Statistique en grande dimension

L. Rouvière

6 décembre 2022

Table des matières

Présentation

Ce tutoriel présente quelques exercices d'application du cours `Modèle linéaire en grande dimension`. On pourra trouver

- les supports de cours associés à ce tutoriel ainsi que les données utilisées à l'adresse suivante https://lrouviere.github.io/page_perso/grande_dim.html ;
- le tutoriel sans les corrections à l'url https://lrouviere.github.io/TUTO_GRANDE_DIM/
- le tutoriel avec les corrigés (à certains moment) à l'url https://lrouviere.github.io/TUTO_GRANDE_DIM/correction.

Il est recommandé d'utiliser **mozilla firefox** pour lire le tutoriel.

Des connaissances de base en R et en statistique (modèles de régression) sont nécessaires. Le tutoriel se structure en 4 parties :

- `Fléau de la dimension` : identification du problème de la dimension pour le problème de régression ;
- `Régression sur composantes` : présentation des algorithmes **PCR** et **PLS** ;
- `Régressions pénalisées`: régularisation à l'aide de pénalités de type **Ridge/Lasso**
- `Modèle additif` : conservation de la structure additive du modèle linéaire mais modélisation non paramétrique des composantes.

partie I

Supervisée

1 Les problèmes de la grande dimension

Nous proposons ici d'illustrer le problème de la grande dimension en régression. On commencera par étudier, à l'aide de simulation, ce problème pour l'estimateur des k plus proches voisins, puis pour les estimateurs des moindres carrés dans le modèle linéaire. Quelques exercices sont ensuite proposées pour calculer les vitesses de convergence de ces estimateurs dans des modèles simples.

1.1 Fléau de la dimension pour les plus proches voisins

La fonction suivante permet de générer un échantillon d'apprentissage et un échantillon test selon le modèle

$$Y = X_1^2 + \cdots + X_p^2 + \varepsilon$$

où les X_j sont uniformes i.i.d de loi uniforme sur $[0, 1]$ et le bruit ε suit une loi $\mathcal{N}(0, 0.5^2)$.

```
simu <- function(napp=300,ntest=500,p=3,graine=1234){
  set.seed(graine)
  n <- napp+ntest
  X <- matrix(runif(n*p),ncol=p)
  Y <- apply(X^2,1,sum)+rnorm(n,sd=0.5)
  Yapp <- Y[1:napp]
  Ytest <- Y[-(1:napp)]
  Xapp <- data.frame(X[1:napp,])
  Xtest <- data.frame(X[-(1:napp),])
  return(list(Xapp=Xapp,Yapp=Yapp,Xtest=Xtest,Ytest=Ytest))
}
df <- simu(napp=300,ntest=500,p=3,graine=1234)
```

La fonction **knn.reg** du package **FNN** permet de construire des estimateurs des k plus proches voisins en régression. On peut par exemple faire du 3 plus proches voisins avec

```
library(FNN)
mod3ppv <- knn.reg(train=df$Xapp,y=df$Yapp,k=3)
```

Parmi toutes les sorties proposées par cette fonction on a notamment

```
mod3ppv$PRESS
```

```
[1] 98.98178
```

qui renvoie la somme des carrés des erreurs de prévision par validation croisée Leave-One-Out (LOO). On peut ainsi obtenir l'erreur quadratique moyenne par LOO

```
mod3ppv$PRESS/max(c(nrow(df$Xapp),1))
```

```
[1] 0.3299393
```

1. Construire la fonction **sel.k** qui admet en entrée :

   - une grille de valeurs possibles de plus proches voisins (un vecteur).
   - une matrice **Xapp** de dimension $n \times p$ qui contient les valeurs variables explicatives.
   - un vecteur **Yapp** de dimension n qui contient les valeurs de la variable à expliquer

   et qui renvoie en sortie la valeur de k dans la grille qui minimise l'erreur LOO présentée ci-dessus.

   Une fois la fonction créée, on peut calculer l'erreur de l'estimateur sélectionné sur un échantillon test avec

   ```
   k.opt <- sel.k(seq(1,50,by=5),df$Xapp,df$Yapp)
   k.opt
   prev <- knn.reg(train=df$Xapp,y=df$Yapp,test=df$Xtest,k=k.opt)$pred
   mean((prev-df$Ytest)^2)
   ```

2. On souhaite comparer les erreurs des règles des k plus proches voisins en fonction de la dimension. On considère 4 dimensions collectées dans le vecteur `DIM` et la grille de valeurs de k suivantes :

   ```
   DIM <- c(1,5,10,50)
   K_cand <- seq(1,50,by=5)
   ```

   Pour chaque valeur de dimension répéter $B = 100$ fois :

   - simuler un échantillon d'apprentissage de taille 300 et test de taille 500
   - calculer la valeur optimale de k dans **K_cand** grâce à **sel.k**
   - calculer l'erreur de l'estimateur sélectionné sur un échantillon test.

On pourra stocker les résultats dans une matrice de dimension $B \times 4$.

3. A l'aide d'indicateurs numériques et de boxplots, comparer la distribution des erreurs en fonction de la dimension.
4. Conclure

1.2 Influence de la dimension dans le modèle linéaire

En vous basant sur l'exercice précédent, proposer une illustration qui peut mettre en évidence la précision d'estimation dans le modèle linéaire en fonction de la dimension. On pourra par exemple considérer le modèle linaire suivant

$$Y = X_1 + 0X_2 + \cdots + 0X_p + \varepsilon$$

et étudier la performance de l'estimateur MCO du coefficient de X_1 pour différentes valeurs de p. Par exemple avec p dans le vecteur

```
DIM <- c(0,50,100,200)
```

Les données pourront être générées avec la fonction suivante

```
n <- 250
p <- 1000
X <- matrix(runif(n*p),ncol=p)
simu.lin <- function(X,graine){
  set.seed(graine)
  Y <- X[,1]+rnorm(nrow(X),sd=0.5)
  df <- data.frame(Y,X)
  return(df)
}
```

1.3 Exercices

Exercice 1.1 (Distances entre deux points)**.** Cet exercice est fortement inspiré de Giraud (2015). Soit $X^{(1)} = (X_1^{(1)}, \ldots, X_p^{(1)})$ et $X^{(2)} = (X_1^{(2)}, \ldots, X_p^{(2)})$ deux variables aléatoires indépendantes de loi uniforme sur l'hypercube $[0,1]^p$. Montrer que

$$\mathbf{E}[\|X^{(1)} - X^{(2)}\|^2] = \frac{p}{6} \quad \text{et} \quad \sigma[\|X^{(1)} - X^{(2)}\|^2] \approx 0.2\sqrt{p}.$$

Exercice 1.2 (Vitesse de convergence pour l'estimateur à noyau)**.** On considère le modèle de régression

$$Y_i = m(x_i) + \varepsilon_i, \quad i = 1, \dots, n$$

où $x_1, \dots, x_n \in \mathbb{R}^d$ sont déterministes et $\varepsilon_1, \dots, \varepsilon_n$ sont des variables i.i.d. d'espérance nulle et de variance $\sigma^2 < +\infty$. On désigne par $\|\,.\,\|$ la norme Euclidienne dans $\mathbb{R}^d$. On définit l'estimateur localement constant de m en $x \in \mathbb{R}^d$ par :

$$\widehat{m}(x) = \operatorname*{argmin}_{a \in \mathbb{R}} \sum_{i=1}^{n} (Y_i - a)^2 K\left(\frac{\|x_i - x\|}{h}\right)$$

où $h > 0$ et pour $u \in \mathbb{R}, K(u) = \mathbf{1}_{[0,1]}(u)$. On suppose que $\sum_{i=1}^n K\left(\frac{\|x_i - x\|}{h}\right) > 0$.

1. Donner la forme explicite de $\widehat{m}(x)$.
2. Montrer que
$$\mathbf{V}[\widehat{m}(x)] = \frac{\sigma^2}{\sum_{i=1}^n K\left(\frac{\|x_i - x\|}{h}\right)}$$
et
$$\mathbf{E}[\widehat{m}(x)] - m(x) = \frac{\sum_{i=1}^n (m(x_i) - m(x)) K\left(\frac{\|x_i - x\|}{h}\right)}{\sum_{i=1}^n K\left(\frac{\|x_i - x\|}{h}\right)}.$$
3. On suppose maintenant que m est Lipschitzienne de constante L, c'est-à-dire que $\forall (x_1, x_2) \in \mathbb{R}^d \times \mathbb{R}^d$
$$|m(x_1) - m(x_2)| \leq L\|x_1 - x_2\|.$$
Montrer que
$$|\text{biais}[\widehat{m}(x)]| \leq Lh.$$
4. On suppose de plus qu'il existe une constante C_1 telle que
$$C_1 \leq \frac{\sum_{i=1}^n \mathbf{1}_{B_h}(x_i - x)}{n\text{Vol}(B_h)},$$
où $B_h = \{u \in \mathbb{R}^d : \|u\| \leq h\}$ est la boule de rayon h dans $\mathbb{R}^d$ et $\text{Vol}(A)$ désigne le volume d'un ensemble $A \subset \mathbb{R}^d$. Montrer que
$$\mathbf{V}[\widehat{m}(x)] \leq \frac{C_2 \sigma^2}{nh^d},$$
où C_2 est une constante dépendant de C_1 et d à préciser.
5. Déduire des questions précédentes un majorant de l'erreur quadratique moyenne de $\widehat{m}(x)$.
6. Calculer h_{opt} la valeur de h qui minimise ce majorant. Que vaut ce majorant lorsque $h = h_{\text{opt}}$? Comment varie cette vitesse lorsque d augmente ? Interpréter.

2 Régression sur composantes

Les performances des estimateurs classiques (MCO) des paramètres du modèle linéaire

$$Y = \beta_0 + \beta_1 X_1 + \cdots + \beta_d X_d + \varepsilon$$

peuvent se dégrader lorsque la dimension d est grande ou en présence de dépendance linéaire entre les variables explicatives. Les régressions sur composantes consistent à trouver de nouvelles composantes $Z_k, j = k, \ldots, q$ avec $q \leq p$ qui s'écrivent le plus souvent comme des combinaisons linéaires des X_j dans l'idée de diminuer le nombre de paramètres du modèle ou la dépendance entre les covariables. Il existe plusieurs façons de construire ces composantes, dans cette partie nous proposons :

- la **régression sous composantes principales (PCR)** : il s'agit de faire simplement une ACP sur la matrice des variables explicatives ;
- la **régression partial least square (PLS)** qui fait intervenir la variable cible dans la construction des composantes.

Nous commençons par un bref rappel sur la sélection de variables.

2.1 Sélection de variables

On considère le jeu de données `ozone.txt` où on cherche à expliquer la concentration maximale en ozone relevée sur une journée (variable `maxO3`) par d'autres variables essentiellement météorologiques.

```
ozone <- read.table("data/ozone.txt")
head(ozone)
```

```
         maxO3   T9  T12  T15 Ne9 Ne12 Ne15     Vx9    Vx12    Vx15 maxO3v
20010601    87 15.6 18.5 18.4   4    4    8  0.6946 -1.7101 -0.6946     84
20010602    82 17.0 18.4 17.7   5    5    7 -4.3301 -4.0000 -3.0000     87
20010603    92 15.3 17.6 19.5   2    5    4  2.9544  1.8794  0.5209     82
20010604   114 16.2 19.7 22.5   1    1    0  0.9848  0.3473 -0.1736     92
20010605    94 17.4 20.5 20.4   8    8    7 -0.5000 -2.9544 -4.3301    114
```

```
20010606    80 17.7 19.8 18.3   6    6    7 -5.6382 -5.0000 -6.0000     94
          vent pluie
20010601  Nord   Sec
20010602  Nord   Sec
20010603   Est   Sec
20010604  Nord   Sec
20010605 Ouest   Sec
20010606 Ouest Pluie
```

1. Ajuster un modèle linéaire avec `lm` et analyser la pertinence des variables explicatives dans le modèle.
2. Expliquer les sorties de la commande

```r
library(leaps)
mod.sel <- regsubsets(maxO3~.,data=ozone,nvmax=14)
summary(mod.sel)
```

```
Subset selection object
Call: regsubsets.formula(maxO3 ~ ., data = ozone, nvmax = 14)
14 Variables  (and intercept)
          Forced in Forced out
T9            FALSE      FALSE
T12           FALSE      FALSE
T15           FALSE      FALSE
Ne9           FALSE      FALSE
Ne12          FALSE      FALSE
Ne15          FALSE      FALSE
Vx9           FALSE      FALSE
Vx12          FALSE      FALSE
Vx15          FALSE      FALSE
maxO3v        FALSE      FALSE
ventNord      FALSE      FALSE
ventOuest     FALSE      FALSE
ventSud       FALSE      FALSE
pluieSec      FALSE      FALSE
1 subsets of each size up to 14
Selection Algorithm: exhaustive
          T9  T12 T15 Ne9 Ne12 Ne15 Vx9 Vx12 Vx15 maxO3v ventNord ventOuest
1  ( 1 )  " " "*" " " " " " " " "  " "  " " " " "  " "   " "     " "       " "
2  ( 1 )  " " "*" " " " " " " " "  " "  " " " " "  " "   "*"     " "       " "
3  ( 1 )  " " "*" " " "*" " " "  " "  " " " " "  " "   "*"     " "       " "
4  ( 1 )  " " "*" " " "*" " " "  " "  "*" " " "  " "   "*"     " "       " "
```

```
5  ( 1 )  " " "*" " " "*" " "  " "  "*" " "  " "  "*"    " "      " "
6  ( 1 )  " " "*" " " "*" " "  " "  "*" " "  "*"  "*"    " "      " "
7  ( 1 )  " " "*" " " "*" " "  " "  "*" " "  "*"  "*"    "*"      " "
8  ( 1 )  " " "*" " " "*" " "  " "  "*" " "  "*"  "*"    " "      "*"
9  ( 1 )  " " "*" " " "*" "*"  " "  "*" " "  "*"  "*"    " "      "*"
10  ( 1 ) " " "*" "*" "*" "*"  " "  "*" " "  "*"  "*"    " "      "*"
11  ( 1 ) " " "*" "*" "*" "*"  " "  "*" "*"  "*"  "*"    " "      "*"
12  ( 1 ) " " "*" "*" "*" "*"  " "  "*" "*"  "*"  "*"    "*"      "*"
13  ( 1 ) "*" "*" "*" "*" "*"  " "  "*" "*"  "*"  "*"    "*"      "*"
14  ( 1 ) "*" "*" "*" "*" "*"  "*"  "*" "*"  "*"  "*"    "*"      "*"
          ventSud pluieSec
1  ( 1 )  " "     " "
2  ( 1 )  " "     " "
3  ( 1 )  " "     " "
4  ( 1 )  " "     " "
5  ( 1 )  " "     "*"
6  ( 1 )  " "     "*"
7  ( 1 )  " "     "*"
8  ( 1 )  "*"     "*"
9  ( 1 )  "*"     "*"
10  ( 1 ) "*"     "*"
11  ( 1 ) "*"     "*"
12  ( 1 ) "*"     "*"
13  ( 1 ) "*"     "*"
14  ( 1 ) "*"     "*"
```

3. Sélectionner le meilleur modèle au sens du R^2. Que remarquez-vous ?
4. Faire de même pour le C_p et le BIC. Que remarquez-vous pour les variables explicatives qualitatives ?
5. Comparer cette méthode avec des modèles sélectionnées par la fonction `step` ou la fonction `bestglm` du package `bestglm`.

2.2 Régression sur composantes principales (méthodo)

L'algorithme **PCP** est une méthode de réduction de dimension, elle consiste à faire un modèle linéaire **MCO** sur les premiers axes de l'**ACP**. On désigne par

- $\mathbb{X}$ la matrice qui contient les valeurs des variables explicatives que l'on suppose centrée réduite.

- $Z_1, \dots, Z_p$ les axes de l'ACP qui s'écrivent comme des combinaisons linéaires des variables explicatives : $Z_j = w_j^t X$.

L'algorithme **PCR** consiste à choisir un nombre de composantes m et à faire une régression MCO sur les m premiers axes de l'ACP :

$$Y = \alpha_0 + \alpha_1 Z_1 + \cdots + \alpha_m Z_m + \varepsilon.$$

Si on désigne par

- $x \in \mathbb{R}^d$ une nouvelle observation que l'on a centrée réduite également;
- $z_1, \dots, z_M$ les coordonnées de x dans la base définie par les m premiers axes de l'ACP ($z_j = w_j^t x$)

l'algorithme **PCR** reverra la prévision

$$\widehat{m}_{\text{PCR}}(x) = \hat{\alpha}_0 + \hat{\alpha}_1 z_1 + \cdots + \hat{\alpha}_m z_m.$$

Cette prévision peut s'écrire également comme une combinaison linéaire des variables explicatives (centrées réduites ou non) :

$$\widehat{m}_{\text{PCR}}(x) = \hat{\gamma}_0 + \hat{\gamma}_1 \tilde{x}_1 + \cdots + \hat{\gamma}_p \tilde{x}_p = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \cdots + \hat{\beta}_p x_p,$$

$\tilde{x}_j$ désignant l'observation brute (non centrée réduite).

L'exercice suivant revient sur cet algorithme et notamment sur le lien entre ces différents paramètres.

Exercice 2.1 (Régression PCR avec R)**.** On considère le jeu de données **Hitters** dans lequel on souhaite expliquer la variable **Salary** par les autres variables du jeu de données. Pour simplifier le problème, on supprime les individus qui possèdent des données manquantes (il ne faut pas faire ça normalement !).

```
library(ISLR)
Hitters <- na.omit(Hitters)
```

1. Parmi les variables explicatives, certaines sont qualitatives. Expliquer comment, à l'aide de la fonction **model.matrix** on peut utiliser ces variables dans un modèle linéaire. On appellera **X** la matrice des variables explicatives construites avec cette variable.
2. Calculer la matrice **Xcr** qui correspond à la matrice **X** centrée réduite. On pourra utiliser la fonction `scale`.

3. A l'aide de la fonction `PCA` du package **FactoMineR**, effectuer l'ACP du tableau **Xcr** avec l'option `scale.unit=FALSE`.
4. Récupérer les coordonnées des individus sur les 5 premiers axes de l'ACP (variables Z dans le cours).
5. Effectuer la régression linéaire sur les 5 premières composantes principales et calculer les estimateurs des MCO ($\widehat{\alpha}_k, k = 1, \dots, 5$ dans le cours).
6. En déduire les estimateurs dans l'espace des données initiales pour les données centrées réduites, puis pour les données brutes. On pourra récupérer les vecteurs propres de l'ACP (w_k dans le cours) dans la sortie **svd** de la fonction **PCA**.
7. Retrouver les estimateurs dans l'espace des données initiales pour les données centrées réduites à l'aide de la fonction `pcr` du package **pls**.
8. On considère les individus suivants

   ```
   df.new <- Hitters[c(1,100,80),]
   ```

   Calculer de 3 façons différentes les valeurs de salaire prédites par la régression sur 5 composantes principales.

Exercice 2.2 (Composantes PCR)**.** On rappelle que les poids w_k des composantes principales s'obtiennent en résolvant le problème :

$$\max_{w \in \mathbb{R}^d} \mathbf{V}(\mathbb{X}w)$$

$$\text{sous les contraintes } \|w\| = 1, w^t \mathbb{X}^t \mathbb{X} w_\ell = 0, \ell = 1, \dots, k-1.$$

1. Montrer w_1 est un vecteur propre associé à la plus grande valeur propre de $\mathbb{X}^t\mathbb{X}$.
2. Calculer w_2.

2.3 Régression PLS : méthodo

La régression **PLS** propose de construire également de nouvelles composantes comme des combinaisons linéaires des variables explicatives. Comme pour l'algorithme **PCR**, les composantes sont calculées les unes après les autres et orthogonales entre elles. La principale différence et qu'on ne cherche pas les composantes qui maximisent la variabilités des observations projetées, mais les composantes qui maximisent la colinéarité avec la cible. L'algorithme est expliqué dans l'exercice suivant.

Exercice 2.3 (Calcul des composantes PLS)**.** On reprend les notations du cours : $\mathbb{Y}$ désigne le vecteur de la variable à expliquer et $\mathbb{X}$ la matrice qui contient les observations des variables explicatives. On la suppose toujours centrée réduite.

1. On pose $\mathbb{Y}^{(1)} = \mathbb{Y}$ et $\mathbb{X}^{(1)} = \mathbb{X}$. On cherche $Z_1 = w_1^t X^{(1)}$ qui maximise
$$\langle \mathbb{X}^{(1)} w_1, \mathbb{Y}^{(1)} \rangle \quad \text{sous la contrainte} \quad \|w\|^2 = 1.$$
Cela revient à cherche la combinaison linéaire des colonnes de $\mathbb{X}^{(1)}$ la plus corrélée à $\mathbb{Y}^{(1)}$. Calculer cette première composante.

2. On pose $Z_1 = w_1^t X^{(1)}$ et $\mathbb{Z}_1 = \mathbb{X}^{(1)} w_1$. On considère le modèle de régression linéaire
$$Y^{(1)} = \alpha_0 + \alpha_1 Z_1 + \varepsilon.$$
Exprimer les estimateurs MCO de $\alpha = (\alpha_0, \alpha_1)$ en fonction de $\mathbb{Z}^{(1)}$ et $\mathbb{Y}^{(1)}$.

3. On passe maintenant à la deuxième composante. On cherche à expliquer la partie résiduelle
$$\mathbb{Y}^{(2)} = P_{Z_1^\perp}(\mathbb{Y}^{(1)}) = \hat{\varepsilon}_1 = \mathbb{Y}^{(1)} - \hat{\mathbb{Y}}^{(1)}$$
par la "meilleure" combinaison linéaire orthogonale à Z_1. On orthogonalise chaque $\tilde{\mathbb{X}}_j^{(1)}$ par rapport à $\mathbb{Z}_1$:
$$\mathbb{X}_j^{(2)} = P_{\mathbb{Z}_1^\perp}(\mathbb{X}_j^{(1)}) = (\mathrm{Id} - P_{\mathbb{Z}_1})(\mathbb{X}_j^{(1)}) = \mathbb{X}_j^{(1)} - \frac{\langle \mathbb{Z}_1, \mathbb{X}_j^{(1)} \rangle}{\langle \mathbb{Z}_1, \mathbb{Z}_1 \rangle} \mathbb{Z}_1.$$
et on déduit w_2 comme w_1 : $w_2 = \tilde{\mathbb{X}}^{(2)'} \mathbb{Y}^{(2)}$. On considère ensuite le modèle $Y^{(2)} = \alpha_2 Z_2 + \varepsilon$. Exprimer l'estimateur des MCO de α_2 en fonction de $\mathbb{Z}_2 = \mathbb{X}^{(2)} w_2$ et $\mathbb{Y}$.

Exercice 2.4 (Régression PLS sur R)**.** On considère les mêmes données que précédemment.

1. A l'aide du vecteur $\mathbb{Y}$ (*Salary*) et de la matrice des $\mathbb{X}$ centrées réduites calculées dans l'Exercice 2.1, calculer la première composante **PLS** $\mathbb{Z}_1$.

2. En déduire le coefficient associé à cette première composante en considérant le modèle
$$Y = \alpha_1 Z_1 + \varepsilon.$$

3. En déduire les coefficients en fonction des variables initiales (centrées réduites) de la régression PLS à une composante
$$Y = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p + \varepsilon.$$

4. Retrouver ces coefficients en utilisant la fonction `plsr`.

2.4 Comparaison : PCR vs PLS.

1. Séparer le jeu de données (`Hitters` toujours) en un échantillon d'apprentissage de taille 200 et un échantillon test de taille 63.
2. Avec les données d'apprentissage uniquement construire les régressions PCR et PLS. On choisira les nombres de composantes par validation croisée.
3. Comparer les deux méthodes en utilisant l'échantillon de validation. On pourra également utiliser un modèle linéaire classique.
4. Comparer ces méthodes à l'aide d'une validation croisée 10 blocs.

3 Régressions pénalisées (ou sous contraintes)

Nous considérons toujours le modèle linéaire

$$Y = \beta_0 + \beta_1 X_1 + \cdots + \beta_d X_d + \varepsilon$$

Lorsque d est grand ou que les variables sont linéairement dépendantes, les estimateurs des moindres carrées peuvent être mis en défaut. Les méthodes pénalisées ou sous contraintes consistent alors à restreindre l'espace sur lequel on minimise ce critère. On va alors chercher le vecteur β qui minimise

$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{d} x_{ij}\beta_j\right)^2 \quad \text{sous la contrainte} \quad \sum_{j=1}^{d}\beta_j^2 \leq t$$

ou de façon équivalente (dans le sens où il existe une équivalence entre t et λ)

$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{d} x_{ij}\beta_j\right)^2 + \lambda\sum_{j=1}^{d}\beta_j^2.$$

Les estimateurs obtenus sont les estimateurs **ridge**. Les estimateurs **lasso** s'obtiennent en remplaçant la contrainte ou la pénalité par une norme 1 ($\sum_{j=1}^{d}|\beta_j|$). Nous présentons dans cette partie les étapes principales qui permettent de faire ce type de régression avec **R**. Le package le plus souvent utilisé est `glmnet`.

3.1 Ridge et lasso avec glmnet

On considère le jeu de données `ozone.txt` où on cherche à expliquer la concentration maximale en ozone relevée sur une journée (variable `maxO3`) par d'autres variables essentiellement météorologiques.

```
ozone <- read.table("data/ozone.txt")
head(ozone)
```

```
         maxO3   T9  T12  T15 Ne9 Ne12 Ne15     Vx9    Vx12    Vx15 maxO3v
20010601    87 15.6 18.5 18.4   4    4    8  0.6946 -1.7101 -0.6946     84
20010602    82 17.0 18.4 17.7   5    5    7 -4.3301 -4.0000 -3.0000     87
20010603    92 15.3 17.6 19.5   2    5    4  2.9544  1.8794  0.5209     82
20010604   114 16.2 19.7 22.5   1    1    0  0.9848  0.3473 -0.1736     92
20010605    94 17.4 20.5 20.4   8    8    7 -0.5000 -2.9544 -4.3301    114
20010606    80 17.7 19.8 18.3   6    6    7 -5.6382 -5.0000 -6.0000     94
          vent pluie
20010601  Nord   Sec
20010602  Nord   Sec
20010603   Est   Sec
20010604  Nord   Sec
20010605 Ouest   Sec
20010606 Ouest Pluie
```

Contrairement à la plupart des autres package **R** qui permettent de faire de l'apprentissage, le package `glmnet` n'autorise pas l'utilisation de `formules` : il faut spécifier explicitement la matrice des X et le vecteur des Y. On peut obtenir la matrice des X et notamment le codage des variables qualitatives avec la fonction `model.matrix`:

```
ozone.X <- model.matrix(maxO3~.,data=ozone)[,-1]
ozone.Y <- ozone$maxO3
```

1. Charger le package `glmnet` et à l'aide de la fonction `glmnet` calculer les estimateurs `ridge` et `lasso`.
2. Analyser les sorties qui se trouvent dans les arguments `lambda` et `beta` de `glmnet`.
3. Visualiser les chemins de régularisation des estimateurs `ridge` et `lasso`. On pourra utiliser la fonction `plot`.
4. Sélectionner les paramètres de régularisation à l'aide de la fonction `cv.glmnet`. On pourra notamment faire un `plot` de l'objet et expliquer le graphe obtenu.
5. On souhaite prédire la variable cible pour de nouveaux individus, par exemple les 25ème et 50ème individus du jeu de données. Calculer les valeurs prédites pour ces deux individus.
6. A l'aide d'une validation croisée, comparer les performances des estimateurs **MCO**, **ridge** et **lasso**. On pourra utiliser les données `ozone_complet.txt` qui contiennent plus d'individus et de variables.

   ```
   ozone1 <- read.table("data/ozone_complet.txt",sep=";") |> na.omit()
   ozone1.X <- model.matrix(maxO3~.,data=ozone1)[,-1]
   ```

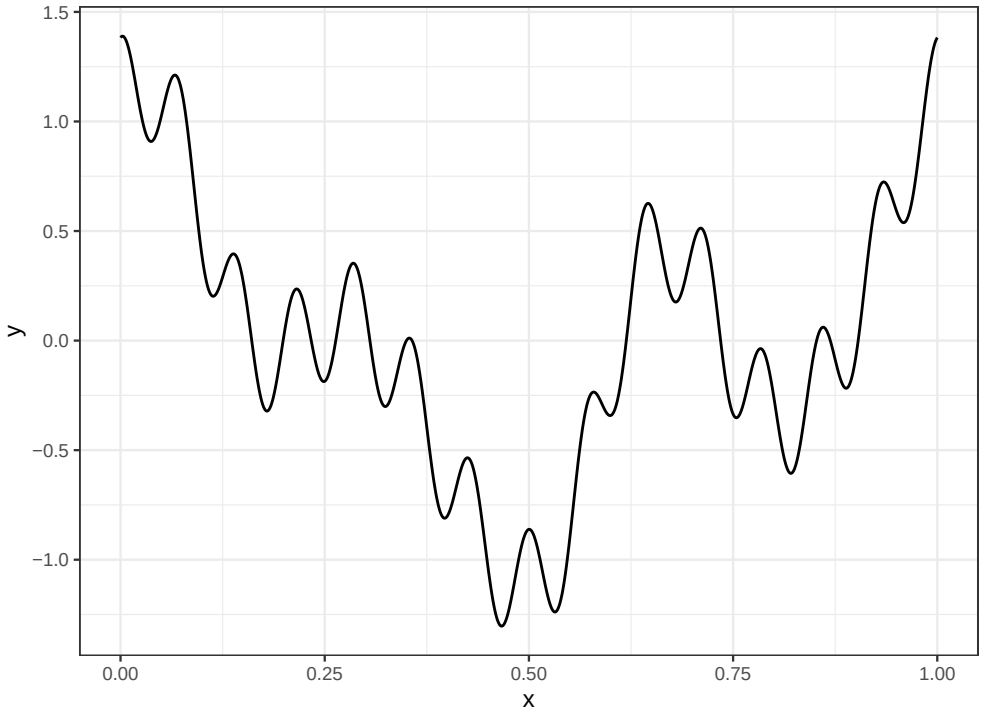
```
ozone1.Y <- ozone1$maxO3
```

7. Refaire la question précédente en considérant toutes les interactions d'ordre 2.

3.2 Reconstruction d'un signal

Le fichier `signal.csv` contient un signal que l'on peut représenter par une fonction $m : \mathbb{R} \to \mathbb{R}$. On le visualise

```
signal <- read_csv("data/signal.csv")
ggplot(signal)+aes(x=x,y=y)+geom_line()
```

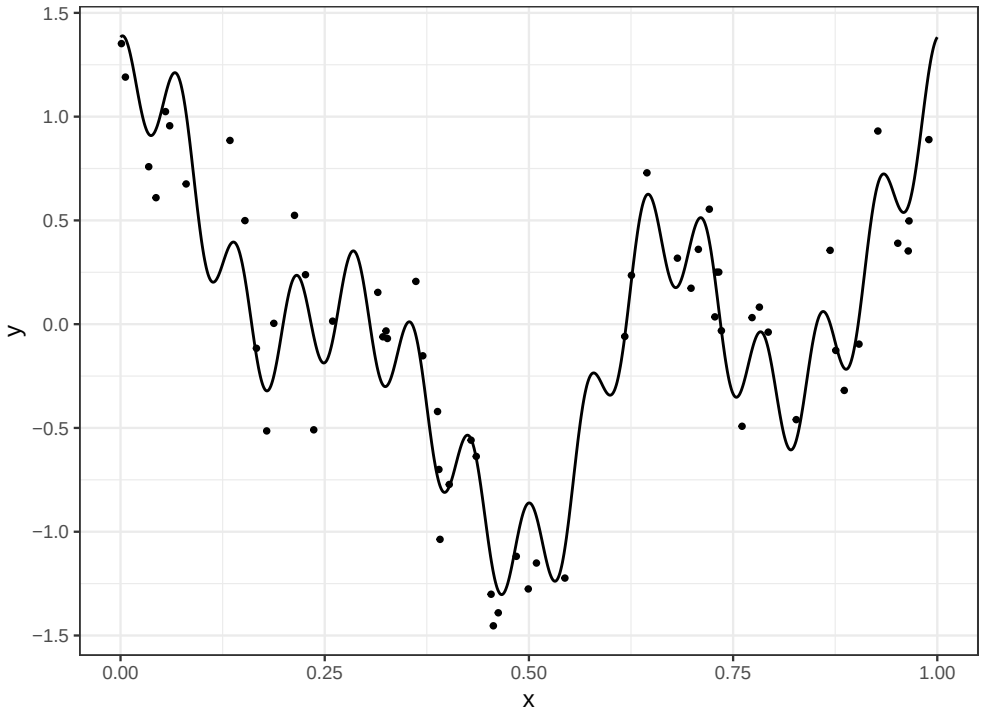


Plaçons nous dans le cas où on ne dispose que d'une version bruitée de ce signal. La courbe n'est pas observée mais on dispose d'un échantillon $(x_i, y_i), i = 1, \dots, n$ généré selon le modèle

$$y_i = m(x_i) + \varepsilon_i.$$

Le fichier `ech_signal.csv` contient $n = 60$ observations issues de ce modèle. On représente les données et la courbe

```r
donnees <- read_csv("data/ech_signal.csv")
ggplot(signal)+aes(x=x,y=y)+geom_line()+
  geom_point(data=donnees,aes(x=X,y=Y))
```



Nous cherchons dans cette partie à reconstruire le signal à partir de l'échantillon. Bien entendu, vu la forme du signal, un modèle linéaire de la forme

$$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i$$

n'est pas approprié. De nombreuses approches en **traitement du signal** proposent d'utiliser une `base` ou `dictionnaire` représentée par une collection de fonctions $\{\psi_j(x)\}_{j=1,\dots,K}$ et de décomposer le signal dans cette base :

$$m(x) \approx \sum_{j=1}^{K} \beta_j \psi_j(x).$$

Pour un dictionnaire donné, on peut alors considérer un **modèle linéaire**

$$y_i = \sum_{j=1}^{K} \beta_j \psi_j(x_i) + \varepsilon_i. \tag{3.1}$$

Le problème est toujours d'estimer les paramètres β_j mais les variables sont maintenant définies par les élements du dictionnaire. Il existe différents types de dictionnaire, dans cet exercice nous proposons de considérer la base de Fourier définie par

$$\psi_0(x) = 1, \quad \psi_{2j-1}(x) = \cos(2j\pi x) \quad \text{et} \quad \psi_{2j}(x) = \sin(2j\pi x), \quad j = 1, \dots, K.$$

1. Écrire une fonction **R** qui admet en entrée :

   - une grille de valeurs de `x` (un vecteur)
   - une valeur de `K` (un entier positif)

   et qui renvoie en sortie une matrice qui contiennent les valeurs du dictionnaire pour chaque valeur de `x`. Cette matrice devra donc contenir `2K` colonnes et le nombre de lignes sera égal à la longueur du vecteur `x`.

2. On fixe `K=25`. Calculer les estimateurs des moindres carrés du modèle (Équation 3.1).

3. Représenter le signal estimé. Commenter le graphe.

4. Calculer les estimateurs **lasso** et représenter le signal issu de ces estimateurs.

5. Identifier les coefficients lasso sélectionnés qui ne sont pas nuls.

6. Ajouter les signaux ajustés par les algorithme PCR et PLS.

3.3 Régression logistique pénalisée

On considère le jeu de données sur la détection d'images publicitaires disponible ici https://archive.ics.uci.edu/ml/datasets/internet+advertisements.

```
ad.data <- read.table("data/ad_data.txt",header=FALSE,sep=",",dec=".",
                      na.strings = "?",strip.white = TRUE)
names(ad.data)[ncol(ad.data)] <- "Y"
ad.data$Y <- as.factor(ad.data$Y)
```

La variable à expliquer est

```
summary(ad.data$Y)
```

```
ad. nonad.
459   2820
```

Cette variable est binaire. On considère une régression logistique pour expliquer cette variable. Le nombre de variables explicatives étant important, comparer les algorithmes du maximum de vraisemblance aux algorithmes de type **ridge/lasso** en faisant une validation croisée 10 blocs. On pourra utiliser comme critère de comparaison l'`erreur de classification`, la `courbe ROC` et l'`AUC`. Il faudra également prendre des décisions pertinentes vis-à-vis des données manquantes…

3.4 Exercices

Exercice 3.1 (Estimateurs ridge pour le modèle linéaire)**.** On considère le modèle de régression

$$Y_i = \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \varepsilon_i$$

où les ε_i sont i.i.d de loi $\mathcal{N}(0, \sigma^2)$. Pour $\lambda \geq 0$, on note $\widehat{\beta}_R(\lambda)$ l'estimateur ridge défini par

$$\widehat{\beta}_R(\lambda) = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^{n} \left(y_i - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2.$$

1. Exprimer $\widehat{\beta}_R(\lambda)$ en fonction de $\mathbb{X}$, $\mathbb{Y}$ et λ.
2. Étudier le biais et la variance de $\widehat{\beta}_R(\lambda)$ en fonction de λ. On pourra également faire la comparaison avec l'estimateur des MCO.
3. On suppose que la matrice $\mathbb{X}$ est orthogonale. Exprimer les estimateurs $\widehat{\beta}_{R,j}(\lambda)$ en fonction des estimateurs des MCO $\widehat{\beta}_j, j = 1, \ldots, p$. Interpréter.

Exercice 3.2 (Estimateurs lasso dans le cas orthogonal)**.** Cet exercice est inspiré de Giraud (2015). On rappelle qu'une fonction $F : \mathbb{R}^n \to \mathbb{R}$ est convexe si $\forall x, y \in \mathbb{R}^n$, $\forall \lambda \in [0, 1]$ on a

$$F(\lambda x + (1 - \lambda)y) \leq \lambda F(x) + (1 - \lambda)F(y).$$

On définit la sous-différentielle d'une fonction convexe F par

$$\partial F(x) = \{w \in \mathbb{R}^n : F(y) \geq F(x) + \langle w, y - x \rangle \text{ pour tout } y \in \mathbb{R}^n\}.$$

On admettra que les minima d'une fonction convexe $F : \mathbb{R}^n \to \mathbb{R}$ sont caractérisés par

$$x^\star \in \underset{x \in \mathbb{R}^n}{\operatorname{argmin}} F(x) \iff 0 \in \partial F(x^\star)$$

et que $\partial F(x) = \{\nabla F(x)\}$ lorsque F est différentiable en x.

1. Montrer que pour $x \in \mathbb{R}$
$$\partial|x| = \begin{cases} \text{signe}(x) & \text{si } x \neq 0 \\ [-1;1] & \text{sinon,} \end{cases}$$
où $\text{signe}(x) = \mathbf{1}_{x>0} - \mathbf{1}_{x\leq 0}$.

2. Soit $x \in \mathbb{R}^n$.

   a. Montrer que
   $$\partial\|x\|_1 = \{w \in \mathbb{R}^n : \langle w, x\rangle = \|x\|_1 \text{ et } \|w\|_\infty \leq 1\}.$$
   On pourra utiliser que pour tout p, q tels que $1/p + 1/q = 1$ on a
   $$\|x\|_p = \sup\left\{\langle w, x\rangle : \|w\|_q \leq 1\right\}.$$

   b. En déduire
   $$\partial\|x\|_1 = \{w \in \mathbb{R}^n : w_j = \text{signe}(x_j) \text{ si } x_j \neq 0, w_j \in [-1, 1] \text{ si } x_j = 0\}.$$

3. Étant données n observations $(x_i, y_i), i = 1, \dots, n$ telles que $x_i \in \mathbb{R}^p$ et $y_i \in \mathbb{R}$ on rappelle que l'estimateur lasso $\hat{\beta}(\lambda)$ est construit en minimisant
$$\mathcal{L}(\beta) = \|Y - \mathbb{X}\beta\|_2^2 + \lambda\|\beta\|_1. \tag{3.2}$$
On admettra que la sous-différentielle $\partial\mathcal{L}(\beta)$ est donnée par
$$\partial\mathcal{L}(\beta) = \left\{-2\mathbb{X}^t(Y - \mathbb{X}\beta) + \lambda z : z \in \partial\|\beta\|_1\right\}.$$
Montrer que $\hat{\beta}(\lambda)$ vérifie
$$\mathbb{X}^t\mathbb{X}\hat{\beta}(\lambda) = \mathbb{X}^t Y - \frac{\lambda}{2}\hat{z}$$
où $\hat{z} \in \mathbb{R}^p$ vérifie
$$\hat{z}_j \begin{cases} = \text{signe}(\hat{\beta}_j(\lambda)) & \text{si } \hat{\beta}_j(\lambda) \neq 0 \\ \in [-1;1] & \text{sinon.} \end{cases}$$

4. On suppose maintenant que la matrice $\mathbb{X}$ est orthogonale.

   a. Montrer que
   $$\text{signe}(\hat{\beta}_j(\lambda)) = \text{signe}(\mathbb{X}_j^t Y) \quad \text{lorsque } \hat{\beta}_j(\lambda) \neq 0$$
   et $\hat{\beta}_j(\lambda) = 0$ si et seulement si $|\mathbb{X}_j^t Y| \leq \lambda/2$.

   b. En déduire
   $$\hat{\beta}_j(\lambda) = \mathbb{X}_j^t Y \left(1 - \frac{\lambda}{2|\mathbb{X}_j^t Y|}\right)_+, \quad j = 1, \dots, p$$
   où $(x)_+ = \max(x, 0)$. Interpréter ce résultat.

Exercice 3.3 (Unicité de l'estimateur lasso)**.** Cet exercice est inspiré de Giraud (2015). Soit $\widehat{\beta}^1(\lambda)$ et $\widehat{\beta}^2(\lambda)$ deux solutions qui minimisent l'Équation 3.2. Soit $\widehat{\beta} = (\widehat{\beta}^1(\lambda) + \widehat{\beta}^2(\lambda))/2$.

1. Montrer que si $\mathbb{X}\widehat{\beta}^1(\lambda) \neq \mathbb{X}\widehat{\beta}^2(\lambda)$ alors

$$\|\mathbb{Y} - \mathbb{X}\widehat{\beta}\|_2^2 + \lambda\|\widehat{\beta}\|_1 < \frac{1}{2}\left(\|\mathbb{Y} - \mathbb{X}\widehat{\beta}^1(\lambda)\|_2^2 + \lambda\|\widehat{\beta}^1(\lambda)\|_1 + \|\mathbb{Y} - \mathbb{X}\widehat{\beta}^2(\lambda)\|_2^2 + \lambda\|\widehat{\beta}^2(\lambda)\|_1\right).$$

   On pourra utiliser la convexité (forte) de $x \mapsto \|x\|_2^2$.

2. En déduire que $\mathbb{X}\widehat{\beta}^1(\lambda) = \mathbb{X}\widehat{\beta}^2(\lambda)$.

4 Modèle additif

Le modèle additif (modèle GAM) peut être vu comme un compromis entre une modélisation linéaire et non paramétrique de la fonction de régression. Il suppose que cette fonction s'écrit

$$m(x) = m(x_1, \dots, x_d) = \alpha + g_1(x_1) + \cdots + g_d(x_d).$$

4.1 Pseudo backfitting

L'algorithme du backfitting est souvent utilisé pour estimer les composantes du modèle additif. Etant donné un échantillon $(x_i, y_i), i = 1, \dots, n$ on note $\bar{\mathbb{Y}}$ le vecteur des y_i et $\mathbb{X}_k$ le vecteur contenant les observations de la variable k pour $k = 1, \dots, d$. L'algorithme se résume ainsi

1. Initialisation : $\hat{\alpha} = \bar{\mathbb{Y}}$, $\hat{g}_k(x_k) = \bar{\mathbb{X}}_k$.
2. Pour $k = 1, \dots, d$:
   - $\mathbb{Y}^{(k)} = \mathbb{Y} - \hat{\alpha} - \sum_{j \neq k} \hat{g}_j(\mathbb{X}_j)$ (résidus partiels)
   - $\hat{g}_k$: lissage non paramétrique de $\mathbb{Y}^{(k)}$ sur $\mathbb{X}_k$.
3. Répéter l'étape précédente tant que les $\hat{g}_k$ changent.

On propose dans cette partie d'utiliser cet algorithme pour estimer les paramètres du modèle linéaire en remplaçant le lissage non paramétrique par un estimateur MCO. On considère le modèle de régression linéaire

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \varepsilon$$

avec X_1 et X_2 de lois uniformes sur $[0, 1]$ et ε de loi $\mathcal{N}(0, 1)$ (ε est indépendante de $(X_1, X_2)'$).

1. Générer un échantillon (x_i, y_i) de taille $n = 300$ selon le modèle ci-dessus pour $\beta_0 = 1, \beta_1 = 3, \beta_2 = 5$.
2. Créer une fonction **R** qui admet en entrée un jeu de données et qui fournit en sortie les estimateurs par la méthode du backfitting.
3. En déduire les estimateurs backfitting pour le problème considéré.
4. Comparer aux estimateurs **MCO**.

4.2 Modèle GAM

On considère les données générées selon

```
n <- 1000
set.seed(1465)
X1 <- 2*runif(n)
X2 <- 2*runif(n)
bruit <- rnorm(n)
Y <- 2*X1+sin(8*pi*X2)+bruit
donnees<-data.frame(Y,X1,X2)
```

1. Écrire le modèle
2. A l'aide du package **gam** visualiser les estimateurs des composantes additives du modèle. On utilisera tout d'abord un lissage par **spline** avec 1 ddl pour la première composante et 24.579 ddl pour la seconde.
3. Faire varier les degrés de liberté, interpréter.
4. Faire le même travail avec le lisseur **loess**. On commencera avec `degree=2` et `span=0.15` puis on fera varier le paramètre `span`.
5. Estimer le degrés de liberté avec la fonction `gam` du package **mgcv** (Il n'est pas nécessaire de charger le package pour éviter les conflits).

4.3 Régression logistique additive

On considère le jeu de données **panne.txt** qui recense des pannes de machine (`etat=1`) en fonction de leur âge et de leur marque.

1. Faire une régression logistique permettant d'expliquer la variable `etat` par la variable `age` uniquement. Critiquer le modèle.
2. Ajuster un modèle additif, toujours avec uniquement la variable `age`.
3. En utilisant le modèle additif, proposer un nouveau modèle logistique plus pertinent.

partie II

Non supervisée

5 Rappels sur le k-means et la CAH

Ces méthodes sont certainement les deux algorithmes les plus utilisés en apprentissage non supervisé.

L'algorithme des k-means propose de trouver un représentant pour chaque classe, appelé centroïde, en minimisant :

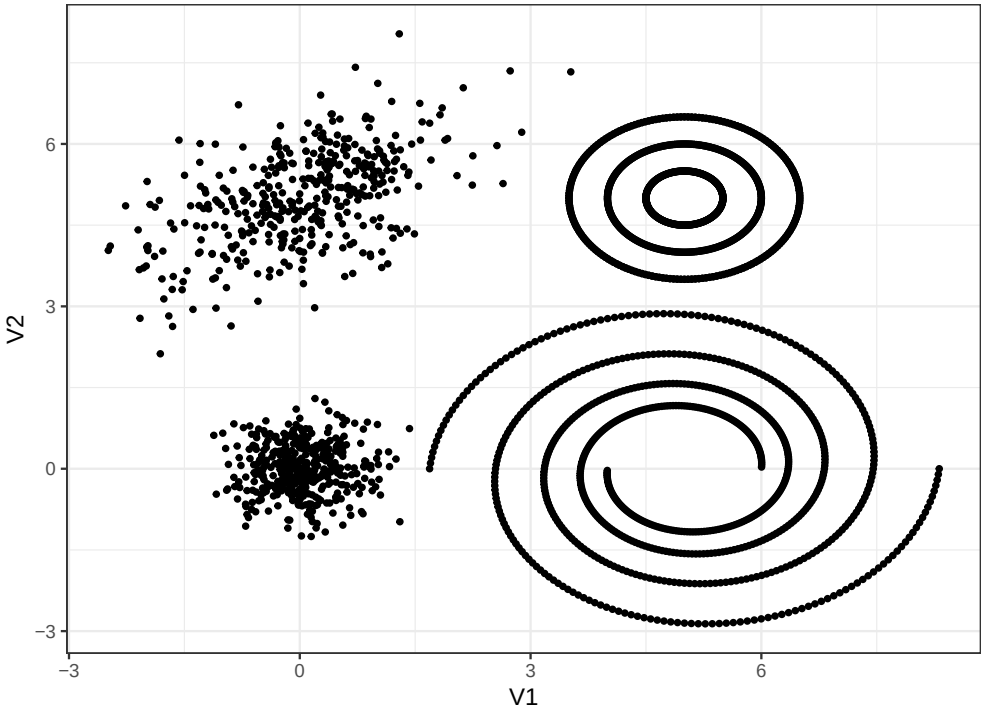
$$\frac{1}{n}\sum_{i=1}^{n}\min_{j=1,\dots,K}\|x_i - c_j\|^2.$$

Plusieurs types d'algorithmes peuvent être utiliser pour trouver des solutions (locales) à ce problème. Une fois la solution obtenue, les clusters s'obtiennent en affectant chaque observation à son centroïde le plus proche.

Une **CAH** va quant à elle définir des clusters de façon récursive en agrégeant à chaqué étape les deux clusters les plus proches au sens d'une mesure de proximité à définir.

Exercice 5.1 (kmeans et CAH sur R)**.** On considère les données

```
tbl <- read_delim("data/donclassif.txt",delim = ";")
ggplot(tbl)+aes(x=V1,y=V2)+geom_point()
```



1. Discuter du nombre de clusters pour ce jeu de données.
2. Tester différents algorithmes k-means, visualiser les résultats et discuter de la capacité de cet algorithme à identifier les différentes structures géométriques des données.
3. Faire le même travail avec la classification ascendente hiérarchique. On commencera par comparer les différentes méthodes d'agglomération en fonction du nombre de cluster, afin d'en déduire une stratégie efficace permettant notamment d'identifier les spirales et les cercles concentriques.

Exercice 5.2 (CAH sur un gros jeu de données)**.** On reprend le même jeu de données mais avec plus d'individus :

```
tbl <- read_delim("data/donclassif2.txt",delim = ";")
dim(tbl)
```

```
[1] 70000     2
```

1. Que se passe t-il lorsque vous faites une CAH ?
2. Proposer une solution pour faire quand même la CAH.

6 Dbscan et clustering spectral

6.1 L'algorithme DBSCAN

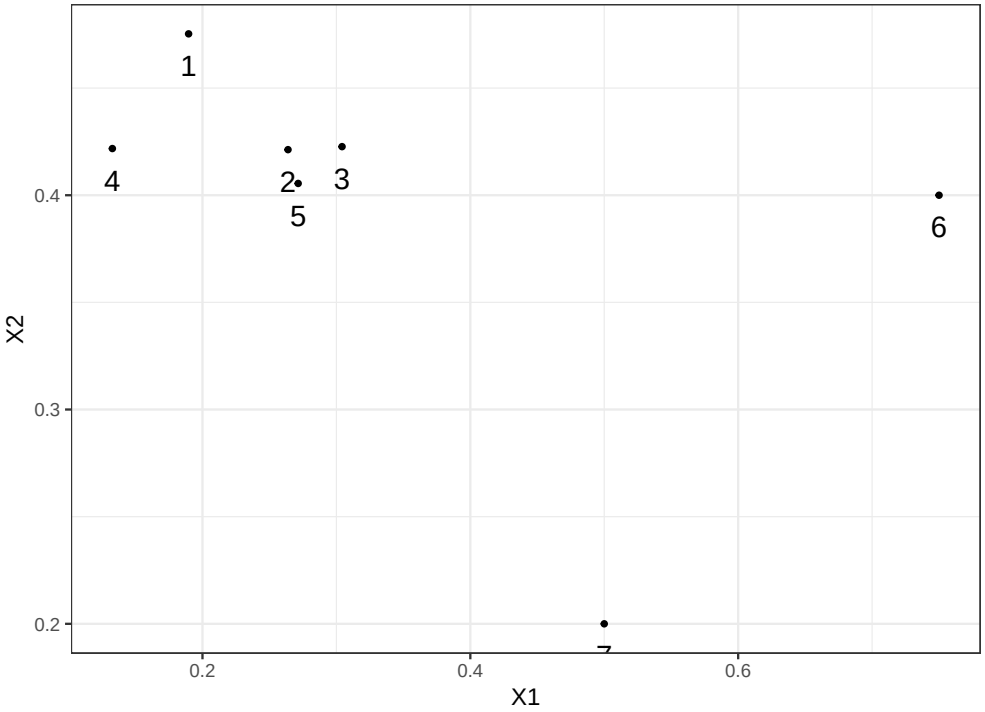
L'algorithme DBSCAN (Density Based Spatial Clustering of Applications with Noise, Ester et al. (1996)) fait partie des méthodes basées sur la densité : les clusters correspondent à des zones de fortes densité séparées par des zones où la densité est plus faible. Ces zones sont définies par deux types de points :

- les **noyaux** : des points qui contiennent plus de `minPts` points à une distance inférieure à `epsilon` ;
- les **points de bordure** : les points qui vont se situés en bordure des clusters ou qui sont isolés.

On charge le package

```
library(dbscan)
```

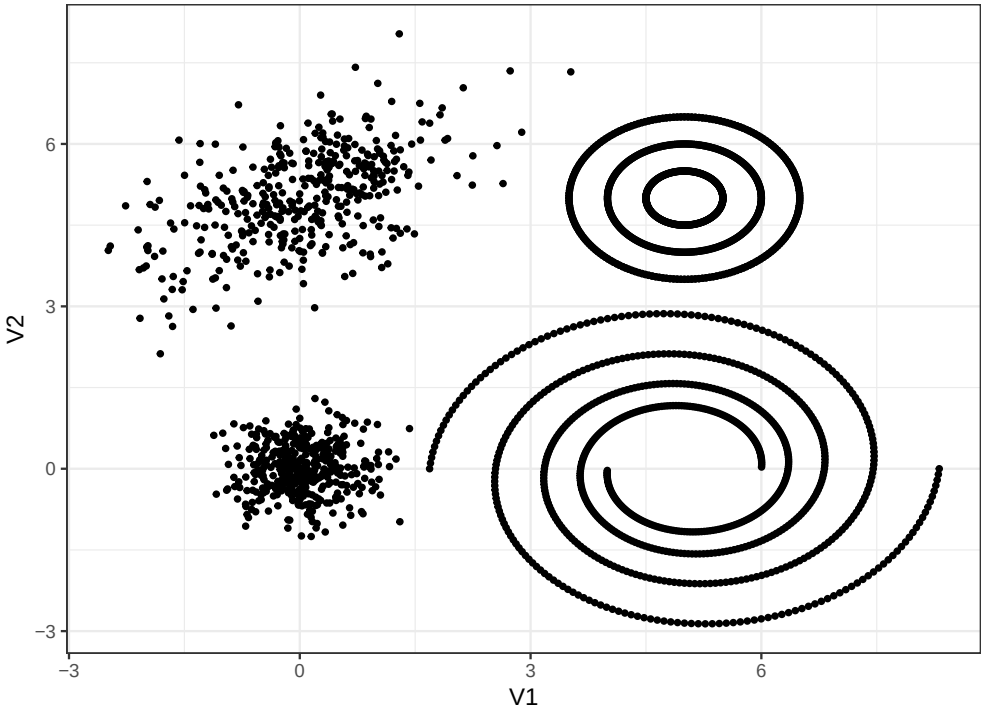
Exercice 6.1 (Noyaux et points de bordure)**.** On considère le "nuage" de points suivant :



1. On fixe `eps=0.13` et `minPts=4`. À l'aide de calculs simples, identifier les noyaux et points de bordure.
2. Retrouver ces résultats à l'aide de la fonction `is.corepoint`.
3. Effectuer l'algorithme **dbscan** avec ces valeurs de paramètre et interpréter.
4. Est-ce que des points de bordure peuvent être affectés à des cluster ? Justifier.

Exercice 6.2 (Calibration de dbscan)**.** On reprend les données de l'Exercice 5.1 :

```
tbl <- read_delim("data/donclassif.txt",delim = ";")
ggplot(tbl)+aes(x=V1,y=V2)+geom_point()
```



En utilisant la stratégie proposée dans l'aide de `dbscan`, calibrer l'algorithme pour essayer d'identifier au mieux les différents clusters. On pourra envisager une deuxième étape pour affecter les petits clusters aux gros…

6.2 Clustering spectral

Le *clustering spectral* est un algorithme de classification non supervisé qui permet de définir des clusters de nœuds sur des graphes ou d'individus pour des données `individus/variables`. L'algorithme est basé sur la décomposition spectrale du Laplacien (normalisé) d'une matrice de similarité, il est résumé ci-dessous :

Entrées :

- tableau de données $n \times p$
- K un noyau
- k le nombre de clusters.

1. Calculer la matrice de **similarités** W sur les données en utilisant le **noyau** K
2. Calculer le **Laplacien normalisé** L_{norm} à partir de W.
3. Calculer les k **premiers vecteurs propres** $u_1, \dots, u_k$ de L_{norm}. On note U la matrice $n \times k$ qui les contient.

4. Calculer la matrice T en **normalisant les lignes** de U : $t_{ij} = u_{ij}/(\sum_\ell u_{i\ell}^2)^{1/2}$.
5. Faire un k-**means** avec les points $y_i, i = 1, \dots, n$ (i-me ligne de T) $\Longrightarrow A_1, \dots, A_k$.

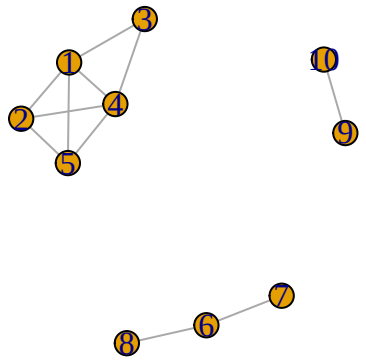
Sortie : clusters $C_1, \dots, C_k$ avec

$$C_j = \{i | y_i \in A_j\}.$$

L'objet de ce chapitre est de travailler sur cet algorithme en le programmant, puis en utilisant la fonction **specc** du package `kernlab`.

On crée tout d'abord un graphe avec trois composantes connexes : on utilise la commande **sample_gnp()** qui permet de créer un graphe selon le modèle d'Erdos-Renyi.

```
library(igraph)
set.seed(1)
n1 <- 5
n2 <- 3
n3 <- 2
n <- n1+n2+n3
# il faut prendre des grandes valeurs de p sinon on risque d'avoir des sous-graphes non co
p1 <- 0.85
p2 <- 0.75
p3 <- 0.7
G1 <- sample_gnp(n1,p1)
G2 <- sample_gnp(n2,p2)
G3 <- sample_gnp(n3,p3)
G <- G1 + G2 + G3 # il cree un graphe avec ces 3 sous-graphes
plot(G)
```



On vérifie le nombre de composantes connexes

```r
components(G)$no
```

```
[1] 3
```

Exercice 6.3 (Laplacien non normalisé)**.**

1. Calculer la matrice d'adjacence de **G** et en déduire le Laplacien normalisé. On pourra utiliser la fonction `as_adj`.
2. Retrouver ce Laplacien avec la fonction **laplacian_matrix**.
3. Calculer les valeurs propres et représenter les sur un graphe. Que remarquez-vous ?
4. Obtenir les trois vecteurs propres associés à la valeur propre nulle. Commenter.
5. Normaliser ces vecteurs. On pourra utiliser la fonction

```r
normalize <- function(x){
  return(x/sqrt(sum(x^2)))
}
```

6. Terminer l'algorithme avec le k-means.

Exercice 6.4 (Clustering spectral avec specc)**.** On reprend les données de l'Exercice 5.1 :

```r
tbl <- read_delim("data/donclassif.txt",delim = ";")
```

Faire le "meilleur" clustering spectral, on pourra utiliser plusieurs noyaux et plusieurs valeurs de paramètres.

1. On commencera par lancer la procédure sur un sous-échantillon de taille 1000
2. On proposera ensuite une autre solution qui permettra de traiter tous les observations.

Références

Ester, M., H. P. Kriegel, J. Sander, et X. Xu. 1996. « A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise ». In.

Giraud, C. 2015. *Introduction to High-Dimensional Statistics*. CRC Press.