

Statistique en grande dimension

L. Rouvière

laurent.rouviere@univ-rennes2.fr

SEPTEMBRE 2022

Présentation

- **Objectifs** : identifier le problème de la grande dimension et adapter les techniques traditionnelles à ce cadre.
- **Pré-requis** : théorie des probabilités, modélisation statistique, régression (linéaire et logistique). R, niveau avancé.
- **Enseignant** : Laurent Rouvière laurent.rouviere@univ-rennes2.fr
 - **Recherche** : statistique non paramétrique, apprentissage statistique
 - **Enseignements** : statistique et probabilités (Université, école d'ingénieur et de commerce, formation continue).
 - **Consulting** : énergie, finance, marketing.

- 12h : 8h CM + 4h TD/TP.

Programme

- 12h : 8h CM + 4h TD/TP.
- Matériel :
 - slides à l'url
https://lrouviere.github.io/page_perso/grande_dim.html
 - tutoriel (compléments de cours + exercices) à l'url
https://lrouviere.github.io/TUTO_GRANDE_DIM/

- 12h : 8h CM + 4h TD/TP.
- Matériel :
 - slides à l'url
https://lrouviere.github.io/page_perso/grande_dim.html
 - tutoriel (compléments de cours + exercices) à l'url
https://lrouviere.github.io/TUTO_GRANDE_DIM/
- 4 parties :
 1. Introduction : le problème de la grande dimension
 2. Régression sur composantes (PCR et PLS) et sélection de variables
 3. Approches régularisées (ridge/lasso)
 4. Modèle additif ou introduction au graph mining

Première partie I

Introduction : le problème de la grande dimension

Quelques exemples

Grande dimension en régression

Approche non paramétrique

Approche paramétrique

Bibliographie

Quelques citations

[Giraud, 2015]

- Over the last twenty years (or so), the dramatic development of data acquisition technologies has enabled devices able to take **thousands (up to million) of measurements simultaneously**.
- Having access to such massive data sounds like a **blessing**.
- Indeed, **separating the useful information from the noise** is generally almost impossible in high dimensional settings.
- This issue is often referred as **the curse of dimensionality**.

[Bühlmann and van de Geer, 2011]

- High-dimensional data are nowadays rule rather than exception in areas like information technology, bioinformatics or astronomy...
- The word "high-dimensional" refers to the situation where the number of unknown parameters which are to be estimated is one or several orders of magnitude larger than the number of samples in the data.
- Classical statistical inference cannot be used for high dimensional problems.

- **Constat** : de plus en plus de données à disposition.

- **Constat** : de plus en plus de données à disposition.

Positif

Beaucoup d'information pour répondre au problème posé.

- **Constat** : de plus en plus de données à disposition.

Positif

Beaucoup d'information pour répondre au problème posé.

Négatif

- Difficile de dissocier l'information pertinente du bruit.
- Modèle de plus en plus complexe \implies de plus en plus de paramètres \implies difficile de bien estimer.

Quelques exemples

Grande dimension en régression

Approche non paramétrique

Approche paramétrique

Bibliographie

Détection de spam

- Sur 4 601 mails, on a pu identifier 1813 spams.
- On a également mesuré sur chacun de ces mails la présence ou absence de 57 mots.

```
> spam %>% select(c(1:8,58)) %>% head()
##   make address  all num3d  our over remove internet type
## 1 0.00    0.64 0.64    0 0.32 0.00    0.00    0.00 spam
## 2 0.21    0.28 0.50    0 0.14 0.28    0.21    0.07 spam
## 3 0.06    0.00 0.71    0 1.23 0.19    0.19    0.12 spam
## 4 0.00    0.00 0.00    0 0.63 0.00    0.31    0.63 spam
## 5 0.00    0.00 0.00    0 0.63 0.00    0.31    0.63 spam
## 6 0.00    0.00 0.00    0 1.85 0.00    0.00    1.85 spam
```

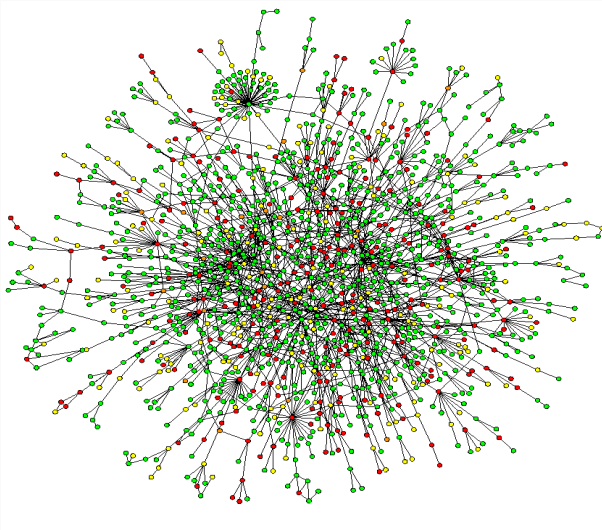
Détection de spam

- Sur 4 601 mails, on a pu identifier 1813 spams.
- On a également mesuré sur chacun de ces mails la présence ou absence de 57 mots.

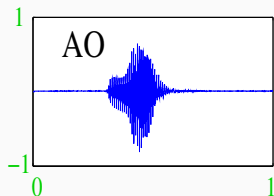
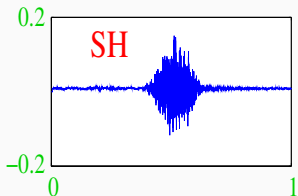
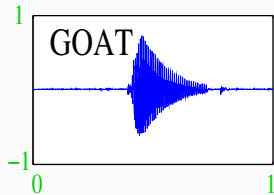
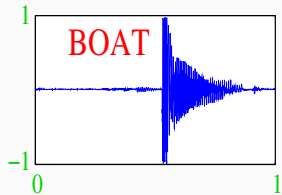
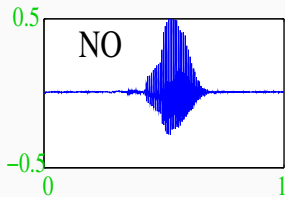
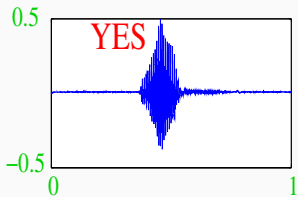
```
> spam %>% select(c(1:8,58)) %>% head()
##   make address  all num3d  our over remove internet type
## 1 0.00     0.64 0.64     0 0.32 0.00    0.00     0.00 spam
## 2 0.21     0.28 0.50     0 0.14 0.28    0.21     0.07 spam
## 3 0.06     0.00 0.71     0 1.23 0.19    0.19     0.12 spam
## 4 0.00     0.00 0.00     0 0.63 0.00    0.31     0.63 spam
## 5 0.00     0.00 0.00     0 0.63 0.00    0.31     0.63 spam
## 6 0.00     0.00 0.00     0 1.85 0.00    0.00     1.85 spam
```

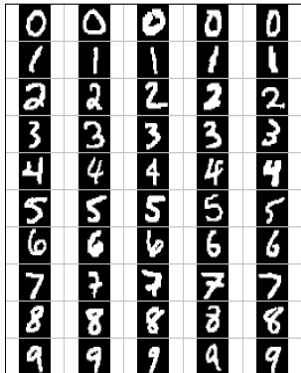
Le problème

Expliquer **type** par $p = 57$ variables.



Données fonctionnelles





Quelques chiffres sur les capacités de stockage [Besse, 2018]

Période	Mémoire	Ordre de grandeur
1940-70	Octet	$n = 30, p \leq 10$
1970	kO	$n = 500, p \leq 10$
1980	MO	Machine Learning
1990	GO	Data-Mining
2000	TO	$p > n$, apprentissage statistique
2010	PO	n explose, cloud, cluster...
2013	serveurs	Big data
2017	??	Intelligence artificielle...

Quelques chiffres sur les capacités de stockage [Besse, 2018]

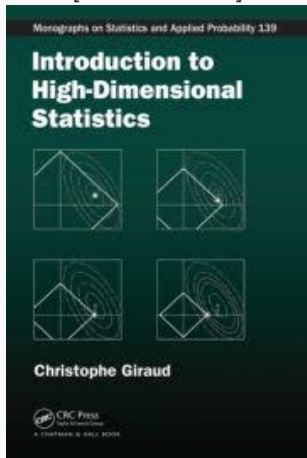
Période	Mémoire	Ordre de grandeur
1940-70	Octet	$n = 30, p \leq 10$
1970	kO	$n = 500, p \leq 10$
1980	MO	Machine Learning
1990	GO	Data-Mining
2000	TO	$p > n$, apprentissage statistique
2010	PO	n explose, cloud, cluster...
2013	serveurs	Big data
2017	??	Intelligence artificielle...

Conclusion

Nécessité d'adapter les techniques traditionnelles à ces données volumineuses.

Références (1)

[Giraud, 2015]

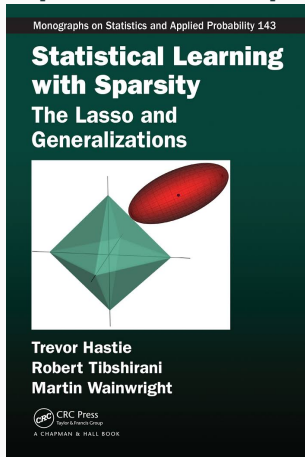


[Cornillon et al., 2019]

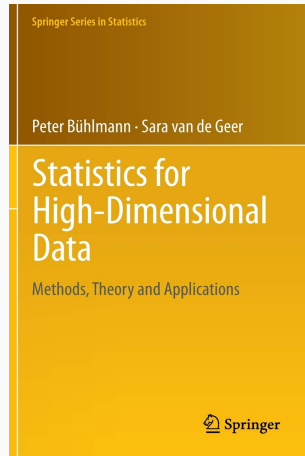


Références (2)

[Hastie et al., 2015]



[Bühlmann and van de Geer, 2011]



Quelques exemples

Grande dimension en régression

Approche non paramétrique

Approche paramétrique

Bibliographie

Le problème de régression

- Les données : $(x_1, y_1), \dots, (x_n, y_n)$ avec $x_i \in \mathbb{R}^p$ et $y_i \in \mathbb{R}$.
- Le modèle

$$y_i = m(x_i) + \varepsilon_i \quad \text{avec} \quad \mathbf{E}[\varepsilon_i] = 0 \text{ et } \mathbf{V}[\varepsilon_i] = \sigma^2.$$

Le problème de régression

- Les données : $(x_1, y_1), \dots, (x_n, y_n)$ avec $x_i \in \mathbb{R}^p$ et $y_i \in \mathbb{R}$.
- Le modèle

$$y_i = m(x_i) + \varepsilon_i \quad \text{avec} \quad \mathbf{E}[\varepsilon_i] = 0 \text{ et } \mathbf{V}[\varepsilon_i] = \sigma^2.$$

Le problème

Estimer $m : \mathbb{R}^p \rightarrow \mathbb{R}$.

Le problème de régression

- Les données : $(x_1, y_1), \dots, (x_n, y_n)$ avec $x_i \in \mathbb{R}^p$ et $y_i \in \mathbb{R}$.
- Le modèle

$$y_i = m(x_i) + \varepsilon_i \quad \text{avec} \quad \mathbf{E}[\varepsilon_i] = 0 \text{ et } \mathbf{V}[\varepsilon_i] = \sigma^2.$$

Le problème

Estimer $m : \mathbb{R}^p \rightarrow \mathbb{R}$.

Différentes approches

- Paramétrique : modèle linéaire et estimation par moindres carrés...
- Non paramétrique : noyau, plus proches voisins...

Quelques exemples

Grande dimension en régression

Approche non paramétrique

Approche paramétrique

Bibliographie

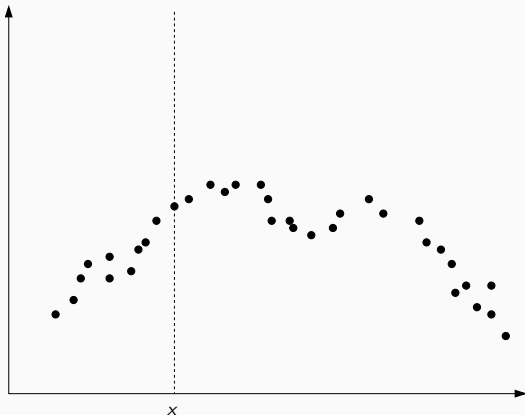
Estimateurs à noyau

- Non paramétriques : moyennes locales $\hat{m}_n(x) = \sum_{i=1}^n W_{ni}(x)y_i$ où les poids $W_{ni}(x)$ vont varier selon les algorithmes.



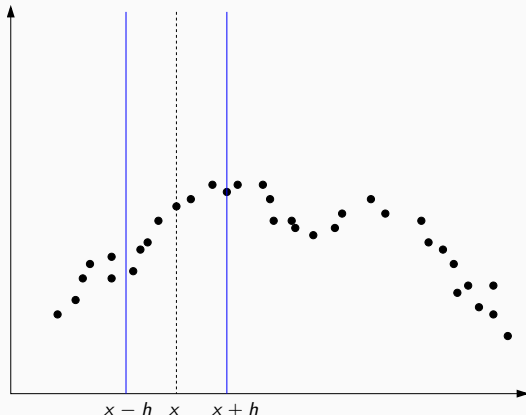
Estimateurs à noyau

- Non paramétriques : moyennes locales $\hat{m}_n(x) = \sum_{i=1}^n W_{ni}(x)y_i$ où les poids $W_{ni}(x)$ vont varier selon les algorithmes.



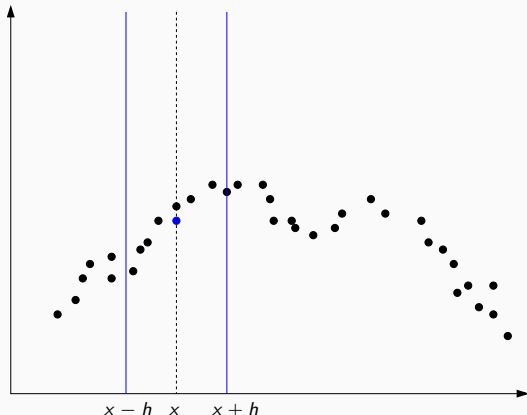
Estimateurs à noyau

- Non paramétriques : moyennes locales $\hat{m}_n(x) = \sum_{i=1}^n W_{ni}(x)y_i$ où les poids $W_{ni}(x)$ vont varier selon les algorithmes.



Estimateurs à noyau

- Non paramétriques : moyennes locales $\hat{m}_n(x) = \sum_{i=1}^n W_{ni}(x)y_i$ où les poids $W_{ni}(x)$ vont varier selon les algorithmes.



- L'estimateur s'écrit

$$\hat{m}_n(x) = \frac{\sum_{i=1}^n \mathbf{1}_{x-h \leq x_i \leq x+h} y_i}{\sum_{i=1}^n \mathbf{1}_{x-h \leq x_i \leq x+h}} = \frac{\sum_{i=1}^n \mathbf{1}_{\left| \frac{x_i - x}{h} \right| \leq 1} y_i}{\sum_{i=1}^n \mathbf{1}_{\left| \frac{x_i - x}{h} \right| \leq 1}}.$$

- L'estimateur s'écrit

$$\hat{m}_n(x) = \frac{\sum_{i=1}^n \mathbf{1}_{x-h \leq x_i \leq x+h} y_i}{\sum_{i=1}^n \mathbf{1}_{x-h \leq x_i \leq x+h}} = \frac{\sum_{i=1}^n \mathbf{1}_{\left| \frac{x_i - x}{h} \right| \leq 1} y_i}{\sum_{i=1}^n \mathbf{1}_{\left| \frac{x_i - x}{h} \right| \leq 1}}.$$

Définition

Soit $h > 0$ et $K : \mathbb{R}^p \rightarrow \mathbb{R}^+$. L'estimateur à noyau de **fenêtre** h et de **noyau** K est défini par

$$\hat{m}_n(x) = \frac{\sum_{i=1}^n K\left(\frac{x_i - x}{h}\right) y_i}{\sum_{i=1}^n K\left(\frac{x_i - x}{h}\right)}.$$

- Noyau usuel dans \mathbb{R}^p :

1. Uniforme : $K(x) = \mathbf{1}_{\|x\| \leq 1}$;
2. Gaussien : $K(x) = \exp(-\|x\|^2)$;
3. Epanechnikov : $K(x) = \frac{3}{4}(1 - \|x\|^2)\mathbf{1}_{\|x\| \leq 1}$.

- Noyau usuel dans \mathbb{R}^p :
 1. Uniforme : $K(x) = \mathbf{1}_{\|x\| \leq 1}$;
 2. Gaussien : $K(x) = \exp(-\|x\|^2)$;
 3. Epanechnikov : $K(x) = \frac{3}{4}(1 - \|x\|^2)\mathbf{1}_{\|x\| \leq 1}$.
- Le choix de h est crucial pour la qualité de l'estimation :
 1. h grand : estimateur « constant », variance faible, biais fort ;
 2. h petit : « interpolation », variance forte, biais faible.

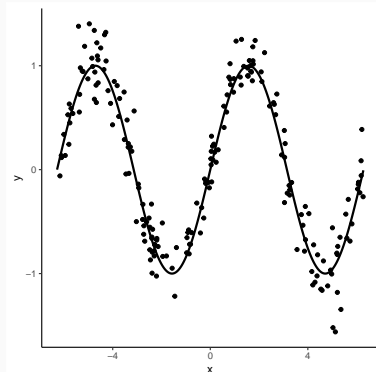
Un exemple

- On génère un échantillon $(X_i, Y_i), i = 1, \dots, n = 200$ selon

$$Y_i = \sin(X_i) + \varepsilon_i, \quad i = 1, \dots, n$$

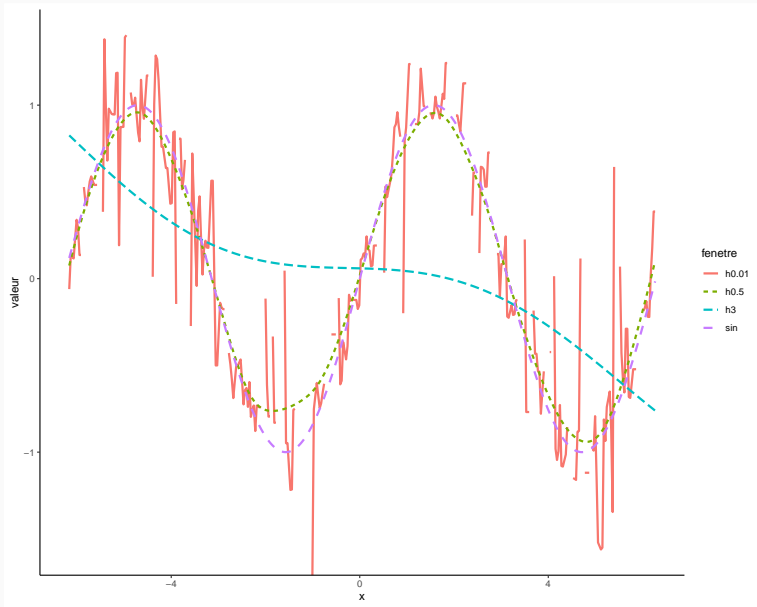
avec X_i uniformes sur $[-2\pi, 2\pi]$, ε_i de loi gaussienne $\mathcal{N}(0, 0.2^2)$.

```
> n <- 200; set.seed(1234)
> X <- runif(n, -2*pi, 2*pi)
> eps <- rnorm(n, 0, 0.2)
> Y <- sin(X) + eps
> df <- data.frame(X=X, Y=Y)
> x <- seq(-2*pi, 2*pi, by=0.01)
> df1 <- data.frame(x=x, y=sin(x))
> ggplot(df1) + aes(x=x, y=y) +
+   geom_line(size=1) +
+   geom_point(data=df, aes(x=X, y=Y)) +
+   theme_classic()
```



Tracé des estimateurs

```
> library("KernSmooth") #package a charger pour la fonction locpoly
> h1 <- 0.5;h2 <- 3;h3 <- 0.01
> fx1 <-locpoly(X,Y,bandwidth=h1)
> fx2 <-locpoly(X,Y,bandwidth=h2)
> fx3 <-locpoly(X,Y,bandwidth=h3)
> df2 <- data.frame(x=fx1$x,"h0.5"=fx1$y,"h3"=fx2$y,"h0.01"=fx3$y) %>%
+   mutate(sin=sin(x)) %>%
+   gather(key="fenetre",value="valeur",-x)
> ggplot(df2)+aes(x=x,y=valeur,color=fenetre,lty=fenetre)+
+   geom_line(size=1)+theme_classic()
```



Algorithme de plus proches voisins

Définition

Soit $k \leq n$ un entier. L'estimateur des k plus proches voisins est défini par

$$\hat{m}_n(x) = \frac{1}{k} \sum_{i \in \text{kppv}(x)} y_i$$

où pour $x \in \mathcal{X}$

$$\text{kppv}(x) = \{i : x_i \text{ fait partie des kppv de } x \text{ parmi } \{x_1, \dots, x_n\}\}.$$

Algorithme de plus proches voisins

Définition

Soit $k \leq n$ un entier. L'estimateur des **k plus proches voisins** est défini par

$$\hat{m}_n(x) = \frac{1}{k} \sum_{i \in \text{kppv}(x)} y_i$$

où pour $x \in \mathcal{X}$

$$\text{kppv}(x) = \{i : x_i \text{ fait partie des kppv de } x \text{ parmi } \{x_1, \dots, x_n\}\}.$$

Remarque

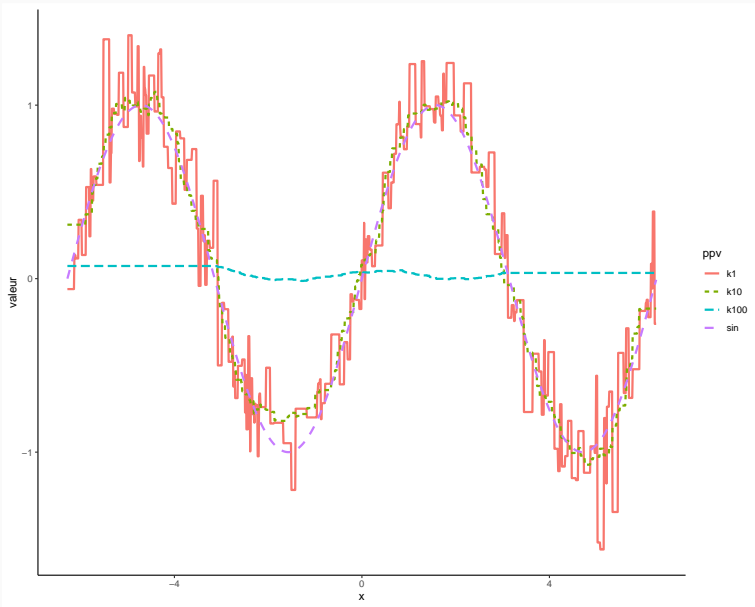
Cette fois, c'est le paramètre k qui est **bleu** crucial pour la qualité de l'estimation :

1. **k grand** : estimateur « constant », variance faible, biais fort ;
2. **k petit** : « sur-ajustement », variance forte, biais faible ;

Exemple

- La fonction `knn.reg` du package `FNN` permet de construire des estimateurs de type k plus proches voisins.

```
> library(FNN)
> k1 <- 10; k2 <- 100; k3 <- 1
> fx1 <- knn.reg(train=X, test=as.matrix(x), y=Y, k=k1)
> fx2 <- knn.reg(train=X, test=as.matrix(x), y=Y, k=k2)
> fx3 <- knn.reg(train=X, test=as.matrix(x), y=Y, k=k3)
> df3 <- data.frame(x=x, "k10"=fx1$pred, "k100"=fx2$pred, "k1"=fx3$pred) %>%
+   mutate(sin=sin(x)) %>%
+   gather(key="ppv", value="valeur", -x)
> ggplot(df3)+aes(x=x, y=valeur, color=ppv, lty=ppv)+
+   geom_line(size=1)+theme_classic()
```



Et que dit la théorie ?

- L'étude des propriétés des estimateurs peut s'effectuer en contrôlant le **risque quadratique**

$$\mathbf{E} \|\hat{m}_n - m\|^2 = \mathbf{E} \left\{ \int (\hat{m}_n(x) - m(x))^2 \mu(dx) \right\}$$

qui se décompose en un terme de **biais** et de **variance**.

- Le contrôle du terme de biais nécessite des **hypothèses sur la régularité** de la fonction à estimer m .
- Nous donnons dans la suite des résultats pour les fonctions **Lipschitziennes** :

$$|m(x) - m(z)| \leq C \|x - z\|, \quad \forall x, z \in \mathbb{R}^p.$$

Théorème [Györfi et al., 2002]

- Pour l'estimateur à **noyau** de fenêtre $h > 0$, on a

$$\mathbf{E}\|\hat{m}_n - m\|^2 \leq C_1^2 h^2 + \frac{C_2}{nh^p}.$$

- Pour l'estimateur des **k ppv**, on a

$$\mathbf{E}\|\hat{m}_n - m\|^2 \leq \frac{C_3}{k} + C_4 \left(\frac{k}{n}\right)^{2/p}.$$

Théorème [Györfi et al., 2002]

- Pour l'estimateur à **noyau** de fenêtre $h > 0$, on a

$$\mathbf{E}\|\hat{m}_n - m\|^2 \leq C_1^2 h^2 + \frac{C_2}{nh^p}.$$

- Pour l'estimateur des **k ppv**, on a

$$\mathbf{E}\|\hat{m}_n - m\|^2 \leq \frac{C_3}{k} + C_4 \left(\frac{k}{n}\right)^{2/p}.$$

Commentaire

- On retrouve bien **l'importance de h et k** dans les vitesses de convergence.
- On voit également que la **dimension p intervient** dans les vitesses de convergence.
- Voir **exercice 1.2 du tuto**.

Corollaire

- La fenêtre et le nombre de ppv **optimaux** sont de l'ordre de

$$h^* = C_5 n^{-\frac{1}{p+2}} \quad \text{et} \quad k^* = C_6 n^{\frac{2}{p+2}}.$$

- Pour ces valeurs optimales, le **risque quadratique** vérifie

$$\mathbf{E} \|\hat{m}_n - m\|^2 \leq C_7 n^{-\frac{2}{p+2}}.$$

Corollaire

- La fenêtre et le nombre de ppv **optimaux** sont de l'ordre de

$$h^* = C_5 n^{-\frac{1}{p+2}} \quad \text{et} \quad k^* = C_6 n^{\frac{2}{p+2}}.$$

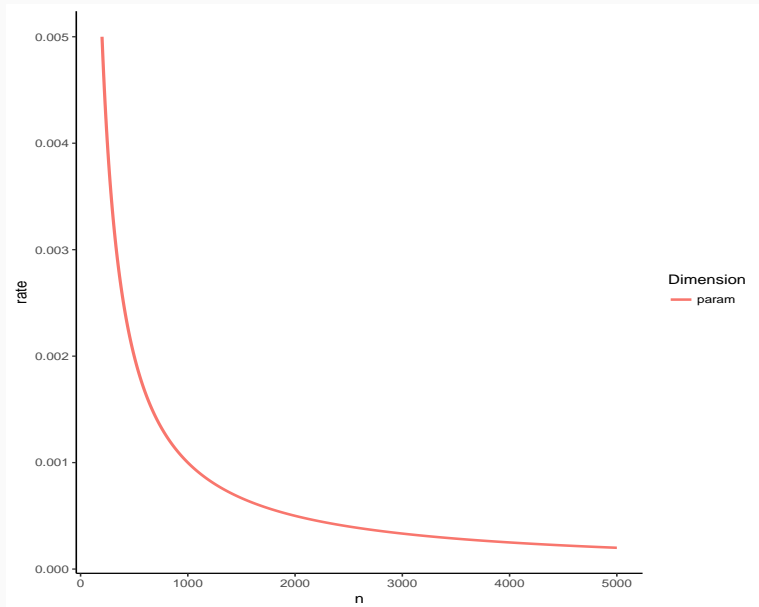
- Pour ces valeurs optimales, le **risque quadratique** vérifie

$$\mathbf{E} \|\hat{m}_n - m\|^2 \leq C_7 n^{-\frac{2}{p+2}}.$$

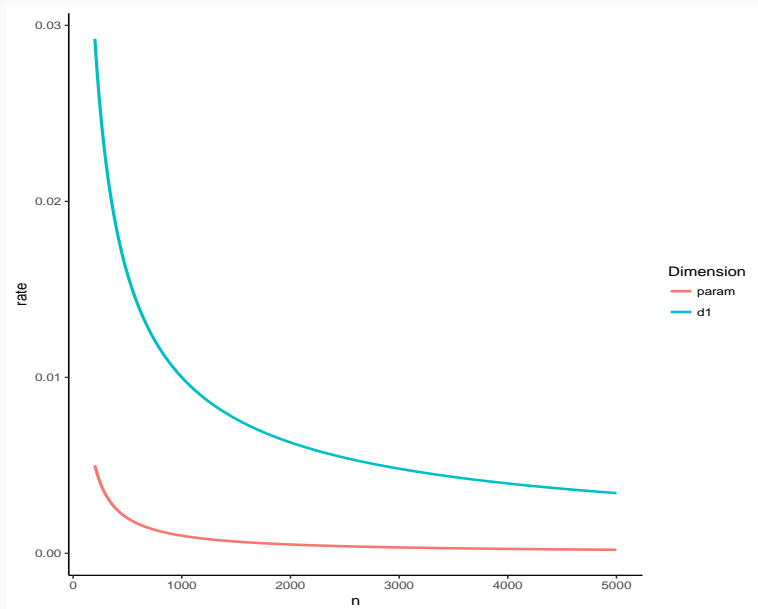
Conséquence

- Lorsque $p \nearrow$, les estimateurs convergent **moins vite** et sont donc **moins précis**.
- C'est le **fléau de la dimension**.

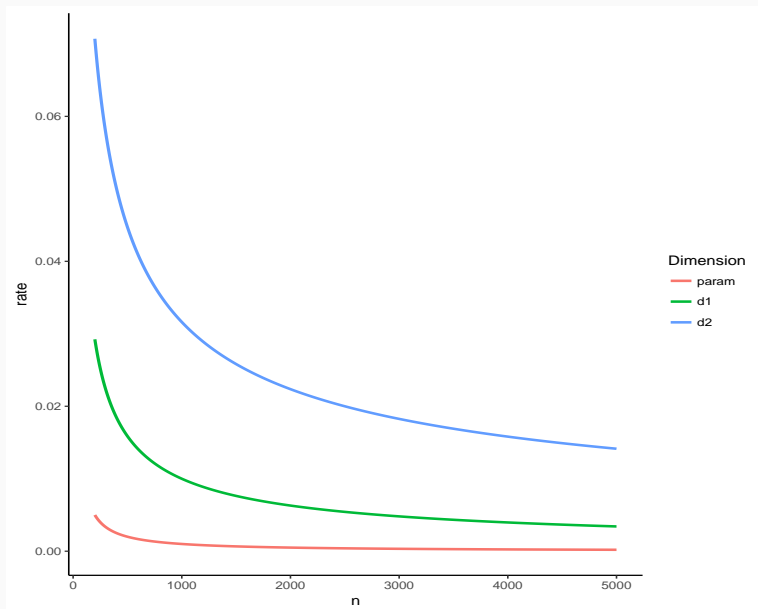
Fléau de la dimension (Illustration)



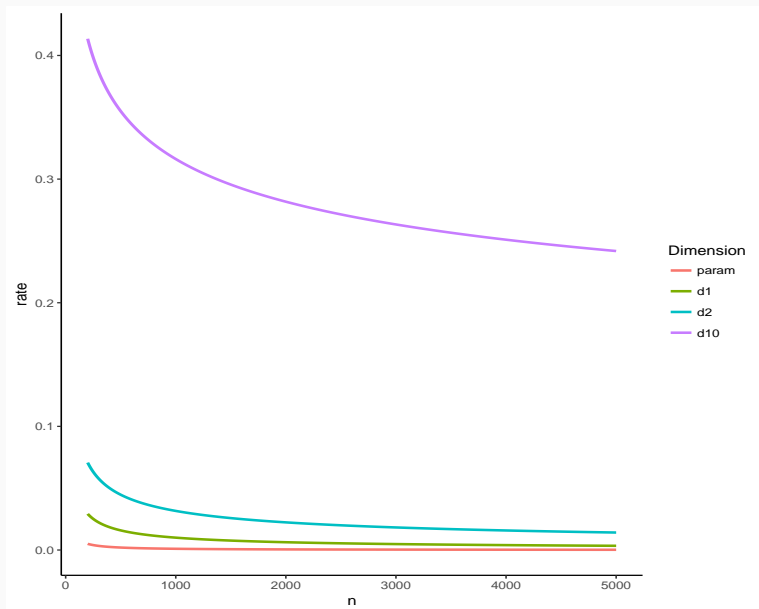
Fléau de la dimension (Illustration)



Fléau de la dimension (Illustration)



Fléau de la dimension (Illustration)



Pourquoi le fléau de la dimension ?

- Les estimateurs non paramétriques présentés sont basés sur des moyennes locales.
- On calcule des moyennes à partir d'observations proches du point où on veut estimer la fonction.

Pourquoi le fléau de la dimension ?

- Les estimateurs non paramétriques présentés sont basés sur des moyennes locales.
- On calcule des moyennes à partir d'observations proches du point où on veut estimer la fonction.
- Lorsque la dimension p augmente, la notion de proximité perd de son sens \implies difficile de trouver des observations proches.
- On dit que les voisinages se vident.

Pourquoi le fléau de la dimension ?

- Les estimateurs non paramétriques présentés sont basés sur des **moyennes locales**.
- On calcule des moyennes à partir d'observations **proches** du point où on veut estimer la fonction.
- Lorsque la dimension p augmente, la notion de proximité perd de son sens \implies difficile de trouver des **observations proches**.
- On dit que les **voisinages se vident**.

Exemple [**Giraud, 2015**]

Soit $X = (X_1, \dots, X_p)$ et $Y = (Y_1, \dots, Y_p)$ 2 vecteurs aléatoires indépendants de distribution uniforme sur $[0, 1]^p$, alors

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

\implies Voir **exercice 1.1 du tuto**.

Quelques exemples

Grande dimension en régression

Approche non paramétrique

Approche paramétrique

Bibliographie

- Le modèle **linéaire** est le modèle **paramétrique** de référence.
- Ce modèle fait **l'hypothèse** que la fonction de régression m est linéaire en ses composantes :

$$y_i = m(x_i) + \varepsilon_i = \beta_1 x_{i1} + \dots + \beta_p x_{ip} + \varepsilon_i$$

avec $\mathbf{E}[\varepsilon_i] = 0$ et $\mathbf{V}[\varepsilon_i] = \sigma^2$.

Modèle linéaire

- Le modèle **linéaire** est le modèle **paramétrique** de référence.
- Ce modèle fait **l'hypothèse** que la fonction de régression m est linéaire en ses composantes :

$$y_i = m(x_i) + \varepsilon_i = \beta_1 x_{i1} + \dots + \beta_p x_{ip} + \varepsilon_i$$

avec $\mathbf{E}[\varepsilon_i] = 0$ et $\mathbf{V}[\varepsilon_i] = \sigma^2$.

Estimation

Estimer m revient à **estimer** $\beta \in \mathbb{R}^p$ (dimension finie \implies **paramétrique**).

Quelques propriétés

- L'approche **moindres carrés** consiste à minimiser

$$\sum_{i=1}^n \varepsilon_i^2 = \sum_{i=1}^n (y_i - \beta_1 x_{i1} + \dots + \beta_p x_{ip})^2$$

qui fournit l'estimateur des moindres carrés

$$\hat{\beta} = (\mathbb{X}^t \mathbb{X})^{-1} \mathbb{X}^t \mathbb{Y}.$$

- La fonction des régression m^* est alors estimée par

$$\hat{m}_n(x) = \hat{\beta}_1 x_1 + \dots + \hat{\beta}_p x_p.$$

Propriété

Sous les hypothèses du modèle linéaire, on a

- $\mathbf{E}[\hat{\beta}] = \beta$ et $\mathbf{V}[\hat{\beta}] = (\mathbb{X}^t \mathbb{X})^{-1} \sigma^2$.
- On déduit (sous certaines hypothèses supplémentaires sur le design)

$$\mathbf{E}[\|\hat{\beta} - \beta\|^2] = O\left(\frac{1}{n}\right) \quad \text{et} \quad \mathbf{E}[(\hat{m}_n(x) - m^*(x))^2] = O\left(\frac{1}{n}\right).$$

Propriété

Sous les hypothèses du modèle linéaire, on a

- $\mathbf{E}[\hat{\beta}] = \beta$ et $\mathbf{V}[\hat{\beta}] = (\mathbb{X}^t \mathbb{X})^{-1} \sigma^2$.
- On déduit (sous certaines hypothèses supplémentaires sur le design)

$$\mathbf{E}[\|\hat{\beta} - \beta\|^2] = O\left(\frac{1}{n}\right) \quad \text{et} \quad \mathbf{E}[(\hat{m}_n(x) - m^*(x))^2] = O\left(\frac{1}{n}\right).$$

Remarque

- On dit que l'estimateur des moindres carrés converge à la vitesse paramétrique $(1/n)$.
- Si on suppose de plus que les erreurs $\varepsilon_i, i = 1 \dots, n$ sont gaussiennes, on déduit la loi des estimateurs des moindres carrés (qui nous permet d'obtenir des intervalles de confiance, des procédures de test...).
- Pour plus de précisions, on pourra se référer à [Grob, 2003, Cornillon et al., 2019].

La dimension en régression linéaire

- La dimension p ne **semble** pas intervenir dans les résultats précédents !

La dimension en régression linéaire

- La dimension p ne **semble** pas intervenir dans les résultats précédents !
- Elle est **bien présente** en réalité (cachée dans les "constantes").

La dimension en régression linéaire

- La dimension p ne **semble** pas intervenir dans les résultats précédents !
- Elle est **bien présente** en réalité (cachée dans les "constantes").

Exemple

- Sous le modèle linéaire, on a

$$\mathbf{E}[\|\hat{\beta} - \beta\|^2] = \text{Tr}((\mathbb{X}^t \mathbb{X})^{-1}) \sigma^2.$$

- Si on suppose de plus que \mathbb{X} est orthonormale, alors

$$\mathbf{E}[\|\hat{\beta} - \beta\|^2] = p \sigma^2.$$

La dimension en régression linéaire

- La dimension p ne **semble** pas intervenir dans les résultats précédents !
- Elle est **bien présente** en réalité (cachée dans les "constantes").

Exemple

- Sous le modèle linéaire, on a

$$\mathbf{E}[\|\hat{\beta} - \beta\|^2] = \text{Tr}((\mathbb{X}^t \mathbb{X})^{-1}) \sigma^2.$$

- Si on suppose de plus que \mathbb{X} est orthonormale, alors

$$\mathbf{E}[\|\hat{\beta} - \beta\|^2] = p \sigma^2.$$

Conséquence

L'erreur d'estimation **augmente** avec la dimension !

Illustration

- On génère des données $(x_i, y_i), i = 1, \dots, 500$ selon le modèle

$$Y = 1X_1 + 0X_2 + \dots + 0X_{q+1} + \varepsilon$$

où $X_2, X_{q+1}, \dots, \varepsilon$ sont i.i.d. de loi $\mathcal{N}(0, 1)$.

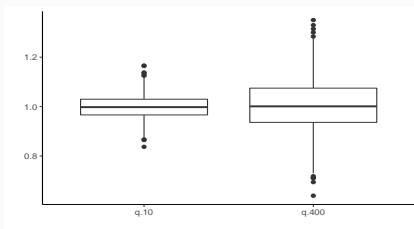
Illustration

- On génère des données $(x_i, y_i), i = 1, \dots, 500$ selon le modèle

$$Y = 1X_1 + 0X_2 + \dots + 0X_{q+1} + \varepsilon$$

où $X_2, X_{q+1}, \dots, \varepsilon$ sont i.i.d. de loi $\mathcal{N}(0, 1)$.

- On calcule l'estimateur de MCO de β_1 sur 1000 répétitions. On trace les boxplot de ces estimateurs pour $q = 10$ et $q = 400$.



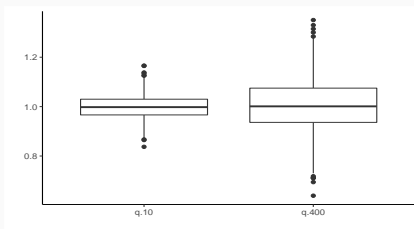
Illustration

- On génère des données $(x_i, y_i), i = 1, \dots, 500$ selon le modèle

$$Y = 1X_1 + 0X_2 + \dots + 0X_{q+1} + \varepsilon$$

où $X_2, X_{q+1}, \dots, \varepsilon$ sont i.i.d. de loi $\mathcal{N}(0, 1)$.

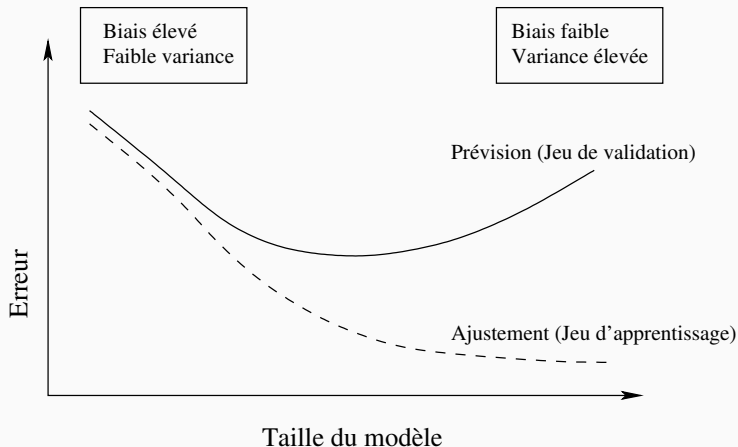
- On calcule l'estimateur de MCO de β_1 sur 1000 répétitions. On trace les boxplot de ces estimateurs pour $q = 10$ et $q = 400$.



Conclusion

Plus de variance (donc moins de précision) lorsque le nombre de variables inutiles augmente.

Taille de modèle



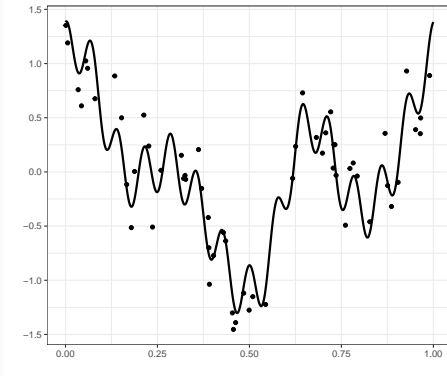
Idem erreur d'estimation (variance) / erreur d'approximation (biais).

Autre exemple : régression fonctionnelle

- On souhaite reconstruire un **signal** à l'aide d'un **échantillon bruité** :

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

- L'échantillon et la vraie fonction se trouvent sur la figure ci-dessous.



- La théorie du signal nous dit qu'une fonction (suffisamment) régulière peut être approchée dans le **domaine de Fourier**.
- Pour p assez grand, on a

$$m(x) = \alpha_0 + \sum_{j=1}^p (\beta_j \cos(2\pi jx) + \gamma_j \sin(2\pi jx)) .$$

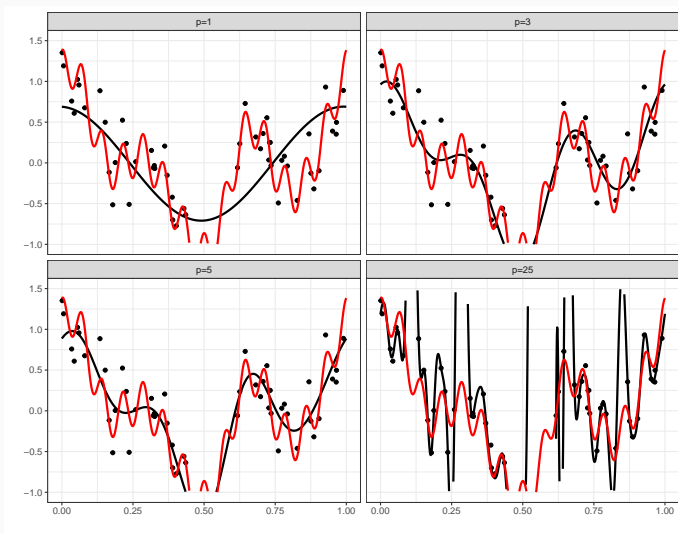
- La théorie du signal nous dit qu'une fonction (suffisamment) régulière peut être approchée dans le **domaine de Fourier**.
- Pour p assez grand, on a

$$m(x) = \alpha_0 + \sum_{j=1}^p (\beta_j \cos(2\pi jx) + \gamma_j \sin(2\pi jx)).$$

- On propose donc de considérer le **modèle linéaire de dimension $2p+1$** :

$$y_i = \alpha_0 + \sum_{j=1}^p (\beta_j \cos(2\pi jx_i) + \gamma_j \sin(2\pi jx_i)) + \varepsilon_i.$$

Résultats pour 4 valeurs de p



On **interpole** si p est trop grand.

Conclusion

- En **grande dimension** les approches traditionnelles sont souvent **peu performantes**.
- Nécessité de les corriger.

Conclusion

- En **grande dimension** les approches traditionnelles sont souvent **peu performantes**.
- Nécessité de les corriger.

Comment ?

- Approches **machine learning** : trouver des algorithmes qui apprennent directement sur les données.
- Méthodes de **réduction de la dimension** : choix de variables ou de combinaisons de variables.
- Approches **régularisées** pour diminuer la variance...

Quelques exemples

Grande dimension en régression

Approche non paramétrique

Approche paramétrique

Bibliographie



Besse, P. (2018).

Science des données - Apprentissage Statistique.

INSA - Toulouse.

http://www.math.univ-toulouse.fr/~besse/pub/Appren_stat.pdf.



Bühlmann, P. and van de Geer, S. (2011).

Statistics for High-Dimensional Data : Methods, Theory and Applications.

Springer.



Cornillon, P., Hengartner, N., Matzner-Løber, E., and Rouvière, L. (2019).

Régression avec R.

EDP Sciences.



Giraud, C. (2015).

Introduction to High-Dimensional Statistics.

CRC Press.



Grob, J. (2003).

Linear regression.

Springer.



Györfi, L., Kohler, M., Krzyzak, A., and Harro, W. (2002).

A Distribution-Free Theory of Nonparametric Regression.

Springer.



Hastie, T., Tibshirani, R., and Wainwright, M. (2015).

Statistical Learning with Sparsity : The Lasso and Generalizations.

CRC Press.

[https:](https://web.stanford.edu/~hastie/StatLearnSparsity_files/SLS.pdf)

[//web.stanford.edu/~hastie/StatLearnSparsity_files/SLS.pdf.](https://web.stanford.edu/~hastie/StatLearnSparsity_files/SLS.pdf)

Deuxième partie II

Réduction de la dimension

Sélections exhaustive et pas à pas

Régression sur composantes

Régression sur composantes principales (PCR)

Rappels ACP

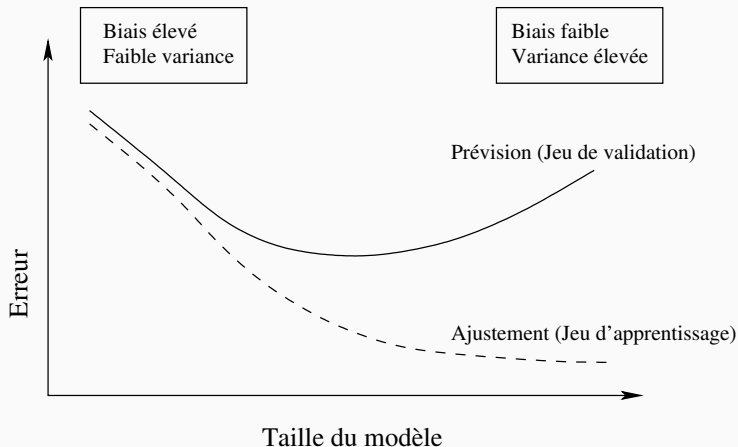
Retour à PCR

Régression PLS

Choix du nombre de composantes

Bibliographie

Rappels



Idem erreur d'estimation (variance) / erreur d'approximation (biais).

- $(X_1, Y_1), \dots, (X_n, Y_n)$ i.i.d. de même loi que (X, Y) à valeurs dans $\mathbb{R}^p \times \mathcal{Y}$;
- Dans ce chapitre, on suppose que $\mathcal{Y} = \mathbb{R}$ ou $\{-1, 1\}$;

- $(X_1, Y_1), \dots, (X_n, Y_n)$ i.i.d. de même loi que (X, Y) à valeurs dans $\mathbb{R}^p \times \mathcal{Y}$;
- Dans ce chapitre, on suppose que $\mathcal{Y} = \mathbb{R}$ ou $\{-1, 1\}$;

Modèle linéaire et logistique

1. Si $\mathcal{Y} = \mathbb{R}$,

$$m(x) = \mathbf{E}[Y|X = x] = \beta_0 + \beta_1 x_1 + \dots + \beta_d x_d = x^t \beta.$$

2. Si $\mathcal{Y} = \{-1, 1\}$,

$$\text{logit } p(x) = \beta_0 + \beta_1 x_1 + \dots + \beta_d x_d = x^t \beta$$

où $p(x) = \mathbf{P}(Y = 1|X = x)$.

- Ces deux modèles font partie des modèles de référence.

Limites

- Principalement 2 motifs d'insatisfaction :

- Principalement 2 motifs d'insatisfaction :
 1. Précision d'estimation : les estimateurs des MCO pour la régression et du MV pour la logistique ont souvent un biais relativement faible mais une variance élevée (notamment lorsque le nombre de variables p est grand).

- Principalement 2 motifs d'insatisfaction :
 1. **Précision d'estimation** : les estimateurs des MCO pour la régression et du MV pour la logistique ont souvent un biais relativement faible mais une variance élevée (notamment lorsque le nombre de variables p est grand).
 2. **Interprétation** : lorsque le nombre de variables p est grand, on ne connaît pas les variables "importantes".

- Principalement 2 motifs d'insatisfaction :
 1. **Précision d'estimation** : les estimateurs des MCO pour la régression et du MV pour la logistique ont souvent un biais relativement faible mais une variance élevée (notamment lorsque le nombre de variables p est grand).
 2. **Interprétation** : lorsque le nombre de variables p est grand, on ne connaît pas les variables "importantes".

Objectifs

- Avec l'augmentation du volume des données ces dernières années, ces deux inconvénients sont de plus en plus visibles.
- Nécessité de développer des procédures de sélection de sous-groupes de variables ou de combinaison de variables.

Sélections exhaustive et pas à pas

Régression sur composantes

Régression sur composantes principales (PCR)

Rappels ACP

Retour à PCR

Régression PLS

Choix du nombre de composantes

Bibliographie

Best subset selection

- $(X_1, Y_1), \dots, (X_n, Y_n)$ i.i.d. à valeurs dans $\mathbb{R}^p \times \mathbb{R}$;
- p variables explicatives $\implies 2^p$ modèles concurrents.

Best subset selection

- $(X_1, Y_1), \dots, (X_n, Y_n)$ i.i.d. à valeurs dans $\mathbb{R}^p \times \mathbb{R}$;
- p variables explicatives $\implies 2^p$ modèles concurrents.

Approche exhaustive

1. Construire les 2^p modèles ;
2. Choisir celui qui optimise un critère donné.

Best subset selection

- $(X_1, Y_1), \dots, (X_n, Y_n)$ i.i.d. à valeurs dans $\mathbb{R}^p \times \mathbb{R}$;
- p variables explicatives $\implies 2^p$ modèles concurrents.

Approche exhaustive

1. Construire les 2^p modèles ;
2. Choisir celui qui optimise un critère donné.

Algorithme : best subset selection

1. Pour $k = 0, \dots, p$:
 - 1.1 Construire les $\binom{p}{k}$ modèles linéaires à k variables ;
 - 1.2 Choisir parmi ces modèles celui qui a le plus grand R^2 . ; On note \mathcal{M}_k le modèle sélectionné.
2. Choisir, parmi $\mathcal{M}_0, \mathcal{M}_1, \dots, \mathcal{M}_p$, le meilleur modèle au sens d'un critère donné.

Exemples de critères (voir [Cornillon et al., 2019])

- **AIC** : Akaike Information Criterion

$$-2\mathcal{L}_n(\hat{\beta}) + 2p.$$

- **BIC** : Bayesian Information Criterion

$$-2\mathcal{L}_n(\hat{\beta}) + \log(n)p.$$

- **R^2 ajusté** :

$$R_a^2 = 1 - \frac{n-1}{n-p+1}(1-R^2) \quad \text{où} \quad R^2 = \frac{SSR}{SST} = \frac{\|\hat{\mathbf{Y}} - \bar{\mathbf{Y}}\mathbf{1}\|^2}{\|\mathbf{Y} - \bar{\mathbf{Y}}\mathbf{1}\|^2}.$$

- **C_p de Mallows** :

$$C_p = \frac{1}{n} \left(\sum_{i=1}^n (Y_i - \hat{Y}_i)^2 + 2p\hat{\sigma}^2 \right).$$

- Ces critères sont constitués de deux parties :
 1. une qui mesure la **qualité d'ajustement** du modèle ;
 2. une autre qui mesure sa **complexité**.

Ajustement complexité

- Ces critères sont constitués de deux parties :
 1. une qui mesure la **qualité d'ajustement** du modèle ;
 2. une autre qui mesure sa **complexité**.

Exemple AIC

- $-2\mathcal{L}_n(\hat{\beta})$ mesure l'ajustement ;
- $2p$ mesure la complexité.

Ajustement complexité

- Ces critères sont constitués de deux parties :
 1. une qui mesure la **qualité d'ajustement** du modèle ;
 2. une autre qui mesure sa **complexité**.

Exemple AIC

- $-2\mathcal{L}_n(\hat{\beta})$ mesure l'ajustement ;
- $2p$ mesure la complexité.

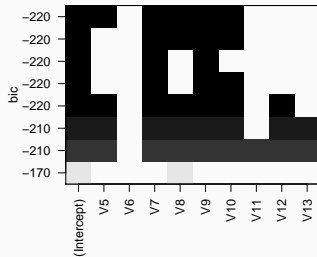
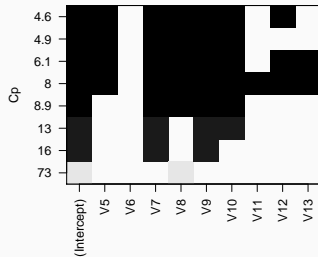
⇒ l'idée est de choisir un modèle de **complexité minimale** qui **ajuste bien** les données.

- La fonction `regsubsets` du package `leaps` permet d'utiliser cette approche exhaustive.

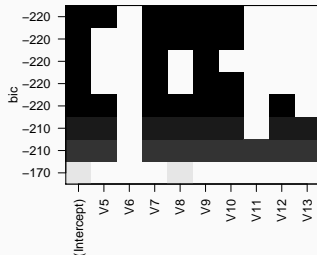
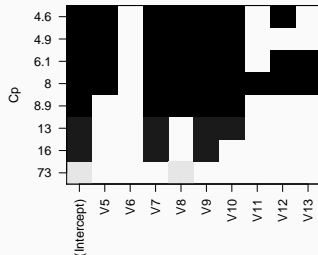
```
> library(mlbench)
> data(Ozone)
> library(leaps)
> reg.fit <- regsubsets(V4~V5+V6+V7+V8+V9+V10+V11+V12+V13,data=Ozone)
> summary(reg.fit)$outmat
```

##		V5	V6	V7	V8	V9	V10	V11	V12	V13
## 1	(1)	" "	" "	" "	"*"	" "	" "	" "	" "	" "
## 2	(1)	" "	" "	"*"	" "	"*"	" "	" "	" "	" "
## 3	(1)	" "	" "	"*"	" "	"*"	"*"	" "	" "	" "
## 4	(1)	" "	" "	"*"	"*"	"*"	"*"	" "	" "	" "
## 5	(1)	"*"	" "	"*"	"*"	"*"	"*"	" "	" "	" "
## 6	(1)	"*"	" "	"*"	"*"	"*"	"*"	" "	"*"	" "
## 7	(1)	"*"	" "	"*"	"*"	"*"	"*"	" "	"*"	"*"
## 8	(1)	"*"	" "	"*"	"*"	"*"	"*"	"*"	"*"	"*"


```
> plot(reg.fit,scale="Cp")
> plot(reg.fit,scale="bic")
```



```
> plot(reg.fit,scale="Cp")
> plot(reg.fit,scale="bic")
```



Conclusion

- C_p sélectionne :

$$Y = \beta_0 + \beta_1 V_5 + \beta_2 V_7 + \beta_3 V_8 + \beta_4 V_9 + \beta_5 V_{10} + \beta_6 V_{12} + \varepsilon.$$

- BIC sélectionne :

$$Y = \beta_0 + \beta_1 V_5 + \beta_2 V_7 + \beta_3 V_8 + \beta_4 V_9 + \beta_5 V_{10} + \varepsilon.$$

Approche pas à pas (stepwise)

- L'avantage de l'approche exhaustive est qu'elle balaie tous les modèles.
- L'inconvénient est que le temps de calcul devient vite très important (résultat long au delà de $p = 30$).

Approche pas à pas (stepwise)

- L'avantage de l'approche exhaustive est qu'elle balaie tous les modèles.
- L'inconvénient est que le temps de calcul devient vite très important (résultat long au delà de $p = 30$).
- Lorsque le nombre de variables est grand, on privilégie souvent les méthodes pas à pas qui consistent à construire les modèles de façon récursive, en ajoutant (ou supprimant) une variable explicative à chaque étape.

Ascendant (forward)

1. Construire \mathcal{M}_0 le **modèle null** (avec uniquement la constante) ;
2. Pour $k = 0, \dots, p - 1$:
 - 2.1 Construire les $p - k$ modèles consistant à ajouter une variable dans \mathcal{M}_k ;
 - 2.2 Choisir, parmi ces $p - k$ modèles, celui qui maximise le $R^2 \rightarrow \mathcal{M}_{k+1}$.
3. Choisir, parmi $\mathcal{M}_0, \dots, \mathcal{M}_p$, le meilleur modèle au sens d'un **critère donné**.

Ascendant (forward)

1. Construire \mathcal{M}_0 le **modèle null** (avec uniquement la constante) ;
2. Pour $k = 0, \dots, p - 1$:
 - 2.1 Construire les $p - k$ modèles consistant à ajouter une variable dans \mathcal{M}_k ;
 - 2.2 Choisir, parmi ces $p - k$ modèles, celui qui maximise le $R^2 \rightarrow \mathcal{M}_{k+1}$.
3. Choisir, parmi $\mathcal{M}_0, \dots, \mathcal{M}_p$, le meilleur modèle au sens d'un **critère donné**.

Descendant (backward)

1. Construire \mathcal{M}_p le **modèle complet** (avec les p variables) ;
2. Pour $k = p, \dots, 1$:
 - 2.1 Construire les k modèles consistant à supprimer une variable dans \mathcal{M}_k ;
 - 2.2 Choisir, parmi ces k modèles, celui qui maximise le $R^2 \rightarrow \mathcal{M}_{k-1}$.
3. Choisir, parmi $\mathcal{M}_0, \dots, \mathcal{M}_p$, le meilleur modèle au sens d'un **critère donné**.

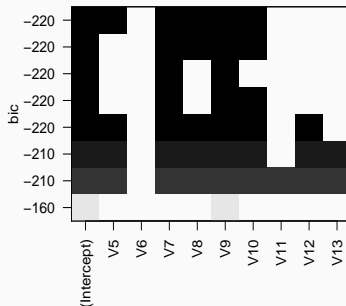
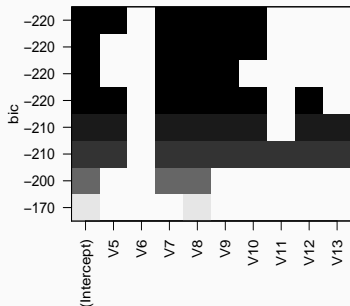
- Il suffit d'ajouter l'argument `method="forward"` ou `method="backward"` dans `regsubsets` pour utiliser ces procédures pas à pas.

```
> reg.fit.for <- regsubsets(V4~V5+V6+V7+V8+V9+V10+V11+V12+V13,  
+                           data=Ozone,method="forward")  
> reg.fit.back <- regsubsets(V4~V5+V6+V7+V8+V9+V10+V11+V12+V13,  
+                             data=Ozone,method="backward")
```

```
> summary(reg.fit.for)$outmat  
##           V5  V6  V7  V8  V9  V10 V11 V12 V13  
## 1  ( 1 ) " " " " " " " " " " " " " " " "  
## 2  ( 1 ) " " " " " " " " " " " " " " " "  
## 3  ( 1 ) " " " " " " " " " " " " " " " "  
## 4  ( 1 ) " " " " " " " " " " " " " " " "  
## 5  ( 1 ) " " " " " " " " " " " " " " " "  
## 6  ( 1 ) " " " " " " " " " " " " " " " "  
## 7  ( 1 ) " " " " " " " " " " " " " " " "  
## 8  ( 1 ) " " " " " " " " " " " " " " " "
```

```
> summary(reg.fit.back)$outmat  
##           V5  V6  V7  V8  V9  V10 V11 V12 V13  
## 1  ( 1 ) " " " " " " " " " " " " " " " "  
## 2  ( 1 ) " " " " " " " " " " " " " " " "  
## 3  ( 1 ) " " " " " " " " " " " " " " " "  
## 4  ( 1 ) " " " " " " " " " " " " " " " "  
## 5  ( 1 ) " " " " " " " " " " " " " " " "  
## 6  ( 1 ) " " " " " " " " " " " " " " " "  
## 7  ( 1 ) " " " " " " " " " " " " " " " "  
## 8  ( 1 ) " " " " " " " " " " " " " " " "
```

```
> plot(reg.fit.for,scale="bic")  
> plot(reg.fit.back,scale="bic")
```



Remarque

Sur cet exemple, les deux algorithmes sélectionnent le même modèle (ce n'est pas forcément toujours le cas).

- Les approches **exhaustive** et **pas à pas** ont été présentées dans un cadre de **régression** ($\mathcal{Y} = \mathbb{R}$) ;

Cas de la discrimination binaire

- Les approches **exhaustive** et **pas à pas** ont été présentées dans un cadre de **régression** ($\mathcal{Y} = \mathbb{R}$) ;
- Elles se **transposent** naturellement à la **discrimination binaire** ($\mathcal{Y} = \{-1, 1\}$).
- Sur **R**, on pourra utiliser :
 - la fonction **bestglm** du package **bestglm** pour l'approche **exhaustive**.
 - la fonction **step** pour les procédures **pas à pas**.

Sélections exhaustive et pas à pas

Régression sur composantes

Régression sur composantes principales (PCR)

Rappels ACP

Retour à PCR

Régression PLS

Choix du nombre de composantes

Bibliographie

Cadre

1. on cherche toujours à expliquer $Y \in \mathbb{R}$ par $X = (X_1, \dots, X_p) \in \mathbb{R}^p$.
2. p grand et/ou fortes corrélations entre les $X_j, j = 1, \dots, p$.

Cadre

1. on cherche toujours à expliquer $Y \in \mathbb{R}$ par $X = (X_1, \dots, X_p) \in \mathbb{R}^p$.
2. p grand et/ou fortes corrélations entre les $X_j, j = 1, \dots, p$.

L'idée

- Réduire la dimension et atténuer l'influence de la corrélation en ne considérant plus les variables initiales...

Cadre

1. on cherche toujours à expliquer $Y \in \mathbb{R}$ par $X = (X_1, \dots, X_p) \in \mathbb{R}^p$.
2. p grand et/ou fortes corrélations entre les $X_j, j = 1, \dots, p$.

L'idée

- Réduire la dimension et atténuer l'influence de la corrélation en ne considérant plus les variables initiales...
- mais un nombre restreint de combinaisons (linéaires) de variables.

La démarche

- Construire de nouvelles variables $Z_k, k = 1, \dots, m$ combinaisons linéaires des variables initiales $X_j, j = 1, \dots, p$

$$Z_k = w_{k1}X_1 + \dots + w_{kp}X_p = w_k^t X.$$

La démarche

- Construire de nouvelles variables $Z_k, k = 1, \dots, m$ combinaisons linéaires des variables initiales $X_j, j = 1, \dots, p$

$$Z_k = w_{k1}X_1 + \dots + w_{kp}X_p = w_k^t X.$$

- On note

$$\mathbb{X} = \begin{pmatrix} x_{11} & \dots & x_{1p} \\ \vdots & & \vdots \\ x_{n1} & \dots & x_{np} \end{pmatrix} \Rightarrow \mathbb{Z} = \begin{pmatrix} z_{11} & \dots & z_{1m} \\ \vdots & & \vdots \\ z_{n1} & \dots & z_{nm} \end{pmatrix},$$

avec

$$\mathbb{Z}_k = w_{k1}\mathbb{X}_1 + \dots + w_{kp}\mathbb{X}_p = \mathbb{X}w_k.$$

La démarche

- Construire de nouvelles variables $Z_k, k = 1, \dots, m$ combinaisons linéaires des variables initiales $X_j, j = 1, \dots, p$

$$Z_k = w_{k1}X_1 + \dots + w_{kp}X_p = w_k^t X.$$

- On note

$$\mathbb{X} = \begin{pmatrix} x_{11} & \dots & x_{1p} \\ \vdots & & \vdots \\ x_{n1} & \dots & x_{np} \end{pmatrix} \Rightarrow \mathbb{Z} = \begin{pmatrix} z_{11} & \dots & z_{1m} \\ \vdots & & \vdots \\ z_{n1} & \dots & z_{nm} \end{pmatrix},$$

avec

$$\mathbb{Z}_k = w_{k1}\mathbb{X}_1 + \dots + w_{kp}\mathbb{X}_p = \mathbb{X}w_k.$$

- Effectuer la régression linéaire par MCO :

$$y_i = \alpha_0 + \alpha_1 z_{i1} + \dots + \alpha_m z_{im} + \varepsilon_i \Rightarrow \hat{\alpha}_0, \dots, \hat{\alpha}_m.$$

Calcul de la prévision

- **Nouvel individu** $x = (x_1, \dots, x_p) \implies z = (z_1, \dots, z_m)$ avec

$$z_k = w_{k1}x_1 + \dots + w_{kp}x_p = w_k^t x, k = 1, \dots, m.$$

Calcul de la prévision

- **Nouvel individu** $x = (x_1, \dots, x_p) \implies z = (z_1, \dots, z_m)$ avec

$$z_k = w_{k1}x_1 + \dots + w_{kp}x_p = w_k^t x, k = 1, \dots, m.$$

- L'algorithme renvoie la **prévision**

$$f_n(x) = \hat{\alpha}_0 + \hat{\alpha}_1 z_1 + \dots + \hat{\alpha}_m z_m.$$

Calcul de la prévision

- **Nouvel individu** $x = (x_1, \dots, x_p) \implies z = (z_1, \dots, z_m)$ avec

$$z_k = w_{k1}x_1 + \dots + w_{kp}x_p = w_k^t x, k = 1, \dots, m.$$

- L'algorithme renvoie la **prévision**

$$f_n(x) = \hat{\alpha}_0 + \hat{\alpha}_1 z_1 + \dots + \hat{\alpha}_m z_m.$$

Retour dans l'espace initial

On a

$$\begin{aligned} f_n(x) &= \hat{\alpha}_0 + \hat{\alpha}_1 w_1^t x + \dots + \hat{\alpha}_m w_m^t x \\ &= \hat{\alpha}_0 + (\hat{\alpha}_1 w_{11} + \dots + \hat{\alpha}_m w_{m1})x_1 + \dots + (\hat{\alpha}_1 w_{1p} + \dots + \hat{\alpha}_m w_{mp})x_p \\ &= \hat{\beta}_0 + \hat{\beta}_1 x_1 + \dots + \hat{\beta}_p x_p. \end{aligned}$$

Calcul de la prévision

- **Nouvel individu** $x = (x_1, \dots, x_p) \implies z = (z_1, \dots, z_m)$ avec

$$z_k = w_{k1}x_1 + \dots + w_{kp}x_p = w_k^t x, k = 1, \dots, m.$$

- L'algorithme renvoie la **prévision**

$$f_n(x) = \hat{\alpha}_0 + \hat{\alpha}_1 z_1 + \dots + \hat{\alpha}_m z_m.$$

Retour dans l'espace initial

On a

$$\begin{aligned} f_n(x) &= \hat{\alpha}_0 + \hat{\alpha}_1 w_1^t x + \dots + \hat{\alpha}_m w_m^t x \\ &= \hat{\alpha}_0 + (\hat{\alpha}_1 w_{11} + \dots + \hat{\alpha}_m w_{m1})x_1 + \dots + (\hat{\alpha}_1 w_{1p} + \dots + \hat{\alpha}_m w_{mp})x_p \\ &= \hat{\beta}_0 + \hat{\beta}_1 x_1 + \dots + \hat{\beta}_p x_p. \end{aligned}$$

\implies **Combinaison linéaire des variables initiales** \implies "Autre façon"
d'**estimer** les paramètres du **modèle linéaire**.

- Algorithme non invariant par changement d'échelle \implies centrer-réduire les données initiales

$$\mathbb{X}_j = \frac{\tilde{X}_j - \mu_j}{\sigma_j}$$

où \tilde{X}_j désigne la variable brute, μ_j sa moyenne et σ_j son écart-type.

- On a alors

$$f_n(x) = \hat{\gamma}_0 + \hat{\gamma}_1 \tilde{x}_1 + \dots + \hat{\gamma}_p \tilde{x}_p$$

avec

$$\hat{\gamma}_0 = \hat{\beta}_0 - \sum_{j=1}^p \hat{\beta}_j \mu_j \quad \text{et} \quad \hat{\gamma}_j = \frac{\hat{\beta}_j}{\sigma_j}.$$

Questions

1. Comment choisir les combinaisons linéaires Z_k ?
2. Comment choisir le nombre de composantes m ?

Sélections exhaustive et pas à pas

Régression sur composantes

Régression sur composantes principales (PCR)

Rappels ACP

Retour à PCR

Régression PLS

Choix du nombre de composantes

Bibliographie

- L'analyse en composantes principales (ACP) fait partie des méthodes standards pour construire des combinaisons linéaires de variables quantitatives.
- L'approche consiste à définir des CL des variables qui restituent "au mieux" l'information d'un tableau de données.
- Outil de statistique descriptive (visualiser des données de "grande dimension" dans des espaces de petite dimension) mais aussi de réduction de dimension.

Sélections exhaustive et pas à pas

Régression sur composantes

Régression sur composantes principales (PCR)

Rappels ACP

Retour à PCR

Régression PLS

Choix du nombre de composantes

Bibliographie

Notations

- Tableau des données

$$\mathbb{X} = \begin{matrix} & \mathbb{X}_1 & \dots & \mathbb{X}_p \\ \begin{matrix} e_1 \\ \vdots \\ e_n \end{matrix} & \begin{pmatrix} x_{1,1} & \dots & x_{1,p} \\ \vdots & & \vdots \\ x_{n,1} & \dots & x_{n,p} \end{pmatrix} \end{matrix}$$

Notations

- Tableau des données

$$\mathbb{X} = \begin{matrix} & \mathbb{X}_1 & \dots & \mathbb{X}_p \\ \begin{matrix} e_1 \\ \vdots \\ e_n \end{matrix} & \begin{pmatrix} x_{1,1} & \dots & x_{1,p} \\ \vdots & & \vdots \\ x_{n,1} & \dots & x_{n,p} \end{pmatrix} \end{matrix}$$

- $e_i = (x_{i,1}, \dots, x_{i,p})$ l'individu i et $\mathbb{X}_j = (x_{1,j}, \dots, x_{n,j})$ la variable j .

Notations

- Tableau des données

$$\mathbb{X} = \begin{matrix} & \mathbb{X}_1 & \dots & \mathbb{X}_p \\ \begin{matrix} e_1 \\ \vdots \\ e_n \end{matrix} & \begin{pmatrix} x_{1,1} & \dots & x_{1,p} \\ \vdots & & \vdots \\ x_{n,1} & \dots & x_{n,p} \end{pmatrix} \end{matrix}$$

- $e_i = (x_{i,1}, \dots, x_{i,p})$ l'individu i et $\mathbb{X}_j = (x_{1,j}, \dots, x_{n,j})$ la variable j .
- $e_i \in \mathbb{R}^p$, la représentation de l'ensemble des individus est un nuage de points dans \mathbb{R}^p , appelé **nuage des individus**, \mathcal{N} .

Notations

- Tableau des données

$$\mathbb{X} = \begin{matrix} & \mathbb{X}_1 & \dots & \mathbb{X}_p \\ \begin{matrix} e_1 \\ \vdots \\ e_n \end{matrix} & \begin{pmatrix} x_{1,1} & \dots & x_{1,p} \\ \vdots & & \vdots \\ x_{n,1} & \dots & x_{n,p} \end{pmatrix} \end{matrix}$$

- $e_i = (x_{i,1}, \dots, x_{i,p})$ l'individu i et $\mathbb{X}_j = (x_{1,j}, \dots, x_{n,j})$ la variable j .
- $e_i \in \mathbb{R}^p$, la représentation de l'ensemble des individus est un nuage de points dans \mathbb{R}^p , appelé **nuage des individus**, \mathcal{N} .
- $\mathbb{X}_j \in \mathbb{R}^n$, la représentation de l'ensemble des variables est un nuage de points dans \mathbb{R}^n , appelé **nuage des variables**, \mathcal{M} .

Remarque

Si l'œil était capable de **visualiser dans \mathbb{R}^n et \mathbb{R}^p** , il n'y aurait pas de problème...

Objectifs

Déterminer un **sous-espace de dimension réduite** qui soit "compréhensible" par l'œil sur lequel projeter le nuage.

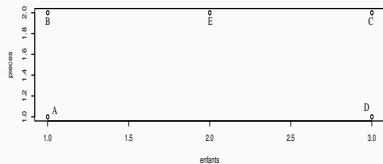
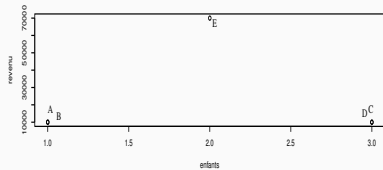
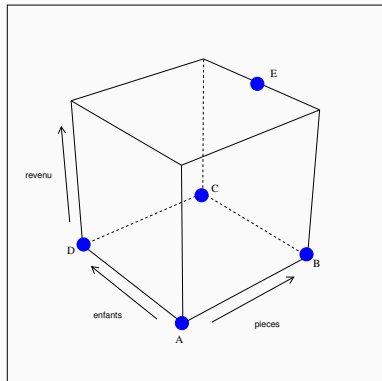
Objectifs

Déterminer un **sous-espace de dimension réduite** qui soit "compréhensible" par l'œil sur lequel projeter le nuage.

Un exemple "jouet"

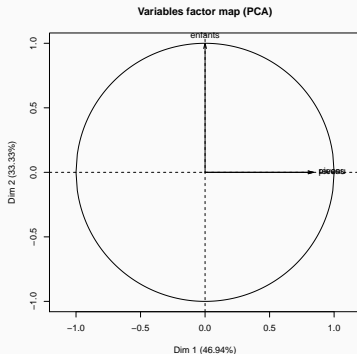
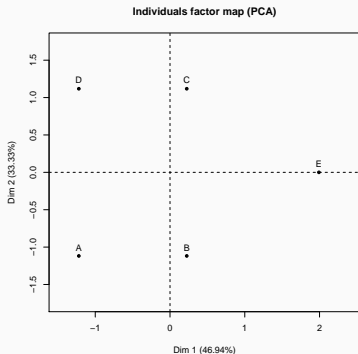
Ménage	Revenu	nb pièces	nb enfants
A	10 000	1	1
B	10 000	2	1
C	10 000	2	3
D	10 000	1	3
E	70 000	2	2

Diverses représentations



Fonction PCA

```
> library(FactoMineR)
> res.pca <- PCA(df)
```



Projection ACP

Le plan de projection est ici défini par $\mathcal{P} = \text{vect}(X_1 + X_2, X_3)$.

Notations

On se place dans l'espace \mathbb{R}^p muni de la distance euclidienne :

- $\langle e_i, e_j \rangle = \sum_{k=1}^p x_{i,k} x_{j,k}$
- $\|e_i\|^2 = \sum_{k=1}^p e_{i,k}^2$
- $d(e_i, e_j)^2 = \sum_{k=1}^p (x_{i,k} - x_{j,k})^2 = \|e_i - e_j\|^2$

Notations

On se place dans l'espace \mathbb{R}^p muni de la distance euclidienne :

- $\langle e_i, e_j \rangle = \sum_{k=1}^p x_{i,k} x_{j,k}$
- $\|e_i\|^2 = \sum_{k=1}^p e_{i,k}^2$
- $d(e_i, e_j)^2 = \sum_{k=1}^p (x_{i,k} - x_{j,k})^2 = \|e_i - e_j\|^2$

Centrage des données :

- Soit $G = \frac{1}{n} \sum_{i=1}^n e_i = (\bar{X}_1, \dots, \bar{X}_p)$ le centre de gravité du nuage des individus.
- Pour simplifier l'écriture de la méthode, on centre le nuage :

$$e_i^c = \begin{pmatrix} x_{i,1} - \bar{X}_1 \\ \vdots \\ x_{i,p} - \bar{X}_p \end{pmatrix} \quad \text{et} \quad \mathcal{N}^c = \{e_1^c, \dots, e_n^c\}.$$

Idée

Chercher à projeter les observations dans un sous-espace \mathcal{F} visible à l'œil qui "restitue au mieux" l'information contenue dans le tableau.

Idée

Chercher à projeter les observations dans un sous-espace \mathcal{F} visible à l'œil qui "restitue au mieux" l'information contenue dans le tableau.

L'inertie

- On appelle **inertie totale** du nuage de points \mathcal{N}

$$I(\mathcal{N}) = \frac{1}{n} \sum_{i=1}^n d(e_i, G)^2 = \frac{1}{n} \sum_{i=1}^n \|e_i - G\|^2 = \frac{1}{n} \sum_{i=1}^n \|e_i^c\|^2 = I(\mathcal{N}^c).$$

Idée

Chercher à projeter les observations dans un sous-espace \mathcal{F} visible à l'œil qui "restitue au mieux" l'information contenue dans le tableau.

L'inertie

- On appelle **inertie totale** du nuage de points \mathcal{N}

$$I(\mathcal{N}) = \frac{1}{n} \sum_{i=1}^n d(e_i, G)^2 = \frac{1}{n} \sum_{i=1}^n \|e_i - G\|^2 = \frac{1}{n} \sum_{i=1}^n \|e_i^c\|^2 = I(\mathcal{N}^c).$$

- On appelle **inertie portée par un sous espace \mathcal{F}** du nuage de points \mathcal{N}

$$I_{\mathcal{F}}(\mathcal{N}) = \frac{1}{n} \sum_{i=1}^n \|P_{\mathcal{F}}(e_i^c)\|^2,$$

où $P_{\mathcal{F}}(.)$ est la projection orthogonale sur \mathcal{F} .

Il est facile de voir que $I_{\mathcal{F}}(\mathcal{N}) \leq I(\mathcal{N})$: projeter fait perdre de l'inertie.

Il est facile de voir que $I_{\mathcal{F}}(\mathcal{N}) \leq I(\mathcal{N})$: projeter fait perdre de l'inertie.

Objectif

Trouver le sous espace \mathcal{F} qui minimise cette perte d'inertie, ou encore trouver le sous espace \mathcal{F} tel que

$I_{\mathcal{F}}(\mathcal{N})$ soit maximale.

Il est facile de voir que $I_{\mathcal{F}}(\mathcal{N}) \leq I(\mathcal{N})$: projeter fait perdre de l'inertie.

Objectif

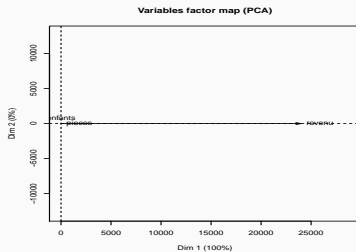
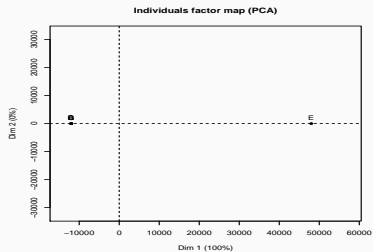
Trouver le sous espace \mathcal{F} qui minimise cette perte d'inertie, ou encore trouver le sous espace \mathcal{F} tel que

$$I_{\mathcal{F}}(\mathcal{N}) \text{ soit maximale.}$$

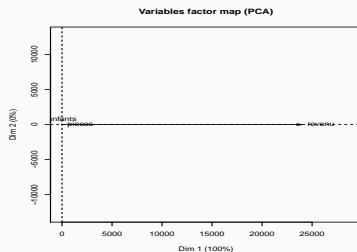
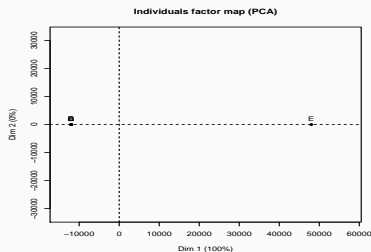
Un "léger" problème

1. Les variables ne sont généralement pas à la même échelle.
2. L'inertie est donc généralement "portée" par un sous groupe de variables.
3. Sur l'exemple, la variable revenu porte à elle seule la quasi totalité de l'inertie...

```
> res.pca1 <- PCA(df, scale.unit = FALSE)
```



```
> res.pca1 <- PCA(df, scale.unit = FALSE)
```



Centrage-réduction

Pour pallier à cette difficulté, on **réduit** les données initiales :

$$\mathbb{X} = \begin{matrix} & \mathbb{X}_1 & \dots & \mathbb{X}_p \\ \begin{matrix} e_1 \\ \vdots \\ e_n \end{matrix} & \begin{pmatrix} \tilde{x}_{11} & \dots & \tilde{x}_{1p} \\ \vdots & \vdots & \vdots \\ \tilde{x}_{n1} & \dots & \tilde{x}_{np} \end{pmatrix} \end{matrix} \quad \text{avec} \quad \tilde{x}_{ij} = \frac{x_{ij} - \bar{\mathbb{X}}_j}{\sigma_j} \quad \text{et} \quad \sigma_j = \sigma(\mathbb{X}_j).$$

Avec un léger abus, on note $x_{ij} = \tilde{x}_{ij}$.

"Meilleur" sous-espace de dimension 1

Il s'agit de chercher une droite vectorielle Δ_1 dirigée par un **vecteur unitaire** $u_1 \in \mathbb{R}^p$ telle que $I_{\Delta_1}(\mathcal{N})$ soit **maximale**.

"Meilleur" sous-espace de dimension 1

Il s'agit de chercher une droite vectorielle Δ_1 dirigée par un **vecteur unitaire** $u_1 \in \mathbb{R}^p$ telle que $I_{\Delta_1}(\mathcal{N})$ soit **maximale**.

Propriété

- $I_{\Delta_1}(\mathcal{N}) = \frac{1}{n} \sum_{i=1}^n \langle e_i, u_1 \rangle^2 = \frac{1}{n} \mathbb{C}'_1 \mathbb{C}_1$ où

$$\mathbb{C}_1 = (\langle e_1, u_1 \rangle, \dots, \langle e_n, u_1 \rangle)' = \mathbb{X}u_1.$$

"Meilleur" sous-espace de dimension 1

Il s'agit de chercher une droite vectorielle Δ_1 dirigée par un **vecteur unitaire** $u_1 \in \mathbb{R}^p$ telle que $I_{\Delta_1}(\mathcal{N})$ soit **maximale**.

Propriété

$$\bullet I_{\Delta_1}(\mathcal{N}) = \frac{1}{n} \sum_{i=1}^n \langle e_i, u_1 \rangle^2 = \frac{1}{n} \mathbb{C}'_1 \mathbb{C}_1 \text{ où}$$

$$\mathbb{C}_1 = (\langle e_1, u_1 \rangle, \dots, \langle e_n, u_1 \rangle)' = \mathbb{X}u_1.$$

Le problème mathématique

Chercher u_1 unitaire qui maximise $I_{\Delta_1}(\mathcal{N})$ revient à résoudre le **problème d'optimisation** suivant :

$$\text{maximiser } \frac{1}{n} u_1' \mathbb{X}' \mathbb{X} u_1 \text{ sous la contrainte } \|u_1\| = 1.$$

Propriété

Un vecteur propre unitaire u_1 rendant l'inertie $I_{\Delta_1}(\mathcal{N})$ maximale est un vecteur propre normé associé à la **plus grande valeur propre** λ_1 de la matrice $\Sigma = \frac{1}{n}\mathbb{X}'\mathbb{X}$.

Propriété

Un vecteur propre unitaire u_1 rendant l'inertie $I_{\Delta_1}(\mathcal{N})$ maximale est un vecteur propre normé associé à la **plus grande valeur propre** λ_1 de la matrice $\Sigma = \frac{1}{n}\mathbb{X}'\mathbb{X}$.

Remarques

- La matrice d'inertie $\Sigma = \frac{1}{n}\mathbb{X}'\mathbb{X}$ étant symétrique et définie positive, elle est diagonalisable et **toutes ses valeurs propres sont positives ou nulles**.
- u_1 est appelé **premier axe factoriel**.

Exemple

- Sur l'exemple "jouet" on a :

```
> df1 <- as.matrix(df)
> n <- nrow(df1)
> Xbar <- apply(df1,2,mean)
> stdX <- sqrt(apply(df1,2,var)*(n-1)/n)
> dfc <- sweep(df1,2,Xbar,FUN="-")
> dfcr <- sweep(dfc,2,stdX,FUN="/")
> 1/nrow(dfcr)*t(dfcr)%*%dfcr
```

##		revenu	pieces	enfants
##	revenu	1.0000000	0.4082483	0
##	pieces	0.4082483	1.0000000	0
##	enfants	0.0000000	0.0000000	1

- Premier axe factoriel :

```
> XX <- 1/nrow(dfcr)*t(dfcr)%*%dfcr
> u1 <- eigen(XX)$vectors[,1]
> u1
## [1] 0.7071068 0.7071068 0.0000000
```

- Coordonnées sur le premier axe :

```
> dfcr %*%u1
##           [,1]
## A -1.2195788
## B  0.2237969
## C  0.2237969
## D -1.2195788
## E  1.9915638
```

- Que l'on retrouve dans les sorties de PCA :

```
> res.pca$ind$coord[,1]
##           A           B           C           D           E
## -1.2195788  0.2237969  0.2237969 -1.2195788  1.9915638
```

Problème

Trouver une droite vectorielle Δ_2 dirigée par un **vecteur normé** u_2 telle que

$$\begin{cases} l_{\Delta_2}(\mathcal{N}) = u_2' \Sigma u_2 \text{ maximale} \\ \|u_2\|^2 = u_2' u_2 = 1 \\ \langle u_2, u_1 \rangle = u_2' u_1 = 0 \end{cases}$$

Solution

Un vecteur unitaire u_2 solution du problème précédent est un vecteur propre normé associé à la **deuxième plus grande valeur propre** λ_2 de la matrice $\Sigma = \frac{1}{n} \mathbb{X}' \mathbb{X}$.

Question

Le plan $\text{vect}(u_1, u_2)$ est-il le meilleur sous-espace de dimension 2 en terme de maximisation d'inertie projetée ?

Question

Le plan $\text{vect}(u_1, u_2)$ est-il le meilleur sous-espace de dimension 2 en terme de maximisation d'inertie projetée ?

Réponse

La réponse est oui ! On déduit ainsi qu'un sous-espace de dimension $q < p$ qui maximise l'inertie projetée est donné par $\text{vect}(u_1, \dots, u_q)$ où u_j est un vecteur normé associé à la $j^{\text{ème}}$ plus grande valeur propre λ_j de $\Sigma = \frac{1}{n} \mathbb{X}' \mathbb{X}$.

Question

Le plan $\text{vect}(u_1, u_2)$ est-il le meilleur sous-espace de dimension 2 en terme de maximisation d'inertie projetée ?

Réponse

La réponse est oui ! On déduit ainsi qu'un sous-espace de dimension $q < p$ qui maximise l'inertie projetée est donné par $\text{vect}(u_1, \dots, u_q)$ où u_j est un vecteur normé associé à la $j^{\text{ème}}$ plus grande valeur propre λ_j de $\Sigma = \frac{1}{n} \mathbb{X}' \mathbb{X}$.

Conclusion : chercher les axes factoriels revient à diagonaliser $\Sigma = \frac{1}{n} \mathbb{X}' \mathbb{X}$.

ACP \approx changement de base

Base canonique

$$X = \begin{matrix} & \mathbb{X}_1 & \dots & \mathbb{X}_p \\ \begin{pmatrix} x_{11} & \dots & x_{1p} \\ \vdots & \vdots & \vdots \\ x_{n1} & \dots & x_{np} \end{pmatrix} \end{matrix}$$

Base $\{u_1, \dots, u_p\}$

$$X = \begin{matrix} & \mathbb{C}_1 & \dots & \mathbb{C}_p \\ \begin{pmatrix} c_{11} & \dots & c_{1p} \\ \vdots & \vdots & \vdots \\ c_{n1} & \dots & c_{np} \end{pmatrix} \end{matrix}$$

ACP \approx changement de base

Base canonique

$$X = \begin{pmatrix} \mathbb{X}_1 & \dots & \mathbb{X}_p \\ x_{11} & \dots & x_{1p} \\ \vdots & \vdots & \vdots \\ x_{n1} & \dots & x_{np} \end{pmatrix}$$

Base $\{u_1, \dots, u_p\}$

$$X = \begin{pmatrix} \mathbb{C}_1 & \dots & \mathbb{C}_p \\ c_{11} & \dots & c_{1p} \\ \vdots & \vdots & \vdots \\ c_{n1} & \dots & c_{np} \end{pmatrix}$$

Propriété

1. $\mathbb{C}_j = \mathbb{X}u_j = \sum_{k=1}^p u_{kj}\mathbb{X}_k$

ACP \approx changement de base

Base canonique

$$X = \begin{pmatrix} \mathbb{X}_1 & \dots & \mathbb{X}_p \\ x_{11} & \dots & x_{1p} \\ \vdots & \vdots & \vdots \\ x_{n1} & \dots & x_{np} \end{pmatrix}$$

Base $\{u_1, \dots, u_p\}$

$$X = \begin{pmatrix} \mathbb{C}_1 & \dots & \mathbb{C}_p \\ c_{11} & \dots & c_{1p} \\ \vdots & \vdots & \vdots \\ c_{n1} & \dots & c_{np} \end{pmatrix}$$

Propriété

1. $\mathbb{C}_j = \mathbb{X}u_j = \sum_{k=1}^p u_{kj}\mathbb{X}_k$
2. \mathbb{C}_j centrée, $\mathbf{V}(\mathbb{C}_j) = \frac{1}{n}\|\mathbb{C}_j\|^2 = \lambda_j = l_{\Delta_j}(\mathcal{N})$ et $\rho(\mathbb{C}_j, \mathbb{C}_k) = 0$, $k \neq j$.

ACP \approx changement de base

Base canonique

$$X = \begin{pmatrix} \mathbb{X}_1 & \dots & \mathbb{X}_p \\ x_{11} & \dots & x_{1p} \\ \vdots & \vdots & \vdots \\ x_{n1} & \dots & x_{np} \end{pmatrix}$$

Base $\{u_1, \dots, u_p\}$

$$X = \begin{pmatrix} \mathbb{C}_1 & \dots & \mathbb{C}_p \\ c_{11} & \dots & c_{1p} \\ \vdots & \vdots & \vdots \\ c_{n1} & \dots & c_{np} \end{pmatrix}$$

Propriété

1. $\mathbb{C}_j = \mathbb{X}u_j = \sum_{k=1}^p u_{kj}\mathbb{X}_k$
2. \mathbb{C}_j centrée, $\mathbf{V}(\mathbb{C}_j) = \frac{1}{n}\|\mathbb{C}_j\|^2 = \lambda_j = l_{\Delta_j}(\mathcal{N})$ et $\rho(\mathbb{C}_j, \mathbb{C}_k) = 0$, $k \neq j$.

Conclusion

L'ACP normée remplace les variables d'origines \mathbb{X}_j par de nouvelles variables \mathbb{C}_j appelées **composantes principales**, de variance maximale, non corrélées deux à deux et qui s'expriment comme combinaison linéaire des variables d'origine.

Sélections exhaustive et pas à pas

Régression sur composantes

Régression sur composantes principales (PCR)

Rappels ACP

Retour à PCR

Régression PLS

Choix du nombre de composantes

Bibliographie

1. Choisir un nombre de composantes m .

Etapas

1. Choisir un nombre de composantes m .
2. Calculer les composantes principales $Z_1 = w_1'X, \dots, Z_m = w_m'X$.

Etapes

1. Choisir un nombre de composantes m .
2. Calculer les composantes principales $Z_1 = w_1'X, \dots, Z_m = w_m'X$.
3. Calculer l'estimateur des MCO dans l'espace des composantes principales.

Etapas

1. Choisir un nombre de composantes m .
2. Calculer les composantes principales $Z_1 = w_1'X, \dots, Z_m = w_m'X$.
3. Calculer l'estimateur des MCO dans l'espace des composantes principales.

Remarque importante

- Comme l'ACP, méthode non invariante par changement d'échelle.

Etapes

1. Choisir un nombre de composantes m .
2. Calculer les composantes principales $Z_1 = w_1'X, \dots, Z_m = w_m'X$.
3. Calculer l'estimateur des MCO dans l'espace des composantes principales.

Remarque importante

- Comme l'ACP, méthode non invariante par changement d'échelle.
- Il est souvent préférable de (centrer)-réduire les données au préalable.

Etapes

1. Choisir un nombre de composantes m .
2. Calculer les composantes principales $Z_1 = w_1'X, \dots, Z_m = w_m'X$.
3. Calculer l'estimateur des MCO dans l'espace des composantes principales.

Remarque importante

- Comme l'ACP, méthode non invariante par changement d'échelle.
- Il est souvent préférable de (centrer)-réduire les données au préalable.
- Souvent fait par défaut par les logiciels.

Algorithme PCR

1. Faire l'ACP du tableau \mathbb{X} (centré/réduit) $\Rightarrow w_1, \dots, w_m$ axes factoriels et $Z_k = \sum_{j=1}^p w_{kj} X_j, k = 1, \dots, m$ composantes principales.

Algorithme PCR

1. Faire l'ACP du tableau \mathbb{X} (centré/réduit) $\Rightarrow w_1, \dots, w_m$ axes factoriels et $Z_k = \sum_{j=1}^p w_{kj} X_j, k = 1, \dots, m$ composantes principales.
2. Effectuer la régression de Y sur les Z_j :

$$Y = \alpha_0 + \alpha_1 Z_1 + \dots + \alpha_m Z_m + \varepsilon$$

Estimateurs MCO

Les composantes principales étant orthogonales entre elles, on a

$$\hat{\alpha}_0 = \bar{Y} \quad \text{et} \quad \hat{\alpha}_k = \frac{\langle Z_k, Y \rangle}{\|Z_k\|^2}.$$

Elle se calcule comme précédemment

$$\begin{aligned}f_n(x) &= \hat{\alpha}_0 + \hat{\alpha}_1 z_1 + \dots + \hat{\alpha}_m z_m \\ &= \hat{\beta}_0 + \hat{\beta}_1 x_1 + \dots + \hat{\beta}_p x_p\end{aligned}$$

avec

$$\hat{\beta}_0 = \hat{\alpha}_0 \quad \text{et} \quad \hat{\beta}_j = \sum_{k=1}^m \hat{\alpha}_k w_{kj}.$$

Elle se calcule comme précédemment

$$\begin{aligned}f_n(x) &= \hat{\alpha}_0 + \hat{\alpha}_1 z_1 + \dots + \hat{\alpha}_m z_m \\ &= \hat{\beta}_0 + \hat{\beta}_1 x_1 + \dots + \hat{\beta}_p x_p\end{aligned}$$

avec

$$\hat{\beta}_0 = \hat{\alpha}_0 \quad \text{et} \quad \hat{\beta}_j = \sum_{k=1}^m \hat{\alpha}_k w_{kj}.$$

Remarque

- Seconde écriture souvent privilégiée pour des raisons d'interprétation.
- C'est généralement cette écriture qui est renvoyée par les logiciels.

Exemple

- Jeu de données **Hitters**

```
> library(ISLR)
> Hitters <- na.omit(Hitters)
> dim(Hitters)
## [1] 263 20
> names(Hitters)
## [1] "AtBat"      "Hits"      "HmRun"     "Runs"      "RBI"      "Walks"
## [7] "Years"     "CAtBat"    "CHits"     "CHmRun"    "CRuns"    "CRBI"
## [13] "CWalks"    "League"    "Division"  "PutOuts"   "Assists"  "Errors"
## [19] "Salary"    "NewLeague"
```

Exemple

- Jeu de données **Hitters**

```
> library(ISLR)
> Hitters <- na.omit(Hitters)
> dim(Hitters)
## [1] 263 20
> names(Hitters)
## [1] "AtBat"      "Hits"       "HmRun"      "Runs"       "RBI"        "Walks"
## [7] "Years"     "CAtBat"     "CHits"      "CHmRun"     "CRuns"      "CRBI"
## [13] "CWalks"    "League"     "Division"   "PutOuts"    "Assists"    "Errors"
## [19] "Salary"     "NewLeague"
```

Le problème

Expliquer **Salary** par les autres variables.

La fonction `pcr`

- La fonction `pcr` du package `pls` permet d'effectuer la régression :

```
> library(pls)
> pcr.fit <- pcr(Salary~.,data=Hitters,scale=TRUE,ncomp=19)
```

La fonction `pcr`

- La fonction `pcr` du package `pls` permet d'effectuer la régression :

```
> library(pls)
> pcr.fit <- pcr(Salary~.,data=Hitters,scale=TRUE,ncomp=19)
```

- On peut obtenir les **coefficients** pour $m = 5$ (dans l'espace des variables initiales) avec :

```
> coefficients(pcr.fit,ncomp=5)
## , , 5 comps
##
##           Salary
## AtBat      28.766042
## Hits       30.447021
## HmRun       25.844498
## Runs       33.000876
```

Attention

- Ces coefficients sont calculés pour les données **réduites** ;
- Il faut diviser par les **écart-types** si on veut les valeurs sur les **variables initiales** ($\hat{\beta}_j$).

Prédiction

- Comme d'habitude, la fonction `predict` permet de calculer des prévisions.
- Pour obtenir les **valeurs prédites (ou ajustées)** des 5 premiers individus, on utilise

```
> predict(pcr.fit,newdata=Hitters[1:5,],ncomp=5)
## , , 5 comps
##
##           Salary
## -Alan Ashby    495.0068
## -Alvin Davis   547.8896
## -Andre Dawson  1010.2236
## -Andres Galarra 409.8232
## -Alfredo Griffin 524.9053
```

Sélections exhaustive et pas à pas

Régression sur composantes

Régression sur composantes principales (PCR)

Rappels ACP

Retour à PCR

Régression PLS

Choix du nombre de composantes

Bibliographie

- Idem à PCR : objectif réduction de dimension.
- On cherche toujours des CL Z_1, \dots, Z_m des variables explicatives.

- Idem à PCR : objectif réduction de dimension.
- On cherche toujours des CL Z_1, \dots, Z_m des variables explicatives.
- Différence avec PCR : les composantes PLS vont être construites en maximisant le lien avec la variable à expliquer Y .

- Y variable à expliquer et $X = (X_1, \dots, X_p)$ variables explicatives.
- $\mathbb{Y} = (y_1, \dots, y_n)$ et \mathbb{X} la matrice $n \times p$ des variables explicatives.
- Comme pour PCR, les covariables sont supposées **centrées réduites**.

Propriété

Les **poids** des composantes principales s'obtiennent en résolvant le problème

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

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

Rappel ACP

Propriété

Les **poids** des composantes principales s'obtiennent en résolvant le problème

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

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

Remarque

- Les **composantes principales** se calculent en cherchant la direction de \mathbb{R}^p qui maximise la **variabilité** des X .

Rappel ACP

Propriété

Les **poids** des composantes principales s'obtiennent en résolvant le problème

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

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

Remarque

- Les **composantes principales** se calculent en cherchant la direction de \mathbb{R}^p qui maximise la **variabilité** des X .
- La **variable à expliquer** Y n'est **pas prise en compte**.

Propriété

Les **poids** des composantes principales s'obtiennent en résolvant le problème

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

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

Remarque

- Les **composantes principales** se calculent en cherchant la direction de \mathbb{R}^p qui maximise la **variabilité** des X .
- La **variable à expliquer** Y n'est **pas prise en compte**.
- L'approche **PLS** va également chercher la **direction la plus corrélée à Y** .

Composantes PLS

Définition

Les poids w_k de la k^e composante PLS sont solution de

$$\max_{w \in \mathbb{R}^d} \text{corr}^2(\mathbb{Y}, \mathbb{X}w) \mathbf{V}(\mathbb{X}w)$$

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

Composantes PLS

Définition

Les poids w_k de la k^e composante PLS sont solution de

$$\max_{w \in \mathbb{R}^d} \text{corr}^2(\mathbb{Y}, \mathbb{X}w) \mathbf{V}(\mathbb{X}w)$$

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

Remarques

- Selon [Hastie et al., 2009], la variance a souvent une place plus importante que la corrélation dans le critère

Composantes PLS

Définition

Les poids w_k de la k^e composante PLS sont solution de

$$\max_{w \in \mathbb{R}^d} \text{corr}^2(\mathbb{Y}, \mathbb{X}w) \mathbf{V}(\mathbb{X}w)$$

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

Remarques

- Selon [Hastie et al., 2009], la variance a souvent une place plus importante que la corrélation dans le critère \implies les deux approches sont souvent proches.

Composantes PLS

Définition

Les poids w_k de la k^e composante PLS sont solution de

$$\max_{w \in \mathbb{R}^d} \text{corr}^2(\mathbb{Y}, \mathbb{X}w) \mathbf{V}(\mathbb{X}w)$$

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

Remarques

- Selon [Hastie et al., 2009], la variance a souvent une place plus importante que la corrélation dans le critère \implies les deux approches sont souvent proches.
- PLS est souvent recommandée lorsque les covariables sont très corrélées et/ou $p > n$.

Composantes PLS

Définition

Les poids w_k de la k^e composante PLS sont solution de

$$\max_{w \in \mathbb{R}^d} \text{corr}^2(\mathbb{Y}, \mathbb{X}w) \mathbf{V}(\mathbb{X}w)$$

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

Remarques

- Selon [Hastie et al., 2009], la variance a souvent une place plus importante que la corrélation dans le critère \implies les deux approches sont souvent proches.
- PLS est souvent recommandée lorsque les covariables sont très corrélées et/ou $p > n$.
- Les composantes se calculent en utilisant des méthodes standard d'optimisation sous contraintes (Lagrangien...).

- Les poids de la première composante sont donnés par

$$w_1 = \frac{X^t Y}{\|X^t Y\|}.$$

- Les poids de la première composante sont donnés par

$$w_1 = \frac{\mathbf{X}^t \mathbf{Y}}{\|\mathbf{X}^t \mathbf{Y}\|}.$$

- Les composantes suivantes s'obtiennent de façon **récursive** en expliquant la **partie résiduelle de \mathbf{Y}** par la "**meilleure**" combinaison linéaire orthogonale à $\text{vect}(\mathbf{Z}_1, \dots, \mathbf{Z}_\ell)$ [Cornillon et al., 2019].

L'algorithme PLS, voir [Hastie et al., 2009]

1. On pose $\hat{\mathbb{Y}}^{(0)} = \bar{\mathbb{Y}}$ et $\mathbb{X}_j^{(0)} = \mathbb{X}_j, j = 1, \dots, p$.

2. Pour $k = 1, \dots, p$:

2.1 $\mathbb{Z}_k = \sum_{j=1}^p w_{kj} \mathbb{X}_j^{(k-1)}$ avec $w_{kj} = \langle \tilde{\mathbb{X}}_j^{(k-1)}, \mathbb{Y} \rangle$.

2.2 $\hat{\alpha}_k = \frac{\langle \mathbb{Z}_k, \mathbb{Y} \rangle}{\langle \mathbb{Z}_k, \mathbb{Z}_k \rangle}$.

2.3 $\hat{\mathbb{Y}}^{(k)} = \hat{\mathbb{Y}}^{(k-1)} + \hat{\alpha}_k \mathbb{Z}_k$.

2.4 Orthogonaliser $\mathbb{X}_j^{(k-1)}$ par rapport à \mathbb{Z}_k :

$$\mathbb{X}_j^{(k)} = \mathbb{X}_j^{(k-1)} - \frac{\langle \mathbb{Z}_k, \mathbb{X}_j^{(k-1)} \rangle}{\langle \mathbb{Z}_k, \mathbb{Z}_k \rangle} \mathbb{Z}_k.$$

3. **Sortie** : les suites de coefficients $(\hat{\alpha}_k)_k$ et de poids $(w_k)_k$.

- Idem à PCR : pour un nombre de composantes m fixé, on prédit

$$\begin{aligned}f_n(x) &= \hat{\alpha}_0 + \hat{\alpha}_1 z_1 + \dots + \hat{\alpha}_m z_m \\ &= \hat{\beta}_0 + \hat{\beta}_1 x_1 + \dots + \hat{\beta}_p x_p\end{aligned}$$

avec

$$\hat{\beta}_0 = \hat{\alpha}_0 \quad \text{et} \quad \hat{\beta}_j = \sum_{k=1}^m \hat{\alpha}_k w_{kj}.$$

- Idem à PCR : pour un nombre de composantes m fixé, on prédit

$$\begin{aligned}f_n(x) &= \hat{\alpha}_0 + \hat{\alpha}_1 z_1 + \dots + \hat{\alpha}_m z_m \\ &= \hat{\beta}_0 + \hat{\beta}_1 x_1 + \dots + \hat{\beta}_p x_p\end{aligned}$$

avec

$$\hat{\beta}_0 = \hat{\alpha}_0 \quad \text{et} \quad \hat{\beta}_j = \sum_{k=1}^m \hat{\alpha}_k w_{kj}.$$

- Ce sont généralement les coefficients $\hat{\beta}_j$ qui sont renvoyés par les logiciels.

- La régression PLS s'effectue à l'aide de la fonction `plsr` du package `pls`.

```
> pls.fit <- plsr(Salary~.,data=Hitters,scale=TRUE)
```

- La régression PLS s'effectue à l'aide de la fonction `plsr` du package `pls`.

```
> pls.fit <- plsr(Salary~.,data=Hitters,scale=TRUE)
```

- On obtient les **coefficients PLS** pour la première composante avec

```
> coefficients(pls.fit,ncomp = 1)
## , , 1 comps
##
##           Salary
## AtBat      25.0420570
## Hits       27.8270677
## HmRun       21.7597795
## Runs       26.6334747
## RBI        28.5110396
```

- Les deux approches permettent de réduire la dimension en considérant un nombre restreint de composantes Z_1, \dots, Z_m .
- Dans les deux cas, ces composantes
 1. sont des combinaisons linéaires des variables X_1, \dots, X_p .
 2. sont orthogonales.
- La seule différence entre les deux approches se situe dans le processus de construction des composantes.

PCR vs PLS

- Les deux approches permettent de **réduire la dimension** en considérant un **nombre restreint de composantes** Z_1, \dots, Z_m .
- Dans les deux cas, ces composantes
 1. sont des **combinaisons linéaires** des variables X_1, \dots, X_p .
 2. sont **orthogonales**.
- La seule **différence** entre les deux approches se situe dans le processus de **construction des composantes**.

Remarque

- **PCR** utilise **uniquement les X** pour construire les composantes.
- **PLS** utilise les **X et Y** .

Sélections exhaustive et pas à pas

Régression sur composantes

Régression sur composantes principales (PCR)

Rappels ACP

Retour à PCR

Régression PLS

Choix du nombre de composantes

Bibliographie

- PCR et PLS construisent des **composantes** Z_1, \dots, Z_p .
- Chaque composante s'écrit comme une **combinaison linéaire** des $X_j, j = 1, \dots, p$.
- Pour une valeur de $m \leq p$, on peut prédire selon

$$f_n^m(x) = \hat{\alpha}_0 + \hat{\alpha}_1 z_1 + \dots + \hat{\alpha}_m z_m.$$

- PCR et PLS construisent des **composantes** Z_1, \dots, Z_p .
- Chaque composante s'écrit comme une **combinaison linéaire** des $X_j, j = 1, \dots, p$.
- Pour une valeur de $m \leq p$, on peut prédire selon

$$f_n^m(x) = \hat{\alpha}_0 + \hat{\alpha}_1 z_1 + \dots + \hat{\alpha}_m z_m.$$

Propriété

Si $m = p$ alors **PCR et PLS sont équivalents au modèle linéaire classique** par MCO avec les p variables X_1, \dots, X_p .

- PCR et PLS construisent des **composantes** Z_1, \dots, Z_p .
- Chaque composante s'écrit comme une **combinaison linéaire** des $X_j, j = 1, \dots, p$.
- Pour une valeur de $m \leq p$, on peut prédire selon

$$f_n^m(x) = \hat{\alpha}_0 + \hat{\alpha}_1 z_1 + \dots + \hat{\alpha}_m z_m.$$

Propriété

Si $m = p$ alors **PCR et PLS sont équivalents** au modèle linéaire classique par MCO avec les p variables X_1, \dots, X_p .

Conséquence

- Si $m = p$ alors PCR et PLS ne sont d'**aucune utilité**.
- Il est donc **crucial** de choisir le "**bon**" nombre de composantes $m \in \{1, \dots, p\}$.

- Pour $m \in \{1, \dots, p\}$ PCR et PLS fournissent un **algorithme de prévision** f_n^m .
- Les méthodes permettant de choisir m sont identiques aux **procédures de calibration en machine learning** :

- Pour $m \in \{1, \dots, p\}$ PCR et PLS fournissent un algorithme de prévision f_n^m .
- Les méthodes permettant de choisir m sont identiques aux procédures de calibration en machine learning :
 1. Choix d'un risque (erreur quadratique de prévision...)

- Pour $m \in \{1, \dots, p\}$ PCR et PLS fournissent un **algorithme de prévision** f_n^m .
- Les méthodes permettant de choisir m sont identiques aux **procédures de calibration en machine learning** :
 1. Choix d'un **risque** (erreur quadratique de prévision...)
 2. Choix d'un **algorithme pour calculer ce risque** (validation hold out, validation croisée, LOO...)

- Pour $m \in \{1, \dots, p\}$ PCR et PLS fournissent un **algorithme de prévision** f_n^m .
- Les méthodes permettant de choisir m sont identiques aux **procédures de calibration en machine learning** :
 1. Choix d'un **risque** (erreur quadratique de prévision...)
 2. Choix d'un **algorithme pour calculer ce risque** (validation hold out, validation croisée, LOO...)
 3. Sélection du paramètre qui **minimise le risque** calculé.

Exemple : validation croisée

- **Risque RMSEP** : $\mathcal{R}(f) = \sqrt{\mathbf{E}[(Y - f(X))^2]}$.
- **Algorithme** : validation croisée K blocs.

Choix du nombre de composantes

Entrées :

- les observations $(x_1, y_1), \dots, (x_n, y_n)$;
- $\{\mathcal{I}_1, \dots, \mathcal{I}_K\}$ une partition de $\{1, \dots, n\}$ en K blocs ;

Pour $m = 1, \dots, p$

- Pour $k = 1, \dots, K$:
 1. Construire la régression sur m composantes en utilisant l'ensemble des données privé du k^e bloc, c'est-à-dire $\{(x_i, y_i) : i \in \{1, \dots, n\} \setminus \mathcal{I}_k\} \implies f_k^m$.
 2. Calculer la valeur prédite par l'algorithme pour chaque observation du bloc k :
 $f_k^m(x_i), i \in \mathcal{I}_k$.

Retourner : pour $m = 1, \dots, p$

$$\hat{\mathcal{R}}(f_n^m) = \frac{1}{n} \sum_{k=1}^K \sum_{i \in \mathcal{I}_k} \sqrt{(y_i - \hat{f}_k^m(x_i))^2}.$$

Exemple : PCR

- Il suffit d'utiliser l'argument `validation="CV"` dans `pcr` :

```
> set.seed(1234)
> pcr.val <- pcr(Salary~.,data=Hitters,scale=TRUE,validation="CV")
> RMSEP(pcr.val)
```

	(Intercept)	1 comps	2 comps	3 comps	4 comps	5 comps	6 comps
## CV	452	353.4	351.8	351.7	349.4	345.4	343.6
## adjCV	452	353.0	351.4	351.3	348.9	344.8	342.8

	7 comps	8 comps	9 comps	10 comps	11 comps	12 comps	13 comps
## CV	343.6	345.3	347.0	349.3	349.4	351.5	355.2
## adjCV	342.9	344.4	346.1	348.1	348.0	350.1	353.8

	14 comps	15 comps	16 comps	17 comps	18 comps	19 comps
## CV	349.4	348.5	339.6	338.7	337.2	339.5
## adjCV	347.6	346.8	337.9	336.9	335.4	337.5

Exemple : PCR

- Il suffit d'utiliser l'argument `validation="CV"` dans `pcr` :

```
> set.seed(1234)
> pcr.val <- pcr(Salary~.,data=Hitters,scale=TRUE,validation="CV")
> RMSEP(pcr.val)
```

##	(Intercept)	1 comps	2 comps	3 comps	4 comps	5 comps	6 comps
## CV	452	353.4	351.8	351.7	349.4	345.4	343.6
## adjCV	452	353.0	351.4	351.3	348.9	344.8	342.8
##	7 comps	8 comps	9 comps	10 comps	11 comps	12 comps	13 comps
## CV	343.6	345.3	347.0	349.3	349.4	351.5	355.2
## adjCV	342.9	344.4	346.1	348.1	348.0	350.1	353.8
##	14 comps	15 comps	16 comps	17 comps	18 comps	19 comps	
## CV	349.4	348.5	339.6	338.7	337.2	339.5	
## adjCV	347.6	346.8	337.9	336.9	335.4	337.5	

Conclusion

On choisira **18 composantes** pour PCR.

Exemple : PLS

```
> set.seed(1234)
> pls.val <- plsr(Salary~.,data=Hitters,scale=TRUE,validation="CV")
> RMSEP(pls.val)
```

##	(Intercept)	1 comps	2 comps	3 comps	4 comps	5 comps	6 comps
## CV	452	348.5	345.6	345.7	345.1	348.4	349.0
## adjCV	452	348.1	344.8	344.7	344.1	346.7	346.9
##	7 comps	8 comps	9 comps	10 comps	11 comps	12 comps	13 comps
## CV	345.7	341.4	341.8	339.8	338.0	336.7	339.0
## adjCV	343.7	339.5	339.9	338.0	336.3	335.0	337.2
##	14 comps	15 comps	16 comps	17 comps	18 comps	19 comps	
## CV	338.9	338.2	338.2	338.2	338.1	339.5	
## adjCV	337.0	336.3	336.4	336.3	336.3	337.5	

Exemple : PLS

```
> set.seed(1234)
> pls.val <- plsr(Salary~.,data=Hitters,scale=TRUE,validation="CV")
> RMSEP(pls.val)
```

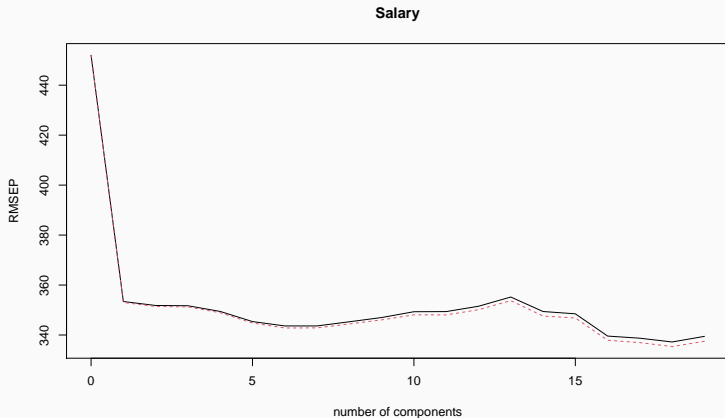
##	(Intercept)	1 comps	2 comps	3 comps	4 comps	5 comps	6 comps
## CV	452	348.5	345.6	345.7	345.1	348.4	349.0
## adjCV	452	348.1	344.8	344.7	344.1	346.7	346.9
##	7 comps	8 comps	9 comps	10 comps	11 comps	12 comps	13 comps
## CV	345.7	341.4	341.8	339.8	338.0	336.7	339.0
## adjCV	343.7	339.5	339.9	338.0	336.3	335.0	337.2
##	14 comps	15 comps	16 comps	17 comps	18 comps	19 comps	
## CV	338.9	338.2	338.2	338.2	338.1	339.5	
## adjCV	337.0	336.3	336.4	336.3	336.3	337.5	

Conclusion

On choisira 12 composantes pour PLS.

- On peut également visualiser les **erreurs** en fonction du **nombre de composantes** avec `validationplot`.

```
> validationplot(pcr.val)
```



- Deux techniques pour réduire la dimension :
 1. sélection de variables.
 2. régression sur composantes.

- Deux techniques pour réduire la dimension :
 1. sélection de variables.
 2. régression sur composantes.
- A utiliser lorsque :
 1. p est grand.
 2. les $X_j, j = 1, \dots, p$ sont "corrélés".

Sélections exhaustive et pas à pas

Régression sur composantes

Régression sur composantes principales (PCR)


Rappels ACP

Retour à PCR

Régression PLS

Choix du nombre de composantes

Bibliographie

 Cornillon, P., Hengartner, N., Matzner-Løber, E., and Rouvière, L. (2019).

Régression avec R.

EDP Sciences.

 Hastie, T., Tibshirani, R., and Friedman, J. (2009).

The Elements of Statistical Learning : Data Mining, Inference, and Prediction.

Springer, second edition.

Troisième partie III

Régularisation

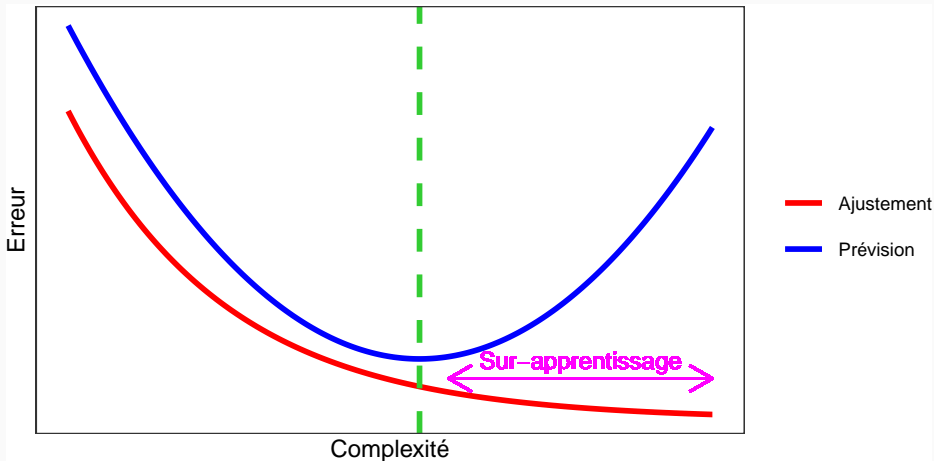
Régression ridge

Régression Lasso

Variantes de ridge/lasso

Discrimination binaire

Bibliographie



Complexité linéaire

Le **nombre de variables** est une mesure de la complexité des algorithmes linéaires.

- On génère des données $(x_i, y_i), i = 1, \dots, 500$ selon le modèle

$$y_i = 1x_{i1} + 0x_{i2} + \dots + 0x_{iq} + \varepsilon_i$$

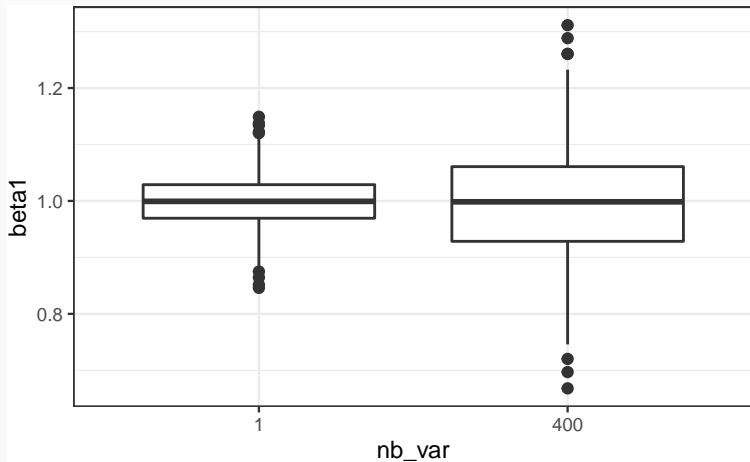
où $x_1, \dots, x_q, \varepsilon$ sont i.i.d. de loi $\mathcal{N}(0, 1)$.

- On génère des données (x_i, y_i) , $i = 1, \dots, 500$ selon le modèle

$$y_i = 1x_{i1} + 0x_{i2} + \dots + 0x_{iq} + \varepsilon_i$$

où $x_1, \dots, x_q, \varepsilon$ sont i.i.d. de loi $\mathcal{N}(0, 1)$.

- Seule X_1 est **explicative**, les $q - 1$ autres variables peuvent être vues comme du **bruit**.
- On calcule l'**estimateur de MCO de β_1** sur 1000 répétitions. On trace les boxplot de ces estimateurs pour $q = 0$ et $q = 400$.



Conclusion

Plus de **variance** (donc **moins de précision**) lorsque le nombre de variables inutiles augmente.

- Lorsque le nombre de variables d est grand, les estimateurs des moindres carrés du modèle linéaire

$$Y = \beta_1 X_1 + \dots + \beta_d X_d + \varepsilon$$

possèdent généralement une grande variance.

- Lorsque le nombre de variables d est grand, les estimateurs des moindres carrés du modèle linéaire

$$Y = \beta_1 X_1 + \dots + \beta_d X_d + \varepsilon$$

possèdent généralement une grande variance.

Idée des méthodes pénalisés

- Contraindre la valeur des estimateurs des moindres carrés de manière à réduire la variance (quitte à augmenter un peu le biais).

- Lorsque le nombre de variables d est grand, les estimateurs des moindres carrés du modèle linéaire

$$Y = \beta_1 X_1 + \dots + \beta_d X_d + \varepsilon$$

possèdent généralement une grande variance.

Idée des méthodes pénalisés

- Contraindre la valeur des estimateurs des moindres carrés de manière à réduire la variance (quitte à augmenter un peu le biais).
- Comment ? En imposant une contrainte sur la valeur des estimateurs des moindres carrés :

$$\hat{\beta}^{pen} = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^n \left(y_i - \sum_{j=1}^d x_{ij} \beta_j \right)^2$$

sous la contrainte $\|\beta\|_? \leq t$.

- Quelle **norme** utiliser pour la contrainte ?

Questions

- Quelle **norme** utiliser pour la contrainte ?
- **Existence/unicité** des estimateurs ? **Solutions explicites** du problème d'optimisation ?

- Quelle **norme** utiliser pour la contrainte ?
- **Existence/unicité** des estimateurs ? **Solutions explicites** du problème d'optimisation ?
- Comment **choisir t** ?
 - t petit \implies estimateurs **contraints** (proche de 0) ;
 - t grand \implies estimateurs des **moindres carrés** (non pénalisés).

Régression ridge

Régression Lasso

Variantes de ridge/lasso

Discrimination binaire

Bibliographie

- La **régression ridge** consiste à minimiser le critère des moindres carrés pénalisé par la norme 2 des coefficients.

Définition

1. Les **estimateurs ridge** $\hat{\beta}^R$ s'obtiennent en minimisant

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

- La **régression ridge** consiste à minimiser le critère des moindres carrés pénalisé par la norme 2 des coefficients.

Définition

1. Les **estimateurs ridge** $\hat{\beta}^R$ s'obtiennent en minimisant

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

2. ou de façon **équivalente**

$$\hat{\beta}^R = \underset{\beta}{\operatorname{argmin}} \left\{ \sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^d x_{ij} \beta_j \right)^2 + \lambda \sum_{j=1}^d \beta_j^2 \right\}. \quad (2)$$

Quelques remarques

- Les définitions (1) et (2) sont **équivalentes** dans le sens où pour tout t il existe un unique λ tels que les solutions aux deux problèmes d'optimisation **coïncident**.

Quelques remarques

- Les définitions (1) et (2) sont **équivalentes** dans le sens où pour tout t il existe un unique λ tels que les solutions aux deux problèmes d'optimisation **coïncident**.
- La **constante** β_0 n'entre généralement **pas** dans la **pénalité**.

Quelques remarques

- Les définitions (1) et (2) sont **équivalentes** dans le sens où pour tout t il existe un unique λ tels que les solutions aux deux problèmes d'optimisation **coïncident**.
- La **constante** β_0 n'entre généralement **pas** dans la **pénalité**.
- L'estimateur **dépend** bien entendu du paramètre t (ou λ) :
$$\hat{\beta}^R = \hat{\beta}^R(t) = \hat{\beta}^R(\lambda).$$

Quelques remarques

- Les définitions (1) et (2) sont **équivalentes** dans le sens où pour tout t il existe un unique λ tels que les solutions aux deux problèmes d'optimisation **coïncident**.
- La **constante** β_0 n'entre généralement **pas** dans la **pénalité**.
- L'estimateur **dépend** bien entendu du paramètre t (ou λ) :
$$\hat{\beta}^R = \hat{\beta}^R(t) = \hat{\beta}^R(\lambda).$$
- Les variables explicatives sont le plus souvent **réduites** pour **éviter les problèmes d'échelle** dans la pénalité.

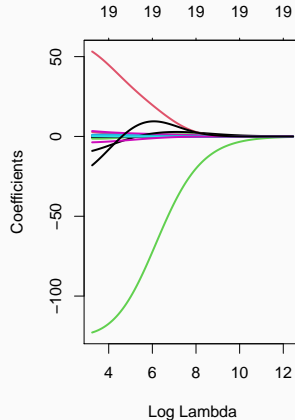
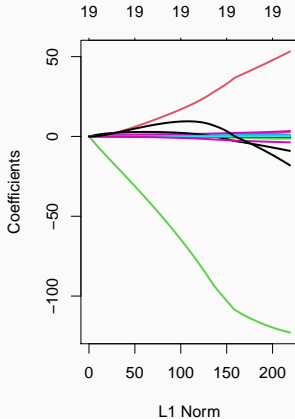
Exemple avec les données Hitters

- Il existe **plusieurs fonctions et packages** qui permettent de faire de la régression pénalisée sur **R**. Nous présentons ici **glmnet**.
- **glmnet** n'accepte pas d'objet **formule**. Il faut spécifier la **matrice** des X et le **vecteur** des Y :

```
> Hitters.X <- model.matrix(Salary~.,data=Hitters)[,-1]
```

Ridge avec glmnet

```
> library(glmnet)
> reg.ridge <- glmnet(Hitters.X, Hitters$Salary, alpha=0)
> par(mfrow=c(1,2))
> plot(reg.ridge, lwd=2)
> plot(reg.ridge, lwd=2, xvar="lambda")
```



Propriétés des estimateurs ridge

Propriétés

1. Lorsque les variables explicatives sont **centrée-réduites**, l'estimateur Ridge solution de (2) s'écrit

$$\hat{\beta}^R = \hat{\beta}^R(\lambda) = (\mathbb{X}^t \mathbb{X} + \lambda \mathbb{I})^{-1} \mathbb{X}^t \mathbb{Y}.$$

2. On déduit

$$\text{biais}(\hat{\beta}^R) = -\lambda(\mathbb{X}^t \mathbb{X} + \lambda \mathbb{I})^{-1} \beta$$

et

$$\mathbf{V}(\hat{\beta}^R) = \sigma^2 (\mathbb{X}^t \mathbb{X} + \lambda \mathbb{I})^{-1} \mathbb{X}^t \mathbb{X} (\mathbb{X}^t \mathbb{X} + \lambda \mathbb{I})^{-1}.$$

Propriétés des estimateurs ridge

Propriétés

1. Lorsque les variables explicatives sont **centrée-réduites**, l'estimateur Ridge solution de (2) s'écrit

$$\hat{\beta}^R = \hat{\beta}^R(\lambda) = (\mathbb{X}^t \mathbb{X} + \lambda \mathbb{I})^{-1} \mathbb{X}^t \mathbb{Y}.$$

2. On déduit

$$\text{biais}(\hat{\beta}^R) = -\lambda(\mathbb{X}^t \mathbb{X} + \lambda \mathbb{I})^{-1} \beta$$

et

$$\mathbf{V}(\hat{\beta}^R) = \sigma^2 (\mathbb{X}^t \mathbb{X} + \lambda \mathbb{I})^{-1} \mathbb{X}^t \mathbb{X} (\mathbb{X}^t \mathbb{X} + \lambda \mathbb{I})^{-1}.$$

Commentaires

- Si $\lambda = 0$, on retrouve le biais et la variance de l'estimateur des **MCO**.
- $\lambda \nearrow \implies \text{biais} \nearrow$ et $\text{variance} \searrow$ et réciproquement lorsque $\lambda \searrow$.

- Il est **crucial** : si $\lambda \approx 0$ alors $\hat{\beta}^R \approx \hat{\beta}^{MCO}$, si λ "grand" alors $\hat{\beta}^R \approx 0$.

Choix de λ

- Il est **crucial** : si $\lambda \approx 0$ alors $\hat{\beta}^R \approx \hat{\beta}^{MCO}$, si λ "grand" alors $\hat{\beta}^R \approx 0$.
- Le choix de λ se fait le plus souvent de façon "classique" :
 1. **Estimation d'un critère** de choix de modèle pour toutes les valeurs de λ ;

Choix de λ

- Il est **crucial** : si $\lambda \approx 0$ alors $\hat{\beta}^R \approx \hat{\beta}^{MCO}$, si λ "grand" alors $\hat{\beta}^R \approx 0$.
- Le choix de λ se fait le plus souvent de façon "classique" :
 1. **Estimation d'un critère** de choix de modèle pour toutes les valeurs de λ ;
 2. Choix du λ qui **minimise** le critère estimé.

Choix de λ

- Il est **crucial** : si $\lambda \approx 0$ alors $\hat{\beta}^R \approx \hat{\beta}^{MCO}$, si λ "grand" alors $\hat{\beta}^R \approx 0$.
- Le choix de λ se fait le plus souvent de façon "classique" :
 1. **Estimation d'un critère** de choix de modèle pour toutes les valeurs de λ ;
 2. Choix du λ qui **minimise** le critère estimé.
- **Exemple** : la fonction `cv.glmnet` choisit la valeur de λ qui minimise l'erreur quadratique moyenne

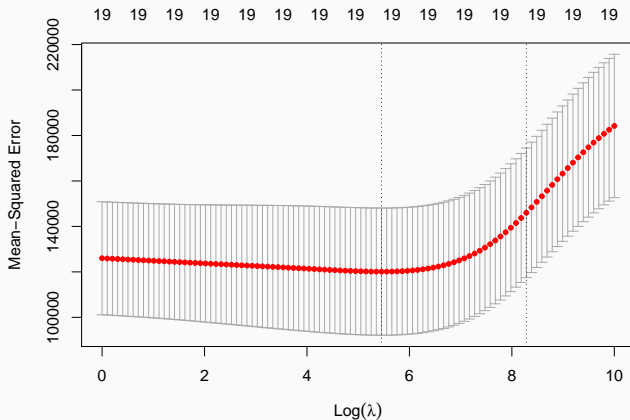
$$\mathbb{E}[(Y - m_{\hat{\beta}^R(\lambda)}(X))^2]$$

estimée par **validation croisée**.

```

> set.seed(321)
> reg.cvridge <- cv.glmnet(Hitters.X,Hitters$Salary,alpha=0,
+                           lambda=exp(seq(0,10,length=100)))
> bestlam <- reg.cvridge$lambda.min
> bestlam
## [1] 233.8186
> plot(reg.cvridge)

```



Régression ridge

Régression Lasso

Variantes de ridge/lasso

Discrimination binaire

Bibliographie

- La **régression lasso** consiste à minimiser le critère des moindres carrés pénalisé par la norme 1 des coefficients.

Définition [**Tibshirani, 1996**]

1. Les **estimateurs lasso** $\hat{\beta}^L$ s'obtiennent en minimisant

$$\sum_{i=1}^n \left(Y_i - \beta_0 - \sum_{j=1}^d X_{ij} \beta_j \right)^2 \quad \text{sous la contrainte} \quad \sum_{j=1}^d |\beta_j| \leq t \quad (3)$$

2. ou de façon **équivalente**

$$\hat{\beta}^L = \underset{\beta}{\operatorname{argmin}} \left\{ \sum_{i=1}^n \left(Y_i - \beta_0 - \sum_{j=1}^d X_{ij} \beta_j \right)^2 + \lambda \sum_{j=1}^d |\beta_j| \right\}. \quad (4)$$

Comparaison Ridge-Lasso

- Dans le cas où la matrice \mathbb{X} est **orthonormée**, on a une **écriture explicite** pour les estimateurs ridge et lasso.

Comparaison Ridge-Lasso

- Dans le cas où la matrice \mathbb{X} est **orthonormée**, on a une **écriture explicite** pour les estimateurs ridge et lasso.

Propriété

Si la matrice de design \mathbb{X} est orthonormée, alors

$$\hat{\beta}_j^R = \frac{\hat{\beta}_j}{1 + \lambda} \quad \text{et} \quad \hat{\beta}_j^L = \text{signe}(\hat{\beta}_j)(|\hat{\beta}_j| - \lambda)_+$$

où $\hat{\beta}_j$ est l'estimateur MCO de β_j .

Comparaison Ridge-Lasso

- Dans le cas où la matrice \mathbb{X} est **orthonormée**, on a une **écriture explicite** pour les estimateurs ridge et lasso.

Propriété

Si la matrice de design \mathbb{X} est orthonormée, alors

$$\hat{\beta}_j^R = \frac{\hat{\beta}_j}{1 + \lambda} \quad \text{et} \quad \hat{\beta}_j^L = \text{signe}(\hat{\beta}_j)(|\hat{\beta}_j| - \lambda)_+$$

où $\hat{\beta}_j$ est l'estimateur MCO de β_j .

Commentaires

- **Ridge** "diminue" l'estimateur MCO de façon **proportionnelle** ;

Comparaison Ridge-Lasso

- Dans le cas où la matrice \mathbb{X} est **orthonormée**, on a une **écriture explicite** pour les estimateurs ridge et lasso.

Propriété

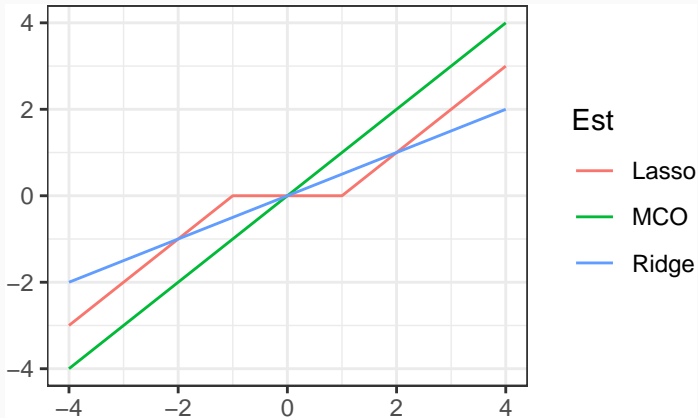
Si la matrice de design \mathbb{X} est orthonormée, alors

$$\hat{\beta}_j^R = \frac{\hat{\beta}_j}{1 + \lambda} \quad \text{et} \quad \hat{\beta}_j^L = \text{signe}(\hat{\beta}_j)(|\hat{\beta}_j| - \lambda)_+$$

où $\hat{\beta}_j$ est l'estimateur MCO de β_j .

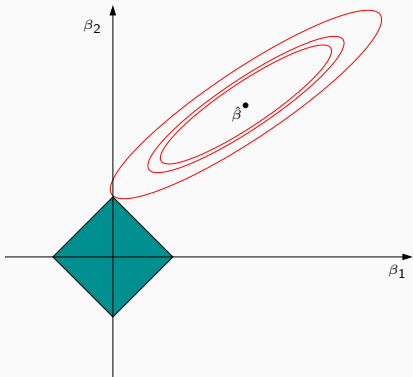
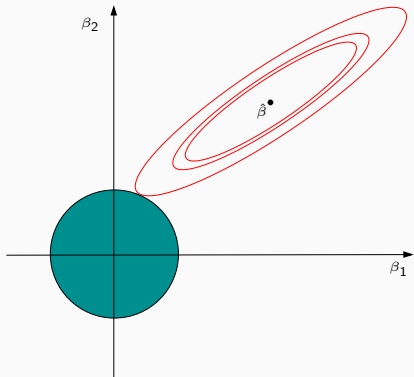
Commentaires

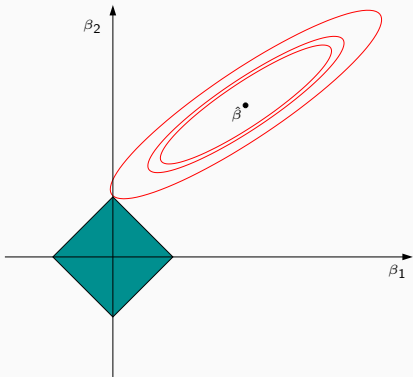
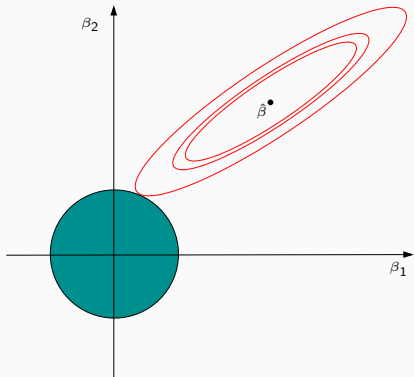
- **Ridge** "diminue" l'estimateur MCO de façon **proportionnelle** ;
- **Lasso** **translate et tronque** l'estimateur MCO (lorsque ce dernier est petit).



Conclusion

Le lasso va avoir tendance à "mettre" des coefficients à 0 et donc à faire de la sélection de variables.





Remarque

Ces approches reviennent (d'une certaine façon) à projeter l'estimateur des MCO sur les boules unités associées à

1. la norme 2 pour la régression ridge ;
2. la norme 1 pour le lasso.

Quelques remarques

- Comme pour la régression ridge :
 - on préfère souvent réduire la matrice de design avant d'effectuer la régression lasso ;

- Comme pour la régression ridge :
 - on préfère souvent réduire la matrice de design avant d'effectuer la régression lasso ;
 - Le choix de λ est crucial (il est le plus souvent sélectionné en minimisant un critère empirique).

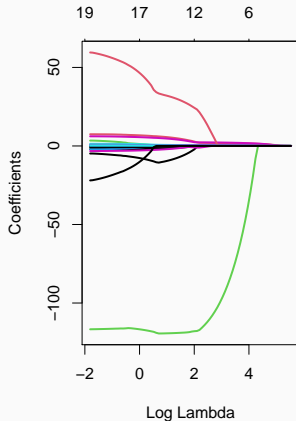
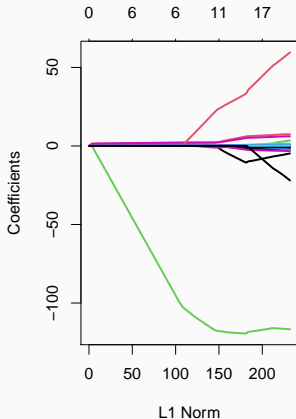
Quelques remarques

- Comme pour la régression ridge :
 - on préfère souvent **réduire la matrice de design** avant d'effectuer la régression lasso ;
 - Le choix de λ est **crucial** (il est le plus souvent sélectionné en minimisant un critère empirique).
 - $\lambda \nearrow \implies$ biais \nearrow et variance \searrow et réciproquement lorsque $\lambda \searrow$.

Quelques remarques

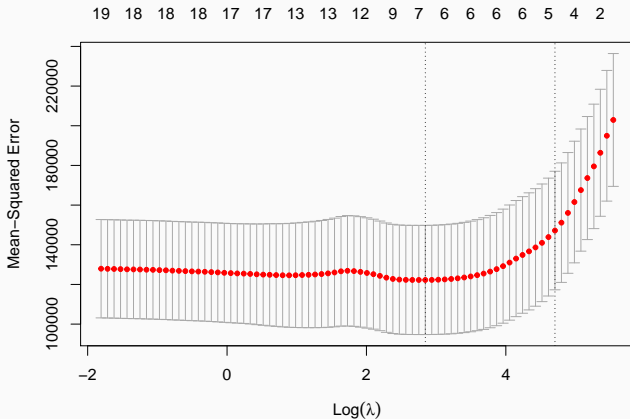
- Comme pour la régression ridge :
 - on préfère souvent **réduire la matrice de design** avant d'effectuer la régression lasso ;
 - Le choix de λ est **crucial** (il est le plus souvent sélectionné en minimisant un critère empirique).
 - $\lambda \nearrow \implies$ biais \nearrow et variance \searrow et réciproquement lorsque $\lambda \searrow$.
- **MAIS**, contrairement à ridge : $\lambda \nearrow \implies$ **le nombre de coefficients nuls augmente** ([Bühlmann and van de Geer, 2011]).

```
> reg.lasso <- glmnet(Hitters.X, Hitters$Salary, alpha=1)
> par(mfrow=c(1,2))
> plot(reg.lasso, lwd=2)
> plot(reg.lasso, lwd=2, xvar="lambda")
```



Sélection de λ

```
> set.seed(321)
> reg.cvlasso <- cv.glmnet(Hitters.X, Hitters$Salary, alpha=1)
> bestlam <- reg.cvlasso$lambda.min
> bestlam
## [1] 17.19108
> plot(reg.cvlasso)
```



- Il existe plusieurs façons de résoudre le problème numérique d'optimisation lasso (ou ridge).
- Un des plus utilisé est l'algorithme de descente de coordonnées [Hastie et al., 2015].

- Il existe plusieurs façons de résoudre le problème numérique d'optimisation lasso (ou ridge).
- Un des plus utilisés est l'algorithme de descente de coordonnées [Hastie et al., 2015].
- On considère le problème lasso

$$\hat{\beta}^L = \underset{\beta}{\operatorname{argmin}} \left\{ \sum_{i=1}^n \left(Y_i - \beta_0 - \sum_{j=1}^d X_{ij} \beta_j \right)^2 + \lambda \sum_{j=1}^d |\beta_j| \right\}$$

avec les variables explicatives centrées-réduites (pour simplifier).

Descente de coordonnées

1. **Initialisation** : $\hat{\beta}_0 = \bar{y}$, $\hat{\beta}_j = \dots, j = 1, \dots, d$.
2. Répéter jusqu'à convergence :
Pour $j = 1, \dots, d$:
 - 2.1 Calculer les **résidus partiels** $r_i^{(j)} = y_i - \sum_{k \neq j} x_{ik} \hat{\beta}_k$;
 - 2.2 Faire la **régression simple** des y_i contre $r_i^{(j)} \Rightarrow \tilde{\beta}_j$;
 - 2.3 **Mettre à jour** $\hat{\beta}_j = \text{signe}(\tilde{\beta}_j)(|\tilde{\beta}_j| - \lambda)_+$
3. **Retourner** : $\hat{\beta}_j, j = 1, \dots, d$.

Régression ridge

Régression Lasso

Variantes de ridge/lasso

Discrimination binaire

Bibliographie

Différentes pénalités

- Les approches **ridge** et **lasso** diffèrent uniquement au niveau de la **pénalité** ajoutée au critère des moindres carrés.
- **Norme 2** pour **ridge** et **norme 1** pour le **lasso**.

Différentes pénalités

- Les approches **ridge** et **lasso** diffèrent uniquement au niveau de la **pénalité** ajoutée au critère des moindres carrés.
- **Norme 2** pour **ridge** et **norme 1** pour le **lasso**.
- Il existe tout un tas d'**autres stratégies** de pénalisations.
- Nous en présentons quelques unes dans cette partie.
- On pourra consulter [**Hastie et al., 2015**] pour plus de détails.

- [Zou and Hastie, 2005] ont proposé de combiner les approches ridge et lasso en proposant une pénalité (appelée elastic net) de la forme

$$\lambda \sum_{j=1}^d ((1 - \alpha)\beta_j^2 + \alpha|\beta_j|)$$

où $\alpha \in [0, 1]$.

- [Zou and Hastie, 2005] ont proposé de combiner les approches ridge et lasso en proposant une pénalité (appelée elastic net) de la forme

$$\lambda \sum_{j=1}^d ((1 - \alpha)\beta_j^2 + \alpha|\beta_j|)$$

où $\alpha \in [0, 1]$.

- Le paramètre α définit le compromis ridge/lasso :
 - $\alpha = 1 \implies$ Lasso ;
 - $\alpha = 0 \implies$ Ridge ;

- [Zou and Hastie, 2005] ont proposé de combiner les approches ridge et lasso en proposant une pénalité (appelée elastic net) de la forme

$$\lambda \sum_{j=1}^d ((1 - \alpha)\beta_j^2 + \alpha|\beta_j|)$$

où $\alpha \in [0, 1]$.

- Le paramètre α définit le compromis ridge/lasso :
 - $\alpha = 1 \implies$ Lasso ;
 - $\alpha = 0 \implies$ Ridge ;
 - Ce paramètre correspond (évidemment) à l'argument **alpha** de la fonction `glmnet`.

- [Zou and Hastie, 2005] ont proposé de combiner les approches ridge et lasso en proposant une pénalité (appelée elastic net) de la forme

$$\lambda \sum_{j=1}^d ((1 - \alpha)\beta_j^2 + \alpha|\beta_j|)$$

où $\alpha \in [0, 1]$.

- Le paramètre α définit le compromis ridge/lasso :
 - $\alpha = 1 \implies$ Lasso ;
 - $\alpha = 0 \implies$ Ridge ;
 - Ce paramètre correspond (évidemment) à l'argument **alpha** de la fonction `glmnet`.
- **Avantage** : on a plus de flexibilité car la pénalité elastic net propose une gamme de modèles beaucoup plus large que lasso et ridge ;

- [Zou and Hastie, 2005] ont proposé de combiner les approches ridge et lasso en proposant une pénalité (appelée elastic net) de la forme

$$\lambda \sum_{j=1}^d ((1 - \alpha)\beta_j^2 + \alpha|\beta_j|)$$

où $\alpha \in [0, 1]$.

- Le paramètre α définit le compromis ridge/lasso :
 - $\alpha = 1 \implies$ Lasso ;
 - $\alpha = 0 \implies$ Ridge ;
 - Ce paramètre correspond (évidemment) à l'argument **alpha** de la fonction `glmnet`.
- **Avantage** : on a plus de flexibilité car la pénalité elastic net propose une gamme de modèles beaucoup plus large que lasso et ridge ;
- **Inconvénient** : en plus du λ il faut aussi sélectionner le α !

- Dans certaines applications, les variables explicatives appartiennent à des groupes de variables prédéfinis.

Group Lasso

- Dans certaines applications, les variables explicatives appartiennent à des groupes de variables prédéfinis.
- Nécessité de "shrinker" ou sélectionner les variables par groupe.

- Dans certaines applications, les variables explicatives appartiennent à des groupes de variables prédéfinis.
- Nécessité de "shrinker" ou sélectionner les variables par groupe.

Exemple : variables qualitatives

- 2 variables explicatives qualitatives X_1 et X_2 et une variable explicative continue X_3 .
- Le modèle s'écrit

$$Y = \beta_0 + \beta_1 \mathbf{1}_{X_1=A} + \beta_2 \mathbf{1}_{X_1=B} + \beta_3 \mathbf{1}_{X_1=C} \\ + \beta_4 \mathbf{1}_{X_2=D} + \beta_5 \mathbf{1}_{X_2=E} + \beta_6 \mathbf{1}_{X_2=F} + \beta_7 \mathbf{1}_{X_2=G} + \beta_8 X_3 + \varepsilon$$

muni des contraintes $\beta_1 = \beta_4 = 0$.

- Dans certaines applications, les variables explicatives appartiennent à des groupes de variables prédéfinis.
- Nécessité de "shrinker" ou sélectionner les variables par groupe.

Exemple : variables qualitatives

- 2 variables explicatives qualitatives X_1 et X_2 et une variable explicative continue X_3 .
- Le modèle s'écrit

$$Y = \beta_0 + \beta_1 \mathbf{1}_{X_1=A} + \beta_2 \mathbf{1}_{X_1=B} + \beta_3 \mathbf{1}_{X_1=C} \\ + \beta_4 \mathbf{1}_{X_2=D} + \beta_5 \mathbf{1}_{X_2=E} + \beta_6 \mathbf{1}_{X_2=F} + \beta_7 \mathbf{1}_{X_2=G} + \beta_8 X_3 + \varepsilon$$

muni des contraintes $\beta_1 = \beta_4 = 0$.

- 3 groupes : $\mathbf{X}_1 = (\mathbf{1}_{X_1=B}, \mathbf{1}_{X_1=C})$, $\mathbf{X}_2 = (\mathbf{1}_{X_2=E}, \mathbf{1}_{X_2=F}, \mathbf{1}_{X_2=G})$ et $\mathbf{X}_3 = X_3$.

Définition

En présence de d variables réparties en L groupes $\mathbf{X}_1, \dots, \mathbf{X}_L$ de cardinal d_1, \dots, d_L . On note $\beta_\ell, \ell = 1, \dots, L$ le vecteur des coefficients associé au groupe \mathbf{X}_ℓ . Les **estimateurs group-lasso** s'obtiennent en minimisant le critère

$$\sum_{i=1}^n \left(y_i - \beta_0 - \sum_{\ell=1}^L \mathbf{x}_{i\ell} \beta_\ell \right)^2 + \lambda \sum_{\ell=1}^L \sqrt{d_\ell} \|\beta_\ell\|_2$$

Remarque

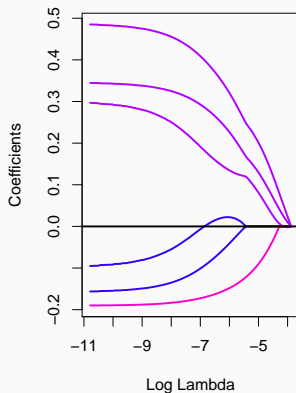
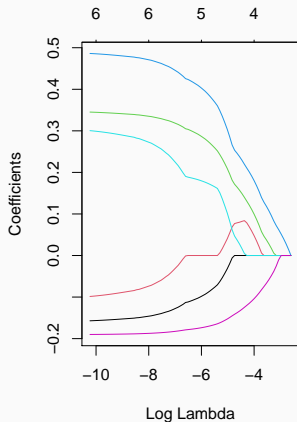
Puisque $\|\beta_\ell\|_2 = 0$ ssi $\beta_{\ell 1} = \dots = \beta_{\ell d_\ell} = 0$, cette procédure encourage la **mise à zéro** des coefficients d'un **même groupe**.

Le coin R

- La fonction `gglasso` du package `gglasso` permet de faire du **groupe lasso** sur R.

```
> summary(donnees)
##           X1                X2                X3                Y
## Length:200      Length:200      Min.   :0.009496  Min.   : -3.23315
## Class :character Class :character 1st Qu.:0.237935 1st Qu.: -0.50404
## Mode  :character Mode  :character Median :0.485563 Median : 0.16759
##                               Mean  :0.483286 Mean  : 0.09792
##                               3rd Qu.:0.734949 3rd Qu.: 0.66918
##                               Max.   :0.998741 Max.   : 3.04377

> D <- model.matrix(Y~.,data=donnees)[,-1]
> model <- glmnet(D,Y,alpha=1)
> groupe <- c(1,1,2,2,2,3)
> library(gglasso)
> model1 <- gglasso(D,Y,group=groupe)
> plot(model1)
```



Remarque

Les coefficients **s'annulent par groupe** lorsque λ augmente (graphe de droite).

Sparse group lasso

- La **norme 2** de la pénalité group-lasso implique que, généralement, tous les coefficients d'un groupe sont **tous nuls** ou **tous non nuls**.
- Dans certains cas, il peut être intéressant de mettre de la **sparsité** dans les groupes aussi. Comment ?

Sparse group lasso

- La **norme 2** de la pénalité group-lasso implique que, généralement, tous les coefficients d'un groupe sont **tous nuls** ou **tous non nuls**.
- Dans certains cas, il peut être intéressant de mettre de la **sparsité** dans les groupes aussi. Comment ?
- En ajoutant **la norme 1** dans la pénalité.

Sparse group lasso

- La **norme 2** de la pénalité group-lasso implique que, généralement, tous les coefficients d'un groupe sont **tous nuls** ou **tous non nuls**.
- Dans certains cas, il peut être intéressant de mettre de la **sparsité** dans les groupes aussi. Comment ?
- En ajoutant **la norme 1** dans la pénalité.

Pénalité sparse group lasso

$$\lambda \sum_{\ell=1}^L [(1 - \alpha) \|\beta_{\ell}\|_2 + \alpha \|\beta_{\ell}\|_1].$$

- Sur **R** : package **SGL**.

Fused lasso

- Utile pour prendre en compte la **spatialité des données**.
- **Idée** : deux coefficients successifs doivent être proches.

Pénalité fused lasso

$$\lambda_1 \sum_{j=1}^d |\beta_j|$$

Fused lasso

- Utile pour prendre en compte la **spatialité des données**.
- **Idée** : deux coefficients successifs doivent être proches.

Pénalité fused lasso

$$\lambda_1 \sum_{j=1}^d |\beta_j| + \lambda_2 \sum_{j=2}^d |\beta_{j+1} - \beta_j|$$

qui peut se re-paramétriser en

$$\lambda \sum_{j=2}^d |\beta_{j+1} - \beta_j|.$$

- Sur **R** : package **genlasso**.

Régression ridge

Régression Lasso

Variantes de ridge/lasso

Discrimination binaire

Bibliographie

- Les méthodes **ridge et lasso** ont été présentées dans un cadre de régression linéaire.
- Ces techniques d'adaptent directement à la **régression logistique** $\mathcal{Y} = \{-1, 1\}$.

- Les méthodes **ridge et lasso** ont été présentées dans un cadre de régression linéaire.
- Ces techniques d'adaptent directement à la **régression logistique** $\mathcal{Y} = \{-1, 1\}$.
- Les **pénalités** sont **identiques**.
- **Seul changement** : le critère moindre carré est remplacé par la déviance \implies ce qui revient à **minimiser l'opposé de la vraisemblance plus la pénalité**.

Définition

On note $\tilde{y}_i = (y_i + 1)/2$.

- On appelle **estimateur ridge** en régression logistique l'estimateur

$$\hat{\beta}^R = \underset{\beta}{\operatorname{argmin}} \left\{ - \sum_{i=1}^n (\tilde{y}_i x_i^t \beta - \log(1 + \exp(x_i^t \beta))) + \lambda \sum_{j=1}^d \beta_j^2 \right\}.$$

- On appelle **estimateur lasso** en régression logistique l'estimateur

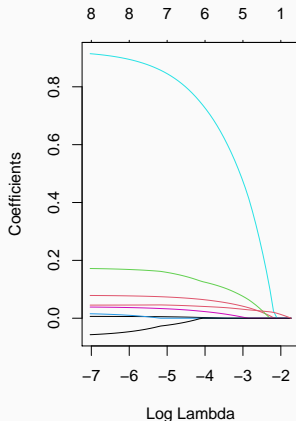
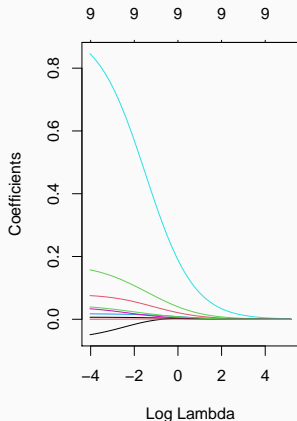
$$\hat{\beta}^L = \underset{\beta}{\operatorname{argmin}} \left\{ - \sum_{i=1}^n (\tilde{y}_i x_i^t \beta - \log(1 + \exp(x_i^t \beta))) + \lambda \sum_{j=1}^d |\beta_j| \right\}.$$

- Pour faire du ridge ou lasso en logistique, il suffit d'ajouter l'argument `family=binomial` dans `glmnet`.
- Tout reste identique pour le reste (tracé du chemin des coefficients, choix du λ ...).
- Exemple : données SAheart

```
> head(SAheart)
##   sbp tobacco   ldl adiposity famhist typea obesity alcohol age chd
## 1 160   12.00 5.73    23.11 Present   49   25.30   97.20 52   1
## 2 144    0.01 4.41    28.61 Absent    55   28.87    2.06 63   1
## 3 118    0.08 3.48    32.28 Present   52   29.14    3.81 46   0
## 4 170    7.50 6.41    38.03 Present   51   31.99   24.26 58   1
## 5 134   13.60 3.50    27.78 Present   60   25.99   57.34 49   1
## 6 132    6.20 6.47    36.21 Present   62   30.77   14.14 45   0
```

- On obtient les chemins de régularisation ridge et lasso avec les commandes suivantes :

```
> SAheart.X <- model.matrix(chd~.,data=SAheart)
> log.ridge <- glmnet(SAheart.X,SAheart$chd,family="binomial",alpha=0)
> log.lasso <- glmnet(SAheart.X,SAheart$chd,family="binomial",alpha=1)
> plot(log.ridge,xvar="lambda")
```



Régression ridge

Régression Lasso

Variantes de ridge/lasso

Discrimination binaire

Bibliographie



Bühlmann, P. and van de Geer, S. (2011).

Statistics for high-dimensional data.

Springer.



Hastie, T., Tibshirani, R., and Friedman, J. (2009).

The Elements of Statistical Learning : Data Mining, Inference, and Prediction.

Springer, second edition.



Hastie, T., Tibshirani, R., and Wainwright, M. (2015).

Statistical Learning with Sparsity : The Lasso and Generalizations.

CRC Press.

[https:](https://web.stanford.edu/~hastie/StatLearnSparsity_files/SLS.pdf)

[//web.stanford.edu/~hastie/StatLearnSparsity_files/SLS.pdf](https://web.stanford.edu/~hastie/StatLearnSparsity_files/SLS.pdf).



Tibshirani, R. (1996).

Regression shrinkage and selection via the lasso.

Journal of the Royal Statistical Society, Series B, 58 :267–288.



Zou, H. and Hastie, T. (2005).

Regularization and variable selection via the elastic net.

Journal of the Royal Statistical Society, Series B, 67 :301–320.

Quatrième partie IV

Modèle additif

Présentation du modèle

Estimation : l'algorithme du backfitting

Lisseurs

Bibliographie

- Toujours le même : modèle de **régression** $\implies (x_i, y_i) \in \mathbb{R}^d \times \mathbb{R}$ tels que

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

- **Rappels** :
 - **Modèle linéaire (paramétrique)** : peu flexible mais convergence rapide ;
 - **Modèle non paramétrique** : flexible mais convergence lente.

- Toujours le même : modèle de **régression** $\implies (x_i, y_i) \in \mathbb{R}^d \times \mathbb{R}$ tels que

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

- **Rappels** :
 - **Modèle linéaire (paramétrique)** : peu flexible mais convergence rapide ;
 - **Modèle non paramétrique** : flexible mais convergence lente.

Idée

Conserver des **vitesse de convergence raisonnables** en **allégeant** l'hypothèse de linéarité en les covariables.

Présentation du modèle

Estimation : l'algorithme du backfitting

Lisseurs

Bibliographie

Le modèle additif

Définition

Un modèle de régression est **additif** si la fonction de régression de y sur $x = (x_1, \dots, x_d)$ s'écrit

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

où $\alpha \in \mathbb{R}$ est inconnu et les fonction g_1, \dots, g_d sont inconnues (et donc **à estimer**).

Le modèle additif

Définition

Un modèle de régression est **additif** si la fonction de régression de y sur $x = (x_1, \dots, x_d)$ s'écrit

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

où $\alpha \in \mathbb{R}$ est inconnu et les fonction g_1, \dots, g_d sont inconnues (et donc à **estimer**).

Remarques

- Hypothèse de **structure additive** en les covariables comme dans le modèle linéaire mais...

Le modèle additif

Définition

Un modèle de régression est **additif** si la fonction de régression de y sur $x = (x_1, \dots, x_d)$ s'écrit

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

où $\alpha \in \mathbb{R}$ est inconnu et les fonction g_1, \dots, g_d sont inconnues (et donc à **estimer**).

Remarques

- Hypothèse de **structure additive** en les covariables comme dans le modèle linéaire mais...
- pas d'hypothèse de **linéarité**.

- Pour l'identifiabilité du modèle on suppose souvent $\mathbf{E}[Y] = \alpha$ et $\mathbf{E}[g_j(x_j)] = 0$.

- Pour l'identifiabilité du modèle on suppose souvent $\mathbf{E}[Y] = \alpha$ et $\mathbf{E}[g_j(x_j)] = 0$.
- On doit estimer d fonctions \implies cadre non paramétrique mais ...

- Pour l'identifiabilité du modèle on suppose souvent $\mathbf{E}[Y] = \alpha$ et $\mathbf{E}[g_j(x_j)] = 0$.
- On doit estimer d fonctions \implies cadre non paramétrique mais ...
- Ces fonctions sont univariées !

- Pour l'identifiabilité du modèle on suppose souvent $\mathbf{E}[Y] = \alpha$ et $\mathbf{E}[g_j(x_j)] = 0$.
- On doit estimer d fonctions \implies cadre non paramétrique mais ...
- Ces fonctions sont univariées !

Conclusion

Si ces fonctions sont convenablement estimées, on peut espérer obtenir des vitesses de convergence proches des vitesses non paramétrique en dimension 1 (qui ne sont pas si mauvaises!).

Présentation du modèle

Estimation : l'algorithme du backfitting

Lisseurs

Bibliographie

Retour au modèle linéaire

- $m(X) = \mathbf{E}[Y|X] = \beta_0 + \beta_1 X_1 + \dots + \beta_d X_d = \sum_{j=0}^d \beta_j X_j$.
- En conditionnant par X_k on obtient :

$$\begin{aligned}\mathbf{E}[Y|X_k = x_k] &= \mathbf{E}[\mathbf{E}[Y|X]|X_k = x_k] \\ &= \mathbf{E}\left[\sum_{j=0}^d \beta_j X_j \middle| X_k = x_k\right] \\ &= \beta_k x_k + \mathbf{E}\left[\sum_{j \neq k} \beta_j X_j \middle| X_k = x_k\right].\end{aligned}$$

Retour au modèle linéaire

- $m(X) = \mathbf{E}[Y|X] = \beta_0 + \beta_1 X_1 + \dots + \beta_d X_d = \sum_{j=0}^d \beta_j X_j$.
- En **conditionnant** par X_k on obtient :

$$\begin{aligned}\mathbf{E}[Y|X_k = x_k] &= \mathbf{E}[\mathbf{E}[Y|X]|X_k = x_k] \\ &= \mathbf{E}\left[\sum_{j=0}^d \beta_j X_j \middle| X_k = x_k\right] \\ &= \beta_k x_k + \mathbf{E}\left[\sum_{j \neq k} \beta_j X_j \middle| X_k = x_k\right].\end{aligned}$$

Propriété

On pose $Y^{(k)} = Y - \sum_{j \neq k} \beta_j X_j$. Alors

$$\mathbf{E}[Y^{(k)}|X_k = x_k] = \beta_k x_k.$$

- **Idée** : estimer β_k en faisant la régression (univariée) de $Y^{(k)}$ sur X_k .
- **Problème** :

- **Idée** : estimer β_k en faisant la régression (univariée) de $Y^{(k)}$ sur X_k .
- **Problème** : $Y^{(k)}$ dépend des $\beta_j, j \neq k$ qui sont inconnus !
- **Solution** :

- **Idée** : estimer β_k en faisant la régression (univariée) de $Y^{(k)}$ sur X_k .
- **Problème** : $Y^{(k)}$ dépend des $\beta_j, j \neq k$ qui sont inconnus !
- **Solution** : utiliser un algorithme itératif.

- **Idée** : estimer β_k en faisant la **régression (univariée)** de $Y^{(k)}$ sur X_k .
- **Problème** : $Y^{(k)}$ dépend des $\beta_j, j \neq k$ qui sont **inconnus** !
- **Solution** : utiliser un **algorithme itératif**.

Algorithme : backfitting pour le modèle linéaire

- Initialisation $\hat{\beta} = (0, \dots, 0), \gamma = (1, \dots, 1)$ (dim $d + 1$).
- Tant que $\min_{1 \leq j \leq d+1} (|\hat{\beta}_j - \gamma_j|) > \varepsilon$ (petit)
 1. $\hat{\beta} = \gamma$
 2. Pour $k = 0, \dots, d$:
 - $\mathbb{Y}^{(k)} = \mathbb{Y} - \sum_{j \neq k} \beta_j \mathbb{X}_j$ (résidus partiels)
 - γ_k : coefficient de la régression univariée sans constante de $\mathbb{Y}^{(k)}$ sur \mathbb{X}_k .
- **Retourner** $\hat{\beta}$.

- Cet algorithme converge vers les estimateurs MCO.
- Aucune utilité pratique puisque ces estimateurs s'obtiennent de façon matricielle !
- Mais...

- Cet algorithme converge vers les **estimateurs MCO**.
- **Aucune utilité pratique** puisque ces estimateurs s'obtiennent de façon matricielle !
- Mais... il ne repose pas sur l'hypothèse de linéarité du modèle, juste sur sa **structure additive**.
- Facilement généralisable au **modèle additif**.

- Retour au **modèle additif**

$$m(X) = \mathbf{E}[Y|X] = \alpha + g_1(X_1) + \dots + g_d(X_d)$$

ou

$$Y = \alpha + g_1(X_1) + \dots + g_d(X_d) + \varepsilon \quad \text{avec} \quad \mathbf{E}[\varepsilon|X] = 0.$$

- Retour au **modèle additif**

$$m(X) = \mathbf{E}[Y|X] = \alpha + g_1(X_1) + \dots + g_d(X_d)$$

ou

$$Y = \alpha + g_1(X_1) + \dots + g_d(X_d) + \varepsilon \quad \text{avec} \quad \mathbf{E}[\varepsilon|X] = 0.$$

Propriété

On pose $Y^{(k)} = Y - \alpha - \sum_{j \neq k} g_j(X_j)$. Alors

$$\mathbf{E}[Y^{(k)}|X_k = x_k] = g_k(x_k).$$

- Retour au **modèle additif**

$$m(X) = \mathbf{E}[Y|X] = \alpha + g_1(X_1) + \dots + g_d(X_d)$$

ou

$$Y = \alpha + g_1(X_1) + \dots + g_d(X_d) + \varepsilon \quad \text{avec} \quad \mathbf{E}[\varepsilon|X] = 0.$$

Propriété

On pose $Y^{(k)} = Y - \alpha - \sum_{j \neq k} g_j(X_j)$. Alors

$$\mathbf{E}[Y^{(k)}|X_k = x_k] = g_k(x_k).$$

Idée

Remplacer l'étape MCO par un **lissage non paramétrique** dans l'algorithme précédent.

L'algorithme du backfitting

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

Backfitting : cas général

L'algorithme du backfitting

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.

Lisseurs non paramétrique

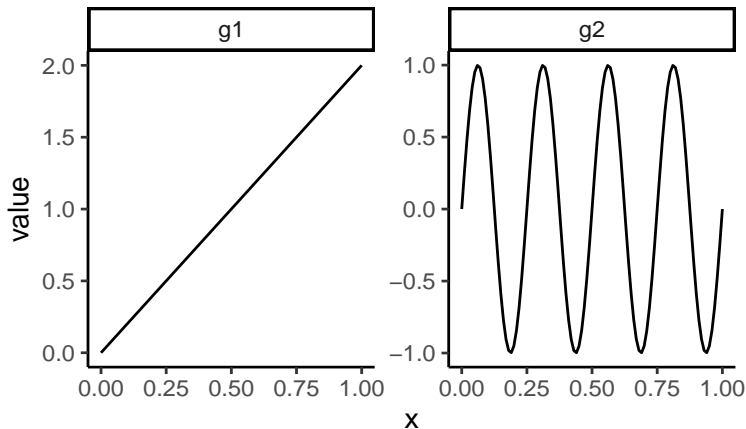
Loess, Nadaraya-Watson, spline de lissage...

Exemple

- On considère le **modèle GAM** :

$$m(X) = 2X_1 + \sin(8\pi X_2)$$

avec X_1 et X_2 indépendantes de lois uniformes sur $[0, 1]$.



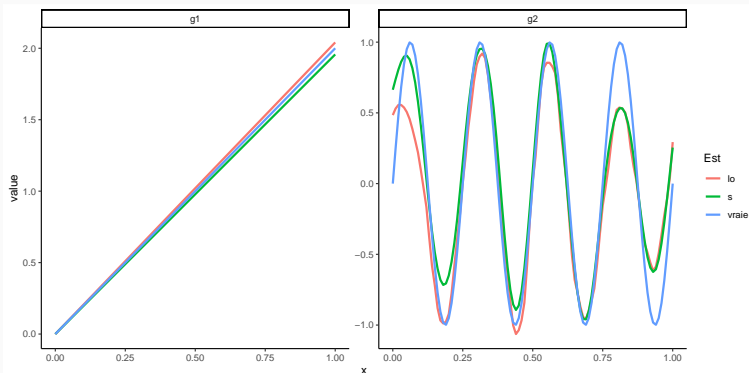
- Sur R, la fonction `gam` du package `gam` permet d'ajuster des modèles additifs :

```
> library(gam)
> model1 <- gam(Y~s(X1,df=1)+s(X2,df=24.579)-1,data=donnees)
> model2 <- gam(Y~lo(X1,span=3)+lo(X2,span=0.15,degree=2)-1,data=donnees)
```

Le coin R

- Sur R, la fonction `gam` du package `gam` permet d'ajuster des modèles additifs :

```
> library(gam)
> model1 <- gam(Y~s(X1,df=1)+s(X2,df=24.579)-1,data=donnees)
> model2 <- gam(Y~lo(X1,span=3)+lo(X2,span=0.15,degree=2)-1,data=donnees)
```



Vitesses de convergence

- Le **backfitting** fournit un estimateur de la fonction de régression

$$\hat{m}(x) = \hat{\alpha} + \hat{g}_1(x_1) + \hat{g}_d(x_d).$$

Théorème [Opsomer, 2000]

Sous certaines hypothèses techniques, notamment les $g_j, j = 1, \dots, d$ de classe C^2 on a

$$\text{Biais}(\hat{m}(x)|X_1, \dots, X_n) = C_1 h^2 + o_p(h^2)$$

et

$$\text{V}(\hat{m}(x)|X_1, \dots, X_n) = \frac{C_2}{nh} + o_p\left(\frac{1}{nh}\right).$$

Vitesses de convergence

- Le **backfitting** fournit un estimateur de la fonction de régression

$$\widehat{m}(x) = \widehat{\alpha} + \widehat{g}_1(x_1) + \widehat{g}_d(x_d).$$

Théorème [Opsomer, 2000]

Sous certaines hypothèses techniques, notamment les $g_j, j = 1, \dots, d$ de classe C^2 on a

$$\text{Biais}(\widehat{m}(x)|X_1, \dots, X_n) = C_1 h^2 + o_p(h^2)$$

et

$$\text{V}(\widehat{m}(x)|X_1, \dots, X_n) = \frac{C_2}{nh} + o_p\left(\frac{1}{nh}\right).$$

Commentaire

Ce sont les **vitesses de convergence classique** des estimateurs **non paramétrique** en dimension 1.

- Les lisseurs non paramétriques dépendent de **paramètres** dont le choix se révèle **crucial** pour la qualité de l'estimation.

- Les lisseurs non paramétriques dépendent de **paramètres** dont le choix se révèle **crucial** pour la qualité de l'estimation.
- **Exemples** : nombre de plus proches voisins, fenêtre de l'estimateur à noyau...

- Les lisseurs non paramétriques dépendent de **paramètres** dont le choix se révèle **crucial** pour la qualité de l'estimation.
- **Exemples** : nombre de plus proches voisins, fenêtre de l'estimateur à noyau...
- Ici encore, il va falloir trouver des **procédures** pour **sélectionner ces paramètres de lissage** dans l'algorithme du backfitting.

Présentation du modèle

Estimation : l'algorithme du backfitting

Lisseurs

Bibliographie

Définition

Soit \hat{m}_λ un estimateur de m qui dépend d'un paramètre λ . \hat{m}_λ est un **lisseur** si il existe une matrice $S_\lambda = S_\lambda(\mathbb{X})$ telle que

$$\hat{\mathbb{Y}} = S_\lambda \mathbb{Y}$$

où $\hat{\mathbb{Y}}$ est le vecteur des valeurs ajustées par \hat{m}_λ et \mathbb{Y} le vecteur des $y_i, i = 1, \dots, n$.

Définition

Soit \hat{m}_λ un estimateur de m qui dépend d'un paramètre λ . \hat{m}_λ est un **lisseur** si il existe une matrice $S_\lambda = S_\lambda(\mathbb{X})$ telle que

$$\hat{\mathbb{Y}} = S_\lambda \mathbb{Y}$$

où $\hat{\mathbb{Y}}$ est le vecteur des valeurs ajustées par \hat{m}_λ et \mathbb{Y} le vecteur des $y_i, i = 1, \dots, n$.

Exemple

- Estimateurs à noyau :

$$S_{ij,h} = \frac{K((x_i - x_j)/h)}{\sum_l K((x_i - x_l)/h)}.$$

Définition

Soit \hat{m}_λ un estimateur de m qui dépend d'un paramètre λ . \hat{m}_λ est un **lisseur** si il existe une matrice $S_\lambda = S_\lambda(\mathbb{X})$ telle que

$$\hat{\mathbb{Y}} = S_\lambda \mathbb{Y}$$

où $\hat{\mathbb{Y}}$ est le vecteur des valeurs ajustées par \hat{m}_λ et \mathbb{Y} le vecteur des $y_i, i = 1, \dots, n$.

Exemple

- Estimateurs à noyau :

$$S_{ij,h} = \frac{K((x_i - x_j)/h)}{\sum_l K((x_i - x_l)/h)}.$$

- Estimateurs des k -ppv :

$$S_{ij,k} = \begin{cases} 1/k & \text{si } x_j \text{ est parmi les } k\text{-ppv de } x_i \\ 0 & \text{sinon.} \end{cases}$$

- L'information sur le **niveau de lissage** est entièrement portée par la **matrice de lissage** S_λ .

- L'information sur le **niveau de lissage** est entièrement portée par la **matrice de lissage** S_λ .

Projecteur

- Lorsque S_λ est un **projecteur** (**modèle linéaire par exemple**), on a

$$\text{tr}(S_\lambda) = \text{rang}(S_\lambda) = \dim \text{esp. où on projette.}$$

- L'information sur le **niveau de lissage** est entièrement portée par la **matrice de lissage** S_λ .

Projecteur

- Lorsque S_λ est un **projecteur** (**modèle linéaire par exemple**), on a

$$\text{tr}(S_\lambda) = \text{rang}(S_\lambda) = \dim \text{esp. où on projette.}$$

- $\text{tr}(S_\lambda)$ est une mesure de la **complexité du lisseur** dans ce cas.

- Par analogie avec la remarque précédente, on pose donc la définition suivante :

Définition

Le nombre de **degrés de liberté** d'un lisseur S_λ est défini par

$$\text{df} = \text{tr}(S_\lambda).$$

- Exemple des **kppv** :

- Par analogie avec la remarque précédente, on pose donc la définition suivante :

Définition

Le nombre de **degrés de liberté** d'un lisseur S_λ est défini par

$$\text{df} = \text{tr}(S_\lambda).$$

- Exemple des **kppv** : $\text{df} = n/k$.
- **Interprétation** :

- Par analogie avec la remarque précédente, on pose donc la définition suivante :

Définition

Le nombre de **degrés de liberté** d'un lisseur S_λ est défini par

$$df = \text{tr}(S_\lambda).$$

- Exemple des **kppv** : $df = n/k$.
- **Interprétation** :
 1. $df \nearrow \implies$ flexibilité \nearrow , biais \searrow et variance \nearrow ;
 2. $df \searrow \implies$ flexibilité \searrow , biais \nearrow et variance \searrow ;

- Le choix du paramètre de lissage est **crucial** pour les estimateurs non-paramétriques.

- Le choix du paramètre de lissage est **crucial** pour les estimateurs non-paramétriques.
- Une approche : **validation croisée LOO**, on choisit λ qui minimise

$$\text{LOO}(\hat{m}_\lambda) = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{m}_\lambda^i(x_i))^2$$

où \hat{m}_λ^i désigne le lisseur calculé sans la i ème observation.

- Le choix du paramètre de lissage est **crucial** pour les estimateurs non-paramétriques.
- Une approche : **validation croisée LOO**, on choisit λ qui minimise

$$\text{LOO}(\hat{m}_\lambda) = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{m}_\lambda^i(x_i))^2$$

où \hat{m}_λ^i désigne le lisseur calculé sans la i ème observation.

Remarque

Cette approche peut se révéler **coûteuse en temps de calcul** car il faut calculer n fois l'estimateurs pour chaque valeur de λ .

- Pour la plupart des **lisseurs** non paramétriques, il n'est pas nécessaire de recalculer n fois l'estimateur.

- Pour la plupart des **lisseurs** non paramétriques, il n'est pas nécessaire de recalculer n fois l'estimateur.
- Le critère **LOO** se déduit directement de la **matrice de lissage** S_λ !

- Pour la plupart des **lisseurs** non paramétriques, il n'est pas nécessaire de recalculer n fois l'estimateur.
- Le critère **LOO** se déduit directement de la **matrice de lissage** S_λ !
- Par exemple, pour l'**estimateur à noyau** on a (voir [Cornillon et al., 2019])

$$\text{LOO}(\hat{m}_h) = \frac{1}{n} \sum_{i=1}^n \left(\frac{y_i - \hat{m}_h(x_i)}{1 - S_{ii,h}} \right)^2,$$

- Pour la plupart des **lisseurs** non paramétriques, il n'est pas nécessaire de recalculer n fois l'estimateur.
- Le critère **LOO** se déduit directement de la **matrice de lissage** S_λ !
- Par exemple, pour l'**estimateur à noyau** on a (voir [Cornillon et al., 2019])

$$\text{LOO}(\hat{m}_h) = \frac{1}{n} \sum_{i=1}^n \left(\frac{y_i - \hat{m}_h(x_i)}{1 - S_{ii,h}} \right)^2 ,$$

- et pour l'estimateur des **kppv**

$$\text{LOO}(\hat{m}_k) = \frac{1}{n} \sum_{i=1}^n \left(\frac{y_i - \hat{m}_{k+1}(x_i)}{1 - S_{ii,k+1}} \right)^2 .$$

- Pour la plupart des **lisseurs** non paramétriques, il n'est pas nécessaire de recalculer n fois l'estimateur.
- Le critère **LOO** se déduit directement de la **matrice de lissage** S_λ !
- Par exemple, pour l'**estimateur à noyau** on a (voir [Cornillon et al., 2019])

$$\text{LOO}(\hat{m}_h) = \frac{1}{n} \sum_{i=1}^n \left(\frac{y_i - \hat{m}_h(x_i)}{1 - S_{ii,h}} \right)^2,$$

- et pour l'estimateur des **kppv**

$$\text{LOO}(\hat{m}_k) = \frac{1}{n} \sum_{i=1}^n \left(\frac{y_i - \hat{m}_{k+1}(x_i)}{1 - S_{ii,k+1}} \right)^2.$$

Remarque

Beaucoup plus **efficace** d'un point de vue **computationnel**.

- Il existe une variante, appelée **Validation Croisée Généralisée (GCV)** au critère LOO, qui peut se révéler plus stable (voir [**Woods, 2006**]).
- Elle consiste à remplacer au dénominateur $S_{ii,\lambda}$ par $\text{tr}(S_\lambda)/n$.

- Il existe une variante, appelée **Validation Croisée Généralisée (GCV)** au critère LOO, qui peut se révéler plus stable (voir [**Woods, 2006**]).
- Elle consiste à remplacer au dénominateur $S_{ii,\lambda}$ par $\text{tr}(S_\lambda)/n$.

Définition

Le critère de **Validation Croisée Généralisée (GCV)** est défini par

$$\text{GCV}(\hat{m}_\lambda) = \frac{n \sum_{i=1}^n (y_i - \hat{m}_\lambda(x_i))^2}{(n - \text{tr}(S_\lambda))^2}.$$

- Il existe une variante, appelée **Validation Croisée Généralisée (GCV)** au critère LOO, qui peut se révéler plus stable (voir [**Woods, 2006**]).
- Elle consiste à remplacer au dénominateur $S_{ii,\lambda}$ par $\text{tr}(S_\lambda)/n$.

Définition

Le critère de **Validation Croisée Généralisée (GCV)** est défini par

$$\text{GCV}(\hat{m}_\lambda) = \frac{n \sum_{i=1}^n (y_i - \hat{m}_\lambda(x_i))^2}{(n - \text{tr}(S_\lambda))^2}.$$

- On remarque, une fois de plus, que ce critère réalise un compromis entre **ajustement** (numérateur) et **complexité** (dénominateur).

- Les composantes du modèle additif ne nécessitent pas toutes le même degré de lissage.
- On utilise un lissage pour chaque composante dans l'algorithme du backfitting.

- Les composantes du modèle additif ne nécessitent pas toutes le même degré de lissage.
- On utilise un lissage pour chaque composante dans l'algorithme du backfitting.
- Il convient donc de sélectionner d paramètres de lissage $\lambda = (\lambda_1, \dots, \lambda_d)$.

- Les composantes du modèle additif ne nécessitent pas toutes le même degré de lissage.
- On utilise un lissage pour chaque composante dans l'algorithme du backfitting.
- Il convient donc de sélectionner d paramètres de lissage $\lambda = (\lambda_1, \dots, \lambda_d)$.
- Il est possible de montrer (voir [Woods, 2006]) que l'estimateur final du modèle additif est également un lisseur dont le degrés de liberté dépend des paramètres de lissage $\lambda_1, \dots, \lambda_d$.

- Les composantes du modèle additif ne nécessitent pas toutes le même degré de lissage.
- On utilise un lissage pour chaque composante dans l'algorithme du backfitting.
- Il convient donc de sélectionner d paramètres de lissage $\lambda = (\lambda_1, \dots, \lambda_d)$.
- Il est possible de montrer (voir [Woods, 2006]) que l'estimateur final du modèle additif est également un lisseur dont le degrés de liberté dépend des paramètres de lissage $\lambda_1, \dots, \lambda_d$.
- C'est pourquoi GCV est fréquemment utilisé pour sélectionner les paramètres de lissage dans les modèles additifs.
- Sur R, on pourra utiliser le package mgcv.

Le coin R

```
> mod.mgcv <- mgcv::gam(Y~s(X1)+s(X2),data=donnees)
> mod.mgcv
##
## Family: gaussian
## Link function: identity
##
## Formula:
## Y ~ s(X1) + s(X2)
##
## Estimated degrees of freedom:
## 1.00 8.56 total = 10.56
##
## GCV score: 1.536021
```

Le coin R

```
> mod.mgcv <- mgcv::gam(Y~s(X1)+s(X2),data=donnees)
> mod.mgcv
##
## Family: gaussian
## Link function: identity
##
## Formula:
## Y ~ s(X1) + s(X2)
##
## Estimated degrees of freedom:
## 1.00 8.56 total = 10.56
##
## GCV score: 1.536021
```

Remarque

Sans surprise, le **degré de liberté** sélectionné pour la deuxième composante est plus **grand** que celui de la première.

Régression logistique additive

- Nous nous sommes restreints à un modèle de régression (Y continue), qui est un modèle GLM particulier.

Régression logistique additive

- Nous nous sommes restreints à un **modèle de régression** (Y continue), qui est un modèle **GLM** particulier.
- Bien entendu, ce type de modélisation s'**étend à l'ensemble des GLM**, d'où le nom de modèle **GAM** (Generalized Additive Model).

Régression logistique additive

- Nous nous sommes restreints à un **modèle de régression** (Y continue), qui est un modèle **GLM** particulier.
- Bien entendu, ce type de modélisation s'**étend à l'ensemble des GLM**, d'où le nom de modèle **GAM** (Generalized Additive Model).

Régression logistique additive

Pour une variable Y binaire, le modèle de **régression logistique additive** s'écrit :

$$\text{logit } p(x) = \log \frac{\mathbf{P}(Y = 1|X = x)}{1 - \mathbf{P}(Y = 1|X = x)} = \alpha_0 + g_1(x_1) + \dots + g_d(x_d).$$

Régression logistique additive

- Nous nous sommes restreints à un **modèle de régression** (Y continue), qui est un modèle **GLM** particulier.
- Bien entendu, ce type de modélisation s'**étend à l'ensemble des GLM**, d'où le nom de modèle **GAM** (Generalized Additive Model).

Régression logistique additive

Pour une variable Y binaire, le modèle de **régression logistique additive** s'écrit :

$$\text{logit } p(x) = \log \frac{\mathbf{P}(Y = 1|X = x)}{1 - \mathbf{P}(Y = 1|X = x)} = \alpha_0 + g_1(x_1) + \dots + g_d(x_d).$$




- La **procédure d'estimation** combine l'algorithme du **backfitting** avec l'algorithme du **score de Fisher** (voir [Hastie et al., 2009]).


Présentation du modèle

Estimation : l'algorithme du backfitting

Lisseurs

Bibliographie

-  Cornillon, P., Hengartner, N., Matzner-Løber, E., and Rouvière, L. (2019).
Régression avec R.
EDP Sciences, 2 edition.
-  Hastie, T., Tibshirani, R., and Friedman, J. (2009).
The Elements of Statistical Learning : Data Mining, Inference, and Prediction.
Springer, second edition.
-  Opsomer, J. (2000).
Asymptotic properties of backfitting estimators.
Journal of Multivariate Analysis, 73 :166–179.

-  Woods, S. (2006).
Generalized additive models.
Chapman & Hall.