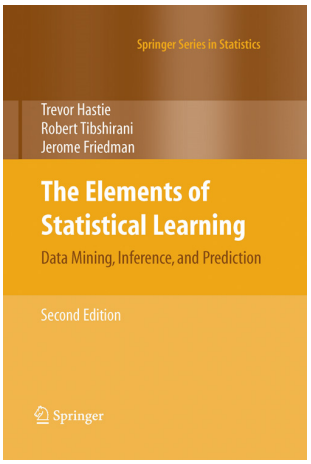
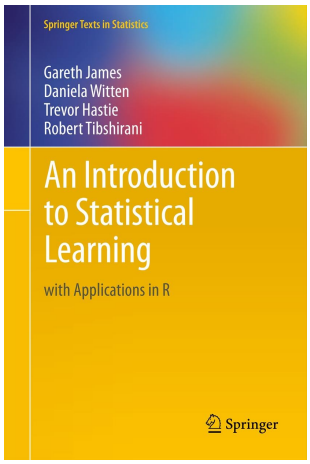
Première partie

Introduction à l'apprentissage statistique

Nous avons ici les deux principales références dont je me suis servi pour rédiger ces notes. Fait rare, ces deux livres sont disponibles gratuitement en pdf sur les sites des auteurs.

Références





1 Introduction générale

L'apprentissage statistique [1] est l'art de programmer les ordinateurs de sorte qu'ils puissent apprendre à partir des données. Voici une définition générale

> L'apprentiſſage automatique eſt la diſcipline donnant aux ordinateurſ la capacité d'apprendre ſanſ qu'ilſ ſoient explicitement programméſ.
>
> Arthur Samuel, 1959

Et une autre plus technique

> Étant donné une tâche T et une meſure de performance P, on dit qu'un programme informatique apprend à partir d'une expérience E ſi leſ réſultatſ obtenuſ ſur T, meſuréſ par P, ſ'améliorent avec l'expérience E.
>
> Tom Mitchell, 1997

1.1 Problèmes d'apprentissage statistique

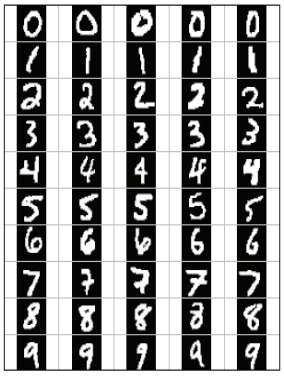
La première application de l'apprentissage statistique ayant véritablement touché un large public, il s'agit du filtre anti-spam. Le filtre anti-spam de nos boites e-mails qui peut apprendre à identifier les e-mails frauduleux à partir d'exemples de pourriels ou « spam » (par exemple, ceux repérés par les utilisateurs) et de messages normaux parfois appelés « ham ». Les exemples de messages utilisés par une méthode d'apprentissage constituent le jeu de données dit **d'entraînement** (*training set* en anglais). Nous donnons ici deux problèmes typiques d'apprentissage

Détection des mails frauduleux :

- Sur 4601 mails, on a pu identifier 1813 spams.
- On a également mesuré sur chacun de ces mails la présence ou absence de 57 mots.
- *Peut-on construire à partir de ces données une méthode de détection automatique de spam ?*

Reconnaissance de chiffres manuscrits : L'apprentissage statistique est présent depuis des décennies dans certaines applications spécialisées telles que la *reconnaissance optique de caractères* ou OCR.

1. Ou apprentissage automatique ou *machine learning* en anglais



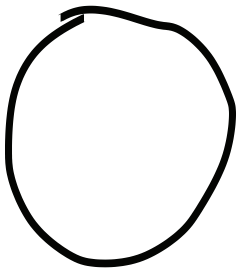


FIGURE 1 – Peut-on mettre en place une méthode automatique de reconnaissance des chiffres manuscrits.

Nous avons ici une petite liste de problèmes d'apprentissage statistique.

1. Identifier les facteurs de risque du cancer de la prostate
2. Prédire si une personne est sujette au crise cardiaque, à partir de mesure clinique, son régime et des données démographiques
3. Classification d'échantillons de tissus dans différents types de cancer, en fonction de données d'expression de gènes
4. Classer des images de tumeurs

1.2 Représentation du problème et vocabulaire

La plupart de ces problèmes peuvent être appréhendés dans un contexte de **régression ou classification supervisée** : on cherche à expliquer une variable Y par d'autres variables dites explicatives $X_1, \ldots, X_p$ souvent représentées par un vecteur une matrice X.

Y	X
Chiffre (OCR)	Image sous forme d'une matrice de pixels
Mot (reconnaissance vocale)	courbe
Spam ou ham	présence/absence d'un ensemble de mots
Type de leucémie	expressions de gênes

— Lorsque la variable à expliquer est quantitative, on parle de **régression**.
— Si elle est qualitative, on parle de **discrimination** ou **classification supervisée** [2].

1.3 Régression

Pour formuler le problème de régression, nous avons besoin des notations suivantes
— Un échantillon i.i.d $(X_1, Y_1), \ldots, (X_n, Y_n)$ à valeurs dans $\mathbb{R}^p \times \mathbb{R}$.

2. Quand il s'agit de clustering, on parle de classification non-supervisée où on cherche à détecter des groupes homogènes d'individus. Le clustering ne fait pas appel à une variable réponse qui est considérée comme le superviseur en classification supervisée.

— Objectif : Prédire ou expliquer la variable Y à partir d'une nouvelle observation X.
— Méthode : construire **une méthode de régression**

$$m : \mathbb{R}^p \mapsto \mathbb{R}.$$

— Critère de performance pour m : l'**erreur de prévision ou erreur quadratique moyenne**

$$\mathbb{E}\left[\big(Y - m(X)\big)^2\right]. \tag{1}$$

Si l'écriture $\mathbb{E}\left[\big(Y - m(X)\big)^2\right]$ semble un peu technique, on peut se contenter de l'interprétation : *la moyenne du carré de l'écart de la variable à expliquer Y et la fonction de régression m dans la population.* On fera souvent appel à sa version empirique dite **erreur d'entraînement** calculée sur le jeu de données d'**entrainement** ou d'**apprentissage** $(x_1, y_1), \ldots, (x_n, y_n)$ définie par

$$\frac{1}{n}\sum_{i=1}^{n}\big(y_i - m(x_i)\big)^2.$$

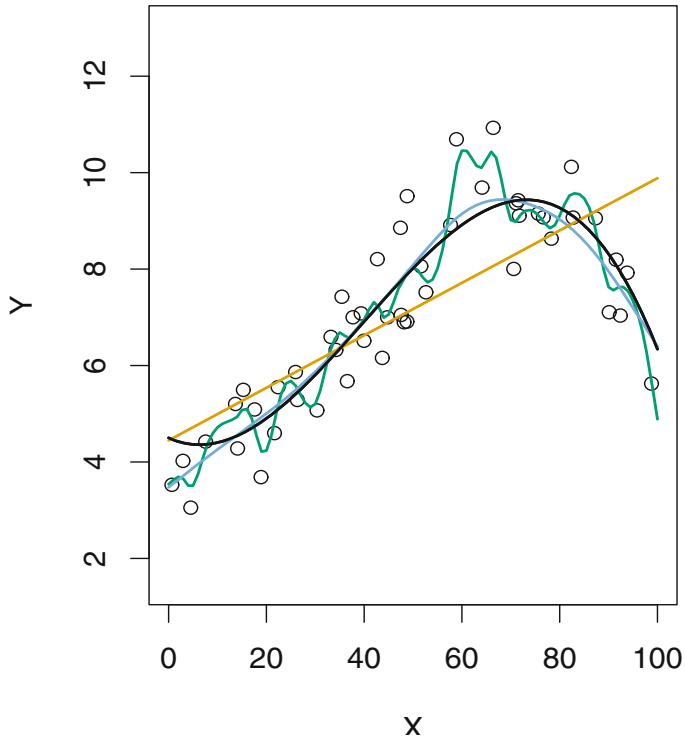


FIGURE 2 – Un exemple jouet en régression : X est en dimension 1 et Y est une variable quantitative. Le nuage de point provient de la courbe en noir (la vérité). Nous avons testé 3 méthodes de régression. Droite orange : une droite de régression (une fonction de la forme $y \approx \beta_0 + \beta_1 x$). En bleu : une fonction de régression en forme de polynôme $y \approx \beta_0 + \beta_1 x + \beta_2 x^2 + \ldots + \beta_p x^p$. En vert : une fonction obtenue par la méthode dite *spline*.

La légende de la figure 2 évoque des notions qui ne sont pas encore étudiées comme le concept de *spline*[3] et nous n'avons aucune information sur les mécanismes de calcul des différents paramètres qui interviennent dans les formules comme les coefficients β_0 et β_1 dans la droite orange. La figure illustre en partie[4] la richesse de la classe de modèles de régression m qu'on rencontre dans un problème de régression. Les trois modèles *droite de régression, polynôme et spline* illustrés sur la figure 2 sont de complexités croissantes[5]. La question du choix de la meilleure fonction de régression se pose naturellement et cela se fait en minimisant l'erreur quadratique moyenne 1.

Un champion Avec un peu de manipulation technique du concept d'espérance conditionnelle, on peut montrer que la meilleure fonction qui minimise le critère 1 est donnée par

$$m^*(x) = \mathbb{E}\big[Y|X = x\big] \tag{2}$$

appelée fonction de régression. Pour toute autre fonction m, nous avons

$$\mathbb{E}\left[\big(Y - m^*(X)\big)^2\right] \leq \mathbb{E}\left[\big(Y - m(X)\big)^2\right].$$

Problème m^*[6] est inconnu en pratique. Il faut construire une méthode de régression $\widehat{m}_n$ à partir des données $(X_1, Y_1), \ldots, (X_n, Y_n)$, tel que

$$\widehat{m}_n(x) \approx m^*(x).$$

$\widehat{m}_n$ peut être choisie parmi les méthodes de régression illustrées sur la figure 2.

Une solution Soit une valeur x_0[7] pour laquelle on souhaite prédire la valeur de y_0 à l'aide $m^*(x_0)$.

- Nous n'avons aucune observation (x_i, y_i) avec $x_i = x_0$.
- On ne sait pas calculer $\mathbb{E}\big[Y|X = x_0\big]$.
- Intuitivement, on peut faire appel à l'approximation

$$\widehat{m}(x_0) = \text{Moyenne}\Big(y_i | x_i \in \mathcal{N}(x_0)\Big)$$

où $\mathcal{N}(x_0)$ représente un certain voisinage de x_0. On peut s'intéresser aux k couples (x_i, y_i) correspondants aux x_i les plus proches de x_0 dans le jeu de données. Il s'agit de l'estimateur des k plus proches voisins[8].

3. Penser à une fonction définie par morceaux par des polynômes.
4. On citer un grand nombre d'autres modèles ...
5. On peut mesurer la complexité d'un modèle par le nombre de paramètre qu'il implique.
6. Si X est la taille d'un individu et Y son poids. $m^*(x)$ correspond à la moyenne du poids des individus de taille fixée à x dans la population.
7. Prenons une valeur sur l'axe des x de la figure 2.
8. On notera k-nn pour *k-nearest neighbors* en anglais.

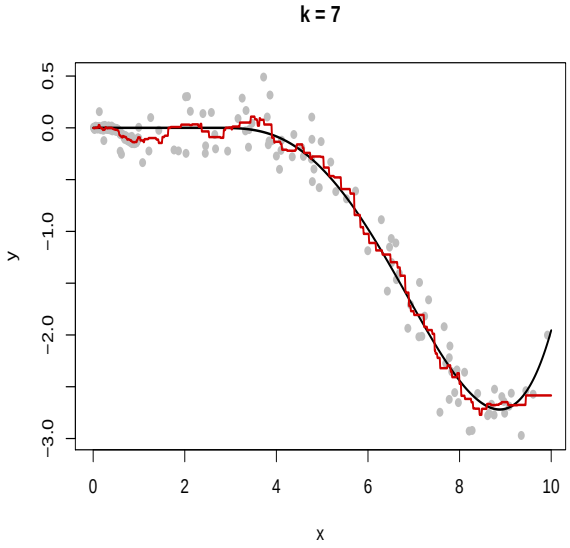


FIGURE 3 – Exemple de régression par k-nn. En noir : la vraie courbe qui correspond au modèle de simulation. En rouge, l'approximation $\widehat{m}_n$ obtenue par la méthode des 7-nn.

1.4 Classification

Commençons par l'introduction des notations, l'objectif de prédiction et le critère de performance en classification.

— Un échantillon i.i.d $(X_1, Y_1), \ldots, (X_n, Y_n)$ à valeurs dans $\mathbb{R}^p \times \{0, 1\}$ [9]. En classification, la variable réponse Y est appelée **label** ou **classe**.

— Objectif : Prédire ou expliquer la variable Y à partir d'une nouvelle observation X.

— Méthode : construire une **règle de classification**

$$g : \mathbb{R}^p \mapsto \{0, 1\}.$$

— Critère de performance pour g : **probabilité d'erreur**

$$\mathbb{P}\big(g(X) \neq Y\big).$$

Intuitivement, cette mesure d'erreur correspond à la proportion théorique d'individus mal classés par la règle de classification g. La version empirique de l'erreur de classification calculée sur le jeu de données d'entraînement, est définie par

$$\frac{1}{n}\sum_{i=1}^{n} \mathbb{1}\Big[y_i \neq g(x_i)\Big].$$

9. Nous allons nous restreindre à la classification binaire ou la variable réponse est à valeurs dans $\{0, 1\}$. Penser à n'importe quelle variable catégorielle à deux modalités.

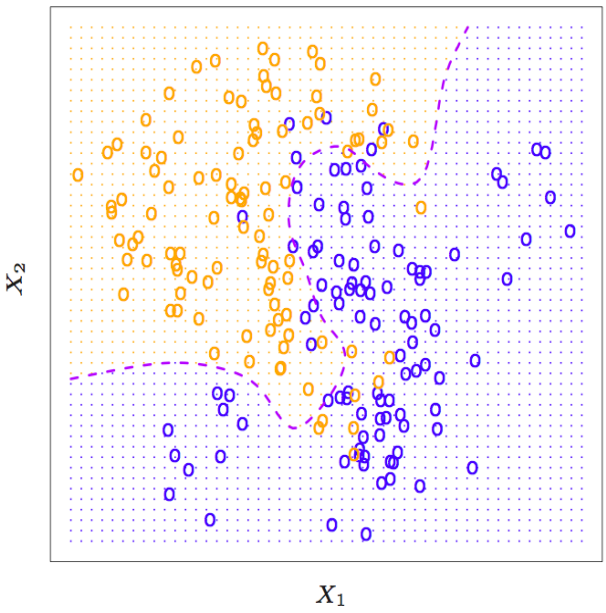


FIGURE 4 – Cette figure illustre un problème de classification simple avec une variable explicative à valeurs dans $\mathbb{R}^2$. Le label y_i de chaque individu correspond à sa couleur. La règle de classification établie est représentée par la courbe appelée frontière de classement. On peut identifier les erreurs de classement commises par le modèle proposé.

Un champion Le règle de classification donnée par :

$$g^*(x) = \begin{cases} 0 & \text{si} \quad \mathbb{P}\big(Y=0|X=x\big) \geq \mathbb{P}\big(Y=1|X=x\big) \\ 1 & \text{sinon,} \end{cases}$$

appelée **règle de Bayes** [10]. Quelque soit la règle de décision g, nous avons

$$\mathbb{P}\big(g^*(X) \neq Y\big) \leq \mathbb{P}\big(g(X) \neq Y\big).$$

Problème la règle de classification g^* est inconnue en pratique. Il faut construire une règle $\widehat{g}_n$ à partir des données d'entraînement $(x_1, y_1), \ldots, (x_n, y_n)$, telle que

$$\widehat{g}_n(x) \approx g^*(x).$$

Un candidat pour approcher $\widehat{g}_n$ est la régression logistique, couramment utilisée comme modèle de classification binaire.

Solution Soit un point x_0 sur la figure 4 pour lequel nous n'avons pas observé de label y.

— On ne sait pas calculer $\mathbb{P}\big[Y|X=x_0\big]$.

— Intuitivement, on peut faire appel à l'approximation

$$\widehat{g}_n(x_0) = \begin{cases} 1 & \text{si} \quad \frac{1}{|\mathcal{N}(x_0)|} \sum\limits_{i \in \mathcal{N}(x_0)} \mathbb{1}\big(y_i = 1\big) > \frac{1}{|\mathcal{N}(x_0)|} \sum\limits_{i \in \mathcal{N}(x_0)} \mathbb{1}\big(y_i = 0\big) \\ 0 & \text{sinon.} \end{cases}$$

10. C'est l'équivalent de la fonction de régression dans le monde de la classification.

où $\mathcal{N}(x_0)$ représente un certain voisinage de x_0 et $|\mathcal{N}(x_0)|$ est le nombre d'individus dans ce voisinage [11]. On peut s'intéresser aux k individus correspondants aux x_i les plus proches de x_0 dans le jeu de données. Il s'agit de l'estimateur des k plus proches voisins [12] pour la classification. Sur l'exemple illustré sur la figure 4, la règle de classification des k-nn correspond au vote majoritaire. Si les points bleu sont majoritaires parmi les k plus proches voisins alors, on affecte un label bleu à x_0 sinon on lui affecte un label orange.

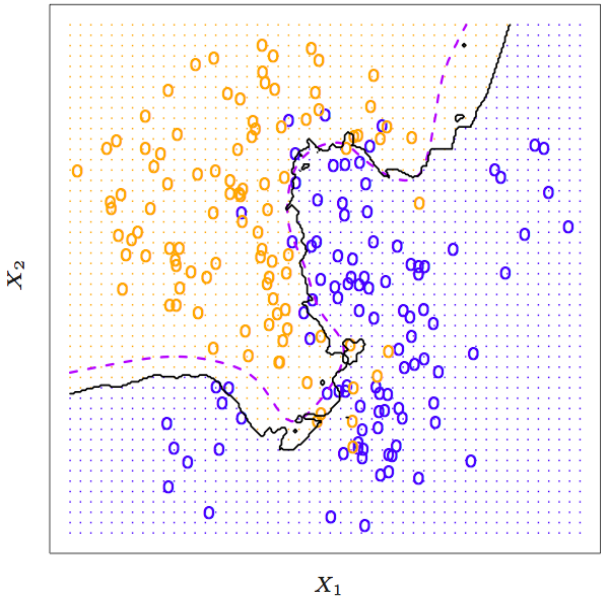


FIGURE 5 – Frontière de classification des 10 plus proches voisins.

2 Complexité de modèle et phénomène de sur-ajustement

La notion de complexité d'un modèle est liée à la famille de modèle à laquelle on s'intéresse pour résoudre un problème d'apprentissage. Sur l'exemple de régression de la figure 2, un modèle d'apprentissage correspond à une des trois courbes représentées. La complexité d'une correspond au nombre de paramètres de la courbe. Dans le cas d'un modèle obtenu par les k plus proches voisins, cette complexité est un peu plus compliquée à conceptualiser. Dans le cadre de cours on admettra que la complexité d'un modèle est inversement proportionnelle aux nombre de voisins k [13].

2.1 Complexité d'un k-nn pour la classification

Commençons par la lecture d'un jeu de données simulé issu du package ElemStatLearn qui permet de retracer une grande partie les jeux de données et les figures disponibles dans le livre [1]. Nous

11. $|\mathcal{N}(x_0)|$ est le cardinal de l'ensemble $\mathcal{N}(x_0)$.
12. On notera k-nn pour *k-nearest neighbors* en anglais.
13. Dans le cas d'une classification, on s'intéressera à la complexité de la frontière de classement. Nous allons illustré ce phénomène sur les exemples simulés que nous allons traiter.

avons besoin d'appeler le package class qui permet d'implémenter un modèle k-nn. Nous allons faire appel au jeu de données simulé mixture.example pour illustrer notre problème.

```
require(ElemStatLearn)
require(class)
data("mixture.example")
names(mixture.example)
x = mixture.example$x
y = mixture.example$y
px1 = mixture.example$px1
px2 = mixture.example$px2
xnew = mixture.example$xnew
plot(x, col=ifelse(y==1,"red", "green"), xlab="x1", ylab="x2")
?knn
```

Le bloc de code suivant permet de mettre en place un modèle de **classification par les** 15 **plus proches voisins**.

```
mod15 = knn(x, xnew, y, k=15, prob=TRUE)
summary(mod15)
prob = attr(mod15, "prob")
# fraction de votants pour la classe majoritaire!
prob = ifelse(mod15=="1", prob, 1-prob)
prob15 = matrix(prob, length(px1), length(px2))
contour(px1, px2, prob15, levels=0.5, labels="", xlab="x1", ylab="x2",
        main="15-nearest neighbour")
points(x, col=ifelse(y==1, "red", "green"))
```

— Expliquer l'objectif de chaque ligne de code du bloc précédent
— Calculer l'erreur de classement des 200 points contenus dans x par le modèle des 15-nn.
— Reprendre le bloc de code précédent pour implémenter le modèle des 1-nn. Tracer la frontière de classement. Calculer l'erreur de classement des 200 points contenus dans x par le modèle des 1-nn. Conclure.

Le bloc de code suivant permet de générer 10000 points du même modèle qui a servi pour générer le jeu de données précédent.

```
require(MASS)
means  = mixture.example$means
set.seed(123)
centers = c(sample(1:10, 5000, replace=TRUE),
            sample(11:20, 5000, replace=TRUE))
means = means[centers, ]
xtest = mvrnorm(10000, c(0,0), 0.2*diag(2)) + means
ytest = c(rep(0, 5000), rep(1, 5000))
K <- c(1,3,5,7,9,11,15,17,23,25,35,45,55,83,101,151)
```

L'objet xtest contient les 10000 points générés. Les 5000 premiers points appartiennent à un groupe et les 5000 restant appartiennent à un autre groupe.

— Pour chaque valeur k dans l'objet K, implémenter un modèle k-nn sur le jeu de données de départ constitué des 200 points de départ.
  — Calculer l'erreur de classement du jeu de données contenu dans x.
  — Calculer l'erreur de classement du jeu de données contenu dans xtest.
— Tracer sur la même figure, les deux erreurs de classement en fonction de la valeur de k.
— Conclure.

Deuxième partie

Régression linéaire

La partie I introduisait les problématiques de régression et de classification ainsi que les notions de complexité de modèle et le phénomène de sur-ajustement. En régression, le modèle linéaire est enseigné dans les parcours de statistique en premier cycle universitaire. La visualisation et la compréhension relativement simple du critère des moindres carrés le rend plus commode pour l'enseignement. L'inférence statistique est simplifié dans le cas d'une régression linéaire. Les sections 3 et 4 expliquent brièvement le modèle de régression linéaire en présence d'une seule variable explicative pour la régression linéaire simple ou plusieurs variables explicatives pour la régression linéaire multiple. On introduit un minimum de notations et de formalisme nécessaire pour décrire la méthode des moindres carrés. Le principe de minimisation des moindres carrés permet d'estimer les paramètres de régression dans les deux cas simple et multiple. L'exemple de régression linéaire simple permet de visualiser et d'assimiler intuitivement le principe des moindres carrés (voir la figure 7). Nous détaillons le calcul de dérivés qui permet de déduire les formules des deux coefficients de régression. Nous illustrons ce même principe via une figure en dimension 3 (voir figure 9). L'écriture matricielle du problème de régression linéaire multiple permet de faciliter le calcul du vecteur des coefficients qui minimise le principe des moindres carrés dans le cas multiple. Comme on peut le constater, l'inférence statistique liée à l'estimation des paramètres de régression ainsi que les diagnostiques de validité d'un modèle de régression linéaire sont volontairement occultés. Pour plus de détails concernant les intervalles de confiance ainsi que les tests statistique dans un modèle de régression linéaire, il est fortement recommandé de lire attentivement les sections 3.1 et 3.2 du livre [2].

3 Régression linéaire simple

Pour introduire la problématique de régression linéaire, nous allons nous servir d'un exemple de jeu de données de cancer de la prostate

3.1 Motivation : données du cancer de la prostate

La tableau 1 résume les variables qui composent le jeu de données du cancer de la prostate [14].

Question : En général, on fait appel à un modèle linéaire pour répondre à la question suivante *Existe-t-il une relation entre* ***lcavol*** *et le* ***lpsa*** *?* Notre premier objectif est de déterminer si les données témoignent d'une association entre **lcavol** et le **lpsa**.

14. Stamey, T.A., Kabalin, J.N., McNeal, J.E., Johnstone, I.M., Freiha, F., Redwine,E.A. and Yang, N. (1989). Prostate specific antigen in the diagnosis and treatment of adenocarcinoma of the prostate : II. radical prostatectomy treated patients, Journal of Urology 141(5), 1076-1083.

lcavol	log(cancer volume)
lweight	log(prostate weight)
age	age
lbph	log(benign prostatic hyperplasia amount)
svi	seminal vesicle invasion
lcp	log(capsular penetration)
gleason	Gleason score
pgg45	percentage Gleason scores 4 or 5
lpsa	log(prostate specific antigen)

TABLE 1 – These data 1 come from a study that examined the correlation between the level of prostate specific antigen and a number of clinical measures in men who were about to receive a radical prostatectomy. It is data frame with 97 rows and 9 columns.

Le modèle linéaire simple permet de répondre à l'ensemble des questions suivantes

- Quelle est la relation entre **lcavol** et le **lpsa** ?
- En supposant qu'il existe une relation entre **lcavol** et le **lpsa**, nous aimerions connaître la force de cette relation. Autrement dit, compte tenu d'un certain **lcavol**, pouvons-nous prédire le **lpsa** ?
- Cette relation est-elle forte ou une prédiction du **lpsa** uniquement sur le **lcavol** serait légèrement mieux qu'une hypothèse aléatoire (relation faible) ?
- Quelles mesures expliquent le **lpsa** ? et comment ?
- La relation est-elle linéaire ? S'il y a approximativement une relation linéaire entre les différentes variables et le **lpsa**, alors la régression linéaire est appropriée. Sinon, il est encore possible de transformer les variables pour que la régression linéaire puisse être utilisée.
- A-t-on des interactions entre les différentes mesures ?

De quoi s'agit-il ?

Une régression (*regression analysis* en anglais) est une méthode pour étudier la relation fonctionnelle entre deux variables. Dans la partie régression linéaire multiple, nous allons étudier la modélisation de la relation entre plusieurs variables (> 2). La régression linéaire simple modélise la relation entre deux variables par une ligne droite, c'est à dire, une variable Y est modélisée comme une fonction linéaire d'une autre variable X.

La figure 6 illustre des modèles de régression simple autour du jeu de données cancer de la prostate.

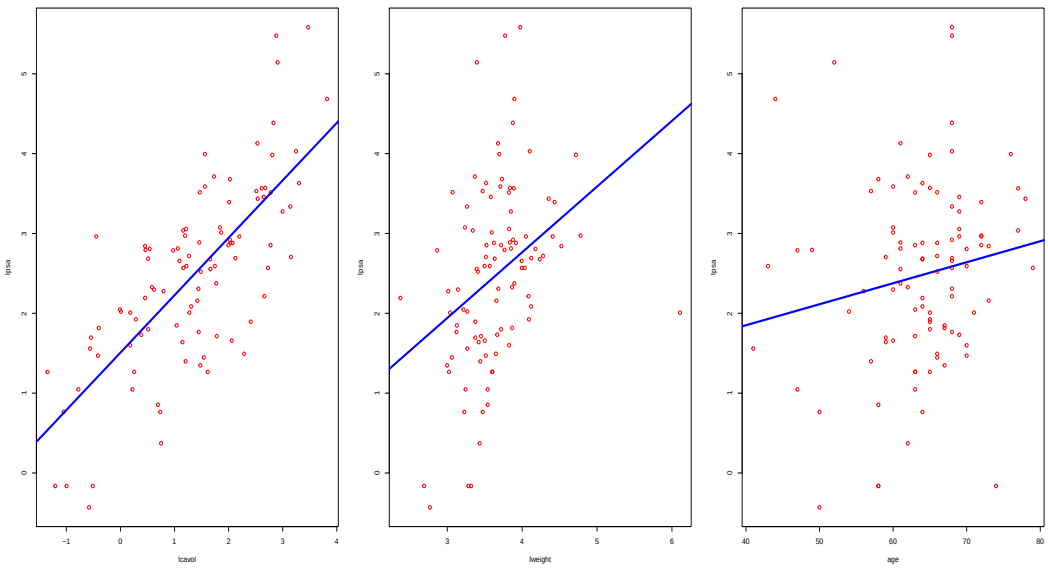


FIGURE 6 – Le nuage de points peut parfois nous donner une idée du type de relation entre les variables (linéaire, quadratique, exponentielle, ...)

3.2 Notations et un peu de formalisme

Rappelons que jeu de données est *souvent* noté

$$(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$$

— x_i est la $i^{\text{ème}}$ valeur de la variable X dite ***explicative*** ou ***covariable***.
— y_i est la $i^{\text{ème}}$ valeur de la variable Y dite ***réponse*** ou ***variable à expliquer***.

Deux situations pour la collecte des valeurs $x_1, x_2, \ldots, x_n$ associées variable X
— Elles sont observées comme dans le cas des données de cancer de la prostate.
— Quand il s'agît d'une étude, elles peuvent être fixées par cette étude.

La régression linéaire est typiquement utilisée pour modéliser la relation entre deux variables Y et X telle que pour une certaine valeur spécifique de $X = x$, on peut prédire la valeur de Y. De manière formelle, la régression d'une variable aléatoire Y sur une variables aléatoire X est

$$\mathbb{E}\big(Y \mid X = x\big),$$

la valeur moyenne de Y lorsque $X = x$.
Par exemple,
— X le **lcavol**
— Y le **lpsa**.

La régression de Y sur X représente le **lpsa** moyen (l'espérance du **lpsa**) pour un **lcavol**.
La régression de Y sur X est linéaire si

$$\mathbb{E}\big(Y \mid X = x\big) = \beta_0 + \beta_1 x$$

où β_0 et β_1 désignent l'***intercept*** et la ***pente*** de la droite de régression.

Supposons que $Y_1, Y_2, \ldots, Y_n$ sont des réalisations indépendantes de la variable aléatoire Y observées aux points $x_1, x_2, \ldots, x_n$ de la variables aléatoires X. Si la régression de Y sur X est linéaire, alors pour $i = 1, 2, \ldots, n$

$$Y_i = \mathbb{E}\big(Y \mid X = x_i\big) + e_i = \beta_0 + \beta_1 x_i + e_i$$

où les e_i sont des erreurs aléatoires associées aux Y_i telles que $\mathbb{E}\big(e_i \mid X\big) = 0$.

Le terme d'erreur :

- L'ajout du terme d'erreur e_i est du à la nature aléatoire de Y. Il y a certainement une certaine variation dans Y qui ne peut être prédite ou expliquée.
- En d'autres termes, toute variation inexpliquée est appelée erreur aléatoire. Ainsi, le terme d'erreur aléatoire e_i ne dépend pas de x et ne contient aucune information sur Y.

On suppose que,

$$\mathrm{Var}\big(Y \mid X = x\big) = \sigma^2.$$

3.3 Estimation de la pente et de l'intercept dans la population

- Supposons que X est la taille, Y est le poids d'un individu choisi aléatoirement dans une population.
- Dans un modèle de régression linéaire simple, le poids moyen d'un individu pour une taille donnée est une fonction linéaire de la taille.
- En pratique, nous n'avons qu'un échantillon de données (un certain nombre de couples (x_i, y_i)) au lieu de la population entière.
- Les valeurs de β_0 et β_1 sont évidemment inconnues.
- On souhaite utiliser les données pour estimer l'intercept (β_0)et la pente (β_1).
- Cela peut être obtenu via une droite qui ajuste au mieux nos données, c'est à dire des valeurs $\hat{\beta}_0$ et $\hat{\beta}_1$ [15] telles que $\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i$ soit aussi proche que possible de y_i.
- La notation $\hat{y}_i$ est utilisée pour noter la valeur donnée par la droite ajustée pour la distinguer de la valeur observée y_i. On dit que $\hat{y}_i$ est la valeur ***prédite*** ou ***ajustée*** de y_i.

3.4 Les résidus

En pratique, on veut minimiser la différence entre les valeurs de y_i et les valeurs prédites $\hat{y}_i$. Cette différence est appelée le résidu, notée $\hat{e}_i$,

$$\hat{e}_i = y_i - \hat{y}_i.$$

La figure 7 illustre les résidus d'une droite de régression.

15. On note $\hat{\beta}$ au lieu de β pour distinguer la estimée du paramètre de sa valeur théorique inconnue.

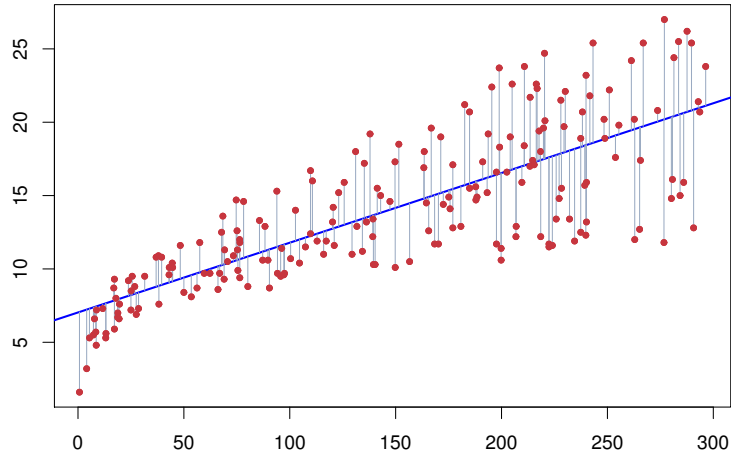


FIGURE 7 – Illustration des résidus d'une droite de régression avec un nuage de points.

3.5 Meilleure approximation au sens des moindres carrés

La méthode couramment utilisée pour choisir les valeurs β_0 et β_1 s'appelle ***méthode des moindres carrés*** . Comme son nom l'indique, β_0 et β_1 sont choisis pour minimiser la somme des résidus notée RSS (*residual sum of squares*),

$$\text{RSS} = \sum_{i=1}^{n} \hat{e}_i^2 = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} (y_i - b_0 - b_1 x_i)^2.$$

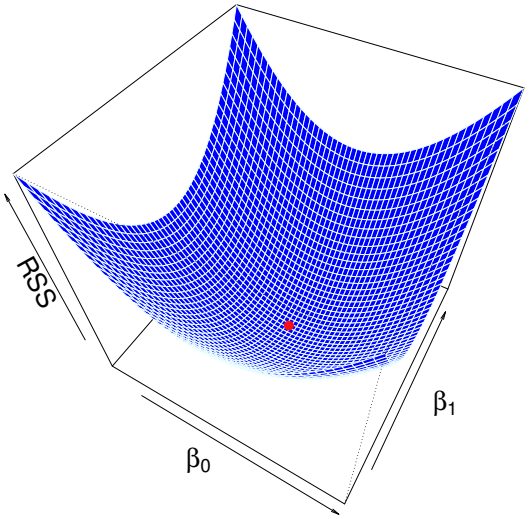
Pour que la somme des moindres carrés RSS soit minimale, il faut que β_0 et β_1 vérifient les équations nous avons les équations

$$\frac{\partial \text{RSS}}{\partial \beta_0} = -2 \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_i) = 0$$

et

$$\frac{\partial \text{RSS}}{\partial \beta_1} = -2 \sum_{i=1}^{n} x_i (y_i - \beta_0 - \beta_1 x_i) = 0$$

La figure suivante illustre la fonction RSS pour différentes valeurs de β_0 et β_1.



En réarrangeant les équations précédentes, on obtient

$$\sum_{i=1}^{n} y_i = \beta_0 n + \beta_1 \sum_{i=1}^{n} x_i$$

et

$$\sum_{i=1}^{n} x_i y_i = \beta_0 \sum_{i=1}^{n} x_i + \beta_1 \sum_{i=1}^{n} x_i^2$$

dites ***équations normales***.
La solution du système composé des deux équations précédentes en β_0 et β_1 nous donne les **estimateurs par les moindres carrés** de l'intercept

$$\hat{\beta}_0 = \bar{y}_n - \hat{\beta}_1 \bar{x}_n$$

et la pente

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n} x_i y_i - n\bar{x}\bar{y}}{\sum_{i=1}^{n} x_i^2 - n\bar{x}^2} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2} = \frac{\text{SXY}}{\text{SXX}}.$$

3.6 Estimation de la variance du terme d'erreur aléatoire

Revenons au modèle de régression linéaire à variance constante décrit précédemment,

$$Y_i = \beta_0 + \beta_1 x_i + e_i \quad (i = 1, 2, \ldots, n)$$

où le l'erreur aléatoire e_i est de moyenne 0 et de variance σ^2. On veut estimer $\sigma^2 = \text{Var}(e)$. Notons que

$$e_i = Y_i - (\beta_0 + \beta_1 x_i) = Y_i - \text{ la droite de régression en } x_i \quad \text{inconnue}$$

Puisque β_0 et β_1 sont inconnues, le mieux qu'on puisse faire, c'est de remplacer ces paramètres inconnus par $\hat{\beta}_0$ et $\hat{\beta}_1$ pour obtenir

$$\hat{e}_i = Y_i - (\hat{\beta}_0 + \hat{\beta}_1 x_i) = Y_i - \text{ la droite de régression estimée au point } x_i.$$

Les résidus peuvent être utilisés pour estimer σ^2. En effet, on peut montrer que

$$S^2 = \frac{\text{RSS}}{n-2} = \frac{1}{n-2}\sum_{i=1}^{n} \hat{e}_i^2$$

est un estimateur sans biais de σ^2. Deux remarques à noter

— $\bar{\hat{e}} = 0$ (puisque $\sum_{i=1}^{n} \hat{e}_i = 0$ car la droite des moindres carrés minimise RSS $= \sum_{i=1}^{n} \hat{e}_i^2$)

— Dénominateur dans S^2 est $n-2$ (nous avons estimé deux paramètres β_0 et β_1)

Le bloc de code suivant illustre l'ajustement d'un modèle de régression linéaire simple sur le jeu de données du cancer de la prostate.

```
## lecture et découverte du jeu de données
 # appel au package qui permet d'accéder au jeu de données
require(lasso2)
 # lecture du jeu de données
data("Prostate")
names(Prostate)
head(Prostate)
dim(Prostate)

## modèle linéaire lpsa en fonction de lcavol
 # ajustement du modèle
lm.fit <- lm(lpsa~lcavol, data=Prostate)
 # résumé du modèle de sortie
summary(lm.fit)
 # les valeurs des coefficients de régression obtenus par moindres carrés
coef(lm.fit)
## visualisation des sorties
attach(Prostate)
 # le nuage de point
plot(lcavol, lpsa, col="red")
 # ajout de la droite de régression au nuage de point
abline(lm.fit, lwd=3, col="blue")
```

4 Régression linéaire multiple

Commençons par le cas particulier d'une régression multiple, connue sous le nom de la régression polynomiale. Dans ce cas, les covariables proviennent de l'observation d'une unique covariable x et ses puissances $(x^2, x^3, \ldots)$. En régression polynomiale, on peut tracer le résultat sur un graphique. La figure 8 illustre un modèle de régression polynomiale de degré 2.

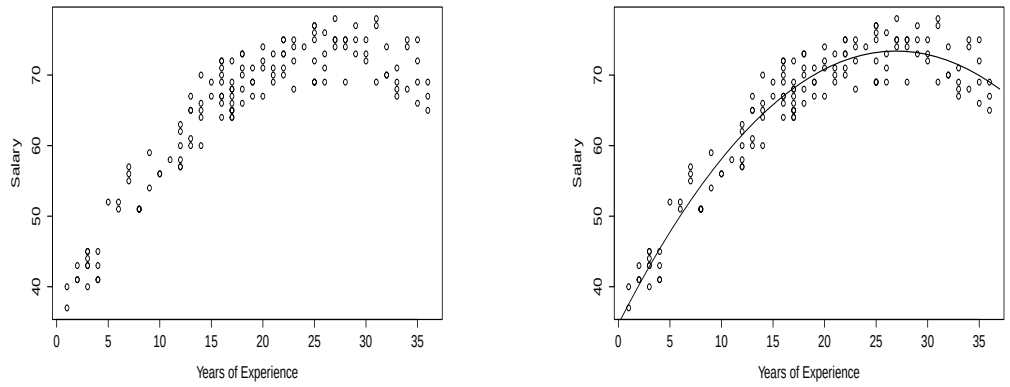


FIGURE 8 – À gauche : Le nuage de points qui suggère une tendance sous forme d'un polynôme. À droite : le modèle de régression quadratique ajusté sur le nuage de points.

La tendance dans le nuage de point suggère un modèle de régression polynomiale

$$Y = \beta_0 + \beta_1 x + \beta_2 x^2 + e$$

4.1 Modèle linéaire multiple

L'estimation des paramètres du modèle de régression revient à choisir les valeurs des paramètres β_0, β_1 et β_2 qui ont permis d'obtenir le polynôme de degré 2 sur la figure 8. Dans cette partie, on introduit un minimum de formalisme nécessaire à l'explication de la méthode des moindres carrés qui permet d'estimer les paramètres du modèle. Dans le modèle de régression multiple, nous avons l'analogue des notations de la régression linéaire simple

$$\mathbb{E}\big(Y \mid X_1 = x_1, X_2 = x_2, \ldots, X_p = x_p\big) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p.$$

Ainsi,

$$Y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} + e_i,$$

où e_i est l'erreur aléatoire dans Y_i telle que $\mathbb{E}\big(e_i \mid X\big) = 0$. Ici, la variable réponse Y (à expliquer) est expliquée par p variables explicatives $X_1, X_2, \ldots, X_p$ et la relation entre Y et $X_1, X_2, \ldots, X_p$ est linéaire en $\beta_0, \beta_1, \ldots, \beta_p$. Dans l'exemple précédent, Y est le salaire, $x_1 = x$ est le nombre d'années d'expérience et $x_2 = x^2$. On cherche l'équation du plan 9

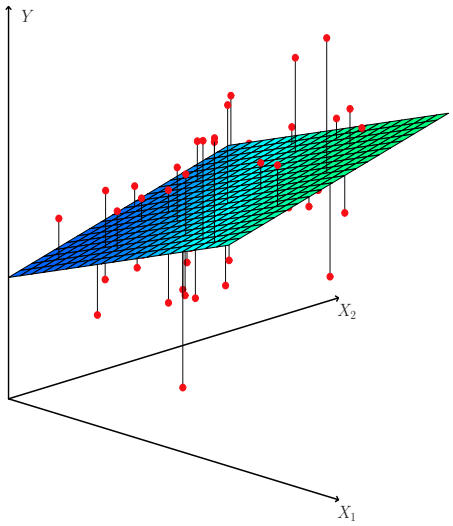


FIGURE 9 – Illustration des moindres carrés en dimension 2.

Les estimateurs des moindres carrés $\hat{\beta}_0, \hat{\beta}_1, \hat{\beta}_2, \ldots, \hat{\beta}_p$ sont les valeurs de $\beta_0, \beta_1, \beta_2, \ldots, \beta_p$ qui minimisent la somme des carrés des résidus

$$\mathrm{RSS}(\boldsymbol{\beta}) = \sum_{i=1}^{n} \hat{e}_i^2 = \sum_{i=1}^{n} \big(y_i - \hat{y}_i\big)^2 = \sum_{i=1}^{n} \big(y_i - \beta_0 - \beta_1 x_{i1} - \beta_2 x_{i2} - \cdots - \beta_p x_{ip}\big)^2.$$

Pour que le $\mathrm{RSS}(\boldsymbol{\beta})$ soit minimal en $\beta_0, \beta_1, \beta_2, \ldots, \beta_p$,

$$\begin{aligned}
\frac{\partial \mathrm{RSS}(\boldsymbol{\beta})}{\partial \beta_0} &= -2 \sum_{i=1}^{n} \big(y_i - \beta_0 - \beta_1 x_{i1} - \beta_2 x_{i2} - \cdots - \beta_p x_{ip}\big) = 0 \\
\frac{\partial \mathrm{RSS}(\boldsymbol{\beta})}{\partial \beta_1} &= -2 \sum_{i=1}^{n} x_{i1} \big(y_i - \beta_0 - \beta_1 x_{i1} - \beta_2 x_{i2} - \cdots - \beta_p x_{ip}\big) = 0 \\
&\vdots \\
\frac{\partial \mathrm{RSS}(\boldsymbol{\beta})}{\partial \beta_p} &= -2 \sum_{i=1}^{n} x_{ip} \big(y_i - \beta_0 - \beta_1 x_{i1} - \beta_2 x_{i2} - \cdots - \beta_p x_{ip}\big) = 0
\end{aligned}$$

Pour pouvoir étudier les propriétés des estimateurs des moindres carrés $\hat{\beta}_0, \hat{\beta}_1, \ldots, \hat{\beta}_p$, nous adoptons des notations vectorielles et matricielles. On définit

— Le vecteur $\mathbf{Y}$ de dimension $(n \times 1)$.
— La matrice $\mathbf{X}$ de taille $n \times (p+1)$.
— Le vecteur de paramètres inconnus (coefficients de régression) $\boldsymbol{\beta}$ de dimension $(p+1) \times 1$.
— Le vecteur des erreurs aléatoires $\mathbf{e}$ de dimension $(n \times 1)$.

Tels que

$$\mathbf{Y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}, \mathbf{X} = \begin{pmatrix} 1 & x_{11} & \cdots & x_{1p} \\ 1 & x_{21} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \cdots & x_{np} \end{pmatrix}, \boldsymbol{\beta} = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{pmatrix} \text{ et } \mathbf{e} = \begin{pmatrix} e_1 \\ e_2 \\ \vdots \\ e_n \end{pmatrix}$$

On peut écrire le modèle de régression multiple sous forme d'équation matricielle

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{e}.$$

De plus, si on note $\mathbf{x}_i'$ la $i^{\text{ème}}$ ligne de la matrice $\mathbf{X}$. Alors

$$\mathbf{x}_i' = \big(1, x_{i1}, x_{i2}, \ldots, x_{ip}\big)$$

est une vecteur ligne $1 \times (p+1)$ et on peut écrire

$$\mathbb{E}\big(Y \mid X = x_i\big) = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} = \mathbf{x}_i'\boldsymbol{\beta}.$$

La somme des carrés des résidus est une fonction de $\boldsymbol{\beta}$ qui s'écrit sous la forme

$$\mathrm{RSS}\big(\boldsymbol{\beta}\big) = \big(\mathbf{Y} - \mathbf{X}\boldsymbol{\beta}\big)'\big(\mathbf{Y} - \mathbf{X}\boldsymbol{\beta}\big).$$

Rappelons que $\big(\mathbf{AB}\big)' = \mathbf{B}'\mathbf{A}'$, nous avons donc

$$\begin{aligned} \mathrm{RSS}\big(\boldsymbol{\beta}\big) &= \mathbf{Y}'\mathbf{Y} + \big(\mathbf{X}\boldsymbol{\beta}\big)'\mathbf{X}\boldsymbol{\beta} - \mathbf{Y}'\mathbf{X}\boldsymbol{\beta} - \big(\mathbf{X}\boldsymbol{\beta}\big)'\mathbf{Y} \\ &= \mathbf{Y}'\mathbf{Y} + \boldsymbol{\beta}'\big(\mathbf{X}'\mathbf{X}\big)\boldsymbol{\beta} - 2\mathbf{Y}'\mathbf{X}\boldsymbol{\beta}. \end{aligned}$$

On sait que pour $v \in \mathbb{R}^d$, $a \in \mathbb{R}^d$ et une matrice $M \in \mathbb{R}^{d\times d}$, nous avons

$$\frac{\partial\big(v'a\big)}{\partial v} = \frac{\partial\big(a'v\big)}{\partial v} = a, \text{ et } \frac{\partial\big(v'Mv\big)}{\partial v} = \big(M + M'\big)v.$$

Sachant que $\mathbf{X}'\mathbf{X}$ est symétrique, à l'aide des deux formules de dérivation vectorielle précédentes, on sait montrer que la solution des moindres carrés qui minimise $\mathrm{RSS}\big(\boldsymbol{\beta}\big)$ est

$$\widehat{\boldsymbol{\beta}} = \big(\mathbf{X}'\mathbf{X}\big)^{-1}\mathbf{X}'\mathbf{Y}. \tag{3}$$

Le bloc de code suivant illustre les résultat d'une régression linéaire multiple sur le jeu de données du cancer de la prostate. Le modèle ajusté explique la variable lpsa en fonction de toutes les autres variables du jeu de données.

```
## lecture et découverte du jeu de données
 # appel au package qui permet d'accéder au jeu de données
require(lasso2)
 # lecture du jeu de données
data("Prostate")
## modèle linéaire lpsa en fonction des autres variables
 # ajustement du modèle
lm.fit <- lm(lpsa~., data=Prostate)
 # résumé du modèle de sortie
summary(lm.fit)
 # les valeurs des coefficients de régression obtenus par moindres carrés
coef(lm.fit)
```

4.2 Complexité d'une régression par polynômes

Nous avons illustré le phénomène de sur-ajustement (*overfitting* en anglais) dans le contexte d'une régression par les k-nn. Dans cette partie, nous allons étudier le phénomène dit de *overfitting* dans le contexte de sélection de modèle [16]. Nous avons un jeu de données qui semble issu d'une fonction polynomiale et nous allons choisir le degré du polynôme pour la régression.

```
# Exemple de données d'apprentissage
# 150 gaussiennes standard
ntr <- 150
x = rnorm(ntr)
# Quadratic Y's
y = 2.5*x^2 - 0.5*x + rnorm(ntr)

# Tracer le nuage de point avec la vraie courbe
plot(x,y,  pch=16, col=8)
curve(2.5*x^2-0.5*x,add=TRUE)

# Ajustons deux modèles basiques
# La constante égale à la moyenne des yi
plot(x,y,  pch=16, col=8)
curve(2.5*x^2-0.5*x,add=TRUE)
y.0 = lm(y ~ 1)
abline(h=y.0$coefficients[1])
# La droite de régression linéaire
d = seq(min(x),max(x),length.out=200) # nous avons besoin d'une grille de x
degree = 1
fm = lm(y ~ poly(x,degree))
assign(paste("y",degree,sep="."), fm)
lines(d, predict(fm,data.frame(x=d)),lty=(degree+1))
```

- Ajuster et ajouter les polynômes allant de 1 à 9
- Calculer et tracer l'erreur d'apprentissage en fonction de la complexité du modèle (ici degré du polynôme)
- Simuler un jeu de données test de taille 150 et l'ajouter à la figure précédente (avec une autre couleur)
- Calculer l'erreur de prédiction sur le jeu de données test en fonction du degré du polynôme et le comparer à l'erreur d'apprentissage

4.3 Pour aller plus loin : validation croisée à 5 folds

- Répliquer 5 fois : simuler un échantillon test et tracer les erreurs de prédiction en fonction du degré du polynôme.
- Simuler un jeu de données de taille 500, à l'aide d'une validation croisée à 5 folds, déterminer le degré optimal du polynôme au sens de l'erreur de prédiction.

16. La complexité d'un modèle de régression par polynôme est donnée par le degré du polynôme.

Troisième partie

Validation croisée et sélection de modèles

Dans cette partie, nous allons discuter de deux méthodes de ***ré-échantillonnage*** : la validation croisée et le bootstrap. Nous faisons le choix de développer la validation croisée. Ces méthodes ré-ajustent le modèle que l'on souhaite sur des échantillons issus de l'échantillon d'apprentissage, dans le but d'obtenir des informations supplémentaires sur ce modèle. Par exemples, ces méthodes fournissent des estimations de l'erreur sur des ensembles de test, le biais et la variance des estimations de paramètres.

5 Erreurs d'entraînement et de test

On rappelle la différence entre ***erreur de test*** et ***erreur d'entraînement***. L'***erreur de test*** est l'erreur moyenne commise par une méthode d'apprentissage statistique pour prédire une réponse sur une nouvelle observation, qui n'a pas été utilisée pour ajuster le modèle. En revanche, l'***erreur d'entraînement*** peut être facilement calculée en appliquant la méthode d'apprentissage sur les données d'entraînement. Mais l'erreur d'entraînement est souvent bien différente de l'erreur de test, et en particulier, l'erreur d'entraînement sous-estime parfois grandement l'erreur de test. On parle d'erreur trop ***optimiste***. La figure 10 illustre ce phénomène.

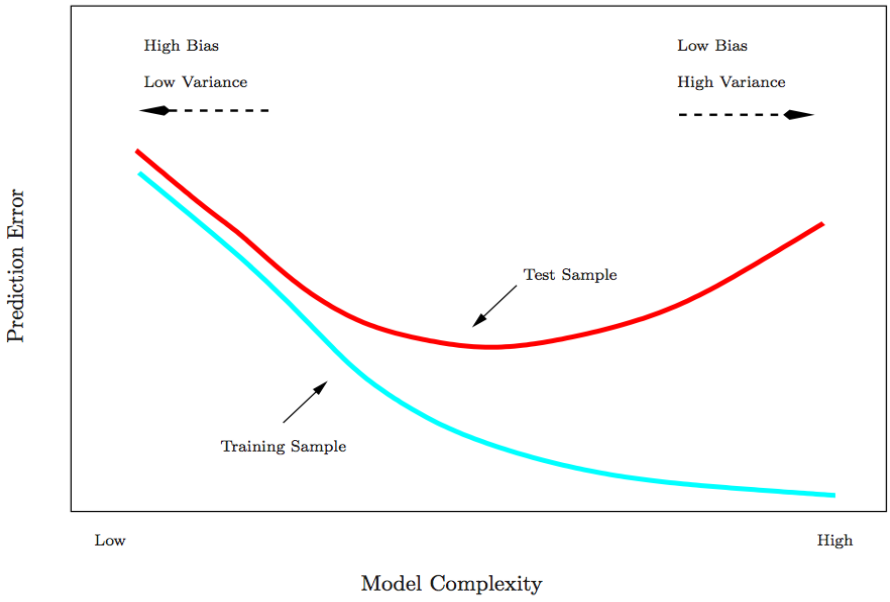


FIGURE 10 – Les deux courbes correspondent à l'erreur de prédiction calculée de deux manières en fonction de la complexité du modèle. Les modèles complexes ont souvent une grande variance tandis que les modèles simples ont un grand biais. La courbe bleu illustre le phénomène de sur-apprentissage :l'optimisme de l'erreur de prédiction calculée sur le jeu de données d'entraînement. La courbe rouge illustre la nécessité d'utiliser un jeu de données test qui n'a pas servi à l'estimation (entrainement) du modèle. La complexité optimale du modèle correspondant au compromis biais-variance.

5.1 Estimation de l'erreur de prédiction

La meilleure solution : un grand ensemble de test clairement désigné. Bien souvent, ce n'est pas disponible. Certaines méthodes permettent de corriger l'erreur d'entraînement pour estimer l'erreur de test, avec des arguments fondés mathématiquement. Cela inclut les C_p **de Mallows**, les critères **AIC** et **BIC** développées dans le contexte de la régression linéaire multiple pour le choix du nombre de variables explicatives à conserver dans le modèle. Le choix de ne pas étudier ces méthodes est délibéré. Dans ces notes, pour la sélection de variables en régression, nous faisons le choix d'étudier les méthodes dites de régularisation qui permettent de traiter de très grand nombre de covariables. La partie IV sera consacrée à la motivation et au développement des méthodes dites de régularisation. Ici, nous nous intéressons à une classe de méthodes qui estime l'erreur de test en mettant de côté un sous-ensemble des données d'entraînement disponibles pour ajuster les modèles, et en appliquant la méthodes ajustée sur ces données mises de côté.

5.2 Approche par ensemble de validation

Cette méthode propose de diviser l'échantillon d'apprentissage en deux (voir figure 11) : un ensemble d'entraînement et un ensemble de validation. Le modèle est ajusté sur l'ensemble d'entraînement, et on l'utilise ensuite pour prédire les réponses sur l'échantillon de validation. L'erreur obtenue en comparant prédiction et observation sur cet échantillon de validation approche l'erreur de test. On utilise typiquement des moindres carrés (MSE) en régression et des taux de mauvaises classification si la réponse est qualitative (ou une fonction de coût d'erreur)

FIGURE 11 – Exemple de partition aléatoire en jeux de données apprentissage-test (ou validation).

Exemple sur les données simulées (degré 2) On veut comparer la régression linéaires à des régressions polynomiales de différents degrés On divise en deux les 200 observations : 100 pour l'entraînement, 100 pour le test.

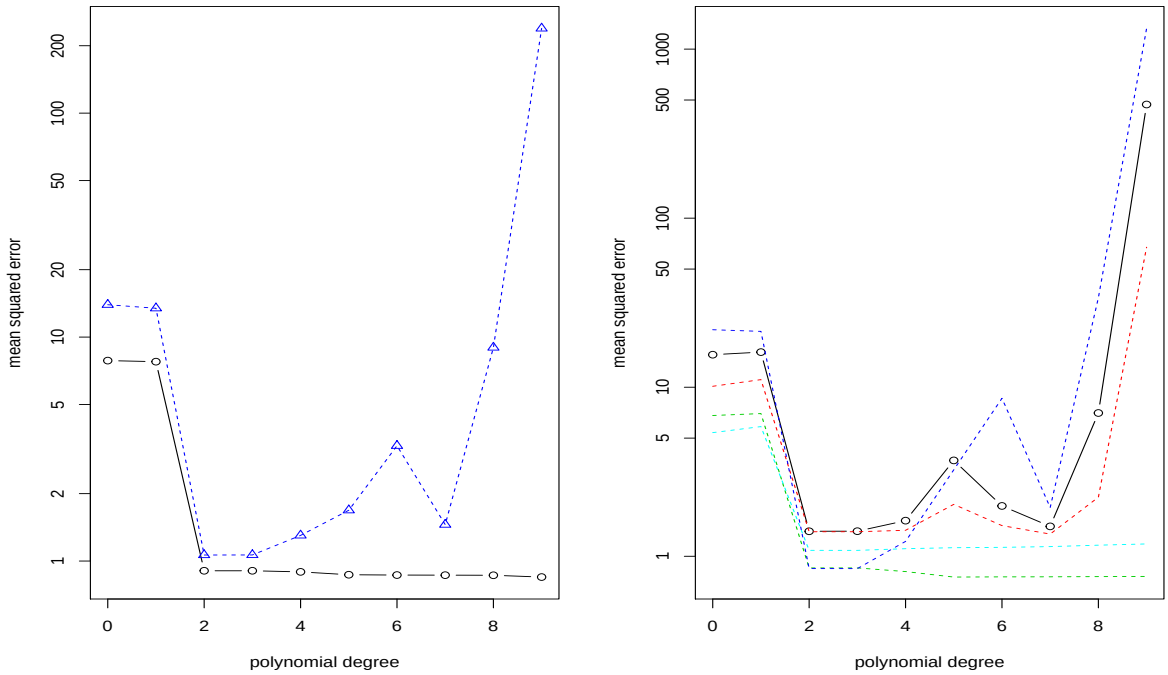


FIGURE 12 – À gauche : l'illustration de l'erreur de test obtenue avec une seule partition aléatoire du jeu de données en apprentissage-test. À droite : l'illustration de la variabilité de l'erreur de test d'une partition aléatoire apprentissage-test à une autre.

Inconvénients de l'approche : L'estimation obtenue par cette méthode peut être très variable, et dépend de la chance ou malchance dans la construction du sous-échantillon de validation. Dans cette approche, seule une moitié des observations est utilisée pour ajuster les modèles — celles qui sont dans l'ensemble d'entraînement. Cela suggère que l'erreur calculée peut surestimer l'erreur de test d'un modèle ajusté sur l'ensemble des données (moins de variabilité d'échantillonnage dans l'inférence des paramètres du modèle). **Déjà mieux :** échanger les rôles entrainement-validation et faire la moyenne des deux erreurs obtenues. On ***croise*** les rôles.

6 Validation croisée à K groupes

C'est la méthode la plus couramment utilisée pour estimer l'erreur de test. L'estimation peut être utilisée pour choisir le meilleur modèle (la meilleure méthode d'apprentissage), ou approcher l'erreur de prédiction du modèle finalement choisi. L'idée est de diviser les données en K groupes de même taille (voir figure 13). On laisse le k-ème bloc de côté, on ajuste le modèle, et on l'évalue sur le bloc laissé de côté. On répète l'opération en laissant de côté le bloc $k = 1$, puis $k = 2, \ldots$ jusqu'à $k = K$. Et on combine les résultats.

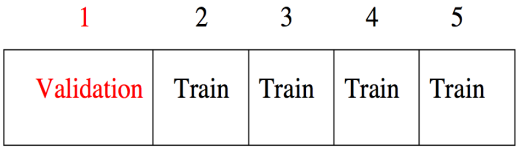


FIGURE 13 – Illustration d'un pas dans une validation croisée à 5 folds.

Pour chacune des observations, on obtient une prédiction $\hat{y}_i = \hat{m}(x_i)$ ou $\hat{g}(x_i)$ au moment où i est dans le groupe mis de côté, et une seule prédiction. On compare alors ces prédictions aux observations comme pour l'erreur de test

$$\text{err}_{(K)} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

ou

$$\text{err}_{(K)} = \frac{1}{n}\sum_{i=1}^{n}\mathbb{1}\Big\{y_i - \hat{g}(x_i)\Big\}$$

Lorsque $K = n$, on parle de « *leave-one out cross-validation* » (LOOCV).

Danger avec le leave-one out : On dit que LOOCV ne secoue pas assez les données. En effet, les règles de classification $\widehat{g}$ ou les fonctions de régression inférées $\widehat{m}$ avec $(n-1)$ données sont très corrélés les uns aux autres. On ne voit plus l'erreur d'échantillonnage, autrement dit la variabilité de l'estimation de la fonction. C'était pourtant tout l'intérêt de la validation croisée. On choisit généralement $K = 5$ ou $K = 10$ blocs. Revenons au jeu de données simulé (voir figure 14). En cas d'égalité, choisir le modèle le plus ***parcimonieux*** car il aura naturellement moins de variance d'estimation dans les coefficients du modèle.

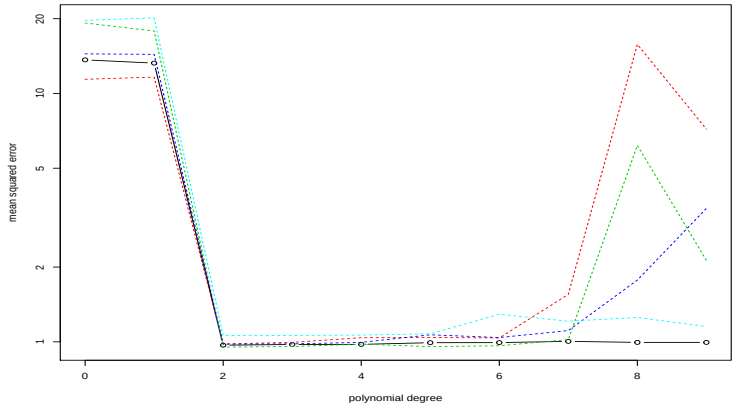


FIGURE 14 – Erreur de test calculée par validation croisée à 5 folds. Les différentes courbes correspondent à des partitions aléatoires en 5 folds.

Quatrième partie

Méthodes de régularisation pour la sélection de variables

7 Motivation

La sélection des variables qui expliquent la variable réponse y est une question cruciale en régression linéaire multiple et en classification. La première raison est numérique, il se trouve que la matrice $(\mathbf{X}'\mathbf{X})$ qui intervient dans la solution des moindres carrés 3 devient non-inversible dès que le nombre de variables explicatives p est supérieur aux nombre d'individus n. Il est donc indispensable de désigner un sous ensemble de variables explicatives liées à y et écarter les variables explicatives qui n'apportent aucune information explicative sur y. Certaines méthodes permettent de choisir le sous-ensemble de variables explicatives à conserver dans le modèle de régression. Ces méthodes reposent sur une recherche dite **pas à pas** ou *stepwise en anglais* pour trouver le modèle qui maximise critère d'information qui réalise un compromis entre l'ajustement aux données et la complexité du modèle. Les critères d'information couramment utilisés sont C_p **de Mallows**, les critères **AIC** et **BIC**. On peut consulter la section 6.1 du livre [2] pour les définitions des critères précédents ainsi les détails des procédures stepwise.
On parle de recherche **forward** en partant du modèle vide comme modèle d'initialisation. On inclut à chaque itération la variable dont l'inclusion réalise la meilleure amélioration du critère d'information. On s'arrête lorsqu'aucune inclusion ne permet d'améliorer le critère ou le modèle complet est atteint. On parle de recherche **backward** en partant du modèle complet incluant toutes les variables comme modèle d'initialisation. On exclut à chaque itération la variable dont l'exclusion du modèle réalise la meilleure amélioration du critère d'information. On s'arrête lorsqu'aucune inclusion n'améliore le critère ou si le modèle vide est atteint. Nous faisons le choix délibéré de ne pas étudier ces méthodes dans ces notes parce que d'une part, un grand nombre de variables explicatives avec une forte corrélation rend impossible la mise en place des procédures de sélection de variables par stepwise en direction **forward** ou **backward**. Un autre problème est du à la *discontinuité* de la sélection de variables par stepwise, c'est à dire une modification infiniment petite des données impacte considérablement le sous-ensemble de variables sélectionnées. On parle aussi d'instabilité de la sélection de variables par stepwise en grande dimension.

8 Régularisation ridge

Le début de réponse à la problématique de sélection de variables en régression en grande dimension se trouve dans les méthodes de régularisation. Cette réponse repose sur les remarques suivantes :

- Si les β_j ne sont pas contraints, ils peuvent prendre de très grandes valeurs et donc avoir une grande variance.
- Pour contrôler la variance, il faut contrôler la taille des coefficients de $\boldsymbol{\beta}$. Cette approche pourrait réduire sensiblement l'erreur de prédiction.

Rappelons que la solution des moindres carrés est obtenue en minimisant

$$\mathrm{RSS}(\boldsymbol{\beta}) = \sum_{i=1}^{n} \hat{e}_i^2 = \sum_{i=1}^{n} \left(y_i - \hat{y}_i\right)^2 = \sum_{i=1}^{n} \left(y_i - \beta_0 - \beta_1 x_{i1} - \beta_2 x_{i2} - \cdots - \beta_p x_{ip}\right)^2.$$

Les coefficients du modèle de régression appelée **ridge** sont obtenus en minimisant le critère suivant

$$\frac{\text{RSS}(\boldsymbol{\beta})}{2} + \lambda \sum_{j=1}^{p} \beta_j^2. \tag{4}$$

À l'instar des moindres carrés, les coefficients de régression ridge ont une expression matricielle

$$\widehat{\boldsymbol{\beta}}_{\lambda}^{\text{ridge}} = \left(\mathbf{X}'\mathbf{X} + \lambda \mathbf{I}_p\right)^{-1} \mathbf{X}' y = \mathbf{H}_{\lambda} y. \tag{5}$$

— Contrairement aux moindres carrés, une solution unique existe toujours si $\lambda > 0$. Dans le cas des moindres carrés $\mathbf{X}'\mathbf{X}$ n'est inversible que si $p < n$.
— Le terme $\lambda \sum_{j=1}^{p} \beta_j^2$ dans l'équation 4 est appelé **terme de pénalité** (ou **terme de régularisation**) où λ est le paramètre de régularisation qui sert à contrôler le compromis (pénalité/ajustement). Ce terme contrôle la taille du vecteur de coefficients $\boldsymbol{\beta}$. La constante β_0 n'est pas concernée par le terme de pénalité dans l'expression 4.
— Il est important de sélectionner la valeur de λ qui réalise une erreur de prédiction minimale.

La fonction lm.ridge du package MASS permet d'ajuster des modèles de régression linéaire ridge pour un ensemble de valeurs de λ à l'aide d'une validation croisée généralisée où chaque individu est considéré comme un **fold**. La valeur de λ qui minimisera l'erreur de prédiction obtenue par validation croisée correspond au modèle retenu. Le bloc de code suivant permet de mettre en place cette procédure sur le jeu de données Prostate du package lasso2

```
rm(list=ls())
require(lasso2)
require(MASS)
data("Prostate")
lam <- seq(0.1,200,len=99)
fit <- lm.ridge(lpsa~., Prostate, lambda=lam)
matplot(fit$lambda, coef(fit)[,-1], col=1:8, type="l", lty=1, lwd=3,
        ylab=~beta, xlab=~lambda, log="x")
legend("topleft",lty=1, lwd=3, col = 1:8, legend = colnames(Prostate[,-9]))
```

8.1 Quelques commentaires

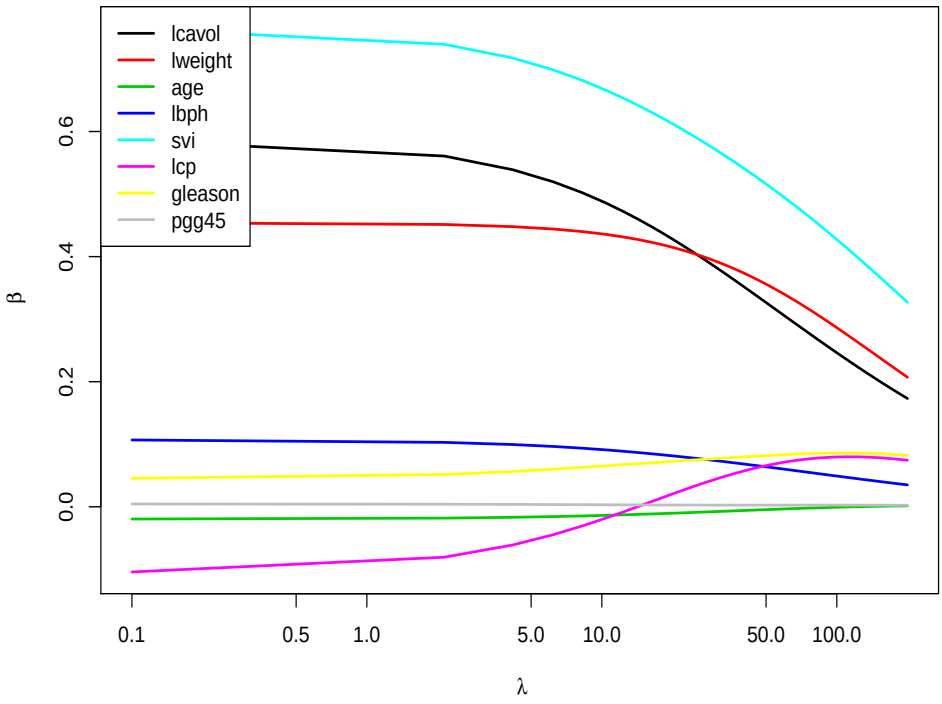


FIGURE 15 – Cette figure est le résultat des deux lignes du bloc de code illustrant la régression ridge sur le jeu de données du cancer de la prostate. Chaque courbe représente la trajectoire du coefficient de régression d'une variable en fonction de la valeur de la pénalité λ.

Détaillons l'exemple illustrant la régression ridge sur les données du cancer de la prostate. La grille lam de valeurs de λ est une séquence uniforme de taille 99 allant de 10^{-1} à 200. Ces valeurs sont choisies à la main pour que la grille puisse contenir des valeurs proches de 0 et des valeurs relativement élevées. Lorsque λ est proche de 0 le coefficient de régression ridge coïncide avec la solution donnée par les moindres carrés dans une régression linéaire multiple sans terme de pénalité. Lorsque λ prend de grandes valeurs, le coefficient de régression ridge est ramené à 0.

Régression ridge : un peu de pratique

— Extraire et stocker dans un objet betabestridge les coefficients de régression correspondants à λ^* qui réalise l'erreur de prédiction minimale obtenue par validation croisée. Commencer par l'identification des noms et des dimensions des objets des champs contenus dans l'objet fit.
— Quelle est la valeur de λ^* ?
— Peut-on identifier dans betabestridge un sous ensemble de variables qui participent à l'explication de y ?

— Peut-on sélectionner des variables à l'aide la régression ridge ?

9 Régularisation lasso

Comme nous l'avons vu précédemment, la régression ridge permet d'ajuster un modèle linéaire multiple pour des cas limites où $n \approx p$, $n < p$ et en présence de corrélation entre les variables explicatives. Seul inconvénient, la régression ridge ne permet pas une sélection de variables et ne parvient pas à fournir un modèle parcimonieux avec peu de paramètres. La solution à ce problème consiste à considérer un estimateur différent des coefficients de régression qui minimise la quantité

$$\frac{\mathbf{RSS}(\boldsymbol{\beta})}{2} + \lambda \sum_{j=1}^{p} |\beta_j| \tag{6}$$

où l'unique différence avec la régression ridge 4 est dans la présence d'une valeur absolue au lieu d'un carré dans le terme de pénalité. Cette légère modification du terme de pénalité est la clé du problème de sélection de variables. La présence de ce terme de pénalité a un impact similaire à la régression ridge sur les coefficients de régression. Cependant, contrairement à la régression ridge, certains coefficients sont mis à zéro ; de telles solutions, avec plusieurs coefficients qui sont identiquement nulles, sont dites **sparse**. La pénalité effectue ainsi une sorte de sélection de variables continue. L'estimateur $\widehat{\boldsymbol{\beta}}_{\lambda}^{\text{lasso}}$ obtenu en minimisant l'expression 6 est appelé **lasso** pour *Least Absolute Shrinkage and Selection Operator*. Le bloc de code suivant permet de mettre en place une procédure de sélection de modèle par lasso avec un choix de la pénalité λ^* optimale par validation croisée [17]. Cette procédure permet de calculer une collection de solutions du problème minimisation 6 où chaque solution correspond à un élément de la grille de valeurs de λ donnée en entrée à l'aide de l'argument lambda de la fonction cv.glmnet du package glmnet.

```
require(glmnet)
lam <- exp(seq(log(1.5),log(0.001),len=100))
y = Prostate[,9]
x = as.matrix(Prostate[,-9])
cvfit <- cv.glmnet(x, y, family="gaussian", lambda = lam)
plot(cvfit$glmnet.fit, "lambda", col=1:8, lwd=2,ylab=~beta)
legend("topright",lty=1, lwd=2, col = 1:8, legend = colnames(Prostate[,-9]))
```

17. Attention le calcul peut prendre quelques minutes si les variables explicatives se comptent par milliers !

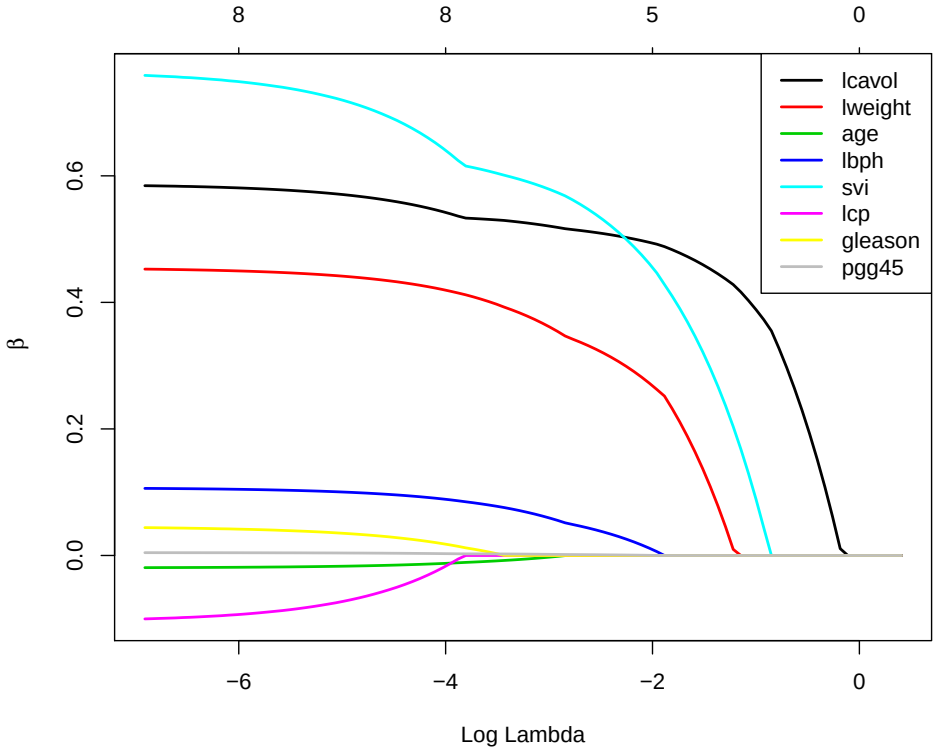


FIGURE 16 – Cette figure est le résultat des deux lignes du bloc de code illustrant la régression lasso sur le jeu de données du cancer de la prostate. Chaque courbe représente la trajectoire du coefficient de régression d'une variable en fonction de la valeur du logarithme de la pénalité λ.

Régression lasso : un peu de pratique

— Décrire les éléments cvm, lambda.min contenus dans l'objet cvfit et les élément beta et df contenus dans le champ glmnet.fit de l'objet cvfit.
— Quel est le nombre de folds retenu pour la procédure de validation croisée ? Tracer l'erreur de validation croisée en fonction de la valeur de λ ? Quelle est la valeur de λ^* qui minimise l'erreur de validation croisée ?
— Quel est le nombre de coefficients non-nuls dans le modèle de régression lasso correspondant à λ^* ?

10 Régularisation elasticnet

On peut faire appel aux deux types de pénalités précédentes en minimisant la quantité

$$\frac{\mathbf{RSS}(\boldsymbol{\beta})}{2} + \lambda\Big(\alpha \sum_{j=1}^{p} |\beta_j| + (1-\alpha) \sum_{j=1}^{p} \beta_j^2\Big) \tag{7}$$

où $\alpha \in]0,1[$ est un second paramètre de régularisation qui permet de répartir les poids des deux pénalités. Le vecteur de coefficients de régression $\widehat{\boldsymbol{\beta}}_\lambda^{\text{net}}$ qui minimise 7 correspond à la régression dite **elasticnet**. Pour faire de la sélection de modèle dans ce cadre, il faut sélectionner le couple (λ^*, α^*) qui minimise l'erreur de prédiction par validation croisée. Pour mettre en place une procédure de sélection de variable par elasticnet, on procède comme suit : Ajouter la valeur de l'argument alpha à la fonction cv.glmnet pour sélectionner le paramètre λ optimale à α fixé. Ensuite, retenir le couple (λ^*, α^*) qui réalise la meilleur erreur de prédiction par validation croisée par les erreurs minimales réalisées.

Régression elasticnet : jeu de données prostate

Reprendre le bloc de code précédent et sélectionner le meilleur modèle par elasticnet ainsi que le couple de valeurs (λ^*, α^*) optimales. Tester les valeurs : $0.1, 0.2, 0.3, \ldots, 0.9$ pour α.

Cinquième partie

Régression logistique

Certains algorithmes de régression peuvent être utilisés pour la classification (et inversement). La **régression logistique** (appelée également régression *logit*) est utilisée couramment pour estimer la probabilité qu'une observation appartienne à une classe particulière (par exemple la probabilité qu'un email soit un spam ?). Si la probabilité estimée est supérieure à 50%, alors le modèle prédit que l'observation appartient à cette classe (appelée la classe positive, l'étiquette « 1 »), sinon il prédit qu'elle appartient à l'autre classe (la classe négative, d'étiquette « 0 »). C'est en fait une règle de classification.

11 Estimation des probabilités

Comment cela fonctionne-t-il ? Tout comme un modèle de régression linéaire, un modèle de régression logistique calcule une somme pondérée $\beta_0 + \sum_{j=1}^{p} \beta_j x_j$ des caractéristiques d'entrée (somme pondérée des variables explicatives + un terme constant). Au lieu de fournir la prédiction $\hat{y}$ à partir de cette somme pondérée, la régression logistique fournit la *logistique* du résultat :

$$\hat{p} = h_{\boldsymbol{\beta}}(\mathbf{x}) = \sigma\Big(\beta_0 + \sum_{j=1}^{p} \beta_j x_j\Big), \tag{8}$$

où $\mathbf{x} = (x_1, \ldots, x_p)$. La fonction logistique (appelé également *logit* et notée $\sigma()$) est une fonction *sigmoïde* (c'est-à-dire en forme « S ») qui renvoie des valeurs comprises entre 0 et 1. Cette fonction est définie par

$$\sigma(t) = \frac{1}{1 + \exp(-t)}. \tag{9}$$

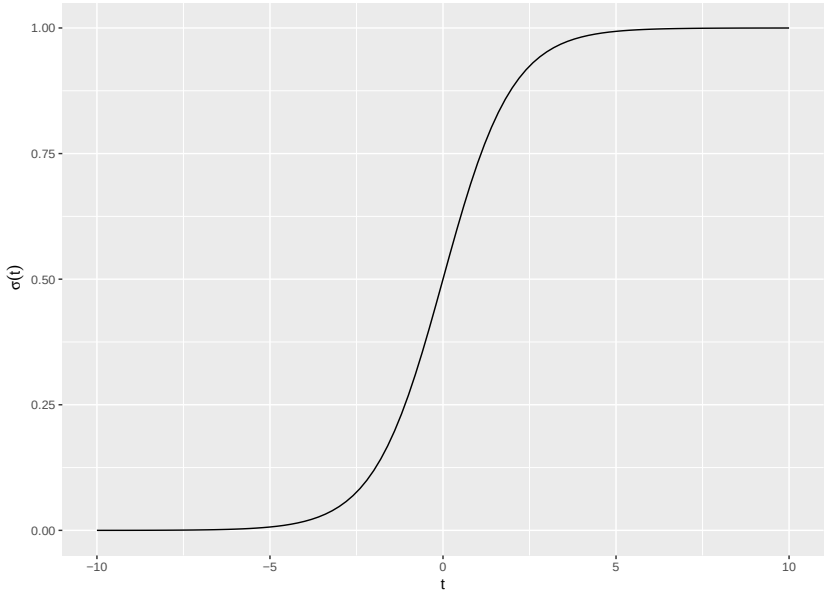


FIGURE 17 – La fonction logistique $\sigma(t)$. Cette fonction permet d'obtenir la probabilité que $y = 1$ à partir de la somme pondérée $\beta_0 + \sum_{j=1}^{p} \beta_j x_j$.

Une fois que le modèle de régression logistique a estimé la probabilité $\hat{p} = h_{\boldsymbol{\beta}}(\mathbf{x})$ qu'une observation $\mathbf{x}$ appartienne à la classe positive, il peut alors faire aisément sa prédiction $\hat{y}$ donné par

$$\hat{y} = \begin{cases} 0 & \text{si} \quad \hat{p} < 0.5. \\ 1 & \text{si} \quad \hat{p} \geq 0.5. \end{cases}$$

Remarquons que $\sigma(t) < 0.5$ lorsque $t < 0$, et $\sigma(t) \geq 0.5$ lorsque $t \geq 0$, c'est pourquoi un modèle de régression logistique prédit 1 si $\beta_0 + \sum_{j=1}^{p} \beta_j x_j$ est positif, ou 0 s'il est négatif.

11.1 Entraînement et fonction de coût

Les estimateurs des coefficients de régression linéaire proviennent d'une minimisation de l'erreur des moindres carrés. L'erreur des moindres carrés $\mathrm{RSS}(\boldsymbol{\beta})$ est aussi appelée **fonction coût**. Revenons maintenant à la régression logistique. Nous savons comment un modèle de régression logistique estime les probabilités et effectue ses prédictions. Mais comment est-il entraîné ? Autrement dit, quel est le mécanisme d'estimation des coefficients de régression $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_p)$ dans le cas d'une régression logistique ?
L'objectif de l'entraînement consiste à définir le vecteur des paramètres $\boldsymbol{\beta}$ afin que le modèle estime des probabilités élevées pour les observations positives ($y = 1$) et des probabilités basses pour des observations négatives ($y = 0$). La fonction de coût suivante traduit cette idée dans le cas d'une unique observation d'apprentissage $\mathbf{x}$

$$c(\boldsymbol{\beta}) = \begin{cases} -\log\ (\hat{p}) & \text{si} \quad y = 1, \\ -\log\ (1 - \hat{p}) & \text{si} \quad y = 0. \end{cases}$$

Cette fonction de coût fonctionne parce que $-\log(t)$ devient très grand lorsque t s'approche de 0, par conséquent le coût sera grand si le modèle estime une probabilité proche de 0 pour une

observation positive ($y = 1$). Le coût sera également très élevé si le modèle estime une probabilité proche de 1 pour une observation négative ($y = 0$). Par ailleurs, $\log(t)$ est proche de 0 lorsque t est proche de 1, et par conséquent, le coût sera proche de 0 si la probabilité estimée est proche de 0 pour une observation négative ou proche de 1 pour une observation positive, ce qui est précisément ce que nous voulons.
La fonction de coût sur l'ensemble du jeu de données d'apprentissage (ou d'entraînement) est simplement le coût moyen sur l'ensemble de ses observations. Comme vous pouvez le vérifier aisément, elle peut s'écrire sous forme d'une simple expression, nommée **perte logistique** (*log loss* en anglais). La fonction de coût de l'ensemble du jeu de données d'apprentissage $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$ est donnée par

$$J(\boldsymbol{\beta}) = -\frac{1}{n}\sum_{i=1}^{n}\Big[y_i \log\big(\hat{p}_i\big) + (1 - y_i)\log\big(1 - \hat{p}_i\big)\Big], \tag{10}$$

où $\hat{p}_i = h_{\boldsymbol{\beta}}(\mathbf{x}_i)$. La mauvaise nouvelle, c'est qu'il n'existe pas de solution analytique connue pour calculer la valeur de $\boldsymbol{\beta}$ qui minimise la fonction de coût 10 comme dans le cas d'une régression linéaire multiple. Mais la bonne nouvelle, c'est que cette fonction de coût est convexe, autrement dit, elle possède les bonnes propriétés mathématiques qui permettent à un algorithme de trouver assurément un minimum global. Cette procédure peut faire l'objet d'une section de ces notes où on pourra détailler l'intuition de l'algorithme d'optimisation appelé **descente du gradient**.
Le bloc de code suivant illustre une régression logistique sur le jeu de données Breast Cancer du package mlbench.

```
# install.packages("mlbench")
data(BreastCancer, package="mlbench")
bc <- BreastCancer[complete.cases(BreastCancer), ]  # create copy
str(bc)
# remove id column
bc <- bc[,-1]
# convert factors to numeric
for(i in 1:9) {
  bc[, i] <- as.numeric(as.character(bc[, i]))
}
bc$Class <- ifelse(bc$Class == "malignant", 1, 0)
bc$Class <- factor(bc$Class, levels = c(0, 1))
mod = glm(Class~., data = bc, family="binomial")
summary(mod)
```

12 Régularisation en régression logistique

La partie IV introduit les méthodes de régularisation *ridge, lasso et elasticnet* pour la régression linéaire multiple. Rappelons que ces méthodes répondent à la problématique de sélection de variables. Ces trois méthodes de régularisation sont parfaitement adaptées au contexte de la régression logistique. Les coefficients de régression d'une régression logistique régularisée sont obtenus en minimisant la fonction de coût pénalisée

$$J(\boldsymbol{\beta}) + \lambda\Big(\alpha\|\boldsymbol{\beta}\|_1 + (1 - \alpha)\|\boldsymbol{\beta}\|_2\Big)$$

où

— $J(\boldsymbol{\beta})$ est la fonction de coût donnée par l'équation 10 qui mesure l'erreur du modèle de régression logistique.
— La partie

$$\lambda\Big(\alpha\|\boldsymbol{\beta}\|_1 + (1-\alpha)\|\boldsymbol{\beta}\|_2\Big)$$

est appelée (le terme de régularisation ou de pénalité).
— Les termes $\|\boldsymbol{\beta}\|_1$ et $\|\boldsymbol{\beta}\|_2$ correspondent à

$$\|\boldsymbol{\beta}\|_1 = \sum_{j=1}^{p} |\beta_j| \quad \text{et} \quad \|\boldsymbol{\beta}\|_2 = \sum_{j=1}^{p} \beta_j^2.$$

— Lorsque $\alpha = 1$, le terme de pénalité correspond à la pénalité lasso.
— Lorsque $\alpha = 0$, le terme de pénalité correspond à la pénalité ridge.

Sélection de variables en régression logistique : jeu de données Breast Cancer

Reprendre le bloc de code précédent et sélectionner le meilleur modèle par elasticnet ainsi que le couple de valeurs (λ^*, α^*) optimales. Tester les valeurs : $0.0, 0.1, 0.2, \ldots, 0.9, 1.0$ pour α. Adapter l'utilisation de la fonction cv.glmnet au contexte de la régression logistique.

Sixième partie

Projet : Utilisation de la libraire caret

Introduction

Au delà de l'objectif d'évaluation, ce projet introduit une librairie qui pourra être d'une grande utilité pour calibrer un grand nombre des méthodes qui seront abordées dans ce cours. On entend par calibration d'un modèle, le choix des hyper-paramètres qui contrôlent la complexité du modèle. L'objectif de ce projet est l'implémentation d'un modèle de régression logistique pour prédire si un patient est atteint ou pas d'un cancer de la prostate à partir de l'observation de 12600 expressions de gènes.

Jeu de données

Le jeu de données provient de l'article [3] et disponible dans le package SIS. Commençons par la lecture du jeu de données par un bloc de code à reprendre en entier pour obtenir la même partition jeux de données apprentissage et test pour l'ensemble des étudiants. *Attention : il est indispensable de conserver l'instruction set.seed(123) qui fixe l'aléa avant le tirage de l'appartenance apprentissage ou test de chaque observation du jeu de données.*

```
rm(list=ls())
require(SIS)
data("prostate.train")
data("prostate.test")
y = c(prostate.train[, 12601], prostate.test[, 12601])
x = rbind(as.matrix(prostate.train[, -12601]),
```

```
          as.matrix(prostate.test[, -12601]))
set.seed(123)
train  = sample(nrow(x), nrow(x)*0.66)
xtrain = x[train,]
xtest = x[-train,]
ytrain  =  factor(y[train])
ytest = factor(y[-train])
```

Choix de α et λ

Mettre en place une procédure de validation croisée pour sélectionner les valeurs optimales de α et λ sur le jeu de données d'apprentissage xtrain et ytrain. *Les paramètres de la validation croisée sont (code à fournir pour chaque étape)*

- La grille de valeurs de α à tester est $0.0, 0.1, \ldots, 0.9, 1.0$.
- Nous n'avons pas besoin d'une grille de valeurs de λ. En cas d'absence d'une grille de valeurs de λ à tester en entrée, les fonctions cv.glmnet et glmnet sont implémentées pour proposer une grille à tester.
- Le nombre de folds à utiliser est de 3. Lire attentivement l'aide de la fonction cv.glmnet pour spécifier les options nécessaires pour une régression logistique avec régularisation.

Exploitation des sorties

Préciser (fournir) le code correspondant à chaque réponse aux questions suivantes.

1. Quelle est la valeur optimale α^*.
2. Le modèle correspondant à α^* fournit deux valeurs de λ données par les champs lambda.min et lambda.1se disponibles dans l'objet renvoyé par la fonction cv.glmnet. Comparer les deux modèles issus des deux valeurs lambda.min et lambda.1se en terme d'erreur de prédiction du jeu de données test xtest et ytest.
3. Expliquer la différence entre lambda.min et lambda.1se.
4. Afficher le nombre de variables (expressions de gènes) sélectionnées par les modèles correspondants à lambda.min et à lambda.1se respectivement. Conclure.

Utilisation du package caret

Si on exécute le code précédent en entier plusieurs fois de suite, on peut constater l'instabilité du modèle final retenu en terme de nombre de variables explicatives sélectionnées et son erreur de prédiction sur le jeu de données test. On peut observer deux situations d'instabilité. Première situation : les modèles sélectionnés (meilleurs modèles obtenus par procédures de validation-croisée successives) sont égaux en terme d'erreur de test et différents en terme du nombre de variables explicatives sélectionnées. Seconde situation : les modèles sélectionnés sont différents en terme d'erreur de test. La première situation témoigne de l'absence d'unicité du modèle optimal en terme d'erreur de test. En d'autres termes, plusieurs modèles réalisent la même erreur de test optimale. Dans ce cas de figure, on se contente de l'un des modèles optimaux observés. La seconde situation témoigne d'une sous-optimalité du modèle sélectionné et nécessite de stabiliser la solution en *secouant* suffisamment les données pour capter le modèle optimal. On soupçonne qu'une seule procédure de validation croisée et plus précisément une seule partition aléatoire

du jeu de données en folds ne suffit pas pour que le modèle optimal soit dans la collection de modèles testés par la procédure. Pour augmenter la taille de la collection de modèles comparés (mis en compétition) par la procédure de validation croisée, la librairie caret permet de faire une multitude de validation croisée en faisant plusieurs partitions aléatoires du jeu de données en folds. L'erreur *"cross-validée"* associée à un couple de paramètres de contrôle de la complexité du modèle (α, λ) est la moyenne des erreurs de validations croisées issues des différentes partitions aléatoires répétées par la procédure. Répéter un grand nombre de fois une procédure de validation croisée peut rendre les temps de calculs trop lents. Afin de réduire le temps de calcul de telles procédures, on peut exploiter les *unités processeurs*[18] dont on peut disposer[19] et pour connaître si notre ordinateur dispose de plus d'une seule unité processeur, il suffit d'utiliser la commande detectCores() de la librairie parallel. Pour utiliser par exemple 3 unités processeurs[20] dans un calcul, il suffit de placer la commande registerDoMC(3) avant les appels aux fonctions de la librairie caret. L'utilisation de la fonction registerDoMC nécessite le chargement de la librairie doMC. Après cette parenthèse sur l'utilisation de plusieurs unités processeurs pour paralléliser les calculs, l'objectif de cette partie du projet est l'utilisation de la librairie caret pour mettre en place 10 répétitions d'une validation-croisée à 3 folds. Quelques remarques utiles pour lancer la procédure

- La fonction trainControl de la librairie caret permet de déclarer la validation croisée à 3 folds répétée 10 fois.
- La fonction expand.grid permet de déclarer la grille de valeurs du couple de paramètres (α, λ). Dans ce projet, nous allons tester les valeurs $0.0, 0.1, \ldots, 1.0$ pour le paramètre alpha et les 100 valeurs fixées par défaut pour le paramètre lambda. Pour récupérer ces 100 valeurs, on peut faire appel à la fonction glmnet[21] pour ajuster une multitude de modèles de régression logistique pénalisée avec une grille par défaut. L'objet renvoyé par cette fonction contient un champ qui correspond à la grille de valeurs par défaut de lambda.
- La fonction train de la librairie caret permet de calibrer une méthode d'apprentissage (ou modèle) en spécifiant une stratégie de calibration déclarée à l'aide de la fonction trainControl[22] et une grille de valeurs déclarées à l'aide de la fonction expand.grid. Dans ce projet, le modèle d'apprentissage correspond à glmnet. Pour que la fonction train puisse fonctionner, il faut que les noms des paramètres déclarés dans la grille de valeurs à tester correspondent aux noms des paramètres dans la méthode glmnet.
- La métrique ou la mesure d'efficacité correspond à Accuracy (taux de bon classement) car la fonction train propose de sélectionné le modèle qui maximise le taux de bon classement. Cela revient à sélectionner le modèle qui minimise l'erreur de classement.
- Il est évident, qu'il faut charger l'ensemble des librairies nécessaires pour utiliser les fonctions appelées dans le code.

Passons à la pratique

5. À l'aide du package caret, mettre en place une procédure de validation-croisée à 3 folds répétée 10 fois avec la grille de paramètres spécifiée ci-dessus. Afficher les valeurs optimales des deux paramètres.

6. Quelle est l'erreur de test du modèle sélectionné.

7. Utiliser la fonction glmnet avec les valeurs optimales des paramètres pour afficher le nombre

18. *cores* en anglais.
19. souvent nos ordinateurs actuels disposent de 2, 4 et pour les plus chanceux 8 cores.
20. Il faut s'assurer que notre ordinateur possède au moins 3 unités processeurs.
21. La fonction glmnet de la librairie du même nom propose une grille par défaut pour lambda et alpha est fixé par défaut à 1. Contrairement à cv.glmnet, elle ne fait pas de validation croisée, elle se contente d'ajuster une suite de modèle lasso correspondants à la suite par défaut de valeurs de lambda sur l'ensemble du jeu de données d'apprentissage.
22. La fonction trainControl appartient au la librairie caret.

de variables sélectionnées par le modèle sélectionné.

Références

[1] Trevor Hastie, Robert Tibshirani et Jerome Friedman. *The elements of statistical learning : data mining, inference and prediction.* 2e éd. Springer, 2009. URL : `http://www-stat.stanford.edu/~tibs/ElemStatLearn/`.

[2] Gareth James et al. *An Introduction to Statistical Learning – with Applications in R.* T. 103. Springer Texts in Statistics. New York : Springer, 2013.

[3] Dinesh Singh et al. "Gene expression correlates of clinical prostate cancer behavior". In : *Cancer Cell* 1.2 (2002), p. 203 -209. ISSN : 1535-6108.