Régression pénalisée :bootstrap

15 septembre 2025

MSP *2025-2026*

Dans ce TP, nous allons appliquer le bootstrap à une régression. Nous allons prédire la variable medv (prix médian dans une région) à partir des autres variables disponibles dans l'ensemble de données Boston.

Découverte du jeu de données

- Lire le jeu de données Boston disponible dans la librairie MASS
- Extraire la variable medv comme variable réponse y et le reste des variables vont constituer la matrice des variables explicatives X.
- Créer une variable n qui correspond au nombre de patientes du jeu de données et p nombre de variables explicatives du jeu de données.
- Centrer et réduire la variable y et les colonnes de la matrice X à l'aide de la fonction scale().
- Utiliser la fonction cv.glmnet de la librairie *glmnet* pour effectuer une sélection de lambda par validation croisée à 5 folds pour la régression LASSO. Afficher les coefficients de régression pour la solution LASSO avec λ égal à lambda.1se.

Nous constatons que quatre coefficients sont fixés à 0. Dans quelle mesure sommes-nous sûrs qu'ils sont bien égaux à 0 ? Voyons si ces quatre coefficients sont également égaux à 0 dans tous/la plupart/certains échantillons bootstrap. Nous allons simplement effectuer un bootstrap sur les lignes de l'ensemble de données Boston, ajuster le modèle LASSO et collecter le paramètre lambda.1st et tous les coefficients. Nous ferons également cela pour lambda.min simultanément.

- Faire $B = 500$ bootstrap de la validation croisée à 5 folds pour le choix du paramètre λ de la régression LASSO.
- Penser à sauvegarder les 500 vecteurs de coefficients de régression de chaque bootstrap avec lambda.1st et 500 vecteurs de coefficients de régression pour lambda.min.
- Penser à sauvegarder les 500 valeurs de lambda.1st et les 500 valeurs de lambda.min.
- Pour chaque variable explicative, calculer la proportion de coefficients de régression avec lambda.1st de valeur absolue supérieure à 10^{-6}.
- Pour chaque variable explicative, calculer la proportion de coefficients de régression avec lambda.min de valeur absolue supérieure à 10^{-6}.
- Utiliser un graphique de type barplot pour comparer les deux proportions pour chaque variable explicative.
- Utiliser un graphique boxplot pour tracer la distribution des 500 réalisation du coefficients associés à chaque variable explicative. Faire le même graphique pour lambda.1st et lambda.min. Commenter les résultats.
- Déduire deux intervalles de confiance (pour lambda.1st et lambda.min) empiriques pour chaque coefficient de régression associé à une variable explicative.

Ces analyses étaient simples à réaliser, même si elles nécessitaient quelques calculs. Cependant, la méthode bootstrap n'est peut-être qu'une approximation grossière de la distribution a posteriori réelle. Nous allons maintenant examiner des résultats plus précis pour la distribution a posteriori d'un modèle LASSO.

Lois a posteriori des coefficients de régression à l'aide de blasso

Nous avons établi que le modèle LASSO revient à une estimation du maximum a posteriori (MAP) d'un modèle bayésien. La propriété centrale de ce modèle bayésien est que la loi a priori sur tous les coefficients est une distribution de Laplace.

Explorons maintenant les distributions a posteriori des coefficients dans le cadre d'un modèle Bayesian LASSOformulé par Park & Casella (2008). La différence avec la version purement « vraisemblance pénalisée » du LASSO est qu'ici, nous introduisons explicitement des lois a priori pour tous les paramètres du modèle et utilisons un MCMC pour estimer les lois a posteriori. Techniquement, cela est plus compliqué que le bootstrap vu précédemment, mais heureusement Robert Gramacy l'a implémenté dans la fonction `blasso()` du package `monomvn`.

Le modèle bayésien derrière `blasso` inclut également les paramètres μ, λ et σ^2, et nécessite donc de leur attribuer aussi des lois a priori explicites. Le Bayesian LASSO est défini de façon hiérarchique : dans un premier temps, μ, λ et σ^2 reçoivent des lois a priori dites *non informatives*, afin de refléter l'idée que, pour ces valeurs, l'information proviendra presque exclusivement des données, et non de l'a priori. Conditionnellement à λ et σ^2, nous définissons ensuite l'a priori LASSO (distribution de Laplace) pour les coefficients. Cet a priori permet un fort rétrécissement des coefficients par rapport à ce que favoriseraient les données seules, cette partie du modèle a donc un caractère très *informatif*. Pour compléter le modèle, nous stipulons que les données observées résultent d'erreurs gaussiennes ajoutées au à la combinaison linéaire, comme dans un modèle linéaire fréquentiste.

En formules, la spécification du modèle est :

- $\mu \sim 1$ (i.e., une loi a priori plate)
- $\sigma^2 \sim 1/\sigma^2$ (i.e., une loi a priori plate pour $\log(\sigma^2)$)
- $\lambda^2 \sim \Gamma$ distribution Gamma «non informative»
- $(\beta_j \mid \sigma, \lambda) \sim \text{Laplace}(0, \sigma/\lambda)$ pour tout $j = 1, \ldots, p$.
- $(y_i \mid \mu, \beta, \sigma^2, x_i) \sim \mathcal{N}(\mu + x_i^T \beta, \sigma^2)$ pour tout $i = 1, \ldots, n$.

Avec ces distributions, la règle de Bayes indique que la loi a posteriori conjointe de tous les paramètres inconnus est proportionnelle au produit de l'a priori et de la vraisemblance :

$$\Pr(\beta, \mu, \sigma^2, \lambda^2 \mid y, X) \propto \Pr(y \mid \beta, \mu, \sigma^2, \lambda^2, X) \cdot \Pr(\beta \mid \sigma^2, \lambda^2) \Pr(\mu) \Pr(\sigma^2) \Pr(\lambda^2),$$

où le premier terme du membre de droite est la vraisemblance gaussienne du modèle linéaire, et les 4 termes restants sont les lois a priori définies de manière hiérarchique (d'abord des lois a priori plates indépendantes sur μ, σ^2 et λ^2, puis la loi a priori de Laplace sur β conditionnellement aux valeurs de σ^2 et λ^2). La fonction blasso() utilise un algorithme MCMC pour échantillonner numériquement cette distribution a posteriori.

- Les options de la fonction blasso() :
  - Elle requiert en entrée le nombre d'itérations MCMC T.
  - Un nombre plus élevé d'itérations conduit à des résultats plus précis, mais demande également plus de temps de calcul.
  - L'option RJ = FALSE correspond au Bayesian LASSO, tandis que RJ = TRUE ajoute une sélection bayésienne de variables via le Reversible Jump MCMC. Pour l'instant, nous utilisons RJ = FALSE.
  - L'algorithme MCMC produit un jeu de valeurs des paramètres à chaque itération, et ces valeurs peuvent être interprétées comme des échantillons de la distribution a posteriori, à l'exception

des toutes premières itérations, où l'algorithme dépend encore de ses conditions initiales. Ainsi, certaines itérations initiales sont souvent éliminées comme période de « burn-in » de l'algorithme. Ici, nous éliminons les 50 premières itérations comme burn-in.

- (**à faire**) Lancer blasso et comparer les distributions a posteriori aux résultats obtenus précédemment par bootstrap (lambda.min) à l'aide de boxplots. Commenter.

Un jeu de données simulé avec plus de sparsité

Nous allons simuler un jeu de données avec $p = 30$ variables explicatives, dont les 3 premières ont des effets non nuls, chacune expliquant environ 5% de la variance de la variable réponse, et considérons $n = 250$ individus.

```
set.seed(122)
p = 30
n = 250
phi = 0.05
b = rep(c(sqrt(phi / (1-phi)), 0), c(3, p-3))
X = matrix(rnorm(n*p), nrow = n)
eps = scale(rnorm(n, 0, 1))
y = scale(X%*%b + eps)
```

- (**à faire**) Lancer blasso et afficher les distributions a posteriori des coefficients de régression à l'aide de boxplots. Comparer les résultats avec l'option RJ=TRUE.