du tableau, définie par :

$$d^2_{\chi^2}(i,j) = \sum_{k=1}^{p} \frac{n}{n_{.k}} \left(\frac{x_{ik} - x_{jk}}{p} \right)^2,$$

où x_{ik} est l'observation de la k-ème variable pour l'individu i, $n_{.k}$ est la somme de la j-ème colonne du tableau disjoinctif complet. D'autres choix de distance sont possibles, comme la distance de Hamming, le coefficient de Hamman, ou la distance de Gower. Cette dernière est définie de la façon suivante :

$$d_{Gower}(i,j) = 1 - \frac{1}{p} \sum_{k=1}^{p} s(x_{ik}, x_{jk}),$$

où $s(x_{ik}, x_{jk})$ est la similarité de Gower entre les points x_{ik} et x_{jk}. Dans le cas où la k-ème variable est quantitative ou qualitative ordinale, on a $s(x_{ik}, x_{jk}) = 1 - |x_{ik} - x_{jk}|$, et dans le cas où la k-ème variable est qualitative non ordinale, on a $s(x_{ik}, x_{jk}) = 1$ si $x_{ik} = x_{jk}$, et $s(x_{ik}, x_{jk}) = 0$ sinon.

Lorsque les variables sont à la fois quantitatives et qualitatives, plusieurs approches sont possibles : (i) on peut transformer les variables quantitatives en variables qualitatives et utiliser l'approche décrite dans le paragraphe précédent, (ii) on peut réaliser une analyse factorielle de données mixtes, pour se ramener à des variables quantitatives, (iii) on peut utiliser une distance qui s'applique aussi bien à des variables qualitatives que quantitatives, par exemple la distance de Gower.

Chapitre 2

Méthodes des k-means ou centres mobiles

Dans ce chapitre, nous supposerons que le nuage de points est constitué de points de $\mathbb{R}^p$ muni d'une distance euclidienne.

2.1 Description de l'algorithme

L'algorithme des k-means est un algorithme de partitionnement qui fournit une partition en k classes d'un ensemble de n points. Le principe de l'algorithme est très simple, ce qui a contribué à sa large utilisation en pratique.

En partant d'un premier ensemble de k points notés $\mu_1, \ldots, \mu_k$ que l'on désigne comme centres des classes (souvent choisis au hasard), on regroupe dans la classe C_k tous les points qui sont plus proches de μ_k que des autres centres. Une fois cette première partition obtenue, on définit les nouveaux centres des classes en calculant les centres de gravités des classes précédemment obtenues. Enfin, on réitère le procédé en affectant les points à la classe dont le centre est le plus proche, puis en recalculant les centres des classes, jusqu'à convergence vers une partition stable, c'est-à-dire lorsqu'elle reste inchangée d'une itération à l'autre. Une version détaillée est donnée dans l'encadré 5.

2.2 Propriétés de l'algorithme

Pour une partition $\mathcal{P} = \{C_1, \ldots, C_k\}$ et un ensemble de points $\theta = \{\mu_1, \ldots, \mu_k\}$, on définit le critère suivant :

$$C(\mathcal{P}, \theta) = \sum_{i=1}^{k} \sum_{x_j \in C_i} \|x_j - \mu_i\|^2. \tag{2.3}$$

En particulier, si l'ensemble θ contient les centres de gravité des classes de la partition $\mathcal{P}$, on retombe sur la somme des inerties intra-classes.

Proposition 6. *À chaque itération de l'algorithme, le critère C est amélioré (c'est-à-dire diminué), et on a :*

$$C(\mathcal{P}_{m+1}, \theta_m) \leq C(\mathcal{P}_m, \theta_m) \tag{2.4}$$

Algorithme 5 : Algorithme des k-means

Soit un ensemble de points $\Omega = \{x_1, \ldots, x_n\}$.

initialisation : on choisit k points dans Ω (au hasard le plus souvent) que l'on définit comme les centres $\mu_1^0, \ldots, \mu_k^0$ des k classes

tant que *la partition n'est pas stable* **faire**

définir la j-ème classe :

$$C_j^m = \{x_i \mid \|x_i - \mu_j^m\| \leq \|x_i - \mu_{j'}^m\|, \forall j' = 1, \ldots, k\} \tag{2.1}$$

mettre à jour les centres des classes :

$$\mu_j^m = \frac{\sum_{x_i \in C_j^m} x_i}{|C_j^m|}, \tag{2.2}$$

où $|C_j^m|$ désigne le cardinal de l'ensemble C_j^m.

$m \leftarrow m + 1$

$$C(\mathcal{P}_{m+1}, \theta_{m+1}) \leq C(\mathcal{P}_{m+1}, \theta_m) \tag{2.5}$$

Démonstration. Montrons d'abord la première inégalité. À la fin de la première sous-étape de l'itération $m+1$ de l'algorithme, on forme une nouvelle partition en affectant les points aux classes dont les centres sont les plus proches. Par définition, en notant C_i^m l'ancienne classe de x_i et C_i^{m+1} sa nouvelle classe, on a, pour tout point x_i :

$$\|x_i - \mu_{C_i^{m+1}}^m\|^2 \leq \|x_i - \mu_{C_i^m}^m\|^2.$$

D'où, en sommant sur les classes et comme les centres restent inchangés, $C(\mathcal{P}_{m+1}, \theta_m) \leq C(\mathcal{P}_m, \theta_m)$.

Montrons maintenant la deuxième inégalité. La deuxième sous-étape de l'itération $m + 1$ de l'algorithme consiste à mettre à jour les centres des classes :

$$\mu_i^{m+1} = \frac{\sum_{x_i \in C_j^{m+1}} x_i}{|C_j^{m+1}|}. \tag{2.6}$$

Les centres $\mu_1^{m+1}, \ldots, \mu_k^{m+1}$ sont donc placés aux centres de gravité des classes $C_1^{m+1}, \ldots, C_k^{m+1}$. On a :

$$\begin{aligned} C(\mathcal{P}_{m+1}, \theta_m) &= \sum_{i=1}^{k} \sum_{x_j \in C_i^{m+1}} \|x_j - \mu_i^m\|^2 \\ &= \sum_{i=1}^{k} \sum_{x_j \in C_i^{m+1}} \|x_j - \mu_i^{m+1} + \mu_i^{m+1} - \mu_i^m\|^2 \\ &= \sum_{i=1}^{k} \sum_{x_j \in C_i^{m+1}} \left(\|x_j - \mu_i^{m+1}\|^2 + \|\mu_i^{m+1} - \mu_i^m\|^2 - 2\langle x_j - \mu_i^{m+1}, \mu_i^{m+1} - \mu_i^m \rangle\right) \end{aligned}$$

$$= \sum_{i=1}^{k} \sum_{x_j \in C_i^{m+1}} \|x_j - \mu_i^{m+1}\|^2 + \sum_{i=1}^{k} \sum_{x_j \in C_i^{m+1}} \|\mu_i^{m+1} - \mu_i^m\|^2$$

$$= C(\mathcal{P}_{m+1}, \theta_{m+1}) + \sum_{i=1}^{k} \sum_{x_j \in C_i^m} \|\mu_i^{m+1} - \mu_i^m\|^2,$$

en remarquant que $\sum_{x_j \in C_i^{m+1}} (x_j - \mu_i^{m+1}) = 0$, et donc que le produit scalaire est nul. Le deuxième terme étant positif ou nul par définition, on obtient bien

$$C(\mathcal{P}_{m+1}, \theta_{m+1}) \leq C(\mathcal{P}_{m+1}, \theta_m).$$

□

Corollaire 1. *L'algorithme des k-means minimise l'inertie intra-classes (et maximise l'inertie inter-classes).*

Démonstration. Immédiate en notant que $C(\mathcal{P}_m, \theta_m) = I_W(\mathcal{P}_m)$. □

Corollaire 2. *La suite numérique $C(\mathcal{P}_m, \theta_m)$ est stationnaire.*

Démonstration. D'après la proposition précédente, la suite $C(\mathcal{P}_m, \theta_m)$ est décroissante. Or, elle ne peut prendre qu'un nombre fini de valeurs car il n'y a qu'un nombre fini de centres possibles θ_m et de partitions $\mathcal{P}_m$. □

Proposition 7. *La suite $(\mathcal{P}_m, \theta_m)$ est stationnaire.*

Démonstration. Notons tout d'abord que la stationnarité de la suite $C(\mathcal{P}_m, \theta_m)$ ne suffit pas à montrer la stationnarité de $(\mathcal{P}_n, \theta_n)$, car il pourrait exister plusieurs combinaisons $(\mathcal{P}_m, \theta_m)$ conduisant à la même valeur de C.

Comme la suite $C(\mathcal{P}_m, \theta_m)$ est stationnaire, on a :

$$\exists N, \forall m > N, C(\mathcal{P}_m, \theta_m) = C(\mathcal{P}_{m+1}, \theta_{m+1}). \tag{2.7}$$

Donc, d'après la proposition 6, comme $C(\mathcal{P}_{m+1}, \theta_{m+1}) \leq C(\mathcal{P}_{m+1}, \theta_m) \leq C(\mathcal{P}_m, \theta_m)$, on a également $C(\mathcal{P}_{m+1}, \theta_{m+1}) = C(\mathcal{P}_{m+1}, \theta_m)$, où les θ_{m+1} sont les centres de gravité des classes de la partition $\mathcal{P}_{m+1}$. Or, pour une partition donnée, le critère C atteint son minimum en les centres de gravité des classes de la partition. Autrement dit, on a

$$C(\mathcal{P}_n, \theta_n) = \min_{\theta} C(\mathcal{P}_n, \theta).$$

De plus, d'après le théorème de Huygens, ce minimum est atteint de façon unique par les centres de gravité des classes de la partition. Donc, on a $\theta_{m+1} = \theta_m$. La partition étant définie de façon unique à partir des centres θ_m, on a donc également $\mathcal{P}_{m+1} = \mathcal{P}_m$.

□

Remarque. *Même si la suite $(\mathcal{P}_m, \theta_m)$ est stationnaire, ce qui implique la convergence de l'algorithme des k-means, cette convergence dépend de l'était initial de l'algorithme et n'est donc que locale. Dans la pratique, il peut être utile de lancer plusieurs fois l'algorithme avec des points de départ différents, et de ne retenir comme partition finale celle correspondant à l'inertie intra-classes minimale.*

Remarque. *On a vu plus haut que l'inertie intra-classes ne pouvait pas être utilisée pour comparer des partitions comportant un nombre différent de classes, car elle décroit lorsque le nombre de classes augmente. En pratique cependant, on peut chercher à identifier le nombre de classes optimal en identifiant un "coude" dans la courbe représentant l'inertie intra-classes en fonction du nombre de classes. Ce point sera abordé dans la section suivante.*

Remarque. *Utiliser la distance euclidienne implique que les classes obtenues sont* sphériques. *On peut généraliser en prenant la distance induite par le produit scalaire sur* $\mathbb{R}^n$ *:* $\langle x, y \rangle_M := x^t My$*, la distance euclidienne étant obtenue en prenant* $M = Id$*. On obtient alors des classes elliptiques. Cependant, certains cas ne seront pas bien traités par l'algorithme et pourront nécessiter une transformation préalable des données.*

2.3 Exemple

On illustre dans cette section le comportement de l'algorithme sur le célèbre jeu de données des iris de Fisher, contenant les longueurs et largeurs des sépales et des pétales de 150 fleurs appartenant à 3 espèces d'iris : *I. setosa*, *I. virginica* et *I. versicolor*. Sachant qu'il y a 3 espèces différentes dans le jeu de données, nous utilisons l'algorithme des k-means avec $k = 3$. Les itérations successives de l'algorithme sont illustrées sur la figure 2.1. On remarque que l'algorithme a très vite convergé, malgré un choix initial pour les centres des classes qui n'était pas optimal.

FIGURE 2.1 – Étapes successives de l'algorithme des k-means. Les points noirs correspondent aux centres des classes, ici choisis par l'utilisateur à l'initialisation, puis calculés par l'algorithme lors des étapes suivantes.

La figure 2.2 montre la diminution de l'inertie intra-classes minimale (i.e. lorsque l'algorithme a convergé), pour différentes valeurs du nombre de classes k. Le critère du "coude" nous conduit à choisir $k = 3$ classes, car l'inertie intra-classes diminue fortement entre 2 et 3 classes, mais ne diminue quasiment pas en passant à 4 classes, puis se stabilise doucement au-delà. Cependant ce critère est empirique et il peut être délicat à interpréter dans certains cas.

2.4 Nuées dynamiques

L'algorithme des nuées dynamiques est une généralisation de l'algorithme des k-means, où chaque classe est représentée non pas par son barycentre (qui peut être extérieur à la classe), mais par un *noyau*, qui peut être un sous-ensemble de points de la classe, un axe principal ou un plan principal, … Ce noyau s'avère parfois plus représentatif de la classe.

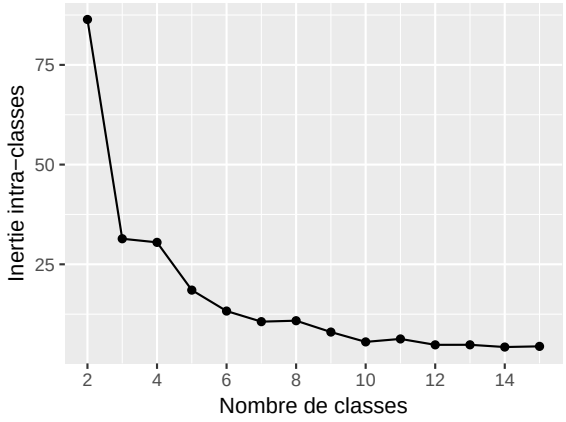


FIGURE 2.2 – Inertie intra-classes en fonction du nombre de classes.

Chapitre 3

Classification Ascendante Hiérarchique (CAH)

3.1 Description de l'algorithme

La classification ascendante hiérarchique (CAH) a pour objectif de construire une suite de partitions emboîtées de l'ensemble des points $\Omega = \{x_1, \ldots, x_n\}$. Plus précisément, et en partant de la partition initiale à n classes, l'étape m consiste à aggréger deux classes entre elles pour former une partition à $n-m$ classes, jusqu'à ce que l'on ait regroupé tous les points dans la même classe. On l'appelle *ascendante* car elle part d'une partition initiale en n singletons pour aboutir à une partition en 1 classe.

Cet algorithme nécessite de définir une notion de distance entre les classes. Plusieurs choix sont possibles et seront discutés dans la section suivante. Pour le moment, supposons qu'une telle distance (ou dissimilarité) a été définie, et notons là D. Une version détaillée de l'algorithme est présentée dans l'encadré 6.

Algorithme 6 : Classification ascendante hiérarchique

Soit un ensemble de points $\Omega = \{x_1, \ldots, x_n\}$.
initialisation : on définit pour $\mathcal{P}_0$ la partition à n points où chaque point constitue une classe à part entière
pour $m = 1, \ldots, n-1$ **faire**
    calculer les distances deux à deux entre les classes de la partition $\mathcal{P}_{m-1}$ à l'aide d'une distance D
    former la partition $\mathcal{P}_m$ en regroupant les classes se trouvant à distance minimale
    $m \leftarrow m + 1$

3.2 Critères d'aggrégation

On discute dans cette section différents choix possibles pour la distance ou la dissimilarité D permettant d'aggréger les classes entre elles. La plus couramment utilisée est la distance de Ward, qui est liée à l'inertie intra-classes.

Critère de Ward

Définition 24. *L'indice de dissimilarité de Ward entre deux parties A et B de centres de gravité respectifs x_A et x_B est défini par :*

$$D_W(A,B) = \frac{n_A n_B}{n_A + n_B}\|x_A - x_B\|^2, \tag{3.1}$$

où n_A est le cardinal de A.

Rappelons que pour un nombre de classes donné, la partition idéale est celle qui minimise l'inertie intra-classes, permettant ainsi de former des classes les plus homogènes possibles. La proposition suivante fait le lien entre le critère de Ward et l'inertie intra-classes.

Remarque. *On a besoin d'une distance euclidienne pour utiliser le critère d'aggrégation de Ward. En présence de variables qualitatives, la distance du chi-deux définie en section 1.2 étant une distance euclidienne, on peut utiliser le critère de Ward si on utilise cette distance.*

Proposition 8. *L'utilisation de l'indice de Ward permet de regrouper à chaque étape de l'algorithme de CAH les classes dont la fusion permet le plus faible gain d'inertie intra-classes, ou de façon équivalente, la plus faible perte d'inertie inter-classes.*

Démonstration. Soit Δ_m la perte d'inertie inter-classes entre les itérations $m-1$ et m de l'algorithme. Notons C_i et C_j les deux classes ayant été regroupées lors de l'itération m. Le reste des classes restant inchangé, la différence d'inertie inter-classes entre les deux partitions provient uniquement de la différence d'inertie inter-classes entre les deux classes fusionnées et les deux classes séparées. Notons x_{C_i} le barycentre de la classe C_i, x_G le barycentre du nuage de points et x_C le barycentre de la classe fusionnée. On a alors :

$$\Delta_m = n_i d^2(x_{C_j}, x_G) + n_j d^2(x_{C_j}, x_G) - (n_i + n_j) d^2(x_C, x_G).$$

Or, par définition du barycentre, on a :

$$x_C = \frac{n_i}{n_i + n_j} x_{C_i} + \frac{n_j}{n_i + n_j} x_{C_j}. \tag{3.2}$$

On a alors :

$$\begin{aligned}
\Delta_m &= n_i\|x_{C_i} - x_G\|^2 + n_j\|x_{C_j} - x_G\|^2 - (n_i + n_j)\|x_C - x_G\|^2 \\
&= n_i\|x_{C_i} - x_G\|^2 + n_j\|x_{C_j} - x_G\|^2 - (n_i + n_j)\left\|\frac{n_i}{n_i + n_j} x_{C_i} + \frac{n_j}{n_i + n_j} x_{C_i} - x_G\right\|^2
\end{aligned}$$

$$\begin{aligned}\Delta_m &= n_i\|x_{C_i} - x_G\|^2 + n_j\|x_{C_j} - x_G\|^2 - \frac{n_i^2}{n_i+n_j}\|x_{C_i} - x_G\|^2 - \frac{n_j^2}{n_i+n_j}\|x_{C_j} - x_G\|^2 \\ &\quad - \frac{2n_in_j}{n_i+n_j}\langle x_{C_i} - x_G, x_{C_j} - x_G\rangle \\ &= \frac{n_in_j}{n_i+n_j}\|x_{C_i} - x_G\|^2 + \frac{n_in_j}{n_i+n_j}\|x_{C_j} - x_G\|^2 - \frac{2n_in_j}{n_i+n_j}\langle x_{C_i} - x_G, x_{C_j} - x_G\rangle \\ &= \frac{n_in_j}{n_i+n_j}\|x_{C_i} - x_{C_j}\|^2.\end{aligned}$$

La perte minimale d'inertie inter-classes est donc bien obtenue lorsque l'on minimise le critère de Ward.

□

Critère du saut minimum (ou "single linkage")

$$D_{min}(A,B) = \min\{d(x,y) \mid x \in A, y \in B\} \tag{3.3}$$

Critère du saut maximum (ou "complete linkage")

$$D_{max}(A,B) = \max\{d(x,y) \mid x \in A, y \in B\} \tag{3.4}$$

Critère de la distance moyenne (ou "average linkage")

$$D_{moy}(A,B) = \frac{1}{n_An_B}\sum_{x\in A}\sum_{y\in B} d(x,y) \tag{3.5}$$

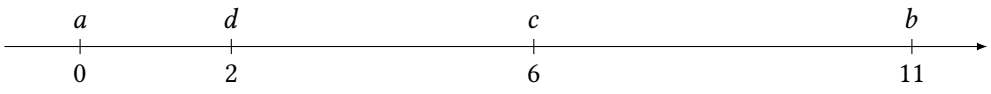
3.3 Dendrogramme

Une façon usuelle de représenter les étapes successives de la CAH est celle du *dendrogramme*. Il s'agit d'une arborescence : les feuilles de l'arbre se trouvent en bas, et représentent les individus de notre nuage de points. Puis, chaque niveau de l'arborescence correspond au regroupement de deux classes du niveau précédent. On représente ainsi les classes fusionnées à l'aide d'une branche les reliant, et la hauteur de la branche est proportionnelle à la distance entre ces classes. Au sommet de l'arbre, on retrouve la classe contenant tous les individus. Un exemple d'application de a CAH et du dendrogramme est donné dans la section suivante.

Remarque. *Attention à ne pas confondre le dendrogramme avec un arbre de décision, comme ceux que l'on peut obtenir avec les méthodes de classification supervisée abordées dans le chapitre 3 de la partie II.*

3.4 Exemple

On considère les points alignés suivants :



On choisit comme mesure de dissimilarité entre points, la distance euclidienne, et on va effectuer une CAH suivant les 4 critères d'aggrégation présentés, D_W, $D_{\min}$, $D_{\max}$ et D_{moy}.

Critère de Ward

On commence par calculer la matrice des distances initiales entre les points, calculées selon le critère de Ward pour des classes réduites à un point. La distance minimale est celle entre les points a et d, que l'on fusionne donc en une classe. On calcule ensuite la distance entre la classe fusionnée $\{a, d\}$ et les points c et b à l'aide de la formule (3.1). Pour cela, on utilise le barycentre de la classe $\{a, d\}$, $x_{\{a,d\}} = 1$.

$$D_W(\{a,d\},c) = \frac{2\times 1}{2+1}\, d^2(1,6) = \frac{50}{3}$$
$$D_W(\{a,d\},b) = \frac{2\times 1}{2+1}\, d^2(1,11) = \frac{200}{3}.$$

	a	b	c	d
a	0			
b	60.5	0		
c	18	12.5	0	
d	***2***	40.5	8	0

	$\{a,d\}$	b	c
$\{a,d\}$	0		
b	66.67	0	
c	16.67	***12.5***	0

	$\{a,d\}$	$\{b,c\}$
$\{a,d\}$	0	
$\{b,c\}$	56.25	0

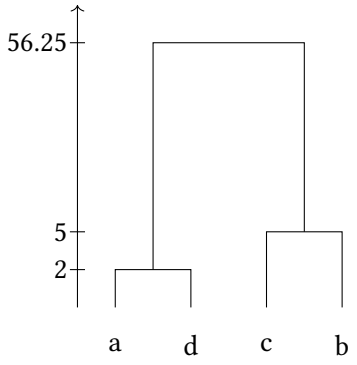


FIGURE 3.1 – Dendrogramme obtenu par le critère de Ward

Critère du saut minimum

On commence par la matrice des distances euclidiennes entre les points. Les points a et d sont les plus proches, on les regroupe donc puis on calcule la valeur du critère $D_{\min}$ entre les 3 classes ainsi obtenues : $\{a, d\}$, b et c.

	a	b	c	d
a	0			
b	11	0		
c	6	5	0	
d	***2***	9	4	0

	$\{a,d\}$	b	c
$\{a,d\}$	0		
b	9	0	
c	***4***	5	0

	$\{a,d,c\}$	b
$\{a,d,c\}$	0	
b	5	0

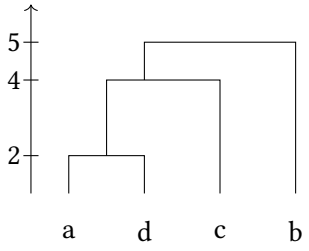


Figure 3.2 – Dendrogramme obtenu par le critère du saut minimum

Critère du saut maximum

	a	b	c	d
a	0			
b	11	0		
c	6	5	0	
d	***2***	9	4	0

	$\{a,d\}$	b	c
$\{a,d\}$	0		
b	11	0	
c	6	***5***	0

	$\{a,d\}$	$\{b,c\}$
$\{a,d\}$	0	
$\{b,c\}$	11	0

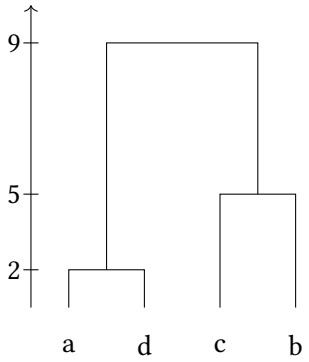


Figure 3.3 – Dendrogramme obtenu par le critère du saut maximum

Critère de la distance moyenne

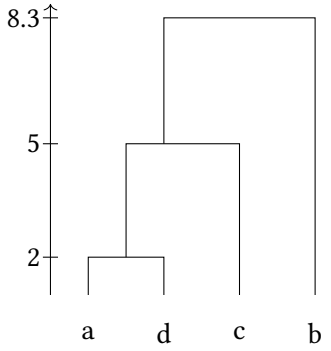
La première étape nous conduit à regrouper les points a et d. par contre, deux choix sont possibles lors de la deuxième itération : soit regrouper $\{a, d\}$ et b, soit regrouper $\{a, d\}$ et c.

	a	b	c	d
a	0			
b	11	0		
c	6	5	0	
d	***2***	9	4	0

	$\{a, d\}$	b	c
$\{a, d\}$	0		
b	10	0	
c	***5***	***5***	0

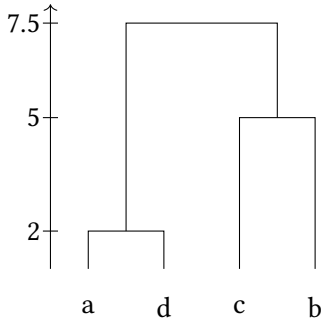
— choix 1 : regroupement de $\{a, d\}$ et b

	$\{a, d, c\}$	b
$\{a, d, c\}$	0	
b	8.33	0



— choix 2 : regroupement $\{a, d\}$ et c.

	$\{a, d\}$	$\{b, c\}$
$\{a, d\}$	0	
$\{b, c\}$	7.5	0



N.B. L'algorithme de CAH peut être difficile à mettre en œuvre lorsque le nombre d'observations est trop important, en particulier car cela nécessite le calcul (et le stockage) de la matrice de distances de taille $n \times n$. Dans ce cas, il peut être intéressant de faire une première étape à l'aide de l'algorithme des k-means, en fixant un nombre assez élevé de classes. Puis, à partir des centres des classes obtenues, on effectue une CAH afin de déterminer un nombre optimal de classes. Cette approche est une façon de réduire la dimension de l'espace des observations. Enfin, on peut relancer un algorithme des k-means sur l'échantillon initial, en utilisant le nombre de classes trouvé par l'algorithme de CAH.

Chapitre 4

Méthodes de partitionnement spectral

Les algorithmes présentés dans les sections précédentes fonctionnent bien surtout lorsque les données sont réparties dans des classes convexes. Les méthodes de partitionnement spectrales, quant à elles, fonctionnent également très bien lorsque les classes ne sont pas forcément convexes. Les algorithmes que nous allons voir dans ce chapitre s'appuyent tous sur une représentation du nuage de points sous forme de graphe, et reformulent la question de la classification en une question de partitionnement de graphe. Ils diffèrent ensuite sur le mode de construction du graphe, et la méthode de partitionnement.

4.1 Notations et définitions

On rappelle que l'on dispose d'un ensemble de points $\{x_1, \dots, x_n\}$, et d'une mesure de similarité entre chaque paire (x_i, x_j). À partir de ces données, on va construire un graphe pondéré et non orienté $G = (V, E)$, où V est l'ensemble des sommets du graphe, c'est-à-dire ici $V = \{x_1, \dots, x_n\}$, et E l'ensemble des arêtes du graphe. On appelle G le *graphe de similarité*.

Matrice d'adjacence

Le fait que G soit pondéré signifie qu'à chaque arête entre deux sommets x_i et x_j, on associe un poids positif $w_{ij} \geq 0$. On note alors $W = (w_{ij})_{i=1,\dots,n,\, j=1,\dots,n}$ la *matrice d'adjacence* du graphe. En particulier, un poids nul entre deux sommets signifie qu'ils ne sont pas reliés par une arête. La matrice W est symétrique, car le graphe G est non orienté, ce qui implique en particulier que $w_{ij} = w_{ji}$.

Degré d'un sommet

Dans un graphe non pondéré, le degré d'un sommet v est simplement le nombre d'arêtes dont l'une des extrémités est égale à v. Dans le cas d'un graphe pondéré, on définit le degré du sommet x_i par :

$$d_i = \sum_{j=1}^{n} w_{ij}.$$

On note alors $D = \text{diag}\,\{d_1, \dots, d_n\}$ la matrice diagonale contenant les degrés des sommets sur sa diagonale.

Graphe connexe, composante connexe

Un sous-ensemble $A \subset V$ est dit *connexe* s'il est possible de relier n'importe quelle paire de sommets de A à l'aide d'un chemin qui ne passe que par des sommets de A. Un sous-ensemble $A \subset V$ est une *composante connexe* du graphe G s'il est connexe, et s'il n'existe aucune connexion entre A et son complémentaire $V \setminus A$.

Enfin, la suite $A_1, \ldots, A_k$ forme une *partition* de G si et seulement si quelques soient i et j, $A_i \cap A_j = \varnothing$ et $A_1 \cup \cdots \cup A_k = V$.

4.2 Construction du graphe de similarité

4.2.1 Mesure de similarité

Il existe plusieurs choix pour la mesure de similarité entre points. Certains ont été évoqués dans l'introduction de la partie III. D'autres sont plus spécifiques aux algorithmes de partitionnement spectral. Citons notamment :

— la fonction de similarité Gaussienne

$$s_{ij} = \exp\left(-\frac{d^2(x_i, x_j)}{2\sigma^2}\right),$$

où $d^2(x_i, x_j)$ est une distance et σ un paramètre d'échelle à définir. Plus σ est grand, plus la similarité entre points diminue vite avec leur distance

— la fonction de similarité cosinus

$$s_{ij} = \frac{\langle x_i, x_j \rangle}{||x_i||\,||x_j||}.$$

4.2.2 Construction du graphe

Une fois que l'on dispose d'une mesure de similarité, il existe plusieurs façons de construire le graphe de similarité.

Graphe entièrement connecté

On relie entre eux tous les points dont la mesure de similarité est positive, et on pondère l'arête reliant x_i et x_j par la mesure de similarité s_{ij} entre ces points. En général, on utilise cette construction lorsque la mesure de similarité choisie reflète elle-même la similarité *locale* entre les points, comme par exemple avec la fonction de similarité Gaussienne.

Graphe des k-plus proches voisins

On ne relie un sommet x_i qu'avec ses k plus proches voisins. La notion de k-voisinage n'étant pas symétrique (un point x_i peut appartenir aux k plus proches voisins de x_j, sans que x_j appartienne aux k plus proches voisins de x_i), cette définition conduit à la construction d'un graphe orienté. On obtient un graphe non orienté en procédant de l'une des deux façons suivantes :

— on relie x_i et x_j si x_i appartient aux k plus proches voisins de x_j, ou si x_j appartient aux k plus proches voisins de x_i. On appelle ce graphe le graphe des k-plus proches voisins.

— on relie x_i et x_j s'ils sont appartiennent tous les deux aux k plus proches voisins de l'autre point. On appelle ce graphe le graphe des k-plus proches voisins mutuels.

Graphe du ε-voisinage

On relie entre eux tous les points dont la distance mutuelle est inférieure au seuil ε. On obtient alors un graphe *binaire* (les poids sont égaux à 1 si les sommets sont connectés, et à 0 sinon).

4.3 Partitionnement du graphe

4.3.1 Laplacien du graphe

Les méthodes de partitionnement spectrales s'appuyent sur le spectre de la matrice de similarité du graphe, et plus précisément sur les valeurs et vecteurs propres de la matrice Laplacienne du graphe G. Il existe plusieurs définitions, dans la littérature, pour la matrice Laplacienne d'un graphe G pondéré, parmi lesquelles :

— Laplacien *non normalisé* : $L = D - W$,

— Laplacien *normalisé* : $\tilde{L} = I - D^{-1}W$, ou la version symétrique $\tilde{L}_{\text{sym}} = I - D^{-1/2}WD^{-1/2}$.

Propriétés de la matrice L (Laplacien non normalisé)

Proposition 9. *La matrice L vérifie les propriétés suivantes :*

1. $\forall q \in \mathbb{R}^n$,
$$q^t Lq = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}(q_i - q_j)^2$$
2. *L est symétrique et semi-définie positive.*
3. *la plus petite valeur propre de L est 0, et le vecteur constant dont toutes les coordonnées sont égales à 1 est un vecteur propre associé à la valeur propre 0.*
4. *L a n valeur propres positives, avec* $0 = \lambda_1 \leq \cdots \leq \lambda_n$.

Démonstration. 1. Par définition de L et de d_i, on a :

$$\begin{aligned} q^t Lq &= q^t Dq - q^t Wq \\ &= \sum_{i=1}^{n} d_i q_i^2 - \sum_{i=1}^{n}\sum_{j=1}^{n} q_i q_j w_{ij} \\ &= \frac{1}{2}\left(\sum_{i=1}^{n} d_i q_i^2 + \sum_{j=1}^{n} d_j q_j^2 - 2\sum_{i=1}^{n}\sum_{j=1}^{n} q_i q_j w_{ij}\right) \end{aligned}$$

$$q^t Lq = \frac{1}{2}\left(\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}q_i^2 + \sum_{j=1}^{n}\sum_{j=1}^{n} w_{ji}q_j^2 - 2\sum_{i=1}^{n}\sum_{j=1}^{n} q_i q_j w_{ij}\right)$$
$$= \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}(q_i - q_j)^2.$$

2. D et W étant symétrique, L l'est aussi. Par le point 1., on en déduit que L est également définie positive car les w_{ij} sont positifs ou nuls.
3. Trivial, par définition de L et de d_i.
4. C'est une conséquence des points 2 et 3.

□

La proposition suivante fait le lien entre le nombre de composantes connexes d'un graphe, et le spectre de la matrice L.

Proposition 10. *Soit G un graphe non orienté pondéré tel que $w_{ij} \geq 0, i = 1, \ldots, n, j = 1, \ldots, n$. La matrice L admet la valeur propre 0 avec une multiplicité k si et seulement si G a k composantes connexes.*

Démonstration. Considérons tout d'abord le cas $k = 1$, et montrons que L admet la valeur propre 0 avec multiplicité 1 ssi G est connexe.

Supposons G connexe, et montrons que la multiplicité de la valeur propre 0 est $k = 1$. Soit v un vecteur propre associé à la valeur propre 0. D'après la proposition précédente,

$$Lv = 0 \Leftrightarrow v^t Lv = 0$$
$$\Leftrightarrow \sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}(v_i - v_j)^2 = 0.$$

Or les poids w_{ij} étant positifs ou nuls, la somme ne peut être nulle que si tous les termes sont nuls. Autrement dit, on doit avoir $w_{ij}(v_i - v_j)^2 = 0$ pour tout couple (i, j). Si les deux sommets x_i et x_j sont connectés, c'est-à-dire si $w_{ij} > 0$, alors on doit avoir $v_i = v_j$. Le vecteur propre v a donc des coordonnées constantes pour les sommets connectés, c'est-à-dire des coordonnées constantes sur toute composante connexe. Or on a supposé que le graphe était connecté, c'est-à-dire que tous les sommets étaient reliés entre eux. Donc on a nécessairement que toutes les coordonnées du vecteur v sont égales. Autrement dit, $v = \mathbf{1}$ est le seul vecteur propre associé à la valeur propre 0. Le sous-espace propre associé à ce vecteur propre est donc de dimension 1, et la multiplicité de la valeur propre associée est donc égale à 1.

Montrons maintenant que si la multiplicité de la valeur propre 0 est égale à 1, alors G est connexe. On va montrer la contraposée : supposons que G n'est pas connexe, et montrons qu'alors la valeur propre 0 a une multiplicité strictement supérieure à 1. G n'étant pas connexe, il peut se décomposer en k composantes connexes $A_1, \ldots, A_k$. Alors, sans perte de généralité, on peut supposer que les sommets sont rangés dans l'ordre de la composante connexe à laquelle ils appartiennent. Comme le poids w_{ij} est non nul si et seulement si les sommets x_i et x_j sont connectés, la matrice W a alors une structure

diagonale par blocs. Il en est alors de même pour la matrice $L = \text{diag}\{L_1, \ldots, L_k\}$, où la taille de chaque bloc est égale au nombre de sommets dans chaque composante. la matrice L_i est alors la matrice Laplacienne de la composante connexe A_i. Donc, d'après la proposition 9, chaque matrice L_i admet la valeur propre 0. La matrice L étant diagonale, son spectre est égale à l'union des spectres des sous-matrices $L_1, \ldots, L_k$, et donc L admet la valeur propre 0 avec un ordre de multiplicité au moins égal à k.

Maintenant, supposons $k > 1$ quelconque. Le graphe G admet k composantes connexes si et seulement si sa matrice laplacienne a une structure diagonale par blocs, $L = \text{diag}\{L_1, \ldots, L_k\}$. D'après ce qui précède, chaque matrice L_i admet la valeur propre 0 avec une multiplicité égale à 1, donc la matrice L admet la valeur propre 0 avec multiplicité exactement égale à k. Réciproquement, si la multiplicité de la valeur propre 0 est égale exactement à k, cela signifie que la matrice L a une structure diagonale par blocs avec exactement k blocs. Ces k blocs correspondent alors à k composantes connexes de G.

□

Propriétés de la matrice $\tilde{L}$ (Laplacien normalisé)

Proposition 11. *1. $\forall q \in \mathbb{R}^n$,*

$$q^t \tilde{L}_{sym} q = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} \left(\frac{q_i}{\sqrt{d_i}} - \frac{q_j}{\sqrt{d_j}} \right)^2$$

2. *$\tilde{L}$ et $\tilde{L}_{sym}$ sont semi-définie positives*
3. *$\tilde{L}$ et $\tilde{L}_{sym}$ admettent toutes les deux la valeur propre 0, le vecteur $\mathbf{1}$ est un vecteur propre associé à la valeur propre 0 pour $\tilde{L}$, et le vecteur propre $D^{1/2}\mathbf{1}$ est un vecteur propre associé à la valeur propre 0 pour $\tilde{L}_{sym}$.*
4. *$\tilde{L}$ et $\tilde{L}_{sym}$ ont les mêmes valeurs propres, et si v est un vecteur propre de $\tilde{L}$, alors $w = D^{1/2}v$ est un vecteur propre de $\tilde{L}_{sym}$ pour la même valeur propre.*

Démonstration. Laissée en exercice. □

La proposition suivante permet de faire le lien, comme dans le cas de la matrice Laplacienne non normalisée, entre la multiplicité de la valeur propre 0 et le nombre de composantes connexes dans le graphe G.

Proposition 12. *Soit G un graphe non orienté pondéré tel que $w_{ij} \geq 0, i = 1, \ldots, n, j = 1, \ldots, n$. Alors la multiplicité k de la valeur propre 0 pour les matrices $\tilde{L}$ et $\tilde{L}_{sym}$ est égale au nombre de composantes connexes dans le graphe.*

Démonstration. Similaire au cas de la matrice non normalisée, et laissée en exercice. □

4.3.2 Algorithmes de partitionnement

Dans la pratique, il est rare que le graphe soit composé d'exactement k composantes connexes, et l'objectif des algorithmes de partitionnement est de "couper" des arêtes afin de créer des composantes connexes. Si le graphe contient bien k composantes connexes, la matrice L est diagonale par blocs, avec k blocs. La matrice contenant les vecteurs propres est alors elle-même diagonale par blocs, d'après la proposition 10. Il s'agit alors d'identifier ces blocs, en appliquant par exemple un algorithme des k-means à cette matrice. C'est le principe sur lequel sont basés les algorithmes de partitionnement spectral présentés dans cette section (voir Algorithme 7).

Algorithme 7 : Partitionnement spectral non normalisé

1 Soit un ensemble de points $\Omega = \{x_1, \ldots, x_n\}$ et une matrice de similarité associée S.
2 **Initialisation** : Construire un graphe de similarité de matrice d'adjacence W
3 Calculer la matrice Laplacienne L, ou $\tilde{L}$.
4 Calculer les vecteurs propres associés aux k plus petites valeurs propres de la matrice Laplacienne, et définir la matrice U contenant les k vecteurs propres en colonnes, i.e.

$$U = (v_1 \mid v_2 \mid \cdots \mid v_k)\,.$$

5 Réaliser une classification des lignes $u_1, \ldots, u_n$ de la matrice U à l'aide d'un algorithme des k-means, pour obtenir les classes $C_1, \ldots, C_k$.
6 Définir les classes $A_1, \ldots, A_k$ par :

$$A_i = \{j \mid u_j \in C_i\}.$$

Algorithme 8 : Partitionnement spectral normalisé

1 Soit un ensemble de points $\Omega = \{x_1, \ldots, x_n\}$ et une matrice de similarité associée S.
2 **Initialisation** : Construire un graphe de similarité de matrice d'adjacence W
3 Calculer la matrice Laplacienne $\tilde{L}_{\text{sym}}$.
4 Calculer les vecteurs propres associés aux k plus petites valeurs propres de la matrice Laplacienne, et définir la matrice U contenant les k vecteurs propres en colonnes
5 Normaliser la matrice U de sorte que la norme par lignes soit égale à 1 et créer la matrice V telle que l'élèment v_{ij} soit défini par $v_{ij} = u_{ij} / \left(\sum_{j=1}^{k} u_{ij}^2\right)^{1/2}$
6 Réaliser une classification des lignes $v_1, \ldots, v_n$ de la matrice V à l'aide d'un algorithme des k-means, pour obtenir les classes $C_1, \ldots, C_k$.
7 Définir les classes $A_1, \ldots, A_k$ par :

$$A_i = \{j \mid v_j \in C_i\}.$$

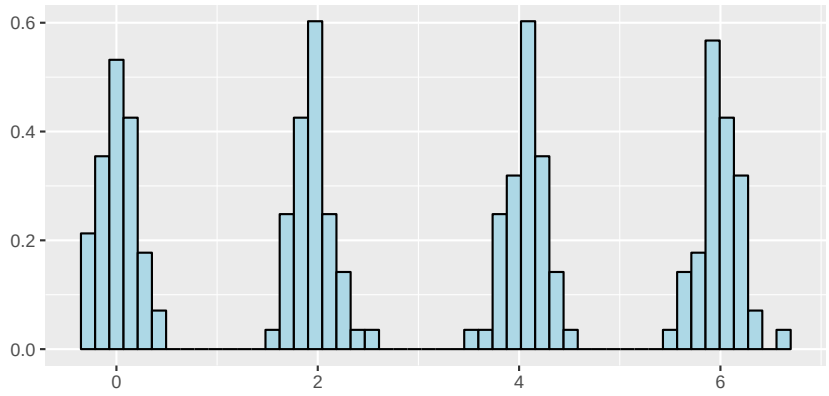


FIGURE 4.1 – Histogramme du jeu de données simulées consistant en un mélange de 4 lois normales univariées.

4.4 Exemple

Illustrons la méthode de partitionnement spectral sur un jeu de données simulées correspondant à un mélange de 4 lois Gaussiennes univariées de moyennes respectives 0, 2, 4 et 6 et d'écart-type 0.2 (voir figure 4.1). En utilisant la fonction de similarité Gaussienne de paramètre σ, on construit la matrice de similarité W et le graphe entièrement connecté correspondant, puis la matrice Laplacienne L.

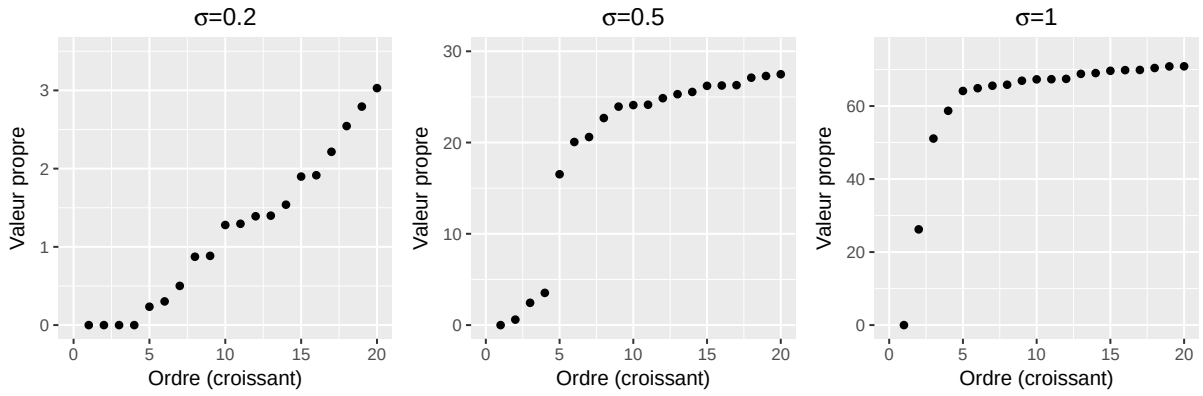
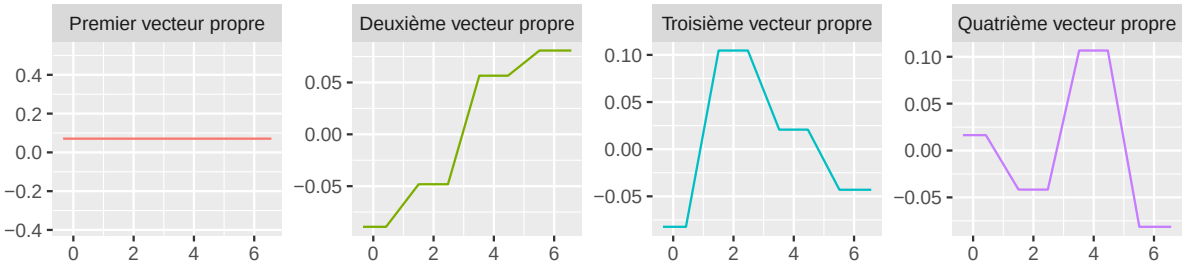


FIGURE 4.2 – Graphe des 20 premières valeurs propres, rangées par ordre croissant.

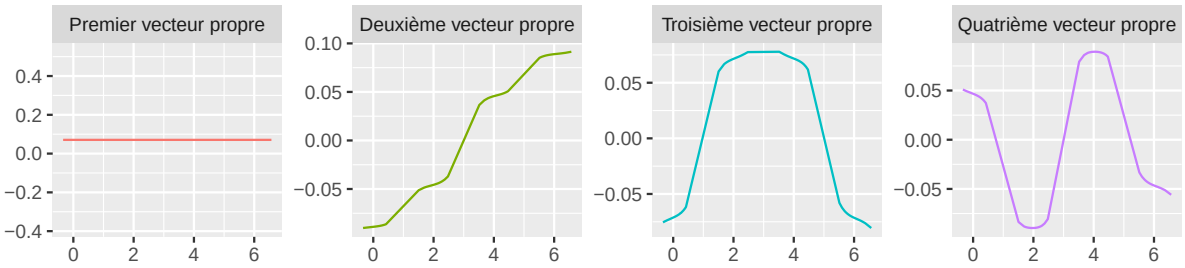
La figure 4.2 représente les 20 premières valeurs propres de la matrice L en fonction de la valeur de σ. On remarque que le choix de σ peut avoir un impact fort sur l'interprétation des résultats. Lorsque $\sigma = 0.2$, on repère visuellement la valeur propre 0 avec une multiplicité 4, ce qui correspond à la présence des 4 classes présentes dans le jeu de données. En réalité, seule la première valeur propre est nulle, les autres étant très faibles (autour de 10^{-1}). En effet, le graphe étant entièrement connecté par construction, la valeur propre 0 devrait apparaître avec la multiplicité 1, cependant la similarité entre points décroît très vite car σ est très faible, ce qui conduit à des valeurs propres proches de 0. Lorsque $\sigma = 0.5$, on obtient toujours la valeur propre 0 avec une multiplicité 1, mais la similarité entre points

décroît moins vite que lorsque $\sigma = 0.2$, ce qui conduit à des valeurs propres plus grandes que dans le cas précédent. On remarque cependant un “saut” à partir de la cinquième valeur propre, ce qui nous conduit également à considérer 4 classes. Lorsque $\sigma = 1$, la similarité décroît encore moins vite, et les points appartenant à des classes différentes n'apparaissent pas aussi éloignés que dans les deux cas précédents. Dans ce cas, le saut dans le graphe des valeurs propres n'est pas aussi visible qu'auparavant.

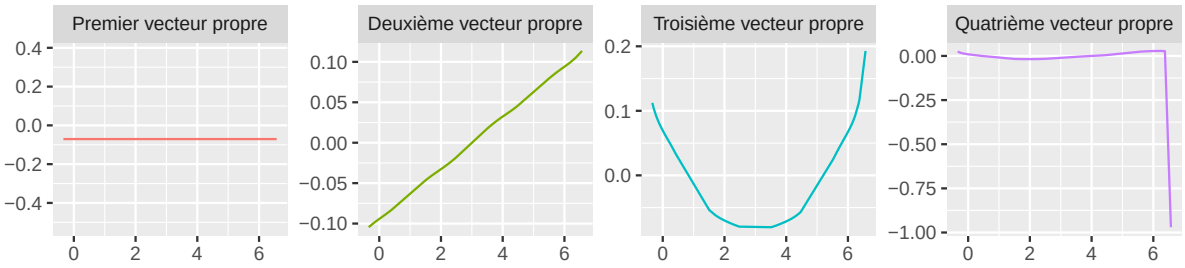
Sur la figure 4.3, on a représenté les 4 premiers vecteurs propres en fonction de x et de σ. Comme attendu, le premier vecteur propre est constant, et les autres permettent une bonne discrimination du jeu de données en 4 classes. Les vecteurs propres sont de plus en plus “lisses” à mesure que σ augmente, et permettent de moins en moins de distinguer les classes du jeu de données initial.



(a) $\sigma = 0.2$



(b) $\sigma = 0.5$



(c) $\sigma = 1$

Figure 4.3 – Graphe des 4 premiers vecteurs propres de la matrice L

La figure 4.4 représente la répartition des points dans les 4 classes obtenues après application de l'algorithme des k-means sur la matrice des 4 premiers vecteurs propres. Avec $\sigma = 0.2$ et $\sigma = 0.5$, on

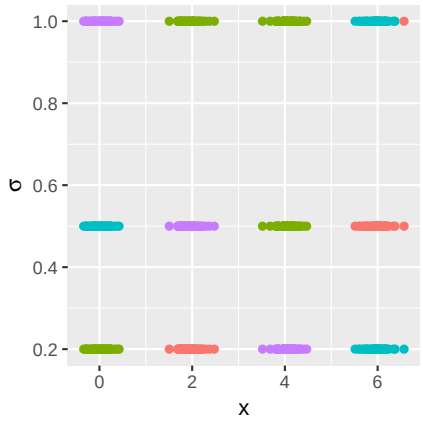


FIGURE 4.4 – Classification des points par partitionnement spectral en fonction de la valeur du paramètre σ de la fonction de similarité Gaussienne (σ est représenté en ordonnée).

récupère bien les 4 classes initiales, mais comme on s'y attendait, lorsque la valeur de σ augmente il est plus difficile de bien identifier les classes.

Chapitre 5

Méthode probabiliste : modèles de mélange

On a déjà rencontré les modèles de mélange et l'approche probabiliste en classification supervisée. Cette méthode peut également être utilisée en classification *non supervisée*, moyennement quelques adaptations. En particulier, le cadre général sera le même que celui présenté au chapitre 2 de la partie II, à la différence (majeure) près que dans le cas de la classification non supervisée, les variables $Z_1, \dots, Z_n$ correspondant à l'identifiant de la classe de l'individu i, ne sont pas observées. L'estimation des paramètres du modèle ne se fait donc pas de la même façon dans le cadre non supervisé.

5.1 Introduction

Nous reprenons ici les notations du modèle de mélange, déjà introduites au chapitre 2 de la partie II. On suppose que l'on dispose d'un échantillon i.i.d. de variables aléatoires $X_1, \dots, X_n$ de loi marginale :

$$f(\mathbf{x}; \theta) = \sum_{k=1}^{K} p_k f_k(\mathbf{x}; \theta_k).$$

L'objectif ici est d'estimer $\theta = (\theta_1, \dots, \theta_K)$, les paramètres des lois du mélange, et également $p = (p_1, \dots, p_K)^t$, les proportions du mélange [1]. Notons $\phi = (\theta, p)$ l'ensemble des paramètres du modèle et $\mathbf{X} = (X_1, \dots, X_n)$. On peut chercher à maximiser la log-vraisemblance suivante :

$$\ell(\phi; \mathbf{X}) = \sum_{i=1}^{n} \ln f(X_i; \phi) \tag{5.1}$$

$$= \sum_{i=1}^{n} \ln \left(\sum_{k=1}^{K} p_k f_k(X_i; \phi) \right). \tag{5.2}$$

Dans le cas de la classification non supervisée, les quantités p_k ne sont pas estimables facilement car on ne connaît pas la répartition des individus en classes, et il n'existe pas de solution analytique au problème de maximisation de $\ell(\phi; \mathbf{X})$.

1. Dans le cadre des modèles de mélange en classification *supervisée*, les p_k sont estimés par les proportions empiriques de chaque classe dans l'échantillon, voire sont connues à l'avance dans certains cas. En classification *non supervisée*, la difficulté vient du fait que l'on n'observe pas la variable de classe Z_i.

Introduisons alors pour chaque individu le vecteur aléatoire Z_i, qui indique le numéro de la classe à laquelle appartient cet individu, comme nous l'avions fait dans le cadre de la classification supervisée. Plus précisément, Z_i est un vecteur de taille K contenant des 0 partout sauf sur la composante correspondant à la classe de l'individu i (par exemple si i appartient à la classe C_k, alors Z_i contient des 0 partout sauf sur la composante k). Notons $Z_i = (Z_{i1}, \dots, Z_{iK})$. Ces variables n'étant pas observées en classification non supervisée, on parle de *variables cachées* ou *variables latentes*. Le fait d'introduire des variables non observées peut paraître contre-intuitif, mais en réalité cela nous permet de nous placer dans le cadre des modèles à variables latentes, et d'utiliser l'algorithme EM (Espérance-Maximisation) pour obtenir l'estimateur du maximum de vraisemblance.

5.2 Modèle à variables latentes

Ré-écrivons le modèle de mélange à l'aide des variables $X_1, \dots, X_n$ et $Z_1, \dots, Z_n$. Comme dans le cas supervisé, on a :

$$X_i \mid Z_{ik} = 1 \sim f_k(\mathbf{x}_i; \phi) \tag{5.3}$$

$$Z_i \sim \mathcal{M}_K(1, (p_1, \dots, p_K)^t) \tag{5.4}$$

La première ligne nous donne $f(\mathbf{X}_i \mid \mathbf{Z}_i; \phi)$, la loi conditionnelle de X_i sachant Z_i, et la deuxième ligne nous donne $f(\mathbf{Z}_i; \phi)$, la loi marginale de Z_i. En notant $p = (p_1, \dots, p_K)^t$, et en utilisant l'expression de la densité de la loi multinomiale par rapport à la mesure de comptage, on peut écrire la densité jointe de (X_i, Z_i) de la façon suivante :

$$f(\mathbf{x}_i, \mathbf{z}_i; \phi) = f(\mathbf{x}_i \mid \mathbf{z}_i; \phi) f(\mathbf{z}_i; p), \tag{5.5}$$

$$= f_k(\mathbf{x}_i; \phi)\, p_1^{z_{i1}} \cdot p_K^{z_{iK}}. \tag{5.6}$$

On appelle log-vraisemblance *observée* la densité jointe des observations $X_1, \dots, X_n$:

$$\ell_{\text{obs}}(\phi; \mathbf{X}) = \sum_{i=1}^{n} \ln f(X_i; \phi) = \sum_{i=1}^{n} \ln \left(\sum_{k=1}^{K} p_k f_k(X_i; \phi) \right), \tag{5.7}$$

et log-vraisemblance *complète* la densité jointe des couples $(X_1, Z_1), \dots, (X_n, Z_n)$:

$$\ell_{\text{comp}}(\phi; \mathbf{X}, \mathbf{Z}) = \sum_{i=1}^{n} \ln f(X_i, Z_i; \phi) = \sum_{i=1}^{n} \sum_{k=1}^{K} Z_{ik} \ln \left(p_k f_k(X_i; \phi) \right). \tag{5.8}$$

Si les variables $Z_1, \dots, Z_n$ étaient connues, la maximisation de (5.8) serait immédiate. Comme ce n'est pas le cas, on va utiliser la probabilité conditionnelle $t_k(\mathbf{x}_i) = \mathbb{P}(Z_{ik} = 1 \mid \mathbf{X}_i = \mathbf{x}_i)$. Plus précisément, l'algorithme EM, qui sera présenté en détails dans la section suivante, procède de façon itérative à l'estimation de cette probabilité conditionnelle (étape E) puis à la maximisation de la vraisemblance complète (étape M) à partir de cette probabilité conditionnelle.

5.3 Estimation des paramètres par l'algorithme EM

L'algorithme EM permet d'obtenir un maximum (local) de la log-vraisemblance en présence de données manquantes ou cachées. Il a été introduit à la fin des années 1970 par (Dempster *et al.*, 1977). Il est particulièrement adapté aux cas où la vraisemblance des données complètes s'écrit plus simplement que la vraisemblance des données observées, et repose sur l'idée suivante : lorsque l'on se trouve en présence de données manquantes, une première intuition est d'estimer ou de remplacer ces données manquantes, puis d'estimer les paramètres du modèle à l'aide des données « augmentées ».

5.3.1 Idée générale

La densité jointe $f(\mathbf{x}, \mathbf{z}; \phi)$ peut se décomposer de deux manières différentes, soit selon la formule donnée dans l'équation (5.5), soit selon la formule suivante :

$$f(\mathbf{x}, \mathbf{z}; \phi) = f(\mathbf{x}; \phi) f(\mathbf{z} \mid \mathbf{x}; \phi), \tag{5.9}$$

et donc

$$\ell_{\text{obs}}(\phi; \mathbf{X}) = \ell_{\text{comp}}(\phi; \mathbf{X}, \mathbf{Z}) - \ln f(\mathbf{Z} \mid \mathbf{X}; \phi), \tag{5.10}$$

Comme $\mathbf{Z}$ est inconnu, l'idée est d'intégrer par rapport à la loi conditionnelle de $\mathbf{Z}$ sachant $\mathbf{X}$, afin d'obtenir une expression qui ne dépende plus de $\mathbf{Z}$. À l'itération m de l'algorithme EM, et étant donné l'estimateur courant de ϕ, ϕ^{m-1}, on calcule l'espérance de chaque membre de l'équation (5.10), par rapport à la loi conditionnelle de $\mathbf{Z}$ sachant $\mathbf{X}$. En remarquant que la log-vraisemblance observée est $\sigma(\mathbf{X})$-mesurable, on obtient :

$$\begin{aligned}
\ell_{\text{obs}}(\phi; \mathbf{X}) &= \mathbb{E}_{\phi^{m-1}} \left(\ell_{\text{comp}}(\phi; \mathbf{X}, \mathbf{Z}) \mid \mathbf{X} \right) - \mathbb{E}_{\phi^{m-1}} \left(\ln f(\mathbf{Z} \mid \mathbf{X}; \phi) \mid \mathbf{X} \right) \\
&= Q(\phi; \phi^{m-1}) + H(\phi; \phi^{m-1}).
\end{aligned}$$

Or, par définition de la fonction H, on a :

$$\begin{aligned}
H(\phi; \phi^m) - H(\phi^m; \phi^m) &= -\mathbb{E}_{\phi^m} \left(\ln f(\mathbf{Z} \mid \mathbf{X}; \phi) - \ln f(\mathbf{Z} \mid \mathbf{X}; \phi^m) \mid \mathbf{Z} \right) \\
&= -\mathbb{E}_{\phi^m} \left(\ln \frac{f(\mathbf{Z} \mid \mathbf{X}; \phi)}{f(\mathbf{Z} \mid \mathbf{X}; \phi^m)} \,\middle|\, \mathbf{X} \right)
\end{aligned}$$

En utilisant la concavité de la fonction ln et l'inégalité de Jensen [2], on en déduit :

$$\begin{aligned}
H(\phi; \phi^m) - H(\phi^m; \phi^m) &\geq -\ln \mathbb{E}_{\phi^m} \left(\frac{f(\mathbf{Z} \mid \mathbf{X}; \phi)}{f(\mathbf{Z} \mid \mathbf{X}; \phi^m)} \,\middle|\, \mathbf{X} \right) \\
&\geq -\ln \int_{\mathbb{R}^{nK}} \frac{f(\mathbf{Z} \mid \mathbf{X}; \phi)}{f(\mathbf{Z} \mid \mathbf{X}; \phi^m)} f(\mathbf{Z} \mid \mathbf{X}; \phi^m) d\mathbf{Z} \\
&\geq -\ln \int_{\mathbb{R}^{nK}} f(\mathbf{Z} \mid \mathbf{X}; \phi) d\mathbf{Z}
\end{aligned}$$

2. Soit f une fonction convexe, et X une variable aléatoire telle que $\mathbb{E}(X)$ et $\mathbb{E}(f(X))$ existent. L'inégalité de Jensen nous dit que $\mathbb{E}(f(X)) \geq f(\mathbb{E}(X))$.

$$\geq 0$$

En particulier, on a $H(\phi;\phi^m) = H(\phi^m;\phi^m)$ si et seulement si $\phi = \phi^m$. Donc à chaque itération m de l'algorithme EM, on est assuré d'augmenter la fonction H. Il suffit donc de s'intéresser à la fonction Q et à la quantité $Q(\phi;\phi^{m-1})$.

Chaque itération de l'algorithme EM se divise alors en deux étapes, l'une dite d'*espérance* (étape E) qui consiste à calculer la quantité $Q(\phi;\phi^{m-1})$ en fonction de la valeur courante de l'estimation ϕ^{m-1}, et une seconde de *maximisation* (étape M), qui consiste à maximiser la fonction Q. L'objectif est de maximiser la log-vraisemblance observée en maximisant Q.

5.3.2 Description de l'algorithme

En partant d'une valeur initiale ϕ^0, l'itération m de l'algorithme consiste à réaliser successivement les deux étapes E et M décrites ci-dessous. Une illustration de l'algorithme est présentée sur la figure 5.1.

Étape E (Espérance)
On calcule la quantité $Q(\phi;\phi^{m-1})$, en fonction de la valeur courante de ϕ^{m-1}. Comme les z_{ik} sont des variables binaires et que $f_k(\mathbf{X}_i;\theta_k)$ est $\sigma(\mathbf{X})$-mesurable, on a :

$$\begin{aligned} Q(\phi;\phi^{m-1}) &= \mathbb{E}_{\phi^{m-1}}\left(\ell_{\text{comp}}(\phi;\mathbf{X},\mathbf{Z}) \mid \mathbf{X}\right) && (5.11)\\ &= \mathbb{E}_{\phi^{m-1}}\left[\sum_{i=1}^{n}\sum_{k=1}^{K} Z_{ik}\ln\left(p_k f_k(\mathbf{X}_i;\theta_k)\right) \,\middle|\, \mathbf{X}\right] && (5.12)\\ &= \sum_{i=1}^{n}\sum_{k=1}^{K} \mathbb{E}_{\phi^{m-1}}[Z_{ik} \mid \mathbf{X}_i]\ln\left(p_k f_k(\mathbf{X}_i;\theta_k)\right) && (5.13)\\ &= \sum_{i=1}^{n}\sum_{k=1}^{K} \mathbb{P}_{\phi^{m-1}}(Z_{ik}=1 \mid \mathbf{X}_i)\ln\left(p_k f_k(\mathbf{X}_i;\theta_k)\right). && (5.14) \end{aligned}$$

Finalement, on obtient la fonction Q suivante :

$$Q(\phi;\phi^{m-1}) = \sum_{i=1}^{n}\sum_{k=1}^{K} t_k(X_i;\phi^{m-1})\ln\left(p_k f_k(X_i;\theta_k)\right) \quad (5.15)$$

$$Q(\phi;\phi^{m-1}) = \sum_{i=1}^{n}\sum_{k=1}^{K} t_k(X_i;\phi^{m-1})\ln p_k + \sum_{i=1}^{n}\sum_{k=1}^{K} t_k(X_i;\phi^{m-1}) f_k(X_i;\theta_k), \quad (5.16)$$

avec :

$$t_k(X_i;\phi^{m-1}) = \frac{p_k^{m-1} f_k(X_i;\theta_k^{m-1})}{\sum_{k=1}^{K} p_k^{m-1} f_k(X_i;\theta_k^{m-1})}.$$

Étape M (Maximisation)
L'étape M consiste à maximiser la fonction Q par rapport à ϕ. Comme la fonction Q peut se décomposer en deux parties, l'une ne dépendant que de p et l'autre ne dépendant que de θ, la maximisation peut se

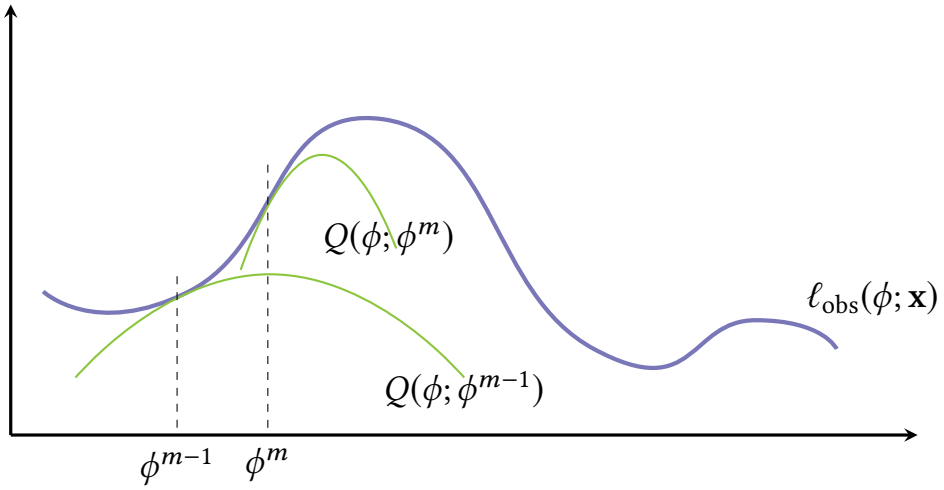


FIGURE 5.1 – Illustration de l'étape m de l'algorithme EM dans le cas où $\phi \in \mathbb{R}$. À partir de ϕ^{m-1}, on obtient la fonction $Q(\phi;\phi^{m-1})$ que l'on maximise afin d'obtenir ϕ^m. Puis, on recalcule la fonction Q à partir de la nouvelle valeur de ϕ^m, et ainsi de suite jusqu'à convergence de l'algorithme.

faire indépendamment pour θ et pour p. En maximisant (5.15) par rapport à p_k, puis par rapport à θ_k, on obtient dans le cas d'un modèle de mélange Gaussien, i.e. pour $\theta_k = (\mu_k, \Sigma_k)$:

$$p_k^m = \frac{1}{n}\sum_{i=1}^{n} t_k(X_i;\phi^{m-1}) \tag{5.17}$$

$$\mu_k^m = \frac{\sum_{i=1}^n t_k(X_i;\phi^{m-1})\, X_i}{\sum_{i=1}^n t_k(X_i;\phi^{m-1})} \tag{5.18}$$

$$\Sigma_k^m = \frac{\sum_{i=1}^n t_k(X_i;\phi^{m-1})\, (X_i-\mu_k^m)(X_i-\mu_k^m)^t}{\sum_{i=1}^n t_k(X_i;\phi^{m-1})} \tag{5.19}$$

Ces deux étapes sont répétées jusqu'à convergence de l'algorithme, c'est-à-dire en pratique jusqu'à ce que la fonction Q se stabilise. Plusieurs variantes de l'algorithme EM ont également été proposées, notamment stochastiques, et reposant sur une approximation de l'étape d'espérance par des méthodes de type Monte Carlo.

Remarque. *Chaque itération de l'algorithme EM permet d'augmenter la valeur de la log-vraisemblance. On peut montrer alors que sous certaines conditions, il converge vers un point stationnaire de la vraisemblance, qui peut être un maximum local, un maximum global, ou un point selle. Dans la pratique il est recommandé de lancer plusieurs fois l'algorithme avec des initialisations différentes, de comparer les résultats obtenus et de retenir celui correspondant à la plus grande vraisemblance.*

5.3.3 Interprétation

Les estimations obtenues dans les équations (5.17) à (5.19) ressemblent beaucoup à celles obtenues pour le mélange Gaussien dans le cadre de la classification supervisée (voir équations (2.17) à (2.19) du

chapitre 2 de la partie II). En effet, on avait dans ce cas :

$$\hat{p}_k^{superv} = \frac{n_k}{n} = \frac{\sum_{i=1}^n Z_{ik}}{n} \tag{5.20}$$

$$\hat{\mu}_k^{superv} = \frac{1}{n_k} \sum_{X_i \in C_k} X_i = \frac{\sum_{i=1}^n Z_{ik} X_i}{\sum_{i=1}^n Z_{ik}} \tag{5.21}$$

$$\hat{\Sigma}_k^{superv} = \frac{1}{n_k} \sum_{\mathbf{X}_i \in C_k} (X_i - \mu_k)^t (X_i - \mu_k)^t = \frac{\sum_{i=1}^n Z_{ik} (X_i - \mu_k)^t (X_i - \mu_k)^t}{\sum_{i=1}^n Z_{ik}} \tag{5.22}$$

Dans le cas de la classification supervisée, chaque observation $\mathbf{x}_i$ est pondérée par la variable binaire indiquant son appartenance à la classe C_k. Dans le cadre de la classification non supervisée, ces variables binaires sont inconnues, et on remplace donc l'étiquette de la classe C_k par la probabilité a posteriori d'appartenir à la classe C_k. On pondère donc, non plus par des variables binaires, mais par des probabilités d'appartenance aux classes.

5.3.4 Estimation de la partition

À partir des estimations obtenues en sortie de l'algorithme EM, on peut déduire une règle d'affectation des individus dans les différentes classes en utilisant le principe du Maximum A Posteriori (MAP). Plus précisément, on propose la règle d'affectation suivante :

$$i \in C_\ell \Leftrightarrow \ell = \underset{k=1,\ldots,K}{\operatorname{argmax}}\, t_k(X_i; \hat{\phi}), \tag{5.23}$$

où $\hat{\phi}$ est l'estimateur obtenu lorsque l'algorithme a convergé. On affecte donc l'individu i à la classe dont la probabilité a posteriori est la plus élevée.

On peut ensuite utiliser les estimations obtenues pour θ_k pour interpréter les classes.

5.4 Classification par l'algorithme CEM

L'approche présentée dans la section précédente a pour objectif principal d'estimer les paramètres du modèle, à savoir p_k et θ_k. On peut également obtenir une partition des données en classes, en définissant une règle d'affectation à partir des estimations obtenues. Le fait de ne pas tenir compte explicitement de l'objectif de classification peut cependant conduire à l'identification de classes qui ne soient pas suffisamment séparées, et rendre ainsi plus difficile l'interprétation des résultats.

Une autre approche, également basée sur un algorithme de type EM, a été proposée par Celeux et Govaert (1992) et intègre de façon explicite cet objectif de classification. En partant de la décomposition obtenue à l'équation (5.10), on obtient :

$$\ell_{\text{obs}}(\phi; \mathbf{X}) = \ell_{\text{comp}}(\phi; \mathbf{X}, \mathbf{Z}) - \sum_{i=1}^n \sum_{k=1}^K Z_{ik} \ln t_k(\mathbf{X}_i; \phi). \tag{5.24}$$

Obtenir une partition idéale, avec des classes parfaitement séparées, signifie que les probabilités a posteriori d'appartenance aux classes sont binaires, il n'y a pas d'ambiguïté quant au choix de la classe.

Dans ce cas, le deuxième terme du membre de gauche dans l'équation (5.24) est nul. Il s'agit en fait d'un terme d'entropie, qui mesure l'écart entre la partition définie par les z_{ik} et les probabilités a posteriori $t_k(\mathbf{x}_i; \phi)$. Autrement dit, la log-vraisemblance observée et la log-vraisemblance complète sont égales en cas de partition idéale. L'algorithme CEM présenté ci-dessous cherche alors à maximiser directement la vraisemblance complète, au lieu de chercher à maximiser la log-vraisemblance observée à l'aide de la fonction Q.

Plus précisément, en partant d'une valeur initiale ϕ^0, l'itération m de l'algorithme CEM (pour Classification Espérance-Maximisation) consiste à réaliser les trois étapes décrites ci-dessous.

Étape E (Espérance)

Identique à l'étape E de l'algorithme EM présenté dans la section précédente. On calcule

$$t_k(X_i; \phi^{m-1}) = \frac{p_k^{m-1} f_k(X_i; \theta_k^{m-1})}{\sum_{k=1}^K p_k^{m-1} f_k(X_i; \theta_k^{m-1})}.$$

Étape C (Classification)

À partir des valeurs de $t_k(X_i; \phi^{m-1})$ calculées à l'étape E, on calcule la partition associée, selon la règle (5.23), c'est-à-dire que l'on estime les z_{ik} de la façon suivante :

$$Z_{ik}^m = \begin{cases} 1 & \text{si } k = \operatorname{argmax}_{\ell=1,\ldots,K} t_\ell(X_i; \phi^{m-1}) \\ 0 & \text{sinon} \end{cases} \tag{5.25}$$

Étape M (Maximisation)

L'étape de maximisation est similaire à l'étape m de l'algorithme EM, sauf que l'on ne cherche pas à maximiser Q, l'espérance conditionnelle de $\ell_{\text{comp}}(\phi; \mathbf{X}, \mathbf{Z})$ sachant $\mathbf{X}$, mais directement $\ell_{\text{comp}}(\phi; \mathbf{X}, \mathbf{Z})$. Plus précisément, on chercher à maximiser :

$$\ell_{\text{comp}}(\phi; \mathbf{X}, \mathbf{Z}^m) = \sum_{i=1}^n \sum_{k=1}^K Z_{ik}^m \ln\left(p_k f_k(X_i; \theta_k)\right).$$

On remarque que cette expression est très proche de celle de la fonction Q, obtenue à l'équation 5.15. On obtient alors les estimations suivantes, dans le cas d'un mélange Gaussien :

$$p_k^m = \frac{1}{n} \sum_{i=1}^n Z_{ik}^m \tag{5.26}$$

$$\mu_k^m = \frac{\sum_{i=1}^n Z_{ik}^m X_i}{\sum_{i=1}^n Z_{ik}^m} \tag{5.27}$$

$$\Sigma_k^m = \frac{\sum_{i=1}^n Z_{ik}^m (X_i - \mu_k^m)(X_i - \mu_k^m)^t}{\sum_{i=1}^n Z_{ik}^m}. \tag{5.28}$$

Dans la pratique, l'algorithme CEM converge souvent plus vite que l'algorithme EM, et peut s'avérer plus efficace pour réaliser un compromis biais-variance, en particulier dans le cas de petits échantillons.

Remarque. *Les estimateurs obtenus avec l'algorithme CEM ne sont pas sans biais, même asymptotiquement.*

5.5 Sélection du nombre de classes

Le nombre de classes de l'échantillon est un paramètre d'entrée des deux algorithmes EM et CEM. Or, ce nombre est souvent inconnu, et peut vouloir identifier le nombre "optimal" de classes. La procédure classique dans ce cas consiste alors à lancer l'algorithme choisi en faisant varier le nombre de classes, et à comparer les résultats obtenus sur des critères de sélection de modèles.

Critères AIC et BIC

Les critères AIC et BIC peuvent être utilisés pour déterminer le nombre de classes à retenir, mais leur objectif est plutôt d'identifier les modèles ajustant le mieux les données. En ce sens, ils peuvent ne pas identifier la classification la mieux séparée. De plus, ils ne peuvent être utilisés que dans le cas de l'algorithme EM, puisque les estimateurs obtenus avec l'algorithme CEM sont biaisés. On rappelle ci-dessous leurs expressions :

$$\begin{aligned} \text{AIC} &= -2\ell_{\text{obs}}(\hat{\phi}; \mathbf{X}) + 2q \\ \text{BIC} &= -2\ell_{\text{obs}}(\hat{\phi}; \mathbf{X}) + q \ln n, \end{aligned}$$

où q est le nombre total de paramètres à estimer, c'est-à-dire le nombre de composantes dans ϕ.

Critère ICL

Un autre critère utilisable est le critère ICL (pour Integrated Complete Likelihood), qui pénalise la log-vraisemblance *complète* à l'aide du même terme de pénalité que le BIC. Plus précisément, on a :

$$\begin{aligned} \text{ICL} &= -2\ell_{\text{comp}}(\hat{\phi}; \mathbf{X}, \hat{\mathbf{Z}}) + q \ln n && (5.29) \\ &= -2\ell_{\text{comp}}(\hat{\phi}; \mathbf{X}) - \sum_{i=1}^{n} \sum_{k=1}^{K} \hat{Z}_{ik} \ln t_k(X_i; \hat{\phi}) + q \ln n && (5.30) \\ &= \text{BIC} - \sum_{i=1}^{n} \sum_{k=1}^{K} \hat{Z}_{ik} \ln t_k(X_i; \hat{\phi}). && (5.31) \end{aligned}$$

Autrement dit, le critère ICL intègre une pénalité concernant la séparation entre classes.

Chapitre 6

Algorithme DBSCAN

L'algorithme DBSCAN (*Density-Based Spatial Clustering of Applications with Noise*) est né au milieu des années 1990. C'est un algorithme de partitionnement qui identifient les zones de forte densité du nuage de points. Il est particulièrement adapté en analyse d'images et plus particulièrement en reconnaissance de formes.

6.1 Définitions

On commence par introduire quelques définitions, qui sont illustrées sur la figure 6.1. On se fixe un seuil $\varepsilon > 0$ et un entier m.

Définition 25. *On appelle* ε-voisinage *d'un point x, et on note $N_\varepsilon(x)$, l'ensemble des points se situant à une distance au plus ε de x :*

$$N_\varepsilon(x) = \{y \in \Omega \mid d(x, y) \leq \varepsilon\}$$

Définition 26. *On appelle* point central *tout point x_c dont le ε-voisinage contient au moins m points. Tous les points appartenant au ε-voisinage d'un point central sont dits* densément atteignables *depuis ce point. On note $\mathcal{P}_c$ l'ensemble des points centraux, avec :*

$$\mathcal{P}_c = \{x \in \Omega \mid \text{Card } N_\varepsilon(x) \geq m\}$$

Définition 27. *Un point y est dit* atteignable *depuis le point x s'il existe un chemin $(x, x_1, x_2, \ldots, x_p, y)$ formés de points tels que x_{i+1} est densément atteignable à partir de x_i.*

Définition 28. *On appelle* point extérieur *un point x dont le ε-voisinage contient strictement moins de m points, mais qui appartient au ε-voisinage d'un point central.*

La définition de point atteignable implique en particulier que tous les points du chemin, à l'exception du point y, sont des points centraux. L'ensemble des points non atteignables forme l'ensemble des points atypiques ou *outliers*.

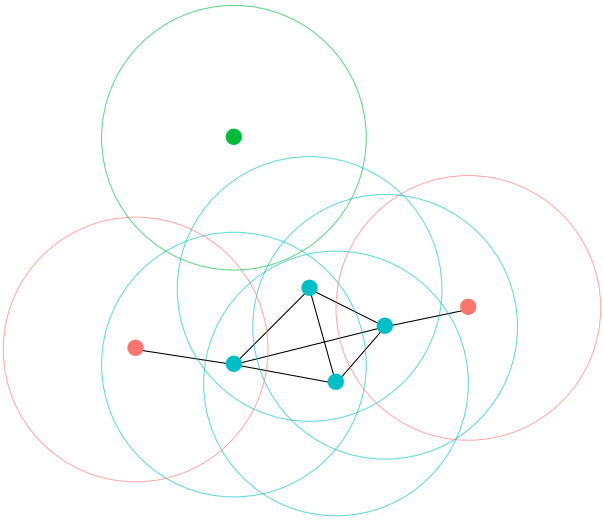


FIGURE 6.1 – Illustration de l'algorithme avec $m = 4$. Les cercles représentent les ε-voisinage de chaque point. Les points centraux *sont ceux ayant au moins 4 points dans leurs ε-voisinage et sont colorés en bleu (●). Ils sont tous densément atteignables les uns des autres. Les points* atteignables *sont colorés en rouge (●). Ce ne sont pas des points centraux car leurs voisinages ne contiennent pas un minimum de 4 points, mais il existe bien un chemin permettant de les relier à chaque point central, et entre eux. Ces points sont des* points extérieurs. *Enfin, le point vert (●) est un point isolé non atteignable.*

6.2 Description de l'algorithme

L'idée principale de l'algorithme DBSCAN est de construire des clusters en regroupant ensemble les points atteignables. Plus précisément, l'algorithme procède de façon itérative en cherchant pour chaque point l'ensemble des points atteignables de puis ce point. Pour cela, on commence par sélectionner un point $x \in \Omega$, puis on construit son ε-voisinage. S'il s'agit d'un point central, on ajoute ses voisins dans le même cluster x, puis on construit le ε-voisinage de ses voisins, et ainsi de suite jusqu'à ce qu'il n'y ait plus de nouveau point à ajouter au cluster. Tous les points ainsi connectés appartiennent au même cluster que x. Par contre, si x n'est pas un point central, autrement dit si son ε-voisinage contient moins de m points, x est considéré comme un point isolé. On ré-itère le processus en partant d'un point qui n'a pas encore été visité par l'algorithme. L'algorithme s'arrête lorsque tous les points ont été visités.

Il est à noter qu'un point considéré comme isolé à une itération donnée de l'algorithme peut être rattaché à un cluster lors d'une itération ultérieure.

À la fin de l'algorithme, on obtient donc k clusters de points, et un ensemble de points isolés que l'on peut interpréter comme du bruit. En cela, l'algorithme DBSCAN est considéré comme étant robuste au bruit dans les données, et ne cherche pas à classer chaque individu dans un cluster.

6.3 Propriétés de l'algorithme

Les principaux avantages de l'algorithme DBSCAN sont les suivants :

— il n'est pas nécessaire de préciser à l'avance le nombre de clusters voulus, l'algorithme les détecte automatiquement

— l'algorithme est robuste à la présence d'outliers, et permet même de les détecter

— il est plus efficace en temps de calcul que certains algorithmes (comme les k-means par exemple)

— il permet de détecter des clusters ayant des formes très variés, de part son exploration de l'espace Ω de proche en proche. Il ne suppose pas de forme prédéfinie (de type sphérique par exemple) pour les clusters.

Ses avantages lui valent d'être assez performant, en particulier en reconnaissance de formes dans l'analyse d'image. Cependant, il existe des cas pour lesquels l'algorithme ne fonctionne pas bien. C'est le cas surtout en grande dimension, à cause du fléau de la dimension. En effet, plus la dimension de l'espace Ω grandit, plus le volume occupé par les points augmente, et plus les points se retrouvent isolés les uns des autres. Les ε-voisinages définis plus hauts se retrouvent alors le plus souvent vides, et il devient difficile de construire des clusters. Par ailleurs, l'algorithme DBSCAN ne détecte pas bien des clusters dont les densités d'occupation de l'espace sont différentes.

Le choix des paramètres ε et m joue également un rôle crucial dans les performances de l'algorithme (tout comme celui de la distance d utilisée pour définir N_ε), et peut être difficile à justifier dans la pratique. En pratique, on peut suivre les recommandations suivantes :

— le paramètre m peut être vu comme une taille minimale de cluster, car c'est ce nombre qui permet de définir les points centraux et de commencer l'accumulation de points dans le cluster. Plusieurs règles empiriques ont été proposées, en particulier $m \geq p+1$, et $m = 2p$. On retiendra qu'une faible valeur de m produit des "petits clusters", ou fonctionne bien lorsque la densité d'occupation dans les clusters est élevée, et qu'une valeur élevée sera en pratique plus robuste en présence d'outliers.

— pour ε, si on ne dispose pas d'information spécifique au jeu de données et au contexte d'application, on peut se baser sur la définition du ε-voisinage. En effet, si deux points appartiennent au même cluster, leurs $(m-1)$ -ème voisins se trouvent environ à la même distance ε. On trace alors le graphe de la distance entre chaque point et son $(m-1)$ -ème voisin, et on cherche un coude dans le graphe obtenu. La distance correspondante pourra être utilisée pour choisir ε (voir Figure 6.2).

6.4 Exemple

On illustre le comportement de l'algorithme, et en particulier l'influence du choix de ε et de m sur le nombre de clusters. Les données utilisées sont représentées sur la figure 6.3. Les données comportent deux clusters ainsi que des observations aléatoires considérées comme du bruit (voir Figure 6.3a), que l'on va essayer de retrouver. Bien sûr, on est dans un cadre de classification non supervisée, donc l'algorithme n'a pas accès à l'information contenue dans 6.3a, mais seulement au jeu de données sans étiquettes 6.3b.

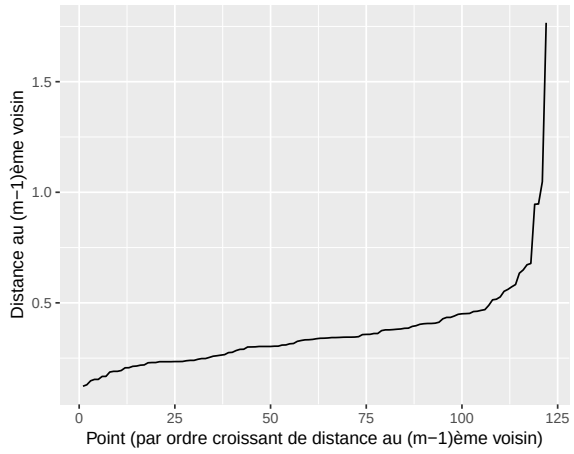
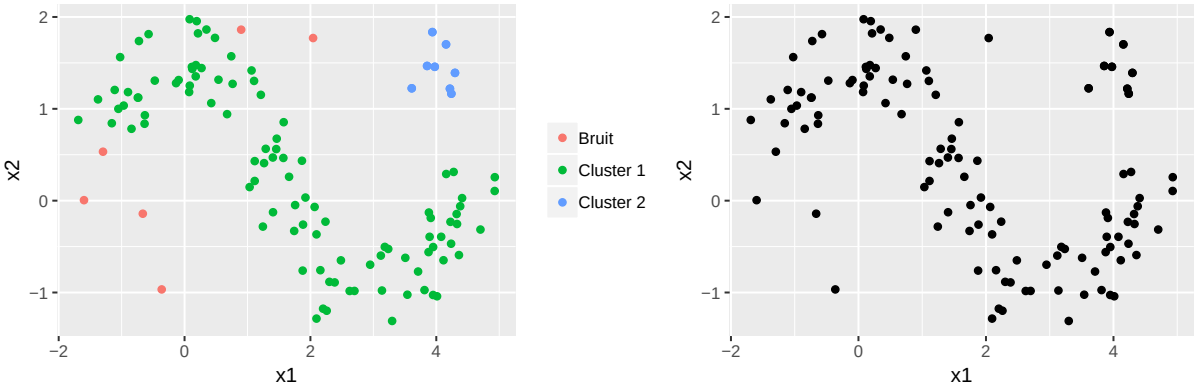


FIGURE 6.2 – Graphe de la distance entre chaque point et son $(m-1)$-ème voisin, par ordre croissant de cette distance. On détecte un coude correspondant à une distance de 0.5, et on pourra alors fixer $\varepsilon = 0.5$



(a) *avec la vraie structure par clusters sous-jacente*

(b) *données sans étiquettes*

FIGURE 6.3 – Données utilisées pour illustrer l'algorithme DBSCAN.

En suivant les recommandations pratiques, on peut se fixer un nombre de voisins $m \geq 2 + 1$, et la règle $m = 2p$ nous donne $m = 4$. On trace alors la distance entre chaque point et son 4-ème plus proche voisin. Le graphe correspondant est celui de la figure 6.2, et un choix de ε guidé par ce graphe se situe aux alentours de 0.5. Sur la figure 6.4, on a représenté les résultats du partitionnement obtenu par l'algorithme pour différentes valeurs de ε et de m.

Quelque soit le nombre de voisins minimum choisis m, de trop faibles valeurs de ε ne permettent pas de retrouver les clusters originaux, et tous les points sont classés comme du bruit (colonne de gauche). Il faut donc augmenter la distance permettant de construire les ε-voisinages. À l'inverse, choisir une valeur trop élevée pour ε a pour conséquence de classer presque tous les points dans le même cluster. En choisissant $\varepsilon = 0.5$, comme suggéré par le graphe des distances au 4 plus proches voisins, on parvient à détecter les clusters. En revanche, même avec cette valeur "raisonnable" pour ε, le choix de m garde une influence non négligeable. Choisir une valeur trop élevée conduit à ne sélectionner que des

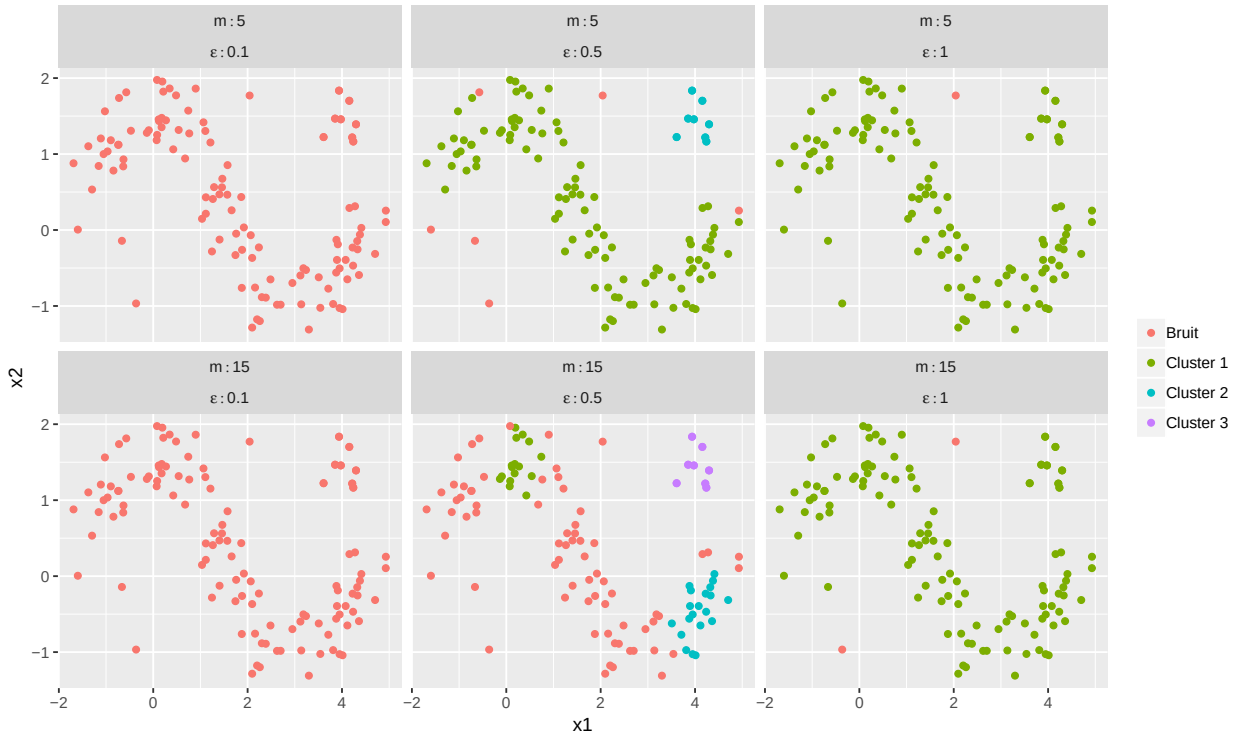


FIGURE 6.4 – Résultats de l'algorithme DBSCAN sur le jeu de données de la figure 6.3, pour différentes valeurs de ε et de m.

clusters de forte densité, ce qui a pour conséquence, sur ce jeu de données, de classer un grand nombre d'observations comme du bruit.

Conclusion

Comme dans le cas de la classification supervisée, les méthodes présentées dans cette partie reposent toutes sur un certain nombre d'hypothèses et ne sont pas toutes aussi performantes sur les mêmes jeux de données. Le tableau 6.1 résume les différentes caractéristiques des méthodes de classification non supervisée que nous avons abordées, et le tableau 6.2 propose différentes implémentations de ces algorithmes sous R et SAS (liste de fonctions non exhaustives, en particulier sous R).

Méthode	Type de variables	Remarques
k-means	mixtes [1]	Fonctionne bien pour des classes sphériques lorsque la distance utilisée est la distance euclidienne. Convergence locale de l'algorithme (dépend de l'initialisation). Nombre de classes k en entrée de l'algorithme
CAH	mixtes	Dépend du critère d'aggrégation choisi Permet d'identifier le nombre de classes *a posteriori* Fonctionne difficilement lorsque n est grand Peut être utilisé comme première étape avant un k-means
Partitionnement spectral	mixtes	Fonctionne bien même si les classes ne sont pas sphériques Dépend de la fonction de similarité choisie Permet d'identifier le nombre de classes *a posteriori*
DBSCAN	quantitatives	Fonctionne pour des classes de forme quelconque Permet de détecter des outliers Permet de détecter le nombre de classes *a posteriori*
Modèle de mélange	quantitatives ou qualitatives	Convergence locale de l'algortihme (dépend de l'initialisation) Nombre de classes k en entrée de l'algorithme

TABLE 6.1 – Caractéristiques des différentes méthodes de classification non supervisée abordées dans le cours

1. en choisissant une distance adaptée au cas mixte, où en transformant les variables quantitatives en variables qualitatives et en utilisant une distance adaptée aux variables qualitatives

Méthode	R	SAS
k-means	fonction `kmeans`	`PROC FASTCLUS;`
CAH	fonction `hclust`	`PROC CLUSTER;` puis `PROC TREE;`
Partitionnement spectral	package `anocva`, fonction `spectralClustering`	à coder ... !
DBSCAN	package `dbscan`	à coder ... !
Modèle de mélange	package `Rmixmod`, fonction `mixmodCluster`	`PROC FMM;`

Table 6.2 – Implémentation des différentes méthodes sous R et SAS

Quatrième partie

Extensions

Chapitre 1

Traitement des données manquantes

Jusqu'à présent dans ce cours, nous avons toujours supposé que la base de données ne comportait aucune donnée manquante. Malheureusement, c'est loin d'être le cas en pratique. La première approche naïve consistant à supprimer les individus ayant au moins une donnée manquante est à proscrire absolument, pour au moins deux raisons : d'une part, si les données manquantes sont réparties uniformément dans la base de données, on risque de supprimer une partie non négligeable des individus, et d'autre part, il n'est JAMAIS bon de supprimer des données, on risque de biaiser les résultats.

Dans un premier temps, on présente les différents types de données manquantes, puis on présentera deux approches utilisables en cas de données MAR : estimation par algorithme EM ou imputation multiple. Il existe bien d'autres approches pour traiter les données manquantes. Le traitement des données manquantes pourrait faire l'objet d'un cours entier, et dépasse donc largement le cadre de ce cours. On se contentera ici de donner un aperçu des problématiques que cela soulève, et de présenter quelques méthodes utilisables dans ce contexte. Pour plus de détails, on pourra se référer au site r-miss-static présentant un ensemble de méthodes, cours, packages `R` sur ce thème.

On supposera dans la suite que l'on s'intéresse à l'estimation d'un vecteur de paramètres que l'on notera θ.

1.1 Type de données manquantes

Notons, comme dans le reste du cours, $X_1, \ldots, X_n$ l'ensemble des observations des n individus, avec $X_i \in \mathbb{R}^p$. On introduit pour chaque individu i, la variable M_i qui vaut 1 si l'individu i possède au moins une donnée manquante, et 0 sinon. On peut alors partitionner l'ensemble des variables en deux sous-ensembles : $\mathbf{X}_{\text{miss}} = \{X_i \mid M_i = 1\}$ et $\mathbf{X}_{\text{obs}} = \{X_i \mid M_i = 0\}$.

On peut classer les données manquantes en trois catégories dues à Donald Rubin (Rubin, 1976), selon que le processus ayant conduit à la présence de données manquantes soit aléatoire ou informatif, ou encore, en fonction de l'indépendance ou non-indépendance entre les variables $\mathbf{X}_{\text{miss}}$, $\mathbf{X}_{\text{obs}}$ et $(M_i)_{i=1,\ldots,n}$.

Données MCAR (missing completely at random)

On dit que les données manquantes suivent un processus MCAR, pour *manquant complètement au hasard*, si la probabilité pour qu'une observation soit manquante est **indépendante** de $\mathbf{X}_{\text{obs}}$ et $\mathbf{X}_{\text{mis}}$. Plus précisément, on a :

$$\mathbb{P}(M_i = 1 \mid \mathbf{X}_{\text{obs}}, \mathbf{X}_{\text{miss}}; \theta) = \mathbb{P}(M_i = 1).$$

C'est le cas idéal, car cela signifie qu'il n'y a pas de raison particulière pour qu'une observation soit manquante.

Quelques exemples de tels processus de données manquantes : étude réalisée sur un sous-échantillon aléatoire d'une grande base de données, bug informatique ayant conduit à perdre une partie des données, appareil de mesure en panne, …

Données MAR (missing at random)

On dit que les données manquantes suivent un processus MAR, pour *manquant au hasard*, si la probabilité pour qu'une observation soit manquante ne dépend pas de la valeur que l'on aurait dû observer, mais qu'elle dépend des variables observées. Autrement dit, on a :

$$\mathbb{P}(M_i = 1 \mid \mathbf{X}_{\text{obs}}, \mathbf{X}_{\text{miss}}; \theta) = \mathbb{P}(M_i = 1 \mid \mathbf{X}_{\text{obs}}).$$

En d'autres termes, les données observées contiennent l'information permettant de savoir si une observation sera manquante ou non.

Quelques exemples de données MAR : une étude où les femmes auraient une probabilité plus élevée de ne pas renseigner leur poids, appareil de mesure qui ne fonctionne pas si la température est trop élevée (et qui mesure autre chose que la température !), …

Données MNAR (missing not at random)

On dit que les données manquantes suivent un processus MNAR, pour *ne manquant pas au hasard*, lorsque la probabilité pour qu'une observation soit manquante dépend de la valeur que l'on aurait dû observer, ou de variables non observées. On ne peut alors pas simplifier les dépendances dans la probabilité $\mathbb{P}(M_i = 1 \mid \mathbf{X}_{\text{obs}}, \mathbf{X}_{\text{miss}}; \theta)$.

Quelques exemples de données MNAR : une étude où les personnes en surpoids ont une plus grande probabilité de ne pas renseigner leur poids, une étude où l'absence de réponse à une question dépend de la catégorie socio-professionnelle de l'individu, celle-ci n'étant pas renseignée dans l'étude (et aucune autre variable ne permet de la prédire), …

Cette catégorie de données manquantes est la plus problématique, car il n'y a alors pas de possibilité de "corriger" ou de prédire le processus de données manquantes.

Il n'existe pas de moyen de tester quel est le mécanisme de données manquantes sous-jacent. Il faudra donc en pratique faire une hypothèse sur ce mécanisme, sans pour autant pouvoir le vérifier sur la base de données. En particulier, il n'est pas possible de distinguer les hypothèses MAR et MNAR à

partir des données observées, puisque l'hypothèse MNAR dépend de quantités non observées et donc non disponibles. Faire l'hypothèse que les données sont MAR doit donc être justifié, que ce soit scientifiquement, ou par rapport au contexte de l'étude.

Si on suppose que les données sont MAR, il est toutefois possible de tester si elles sont en fait MCAR, en testant par exemple si la distribution des données est la même chez les individus sans données manquantes et chez les individus avec données manquantes. Cependant, cette approche nécessite au moins que l'hypothèse MAR soit valide, ce qui est invérifiable en pratique.

1.2 Estimation par algorithme EM

Sous l'hypothèse MAR, l'algorithme EM est le plus adapté pour estimer les paramètres d'un modèle par maximum de vraisemblance. En effet, cet algorithme a été spécialement conçu pour traiter le cas où une partie des variables n'est pas observée. Il s'agit alors d'adapter la méthodologie que l'on souhaite appliquer au jeu de données, au formalisme de l'algorithme EM (écrire la vraisemblance observée, la vraisemblance complète, la fonction Q de l'étape E, ...).

De façon générale, l'algorithme EM procède de façon itérative en calculant puis maximisant la fonction Q définie comme l'espérance conditionnelle de la log-vraisemblance complète, sachant les données manquantes et la valeur courante du paramètre. Dans notre cadre de données manquantes, cela conduit, à l'itération m de l'algorithme, aux deux étapes E et M suivantes :

— **étape E** : calculer la fonction Q :

$$Q(\theta;\theta^{m-1}) = \mathbb{E}_{\theta^{m-1}}\left[\ell_{\text{comp}}(\mathbf{X}_{\text{obs}},\mathbf{X}_{\text{miss}};\theta) \mid \mathbf{X}_{\text{obs}}\right] = \int \ell_{\text{comp}}(\mathbf{X}_{\text{obs}},\mathbf{X}_{\text{miss}};\theta) f(\mathbf{X}_{\text{miss}} \mid \mathbf{X}_{\text{obs}};\theta) d\mathbf{X}_{\text{miss}}$$

— **étape M** : maximiser la fonction Q :

$$\theta^m = \arg\max_{\theta} Q(\theta;\theta^{m-1})$$

Sous R, il existe des packages permettant de traiter le modèle de mélange Gaussien avec données manquantes, ou le modèle de régression logistique avec données manquantes (package `misaem`).

N.B. Une fois les paramètres du modèle estimés à l'aide d'un algorithme de type EM, l'autre difficulté réside dans l'estimation de la variance de ces estimateurs. Comme l'algorithme EM permet d'approcher (voire même d'obtenir dans certains cas) l'estimateur du maximum de vraisemblance, et celui-ci étant asymptotiquement de variance minimale, on peut utiliser la matrice d'information de Fisher. En présence de données manquantes, plusieurs méthodes sont utilisables pour calculer la matrice de Fisher des données observées : soit par le principe de Louis (implémenté dans le package `misaem`), soit par des approches de type bootstrap.

En pratique, il y a au moins deux inconvénients qui peuvent conduire à préférer d'autres méthodes, comme celles qui seront développées dans la section suivante. D'une part, il peut être difficile de mettre

en œuvre l'algorithme, soit parce que l'intégrale intervenant à l'étape E n'est pas calculable analytiquement, soit parce que l'étape M n'est pas explicite. D'autre part, il n'est pas possible d'avoir un algorithme générique pouvant s'adapter dans toutes les situations, car il faut construire une version EM de chaque nouvelle méthode que l'on souhaiterait appliquer.

1.3 Imputation multiple

Toujours sous l'hypothèse MAR, l'autre approche que nous abordons ici pour traiter les données manquantes est celle de l'imputation multiple. Celle-ci se décompose en trois étapes : 1) à l'aide d'un *modèle d'imputation*, on génère des valeurs pour remplacer les données manquantes, de façon à obtenir M jeux de données complétés, 2) à l'aide d'un *modèle d'analyse* [1] (ex. : régression logistique, modèle de mélange, ...) on estime θ sur les M jeux de données imputés et 3) on aggrège les résultats obtenus. En utilisant les résultats de Rubin, et en notant $\hat{\theta}_m$ l'estimateur de θ obtenu sur le m-ème jeu de données imputé, on obtient :

$$\hat{\theta} = \frac{1}{M} \sum_{m=1}^{M} \hat{\theta}_m, \tag{1.1}$$

et

$$\hat{\mathrm{Var}}(\hat{\theta}) = \frac{1}{M} \sum_{m=1}^{M} \hat{\mathrm{Var}}(\hat{\theta}_m) + \left(1 + \frac{1}{M}\right) \frac{1}{M-1} \sum_{im1}^{M} (\hat{\theta}_m - \hat{\theta})^2, \tag{1.2}$$

où $\hat{\mathrm{Var}}(\hat{\theta}_m)$ est la variance estimée de l'estimateur $\hat{\theta}_m$, obtenue en appliquant le modèle d'analyse au jeu de données imputé m.

L'imputation multiple est à préférer à l'imputation simple, qui consiste à générer une unique valeur en remplacement de chaque donnée manquante, car celle-ci a pour conséquence de sous-estimer l'incertitude entourant l'observation manquante. Cependant, les méthodes d'imputation multiples sont souvent plus difficiles et coûteuses à mettre en place que les méthodes d'imputation simple. Elles reposent en particulier sur le choix d'un modèle d'imputation, et ce choix peut être délicat et avoir une forte influence sur la qualité des résultats.

Imputation à l'aide d'une loi normale multivariée

On choisit ici un modèle d'imputation de type gaussien multivarié, i.e. on suppose que la loi jointe des données est une une loi gaussienne multivariée de paramètres μ et Σ, i.e. $X_i \sim \mathcal{N}(\mu, \Sigma)$. *Attention, il s'agit du modèle permettant d'imputer les données, mais pas du modèle qui servira à analyser les données et à construire par exemple un classifieur.*

Les paramètres μ et Σ du modèle d'imputation sont inconnus, et on va donc avoir recours à la procédure suivante pour imputer les données :

1. le modèle d'imputation et le modèle d'analyse sont deux choses différentes : l'un sert à compléter le jeu de données, et peut-être "oublié" par la suite, et l'autre correspond à la méthode que l'on veut mettre en place sur notre jeu de données : dans le cadre de ce cours, il s'agira typiquement d'une méthode de classification supervisée ou non supervisée

1. générer M échantillons bootstrap de l'échantillon initial (on obtient donc M jeux de données contenant chacun des données manquantes)
2. sur chacun de ces échantillons bootstrap, estimer les paramètres μ et Σ à l'aide d'un algorithme EM : on obtient $\hat{\mu}^m$ et $\hat{\Sigma}^m$
3. utiliser les M valeurs estimées à l'étape précédente pour générer des valeurs en remplacement des données manquantes à l'aide de la loi conditionnelle $f(\mathbf{X}_{\text{miss}} \mid \mathbf{X}_{\text{obs}}; \hat{\mu}^m, \hat{\Sigma}^m)$ (qui sera gaussienne ici)

On obtient alors M jeux de données imputés, sur lesquels on peut lancer le modèle d'analyse de notre choix. On aggrègera ensuite les résultats obtenus selon (1.1) et (1.2).

Cette approche est implémentée dans le package R `Amelia`.

N.B. On pourrait penser qu'il suffit d'utiliser l'algorithme EM sur le jeu de données initial, afin d'obtenir un estimateur de μ et de Σ, que l'on pourrait ensuite utiliser pour imputer les valeurs manquantes. Cependant, dans ces travaux sur les données manquantes, Rubin a formalisé la notion d'*imputation propre* (proper imputation), permettant notamment d'obtenir des estimateurs sans biais de la variance. Même si cela dépasse le cadre de ce cours, on peut toutefois noter qu'une des conditions que doit vérifier le processus d'imputation est celle de tenir compte de l'incertitude sur les paramètres. En particulier, cela implique que tous les modèles d'imputation utilisant une unique valeur de paramètres pour générer des valeurs de remplacement ne remplissent pas les conditions nécessaires.

Imputation multiple par équations chaînées

Les méthodes d'imputation multiple par équations chaînées reposent sur une décomposition de la loi jointe des données en un ensemble de lois conditionnelles univariées. Lorsque la dimension des variables est grande, cette approche peut être beaucoup plus efficace que celles reposant directement sur la loi jointe. On parle aussi de méthode FCS (pour Fully Conditional Specification).

Pour $m = 1, \ldots, M$, on répète le processus d'imputation suivant :

1. générer un premier ensemble de valeurs pour remplacer les valeurs manquantes $\mathbf{X}_{\text{miss}}$: on obtient un jeu de données imputé noté $\mathbf{X}^{(m)}$
2. pour chaque variable $\mathbf{X}_{j,\text{miss}}$, avec $j = 1, \ldots, p$, utiliser une méthode d'imputation *univariée* pour générer des valeurs de remplacement des données manquantes, à l'aide d'une distribution de la forme $f(\mathbf{X}_{j,\text{miss}} \mid \mathbf{X}_{j,\text{obs}}, \mathbf{X}^{(m)}_{(-j)})$, et remplacer la variable $\mathbf{X}^{(m)}_j$ par la version imputée, où $\mathbf{X}^{(m)}_{(-j)}$ correspond à l'ensemble des variables privé de la j-ème variable.
3. ré-itérer l'étape 2. jusqu'à convergence, puis renvoyer le jeu de données $\mathbf{X}^{(m)}$

Cette approche est implémentée dans le package R `mice`, et dans la procédure `PROC MI` de SAS.

1.4 Autres approches

Il existe d'autres approches possibles, comme celles évoquées d'imputation simple (même si ces dernières sont à éviter s'il est possible d'avoir recours aux méthodes d'imputation multiple). Des versions de l'ACP adaptées à la présence de données manquantes ont également été développées. Enfin, il est possible d'utiliser des méthodes d'analyse robustes à la présence de données manquantes. C'est le cas notamment des arbres de décision CART (et donc des forêts aléatoires), et de l'implémentation XGBoost, qui permet de prendre en compte les données manquantes sans les supprimer de l'analyse.

Chapitre 2

Classification semi-supervisée

Dans ce chapitre, on s'intéresse au cas où les données sont partiellement étiquetées, c'est-à-dire que l'on connaît la répartition en classes pour une partie seulement des individus, et on souhaite étendre cette répartition en classes à toute la population. C'est ce que l'on appelle de la classification *semi-supervisée*. On traitera cette approche à travers un exemple.

On peut voir la classification semi-supervisée comme un cas particulier de classification supervisée avec des données manquantes sur la variable cible Y.

2.1 Motivation

Dans le cadre d'une étude médicale, on souhaite étudier le lien entre différents marqueurs biologiques et la survenue d'une maladie. Le diagnostic de cette maladie repose sur un test de dépistage très coûteux, et on cherche donc à établir un critère permettant d'identifier les patients avec une forte probabilité d'être malade, auxquels on fera subir le test.

La base de données dont on dispose concerne n patients, pour lesquels on a mesuré la concentration sanguine de $p = 5$ différents marqueurs. Sur ces n patients, n_l, choisis au hasard, ont passé le test de dépistage. Pour ces n_l patients, on sait donc s'ils sont atteints de la maladie en question.

On souhaite utiliser **toute l'information** du jeu de données afin d'établir un critère permettant d'estimer la probabilité pour un individu d'être malade.

2.1.1 Modélisation

On note $X_i, i = 1, \dots, n$ le vecteur correspondant aux observations du i-ème patient, et $Z_i = (Z_{i1}, Z_{i2})$ le vecteur aléatoire indiquant le statut du patient, où $Z_i = (1, 0)$ signifie que le patient i est malade (groupe 1) et $Z_i = (0, 1)$ signifie que le patient est sain (groupe 2). On a donc $X_i \in \mathbb{R}^p$ et $Z_i \in \{0, 1\}^2$.

On appelle *données étiquetées* les n_l observations pour lesquelles on a observé la variable Z_i, et *données non étiquetées* les n_u observations restantes. On a donc $n = n_l + n_u$. On note $(\mathbf{x}_l, \mathbf{z}_l) := \{(x_1, z_1), \dots, (x_{n_l}, z_{n_l})\}$ les observations correspondant aux données étiquetées, et $\mathbf{x}_u := (x_{n_l+1}, \dots, x_n)$

les observations correspondant aux données non étiquetées. Enfin, on note p_k, $k = 1, 2$ la probabilité d'appartenir à la classe k, $f_k(\cdot; \theta_k)$ la densité conditionnelle de X_1 sachant qu'il appartient au groupe k, et $\theta := (\theta_1, \theta_2, p_1, p_2)$ l'ensemble des paramètres du modèle.

On se place dans un **modèle de mélange Gaussien**, c'est-à-dire que l'on suppose que $f_k, k = 1, 2$ est la densité d'une loi Gaussienne de dimension $p = 5$, de moyenne μ_k et de matrice de covariance Σ_k. On a donc $\theta_k = (\mu_k, \Sigma_k)$.

On suppose que les données étiquetées proviennent de n_l réalisations indépendantes et identiquement distribuées du couple (X_1, Z_1), et que les données non étiquetées proviennent de n_u réalisations indépendantes et identiquement distribuées de X_1.

La variable aléatoire Z_i suit une loi multinomiale $\mathcal{M}(1, (p_1, p_2))$, c'est-à-dire, pour $k = 1, 2$:

$$\mathbb{P}(Z_{ik} = 1, Z_{ik'} = 0, k' \neq k) = p_k.$$

En remarquant que Z_{ik} vaut 0 ou 1, on peut ré-écrire cette loi sous la forme :

$$\mathbb{P}(Z_{ik} = 1, Z_{ik'} = 0, k' \neq k) = \mathbb{P}(Z_{i1} = z_{i1}, Z_{i2} = z_{i2}) = p_1^{z_{i1}} p_2^{z_{i2}} = \prod_{k=1}^{2} p_k^{z_{ik}}.$$

La loi jointe du couple (X_i, Z_i) s'écrit :

$$f_{X_i, Z_i}(x, k; \theta_k) := f_{X_i|Z_i}(x; \theta_k)\mathbb{P}(Z_{ik} = 1, Z_{ik'} = 0, k' \neq k) = p_k f_k(x; \theta_k).$$

On peut aussi l'écrire sous la forme :

$$f_{X_i, Z_i}(x, z_{i1}, z_{i2}; \theta_k) = f_{X_i|Z_i}(x; \theta_k)\mathbb{P}(Z_i = (z_{i1}, z_{i2})) = \prod_{k=1}^{2} (p_k f_k(x; \theta_k))^{z_{ik}}.$$

La loi marginale de X_i s'obtient en intégrant la loi jointe par rapport à Z_i et s'écrit alors :

$$f_{X_i}(x; \theta_k) = \sum_{k=1}^{2} p_k f_k(x; \theta_k) = p_1 f_1(x; \theta_k) + p_2 f_2(x; \theta_k).$$

La loi conditionnelle de Z_i sachant X_i est une loi multinomiale de paramètres 1 et $t(x; \theta) = (t_1(x; \theta), t_2(x; \theta))$, où $t_k(x; \theta) = \mathbb{P}(Z_{ik} = 1, Z_{ik'} = 0, k' \neq k \mid X_i = x)$. Elle s'obtient à l'aide de la loi jointe du couple (X_i, Z_i) et de la loi marginale de X_i, et on a :

$$t_k(x; \theta) = \frac{f_{X_i, Z_i}(x, k; \theta_k)}{f_{X_i}(x; \theta_k)} = \frac{p_k f_k(x; \theta_k)}{p_1 f_1(x; \theta_k) + p_2 f_2(x; \theta_k)}.$$

Par définition de la loi multinomiale, la loi conditionnelle de Z_{ik} sachant X_i est une loi de Bernoulli de paramètre $t_k(x; \theta)$. On a donc :

$$\mathbb{E}(Z_{ik} \mid X_i = x) = t_k(x; \theta)$$

2.1.2 Estimation des paramètres

On cherche maintenant à construire la vraisemblance du modèle. Par hypothèse, les données étiquetées proviennent de n_ℓ réalisations indépendantes et identiquement distribuées du couple (X_1, Z_1), et les données non étiquetées proviennent de n_u réalisations indépendantes et identiquement distribuées de X_1. La vraisemblance étant la densité jointe des observations, elle peut donc se décomposer en deux termes, l'un correspondant aux n_ℓ couples de données étiquetées, et l'autre correspondant aux n_u données non étiquetées. Cela se traduit par la log-vraisemblance suivante :

$$\begin{aligned}\ell_{\text{obs}}(\theta; \mathbf{x}_l, \mathbf{z}_l, \mathbf{x}_u) &= \sum_{i=1}^{n_\ell} \ln\left(\prod_{k=1}^{2}(p_k f_k(x_i;\theta_k))^{z_{ik}}\right) + \sum_{i=1}^{n_u}\left(\ln\sum_{k=1}^{2} p_k f_k(x;\theta_k)\right)\\ &= \sum_{i=1}^{n_\ell}\sum_{k=1}^{2} z_{ik}\ln\left(p_k f_k(x_i;\theta_k)\right) + \sum_{i=n_\ell+1}^{n}\left(\ln\sum_{k=1}^{2} p_k f_k(x;\theta_k)\right)\end{aligned}$$

Comme dans le cas non supervisée, le fait que certaines des variables Z_i ne soient pas observées ne permet pas d'obtenir une maximisation explicite de la vraisemblance. Cependant, on est dans le cadre idéal pour appliquer l'algorithme EM, adapté aux modèles à variables latentes ou cachées. Chaque itération de l'algorithme se décompose en deux étapes : une première étape d'estimation (étape E), pendant laquelle on calcule l'espérance conditionnelle de la log-vraisemblance des données complètes, sachant les observations, et une deuxième étape de maximisation (étape M) pendant laquelle on maximise l'espérance calculée à l'étape E.

Ici, les données observées sont $(\mathbf{x}_l, \mathbf{z}_l, \mathbf{x}_u)$, et les données complètes $(\mathbf{x}_l, \mathbf{z}_l, \mathbf{x}_u, \mathbf{z}_u)$. La log-vraisemblance complète est la vraisemblance des n couples (X_i, Z_i), pour $i = 1, \ldots, n$. Dans ce cas, par indépendance des observations, la vraisemblance complète s'écrit comme un produit de n termes, où chaque terme correspond à la densité du couple (X_i, Z_i) évaluée en (x_i, z_i). En passant au logarithme, on a :

$$\begin{aligned}\ell_{\text{comp}}(\theta; \mathbf{x}_l, \mathbf{z}_l, \mathbf{x}_u, \mathbf{z}_u) &= \sum_{i=1}^{n} \ln\left(\prod_{k=1}^{2}(p_k f_k(x_i;\theta_k))^{z_{ik}}\right)\\ &= \sum_{i=1}^{n}\sum_{k=1}^{2} z_{ik}\ln(p_k f_k(x_i;\theta_k))\\ &= \sum_{i=1}^{n_\ell}\sum_{k=1}^{2} z_{ik}\ln(p_k f_k(x_i;\theta_k)) + \sum_{i=n_\ell+1}^{n}\sum_{k=1}^{2} z_{ik}\ln(p_k f_k(x_i;\theta_k)).\end{aligned}$$

Etape E

On a, par linéarité de l'espérance conditionnelle, et en utilisant le fait que $(\mathbf{x}_l, \mathbf{z}_l, \mathbf{x}_u)$ est $\sigma(\mathbf{x}_l, \mathbf{z}_l, \mathbf{x}_u)$-mesurable, et que Z_i est indépendant de $X_{i'}$, pour $i' \neq i$:

$$\begin{aligned}Q(\theta;\theta^m) &= \mathbf{E}_{\theta^m}[\ell_{\text{comp}}(\theta; \mathbf{x}_l, \mathbf{z}_l, \mathbf{x}_u, \mathbf{z}_u) \mid \mathbf{x}_l, \mathbf{z}_l, \mathbf{x}_u]\\ &= \mathbf{E}_{\theta^m}\left[\sum_{i=1}^{n_\ell}\sum_{k=1}^{2} z_{ik}\ln(p_k f_k(x_i;\theta_k)) + \sum_{i=n_\ell+1}^{n}\sum_{k=1}^{2} z_{ik}\ln(p_k f_k(x_i;\theta_k)) \,\middle|\, \mathbf{x}_l, \mathbf{z}_l, \mathbf{x}_u\right]\end{aligned}$$

$$
\begin{aligned}
&= \sum_{i=1}^{n_\ell}\sum_{k=1}^{2} \mathbf{E}_{\theta^m}\left[z_{ik}\ln(p_k f_k(x_i;\theta_k)) \mid \mathbf{x}_l, \mathbf{z}_l, \mathbf{x}_u\right] + \sum_{i=n_\ell+1}^{n}\sum_{k=1}^{2} \mathbf{E}_{\theta^m}\left[z_{ik}\ln(p_k f_k(x_i;\theta_k)) \mid \mathbf{x}_l, \mathbf{z}_l, \mathbf{x}_u\right] \\
&= \sum_{i=1}^{n_\ell}\sum_{k=1}^{2} z_{ik}\ln(p_k f_k(x_i;\theta_k)) + \sum_{i=n_\ell+1}^{n}\sum_{k=1}^{2} \ln(p_k f_k(x_i;\theta_k))\mathbf{E}_{\theta^m}\left[z_{ik} \mid \mathbf{x}_l, \mathbf{z}_l, \mathbf{x}_u\right] \\
&= \sum_{i=1}^{n_\ell}\sum_{k=1}^{2} z_{ik}\ln(p_k f_k(x_i;\theta_k)) + \sum_{i=n_\ell+1}^{n}\sum_{k=1}^{2} \ln(p_k f_k(x_i;\theta_k))\mathbf{E}_{\theta^m}\left[z_{ik} \mid X_i = x_i\right] \\
&= \sum_{i=1}^{n_\ell}\sum_{k=1}^{2} z_{ik}\ln(p_k f_k(x_i;\theta_k)) + \sum_{i=n_\ell+1}^{n}\sum_{k=1}^{2} \ln(p_k f_k(x_i;\theta_k)) t_k(x_i;\theta^m)
\end{aligned}
$$

On a également l'expression exacte de $Q(\theta;\theta^m)$ dans le cas Gaussien :

$$
\begin{aligned}
Q(\theta;\theta^m) &= \sum_{i=1}^{n_\ell}\sum_{k=1}^{2} z_{ik}\left[\ln p_k - \frac{p}{2}\ln(2\pi) - \frac{1}{2}\ln|\Sigma_k| - \frac{1}{2}(x_i-\mu_k)^t\Sigma_k^{-1}(x_i-\mu_k)\right] \\
&\quad + \sum_{i=n_\ell+1}^{n}\sum_{k=1}^{2} t_k(x_i;\theta^m)\left[\ln p_k - \frac{p}{2}\ln(2\pi) - \frac{1}{2}\ln|\Sigma_k| - \frac{1}{2}(x_i-\mu_k)^t\Sigma_k^{-1}(x_i-\mu_k)\right] \\
&= C^{ste} + \sum_{i=1}^{n_\ell}\sum_{k=1}^{2} z_{ik}\ln p_k + \sum_{i=n_\ell+1}^{n}\sum_{k=1}^{2} t_k(x_i;\theta^m)\ln p_k \\
&\quad -\frac{1}{2}\sum_{i=1}^{n_\ell}\sum_{k=1}^{2} z_{ik}\left[\ln|\Sigma_k| + (x_i-\mu_k)^t\Sigma_k^{-1}(x_i-\mu_k)\right] \\
&\quad -\frac{1}{2}\sum_{i=n_\ell+1}^{n}\sum_{k=1}^{2} t_k(x_i;\theta^m)\left[\ln|\Sigma_k| + (x_i-\mu_k)^t\Sigma_k^{-1}(x_i-\mu_k)\right] \\
&= C^{ste} + \sum_{k=1}^{2} n_k^m \ln p_k - \frac{1}{2}\sum_{i=1}^{n_\ell}\sum_{k=1}^{2} z_{ik}\left[\ln|\Sigma_k| + (x_i-\mu_k)^t\Sigma_k^{-1}(x_i-\mu_k)\right] \\
&\quad -\frac{1}{2}\sum_{i=n_\ell+1}^{n}\sum_{k=1}^{2} t_k(x_i;\theta^m)\left[\ln|\Sigma_k| + (x_i-\mu_k)^t\Sigma_k^{-1}(x_i-\mu_k)\right].
\end{aligned}
$$

Etape M

Pour l'étape de maximisation, on dérive $Q(\theta;\theta^m)$ par rapport à p_k^{m+1}, μ_k^{m+1} et Σ_k^{m+1}, en notant bien que les p_k sont reliés par la relation $p_1 + p_2 = 1$, et que l'on a $n_1^m + n_2^m = n$. Autrement dit, on a $\sum_{k=1}^{2} n_k^m \ln p_k = n_1^m \ln p_1 + (n - n_1^m)\ln(1-p_1)$. On note aussi que $(x_i-\mu_k)^t\Sigma_k^{-1}(x_i-\mu_k) \in \mathbb{R}$, et donc on a en particulier $(x_i-\mu_k)^t\Sigma_k^{-1}(x_i-\mu_k) = \mathrm{Trace}((x_i-\mu_k)^t\Sigma_k^{-1}(x_i-\mu_k)) = \mathrm{Trace}((x_i-\mu_k)(x_i-\mu_k)^t\Sigma_k^{-1})$. D'où :

$$
\begin{aligned}
\frac{\partial Q(\theta;\theta^m)}{\partial p_1} &= \frac{n_1^m}{p_1} - \frac{n-n_1^m}{1-p_1} \\
\frac{\partial Q(\theta;\theta^m)}{\partial \mu_k} &= -\sum_{i=1}^{n_\ell} z_{ik}\Sigma_k^{-1}(x_i-\mu_k) - \sum_{i=n_\ell+1}^{n} t_k(x_i;\theta^m)\Sigma_k^{-1}(x_i-\mu_k) \\
&= -\Sigma_k^{-1}\left(\sum_{i=1}^{n_\ell} z_{ik}(x_i-\mu_k) + \sum_{i=n_\ell+1}^{n} t_k(x_i;\theta^m)(x_i-\mu_k)\right)
\end{aligned}
$$

$$= -\Sigma_k^{-1}\left(\sum_{i=1}^{n_\ell} z_{ik}x_i + \sum_{i=n_\ell+1}^{n} t_k(x_i;\theta^m)x_i - n_k^m\mu_k\right)$$

Attention, ici on dérive par rapport à Σ_k^{-1} :

$$\begin{aligned}\frac{\partial Q(\theta;\theta^m)}{\partial(\Sigma_k^{-1})} &= \frac{1}{2}\sum_{i=1}^{n_\ell} z_{ik}\Sigma_k - \frac{1}{2}\sum_{i=1}^{n_\ell} z_{ik}(x_i-\mu_k)(x_i-\mu_k)^t \\ &\quad + \frac{1}{2}\sum_{i=n_\ell+1}^{n} t_k(x_i;\theta^m)\Sigma_k - \frac{1}{2}\sum_{i=n_\ell+1}^{n} t_k(x_i;\theta^m)(x_i-\mu_k)(x_i-\mu_k)^t \\ &= -\frac{1}{2}\left(\sum_{i=1}^{n_\ell} z_{ik}(x_i-\mu_k)(x_i-\mu_k)^t + \sum_{i=n_\ell+1}^{n} t_k(x_i;\theta^m)(x_i-\mu_k)(x_i-\mu_k)^t - n_k^m\Sigma_k\right)\end{aligned}$$

En annulant chaque dérivée partielle, on obtient :

$$\begin{aligned}p_1^{m+1} &= \frac{n_1^m}{n}, \qquad p_2^{m+1} = 1 - p_1^{m+1} = \frac{n_2^m}{n} \\ \mu_k^{m+1} &= \frac{\sum_{i=1}^{n_\ell} z_{ik}x_i + \sum_{i=n_\ell+1}^{n} t_k(x_i;\theta^m)x_i}{n_k^m} \\ \Sigma_k^{m+1} &= \frac{\sum_{i=1}^{n_\ell} z_{ik}(x_i-\mu_k)(x_i-\mu_k)^t + \sum_{i=n_\ell+1}^{n} t_k(x_i;\theta^m)(x_i-\mu_k)(x_i-\mu_k)^t}{n_k^m}\end{aligned}$$

2.2 Application

On applique les résultats obtenus dans la section précédente. On commence par une analyse descriptive des données, en regardant la distribution de chacun des 5 marqueurs biologiques dans la population (voir Table 2.1). Puis, on applique l'algorithme EM précédent : les résultats de la classification semi-supervisée sont présentés dans la Table 2.2. On appelle ce modèle **M1**.

Marqueur1	Marqueur2	Marqueur3	Marqueur4	Marqueur5	MaladeOuiNon
Min. :-5.4920	Min. :-5.312	Min. :-6.11629	Min. :-1.988	Min. :-6.848	Malade: 5
1st Qu.:-0.2960	1st Qu.: 1.145	1st Qu.:-1.69015	1st Qu.: 1.988	1st Qu.: 2.281	Sain : 50
Median : 0.7808	Median : 2.357	Median :-0.01649	Median : 2.940	Median : 4.974	NA's :312
Mean : 0.8499	Mean : 2.410	Mean : 0.04692	Mean : 2.945	Mean : 4.827	
3rd Qu.: 1.9661	3rd Qu.: 3.783	3rd Qu.: 1.68423	3rd Qu.: 3.874	3rd Qu.: 7.497	
Max. : 7.4918	Max. :10.564	Max. : 8.68274	Max. : 8.065	Max. :16.291	

TABLE 2.1 – Statistiques descriptives de la base de données

Questions

9. Que vaut n ? que vaut n_ℓ ?
10. Peut-on identifier des marqueurs qui sembleraient, à première vue, jouer un rôle dans la discrimination des patients sains et malades ?
11. Donner les évaluations numériques des estimateurs de p_k, μ_k et Σ_k.

```
************************************************************
* Number of samples    =  367
* Problem dimension    =  5
************************************************************
*       Number of cluster =  2
*              Model Type =  Gaussian_pk_Lk_C
*               Criterion =  BIC(7046.1245)
*              Parameters =  list by cluster
*                  Cluster  1 :
                         Proportion =  0.0585
                              Means =  -0.3104 2.2205 0.2550 3.0156 3.7446
                          Variances = |     3.1283     0.5578     1.9440     2.0947     1.8808 |
                                      |     0.5578     4.4884     0.0569     2.0353     1.5578 |
                                      |     1.9440     0.0569     6.1006     1.2262     0.7601 |
                                      |     2.0947     2.0353     1.2262     2.1376     1.0291 |
                                      |     1.8808     1.5578     0.7601     1.0291    16.8710 |
*                  Cluster  2 :
                         Proportion =  0.9415
                              Means =  0.9219 2.4219 0.0340 2.9409 4.8941
                          Variances = |     3.0450     0.5430     1.8922     2.0389     1.8307 |
                                      |     0.5430     4.3689     0.0554     1.9811     1.5163 |
                                      |     1.8922     0.0554     5.9381     1.1935     0.7399 |
                                      |     2.0389     1.9811     1.1935     2.0806     1.0016 |
                                      |     1.8307     1.5163     0.7399     1.0016    16.4216 |
*          Log-likelihood =  -3443.3399
************************************************************
```

*TABLE 2.2 – Résultats de la première classification semi-supervisée (modèle **M1**)*

Suite à une augmentation du budget de l'étude, on propose à plusieurs patients n'ayant pas été dépistés précédemment de passer le test. On choisit aléatoirement 255 nouveaux patients. Les résultats de cette nouvelle classification semi-supervisée sont présentés dans la Table 2.3. On appelle ce modèle **M2**.

Puis, suite à une dernière levée de fonds, on peut finalement proposer à tous les patients restants de passer le test. Les résultats de l'analyse correspondante se trouvent dans la Table 2.4. On appelle ce modèle **M3**

Suite aux résultats de cette étude, un autre groupe de recherche travaillant sur la même maladie souhaite tester le modèle sur ses propres patients. Pour cela, il dispose d'un échantillon de 113 individus, pour lesquels les mêmes marqueurs biologiques ont été mesurés, et qui ont tous passé le test de dépistage. Les résultats de la comparaison du modèle 1 (classification semi-supervisée de la Table 2.2) et du modèle 3 (classification de la Table 2.4) sont présentés dans la Table 2.5.

```
************************************************************
* Number of samples    =  367
* Problem dimension    =  5
************************************************************
*       Number of cluster =  2
*              Model Type =  Gaussian_pk_Lk_C
*               Criterion =  BIC(7066.3242)
*              Parameters =  list by cluster
*                  Cluster  1 :
                         Proportion =  0.0697
                              Means =  -0.4937 1.9488 0.6097 2.7590 3.2519
                          Variances = |     3.0383     0.5183     1.9632     2.0454     1.7778 |
                                      |     0.5183     4.4217     0.0738     2.0040     1.4963 |
                                      |     1.9632     0.0738     6.0075     1.2206     0.8055 |
                                      |     2.0454     2.0040     1.2206     2.1097     0.9899 |
                                      |     1.7778     1.4963     0.8055     0.9899    16.5581 |
*                  Cluster  2 :
                         Proportion =  0.9303
                              Means =  0.9506 2.4447 0.0048 2.9593 4.9450
                          Variances = |     2.9945     0.5108     1.9349     2.0159     1.7522 |
                                      |     0.5108     4.3580     0.0728     1.9752     1.4747 |
                                      |     1.9349     0.0728     5.9209     1.2030     0.7939 |
                                      |     2.0159     1.9752     1.2030     2.0793     0.9757 |
                                      |     1.7522     1.4747     0.7939     0.9757    16.3195 |
*          Log-likelihood =  -3453.4397
************************************************************
```

*TABLE 2.3 – Résultats de la deuxième classification semi-supervisée (modèle **M2**)*

Questions

12. Comparer les résultats des deux classifications semi-supervisées (Tables 2.2 et 2.3). Quels sont les taux de données *non étiquetées* dans chaque cas ?
13. Quelle analyse a t-on effectuée dans la Table 2.4 ? Au vu des résultats, à quel autre modèle pourrait-on comparer les résultats ? Comment ferait-on ensuite pour choisir entre ces deux modèles concurrents ?
14. Comparer les résultats de la Table 2.4 avec ceux des deux classifications semi-supervisées. Commenter.
15. Qu'a t-on représenté dans la Table 2.5 ? Donner une estimation de l'erreur de mauvais classement obtenue avec chaque modèle.
16. Comment a-t-on affecté chaque patient à la catégorie "Malade" ou "Sain" ? Quelles hypothèses a t-on faites ? Qu'aurait-on pu faire autrement ?
17. Si nous n'avions pas eu accès aux données de l'autre groupe de recherche, comment aurait-on pu évaluer l'erreur de mauvais classement ?

```
************************************************************
* Number of samples    =  367
* Problem dimension    =  5
************************************************************
*       Number of cluster =  2
*              Model Type =  Gaussian_pk_Lk_C
*               Criterion =  BIC(7074.3668)
*              Parameters =  list by cluster
*                  Cluster  1 :
                         Proportion =  0.0736
                              Means =  -0.4764 1.9610 0.5427 2.7488 3.6000
                          Variances = |     2.9939     0.5118     1.9333     2.0170     1.7850 |
                                      |     0.5118     4.3670     0.0710     1.9793     1.4884 |
                                      |     1.9333     0.0710     5.9354     1.2042     0.7769 |
                                      |     2.0170     1.9793     1.2042     2.0826     0.9809 |
                                      |     1.7850     1.4884     0.7769     0.9809    16.4111 |
*                  Cluster  2 :
                         Proportion =  0.9264
                              Means =  0.9552 2.4458 0.0075 2.9609 4.9244
                          Variances = |     2.9917     0.5115     1.9319     2.0155     1.7837 |
                                      |     0.5115     4.3639     0.0709     1.9779     1.4873 |
                                      |     1.9319     0.0709     5.9311     1.2034     0.7763 |
                                      |     2.0155     1.9779     1.2034     2.0811     0.9802 |
                                      |     1.7837     1.4873     0.7763     0.9802    16.3992 |
*          Log-likelihood =  -3457.4610
************************************************************
```

*TABLE 2.4 – Résultats de l'analyse finale (tous les patients ont passé le test de dépistage, modèle **M3**).*

Modèle **M1**

Prédit \ réel	Malade	Sain
Malade	4	1
Sain	2	106

Modèle **M3**

Prédit \ réel	Malade	Sain
Malade	3	1
Sain	3	106

TABLE 2.5 – Comparaison des prédictions obtenues avec chaque modèle

Bibliographie

J. P. Benzecri : Sur le calcul des taux d'inertie dans l'analyse d'un questionnaire, addendum et erratum à [bin. mult.]. *Cahiers de l'Analyse des Données*, 4(3) :377–378, 1979.

L. Breiman, J. Friedman, R. Olshen et C. Stone : Classification and regression trees. wadsworth int. *Group*, 37(15) :237–251, 1984.

G. Celeux et G. Govaert : A classification em algorithm for clustering and two stochastic versions. *Computational statistics & Data analysis*, 14(3) :315–332, 1992.

G. Celeux et J.-P. Nakache : *Analyse discriminante sur variables qualitatives.* Politechnica, 1994.

N. V. Chawla, K. W. Bowyer, L. O. Hall et W. P. Kegelmeyer : Smote : synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16 :321–357, 2002.

T. Chen et C. Guestrin : Xgboost : A scalable tree boosting system. 2016. URL http://arxiv.org/abs/1603.02754.

A. Dempster, N. M. Laird et D. Rubin : Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society – Series B Statistical Methodology*, 39(1) :1–38, 1977.

B. Efron : Bootstrap methods : Another look at the jackknife. *The Annals of Statistics*, 7(1) :1–26, 1979.

Y. Freund et R. E. Schapire : A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1) :119–139, 1997.

M. J. Greenacre : Biplots in correspondence analysis. *Journal of Applied Statistics*, 20(2) :251–269, 1993.

Haibo He, Yang Bai, E. A. Garcia et Shutao Li : Adasyn : Adaptive synthetic sampling approach for imbalanced learning. *In 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, p. 1322–1328, 2008.

T. Hastie, R. Tibshirani et J. Friedman : *The Elements of Statistical Learning.* NY : Springer, 2001.

D. B. Rubin : Inference and missing data. *Biometrika*, 63(3) :581–592, 1976.

G. Saporta : *Probabilités, analyse des données et statistique.* Editions Technip, 2006.

A. J. Scott et M. J. Symons : Clustering methods based on likelihood ratio criteria. *Biometrics*, p. 387–397, 1971.

U. von Luxburg : A tutorial on spectral clustering, 2006. Technical report.

Annexes

Annexe A

Algorithme de Newton-Raphson

L'algorithme de Newton-Raphson est un algorithme itératif dont l'objectif est de trouver le zéro d'une fonction f deux fois différentiable, définie de $\mathbb{R}^n$ dans $\mathbb{R}^p$. On utilise pour cela le développement en série de Taylor à l'ordre 1 au voisinage d'un point $\boldsymbol{x}_0$:

$$f(\boldsymbol{x}) = f(\boldsymbol{x}_0) + J_f(\boldsymbol{x}_0)(\boldsymbol{x} - \boldsymbol{x}_0) + o(\|\boldsymbol{x} - \boldsymbol{x}_0\|) \tag{A.1}$$

On définit sa matrice jacobienne par :

$$J_f = \frac{\partial f}{\partial \boldsymbol{x}} = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{pmatrix}. \tag{A.2}$$

On cherche la valeur de $\boldsymbol{x}$ telle que $f(\boldsymbol{x}) = 0$, ce qui donne, une fois ré-injecté dans l'équation (A.1), l'équation de récurrence suivante :

$$\boldsymbol{x}_k = \boldsymbol{x}_{k-1} - J_f^{-1}(\boldsymbol{x}_{k-1}) f(\boldsymbol{x}_{k-1}),$$

en partant d'une valeur initiale x_0.

Lorsque l'on cherche à maximiser une fonction de vraisemblance (ou de log-vraisemblance), on cherche souvent les valeurs qui annulent la ***dérivée*** de la (log-)vraisemblance. Dans ce cas, on obtient l'équation de récurrence suivante :

$$\boldsymbol{x}_k = \boldsymbol{x}_{k-1} - H^{-1}(\boldsymbol{x}_{k-1}) J(\boldsymbol{x}_{k-1}),$$

où H est la matrice Hessienne de la (log-)vraisemblance, définie par :

$$H_f = \frac{\partial^2 f}{\partial \boldsymbol{x} \partial \boldsymbol{x}^t} = \begin{pmatrix} \frac{\partial^2 f_1}{\partial x_1^2} & \cdots & \frac{\partial^2 f_1}{\partial x_n \partial x_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f_m}{\partial x_1 \partial x_n} & \cdots & \frac{\partial^2 f_m}{\partial^2 x_n} \end{pmatrix}. \tag{A.3}$$

Annexe B

Bootstrap

La méthode du bootstrap a été introduite en 1979 par Bradley Efron (Efron, 1979). Il s'agit d'une méthode de ré-échantillonnage, particulièrement adaptée aux cas où la taille de l'échantillon ne permet pas d'utiliser des résultats asymptotiques pour obtenir la loi d'un estimateur, mais également aux cas où l'on ne connaît pas la loi exacte d'un estimateur.

On suppose que l'on dispose d'un échantillon $\mathbb{X}$ de variables aléatoires i.i.d. $X_1, \ldots, X_n$, de fonction de répartition F inconnue. On se place dans le cadre paramétrique, c'est-à-dire que F appartient à une famille de lois connue, mais dépend d'un paramètre θ inconnu. Soit $T(\mathbb{X}) = T(X_1, \ldots, X_n) = \hat{\theta}$ un estimateur de θ.

B.1 Échantillons bootstrap

On construit un échantillon bootstrap en procédant à un tirage aléatoire avec remise de l'échantillon initial, et où la probabilité de tirer X_i vaut $\frac{1}{n}$ pour tout i. Cela revient à tirer les indices des observations qui appartiendront à l'échantillon bootstrap, c'est-à-dire à construire une suite d'entiers aléatoires $\{m_j, j = 1, \ldots, n\}$ i.i.d. indépendante de l'échantillon initial, et telle que $\mathbb{P}(m_j = i) = \frac{1}{n}, i = 1, \ldots, n, j = 1, \ldots, n$. Autrement dit, les $\{m_j\}$ suivent une loi multinomiale de paramètres $n = 1$ et $p = (\frac{1}{n}, \ldots, \frac{1}{n})$. Par exemple, supposons que $n = 4$. Si on tire l'échantillon $m_1 = 2$, $m_2 = 1$,$m_3 = 4$, $m_4 = 4$, on obtient l'échantillon bootstrap (X_2, X_1, X_4, X_4).

On construit ainsi B échantillons bootstrap, que l'on note $\mathbb{X}_1^* \ldots, \mathbb{X}_B^*$, avec $\mathbb{X}_b^* = (X_{b1}^*, \ldots, X_{bn}^*)$ (en pratique B est choisi suffisamment grand), et sur chacun de ces échantillons bootstrap, on calcule l'estimateur bootstrap de θ, $\hat{\theta}_b^* = T(\mathbb{X}_b^*) = T(X_{b1}^*, \ldots, X_{bn}^*)$, pour $b = 1, \ldots, B$. On dispose alors de B observations de l'estimateur de $\hat{\theta}^*$: $\hat{\theta}_1^*, \ldots, \hat{\theta}_B^*$.

B.2 Loi de l'estimateur bootstrap

Par construction des échantillons bootstrap, $\mathbb{X}^*$ a pour fonction de répartition la fonction de répartition empirique F_n de l'échantillon initial, définie par :

$$F_n(x) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}_{X_i \leq x}.$$

La loi de $\hat{\theta}^*$ dépend de F_n, donc elle est connue mais peut être difficile à calculer en pratique. Cependant on dispose de B observations de $\hat{\theta}^*$, donc on peut estimer cette loi. On obtient alors la fonction de répartition empirique de $\hat{\theta}^*$ calculée à partir des B échantillons bootstrap :

$$G^*_{n,B}(x) = \frac{1}{B}\sum_{b=1}^{B} \mathbf{1}_{\hat{\theta}^*_b \leq x}.$$

Grâce à la loi empirique $G^*_{n,B}$, et si cette loi est suffisamment proche de G_n, la loi de $\hat{\theta}$, on pourra construire par exemple des intervalles de confiance, ou des tests sur θ.

B.3 Exemples d'utilisations bootstrap

B.3.1 Estimateur bootstrap du biais

Soit θ un paramètre inconnu, et $\hat{\theta}$ un estimateur de θ. On s'intéresse au biais de l'estimateur, c'est-à-dire à la quantité

$$b(\hat{\theta}) = \mathbb{E}_F(\hat{\theta}) - \theta.$$

Cependant, on ne connaît pas la valeur de θ, et on ne connaît donc pas non plus F (qui dépend de θ). On définit alors l'estimateur boostrap du biais par :

$$\hat{b}^*(\hat{\theta}) = \frac{1}{B}\sum_{b=1}^{B} \hat{\theta}^*_b - \hat{\theta},$$

où $\hat{\theta}$ est l'estimateur obtenu sur l'échantillon initial.

B.3.2 Estimateur bootstrap de la variance

De la même façon, on peut proposer l'estimateur bootstrap de la variance :

$$\hat{\mathrm{Var}}^*(\hat{\theta}) = \frac{1}{B}\sum_{b=1}^{B}\left(\hat{\theta}^*_b - \frac{1}{B}\sum_{b=1}^{B}\hat{\theta}^*_b\right)^2.$$

B.3.3 Intervalle de confiance bootstrap percentile

Pour construire un intervalle de confiance pour θ à partir de l'estimateur $\hat{\theta}$, on utilise les quantiles de la loi de $\hat{\theta}$. Cette loi n'est pas connue en général. On rappelle que l'on dispose de B observations

de $\hat{\theta}^*$, notées $\hat{\theta}_1^*, \ldots, \hat{\theta}_B^*$. On note $\hat{\theta}_{(1)}^*, \ldots, \hat{\theta}_{(B)}^*$ les statistiques d'ordre associées (c'est-à-dire après avoir ordonné les valeurs $\hat{\theta}_1^*, \ldots, \hat{\theta}_B^*$ par ordre croissant : $\hat{\theta}_{(1)}^*$ est donc le minimum, et $\hat{\theta}_{(B)}^*$ le maximum).

On peut alors proposer l'intervalle de confiance bootstrap percentile de niveau $1 - \alpha$ suivant :

$$\hat{IC}_{\text{perc}}^*(\alpha) = \left[\hat{\theta}_{(\lceil B\frac{\alpha}{2}\rceil)}^*; \hat{\theta}_{(\lceil B(1-\frac{\alpha}{2})\rceil)}^*\right].$$

Autrement dit, l'intervalle de confiance bootstrap percentile est construit à partir des quantiles empiriques des $\hat{\theta}_1^*, \ldots, \hat{\theta}_B^*$.