TD3 : Régression en grande dimension

13/03/2025

Dans ce TD, on explore trois techniques de re-échantillonnage pour l'estimation de l'erreur théorique de prévision d'un modèle de régression. On utilise le jeu de données `Auto`, du package `ISLR`, qui est constituée de $n = 392$ observations et 9 variables. Dans la suite nous allons considérer trois modèles de régression, pour expliquer la variable `mpg` en fonction de la variable `horsepower`, et nous évaluons l'erreur théorique de prévision de chacun des modèles.

La méthode de l'ensemble de validation (Apprentissage/Validation)

Nous commençons par utiliser la fonction `sample()` pour diviser la base de données `Auto` en deux parties, en tirant de manière aléatoire sans remise $(2/3) * n$ observations (qui vont servir pour l'apprentissage) parmi les $n = 392$ observations de la base de données. Les observations restantes vont être utilisées pour évaluer l'erreur théorique de prévision. L'ensemble d'apprentissage sera noté dans la suite `ensemble_apprentissage` et l'ensemble de validation `ensemble_validation`.

```r
library(ISLR)
str(Auto)
```

```
## 'data.frame':    392 obs. of  9 variables:
##  $ mpg         : num  18 15 18 16 17 15 14 14 14 15 ...
##  $ cylinders   : num  8 8 8 8 8 8 8 8 8 8 ...
##  $ displacement: num  307 350 318 304 302 429 454 440 455 390 ...
##  $ horsepower  : num  130 165 150 150 140 198 220 215 225 190 ...
##  $ weight      : num  3504 3693 3436 3433 3449 ...
##  $ acceleration: num  12 11.5 11 12 10.5 10 9 8.5 10 8.5 ...
##  $ year        : num  70 70 70 70 70 70 70 70 70 70 ...
##  $ origin      : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ name        : Factor w/ 304 levels "amc ambassador brougham",..: 49 36 231 14 161 141 54 223 241
```

```r
n <- nrow(Auto) #le nombre d'observations
indices <- sample(x = n, size = trunc((2/3)*n), replace = FALSE)
#trunc() pour prendre la partie entière
ensemble_apprentissage <- Auto[indices, ]
ensemble_validation <- Auto[- indices, ]
```

Questions

1 - Afficher la structure de ces deux ensembles. Commenter;

2 - Ajuster un modèle de régression linéaire simple pour expliquer la variable `mpg` en fonction de la variable `horsepower` en utilisant l'ensemble d'apprentissage (nommer ce modèle `modele1`);

3 - Donner les valeurs prédites de la variable `mpg` de l'ensemble de validation. Stocker les résultats dans un objet R que vous appellerez `valeurs_predites`;

4 - Donner une estimation de l'erreur théorique de prévision du modèle `modele1` en utilisant la méthode de l'ensemble de validation. Stocker le résultat dans `estimation_erreur_modele1`;

5 - Relancer le code précédent plusieurs fois et comparer les résultats des estimations de l'erreur que vous obtenez. Expliquer la variation de ces estimations.

6 - Ajuster un modèle polynomial de degré 2 (`modele2`) et un modèle polynomial de degré 3 (`modele3`) pour expliquer `mpg` en fonction de `horsepower`, en utilisant l'ensemble d'apprentissage;

7 - Donner une estimation de l'erreur de prévision de chacun des modèles `modele2` et `modele3` en utilisant la méthode de l'ensemble de validation. Stocker les résulats dans `estimation_erreur_modele2` et `estimation_erreur_modele3`.

8 - Parmi les trois modèles, lequel choisiriez-vous? Justifier votre choix.

La méthode leave-one-out Cross-Validation (LOOCV)

L'estimation de l'erreur d'un modèle linéaire (généralisé) par la méthode LOOCV peut étre calculée automatiquement à l'aide de la fonction `cv.glm()`, du package `boot`, si le modèle linéaire est construit à l'aide de la fonction `glm()` (au lieu de la fonction `lm()`)

Questions

1 - Ajuster un modèle de régression linéaire simple `modele1` pour expliquer la variable `mpg` en fonction de la variable `horsepower`, en utilisant toute la base `Auto`, à l'aide de la fonction `glm()`;

2 - Donner une estimation de l'erreur de prévision du modèle `modele1` par la méthode LOOCV, à l'aide de la fonction `cv.glm()`;

3 - Reprendre la question 2 et comparer les deux résultats? Expliquer;

4 - Ajuster d'autres modèles, un polynomial de degré 2 `modele2` et un polynomial de degré 3 `modele3`, à l'aide de la fonction `glm()`;

5 - Donner une estimation de l'erreur de chacun des modèles précédents par la méthode LOOCV, à l'aide de la fonction `cv.glm()`;

6 - Quel modèle choisiriez-vous? Jusitifer;

La méthode K-fold Cross-Validation (K-fold CV)

L'estimation de l'erreur de prédiction peut ête également obtenue par cette méthode à l'aide de la fonction `cv.glm()` du package `boot`. Nous prenoms pour la suite $K = 10$.

Questions

1 - Evaluer l'erreur de chacun des trois modèles précédents, par la méthode K-fold CV, à l'aide de la fonction `cv.glm()`;

2 - Relancer le code R de la question précédente, et comparer les résulats. Expliquer;

3 - Quel modèle choisiriez-vous? Justifier.

Le bootstrap

Dans ce paragraphe, on présente comment utiliser sous R la technique du boostrap pour estimer la variance et le biais d'un estimateur du paramètre d'un modèle donné. Cela peut se faire automatiquement à l'aide de la fonction `boot()` du package `boot`. À titre d'exemple, nous allons considérer la base de données `Auto` précédente, et le modèle de régression linéaire simple de la variable $y =$`mpg` en fonction du régresseur $x =$`horsepower`. Rappelons que le modèle s'écrit $y = w_0 + w_1 x + \varepsilon$. Nous considérons les estimateurs classiques des moindres-carrés $\widehat{w}_0$ et $\widehat{w}_1$ (donnés par la fonction `lm()` ou `glm()`), et nous allons estimer leur variances et biais par le bootstrap. Rappellons que la fonction `lm()` donne des estimations, $\widehat{sd}(\widehat{w}_0)$ et $\widehat{sd}(\widehat{w}_1)$, des écarts-types (donc des variances), conditionnels, de chacun de ces estimateurs, et qui sont basées sur les formules suivantes

$$Var(\widehat{w}_0) = \sigma^2 \left[\frac{1}{n} + \frac{\overline{x}^2}{\sum_{i=1}^n (x_i - \overline{x})^2}\right], \quad Var(\widehat{w}_1)^2 = \frac{\sigma^2}{\sum_{i=1}^n (x_i - \overline{x})^2},$$

où $\sigma^2 := \mathrm{Var}(\varepsilon)$. Ces formules sont valables sous l'hypothèse de non-corrélation des résidus ε_i et le fait que $\mathrm{Var}(\varepsilon_i) = \sigma^2, \forall i = 1, \ldots, n$ (hypothèse d'homoscédasticité). Les estimateurs fournis par la fonction `lm()` ou `glm()` sont calculés à partir des formules précédentes en estimant σ^2 par $\widehat{\sigma}^2 := \frac{1}{n-2}\sum_{i=1}^n (\varepsilon_i - \overline{\varepsilon})^2$, i.e.,

$$\widehat{sd}(\widehat{w}_0) := \sqrt{\widehat{\sigma}^2 \left[\frac{1}{n} + \frac{\overline{x}}{\sum_{i=1}^n (x_i - \overline{x})^2}\right]}, \quad \widehat{sd}(\widehat{w}_1) := \sqrt{\frac{\widehat{\sigma}^2}{\sum_{i=1}^n (x_i - \overline{x})^2}}.$$

Pour calculer les estimateurs des variances et biais, de $\widehat{w}_0$ et $\widehat{w}_1$, par bootstrap, et qui s'affranchit des hypothèses de non corrélation et d'homoscédasticité, nous allons d'abord définir une fonction qui donne les estimateurs $\widehat{w}_0$ et $\widehat{w}_1$, et qui aura comme entrées une base de données `data` et un vecteur d'indices `index` qui permet d'extraire les observations à utiliser pour la construction de ces estimateurs

```
f_estimateurs_w <- function(data, index){return(coef(lm(formula = mpg ~ horsepower,
                                                       data = data, subset = index)))}
```

```
f_estimateurs_w(data = Auto, index = 1:392)
```

```
## (Intercept)  horsepower
##  39.9358610  -0.1578447
```

Cette fonction permet de calculer des estimateurs bootstrapés par échantillonnage aléatoire avec remise dans la base des données. Ici deux exemples

```
f_estimateurs_w(data = Auto, index = sample(x = 392, size = 392, replace = T))
```

```
## (Intercept)  horsepower
##  40.9987637  -0.1678964
```

```
f_estimateurs_w(data = Auto, index = sample(x = 392, size = 392, replace = T))
```

```
## (Intercept)  horsepower
##  41.0862479  -0.1644868
```

Maintenant, nous allons donner les estimateurs bootstrap de l'écart-type et du biais de chacun des estimateurs $\widehat{w}_0$ et $\widehat{w}_1$, à l'aide de la fonction `boot()`, sur la base de $R = 1000$ échantillons bootstrapés

```
library(boot)
boot(data = Auto, statistic = f_estimateurs_w, R = 1000)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Auto, statistic = f_estimateurs_w, R = 1000)
##
##
## Bootstrap Statistics :
##       original        bias    std. error
## t1* 39.9358610  0.0971801945 0.858037357
## t2* -0.1578447 -0.0008596443 0.007548649
```

Nous pouvons comparer avec les estimations données par `lm()`

```r
summary(lm(formula = mpg ~ horsepower, data = Auto))
```

```
##
## Call:
## lm(formula = mpg ~ horsepower, data = Auto)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13.5710  -3.2592  -0.3435   2.7630  16.9240
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 39.935861   0.717499   55.66   <2e-16 ***
## horsepower  -0.157845   0.006446  -24.49   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.906 on 390 degrees of freedom
## Multiple R-squared:  0.6059, Adjusted R-squared:  0.6049
## F-statistic: 599.7 on 1 and 390 DF,  p-value: < 2.2e-16
```

Question

Comparer et commenter les résultats précédents.