Analyse des données prostate: quelques méthodes de régularisation ℓ_1

julien.chiquet@genopole.cnrs.fr

Module MPR – option modélisation, 8 décembre 2009

Table des matières

1 Motivations

Soit le modèle linéaire défini par

$$y = \beta_0 + \sum_{i=1}^{p} X_i \beta_i + \varepsilon,$$

où y est la réponse à expliquer par le vecteur de variables $X = (X_1, \ldots, X_p)^\intercal$, avec $\varepsilon \sim \mathcal{N}(0, \sigma^2)$. Notons $\mathbf{y}$ un vecteur de taille n et $\mathbf{X}$ la matrice de taille $n \times p$ des n observations.

Dans la suite, afin de simplifier l'écriture, on suppose que l'intercept β_0 est estimé par $\bar{\mathbf{y}} = n^{-1} \sum_{i=1}^{n} y_i$, que le vecteur $\mathbf{y}$ est centré et que la matrice $\mathbf{X}$ est centrée-réduite. Ainsi l'estimation ne porte plus que sur le vecteur $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)^\intercal$ de $\mathbb{R}^p$. Dans ce cas, le modèle s'écrit en terme matriciel sous la forme

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}.$$

Par rapport aux moindres carrés ordinaires, la régression *ridge* estime le paramètre $\boldsymbol{\beta}$ avec un léger biais mais permet de contrôler la variance. Pour un choix judicieux[1] du paramètre de pénalisation λ, l'erreur de prédiction peut être meilleure avec la régression ridge qu'avec les moindres carrés.

Nous présentons ici d'autres méthodes de régularisation qui induisent également un biais et une baisse de la variance. Elles sont fondées sur la norme ℓ_1, définie, pour un vecteur $\mathbf{x} = (x_1, \ldots, x_p)^\intercal$, par

$$\|\mathbf{x}\|_{\ell 1} = \sum_{i=1}^{p} |x_i|.$$

Ces méthodes ont un double avantage

1. elles permettent d'annuler *exactement* certaines valeurs des coefficients estimés pour des valeurs suffisamment fortes du paramètre λ : en *sélectionnant* automatiquement les variables, le modèle gagne en interprétabilité ;
2. le problème d'optimisation à résoudre reste convexe, c'est-à-dire relativement facile résoudre, même s'il n'existe pas de forme analytique de la solution dans le cas général.

2 Méthodes de régularisation s'appuyant sur la norme ℓ_1

Dans la suite, on note (1)

$$\mathrm{RSS}(\boldsymbol{\beta}) = \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 ,$$

la somme du carré des résidus du problème.

2.1 Le Lasso

La régression Lasso, très populaire, a été proposée par [Tib96], s'inspirant de méthodes de traitement du signal. Elle consiste à ajouter une pénalité en norme ℓ_1 du vecteur des coefficients (*i.e.*, la somme des valeurs absolues des éléments du vecteur). Ainsi, l'estimateur est défini par

$$\widehat{\boldsymbol{\beta}}^{\text{lasso}} = \arg\min_{\boldsymbol{\beta}\in\mathbb{R}^p} \left\{ \frac{1}{2}\mathrm{RSS}(\boldsymbol{\beta}) + \lambda\|\boldsymbol{\beta}\|_1 \right\}. \tag{1}$$

La Lasso (*Least Absolute Shrinkage and Selection Operator*) a la double propriété de *rétrécir* les coefficients de $\boldsymbol{\beta}$, en limitant leur taille à la manière de la régression ridge, mais surtout de *sélectionner* ces coefficients en en annulant certains pour des valeurs suffisamment forte du paramètre de pénalité λ.

Chemin de régularisation. Si l'on construit le chemin de solutions du vecteur $\boldsymbol{\beta}$ en fonction du paramètre λ, on s'aperçoit qu'il est *linéaire par morceaux*. Ainsi, on peut construire une suite de $p+1$ valeurs croissantes de $\lambda_0 = 0 < \lambda_1 < \ldots \lambda_p$ telles

[1] typiquement, une valeur qui minimise l'erreur de prédiction calculée sur un échantillon test

que $\boldsymbol{\beta}_{\lambda_0}^{\text{lasso}}$ soit l'estimateur des moindres carrés, $\boldsymbol{\beta}_{\lambda_1}^{\text{lasso}}$ ait exactement un coefficient nul, $\boldsymbol{\beta}_{\lambda_2}^{\text{lasso}}$ deux coefficients nuls, etc. jusqu'à $\boldsymbol{\beta}_{\lambda_p}^{\text{lasso}}$ égal au vecteur nul.

La détermination de cette suite de valeurs de λ et des $\boldsymbol{\beta}$ correspondant se fait par l'algorithme LARS dont il existe un package `R`.

2.2 Quelques variantes

De nombreuses variantes du LASSO ont été proposées, impliquant la plupart du temps de nouveaux algorithmes de calcul.

2.2.1 L'Elastic-Net

L'elastic-net de [ZH05] pénalise la somme des carrés résiduelles par un mélange de pénalité ridge et de pénalité LASSO :

$$\widehat{\boldsymbol{\beta}}^{\text{e-net}} = \arg\min_{\boldsymbol{\beta}\in\mathbb{R}^p} \left\{ \frac{1}{2}\text{RSS}(\boldsymbol{\beta}) + \lambda \left(\alpha\|\boldsymbol{\beta}\|_1 + (1-\alpha)\|\boldsymbol{\beta}\|_2^2\right) \right\}, \tag{2}$$

2.2.2 L'adaptive Lasso

L'adaptive-LASSO permet de pénaliser différemment les différents coefficients du vecteur $\boldsymbol{\beta}$.

$$\widehat{\boldsymbol{\beta}}^{\text{lasso}} = \arg\min_{\boldsymbol{\beta}\in\mathbb{R}^{p+1}} \left\{ \frac{1}{2}\text{RSS}(\boldsymbol{\beta}) + \lambda\|\mathbf{w}\circ\boldsymbol{\beta}\|_1 \right\}. \tag{3}$$

où $\mathbf{w}$ est un vecteur de poids permettant de doser le niveau de pénalité de chaque coefficient de $\boldsymbol{\beta}$. La notation $\circ$ est utilisée pour désigner le produit terme à terme entre deux vecteurs ou deux matrices.

2.2.3 Le group-Lasso

Le group-LASSO de [YL06] pénalise les variables par groupes, ainsi les variables sont activées (sélectionnées) par groupes :

$$\widehat{\boldsymbol{\beta}}^{\text{grp}} = \arg\min_{\boldsymbol{\beta}\in\mathbb{R}^p} \left\{ \frac{1}{2}\text{RSS}(\boldsymbol{\beta}) + \lambda \sum_{g=1}^{G} \|\boldsymbol{\beta}^g\|_1 \right\}, \text{ où } \boldsymbol{\beta} = \left(\boldsymbol{\beta}^1, \ldots, \boldsymbol{\beta}^g, \ldots, \boldsymbol{\beta}^G\right)^\intercal, \tag{4}$$

avec $\text{card}(\boldsymbol{\beta}^g) = p_g$, $\sum_g p_g = p$. Les groupes sont connus selon ce que représentent les variables explicatives.

3 Code `R` commenté

Les données prostate contiennent une matrice `x`, un vecteur `y` et un vecteur de booléens `set` indiquant les données de l'ensemble d'apprentissage. On charge les données et les fonctions qui sont stockées dans le fichier `functions.R`, puis on centre et on réduit la matrice $\mathbf{X}$. On construit les données de test et d'apprentissage comme pour l'étude de la régression ridge.

```
> source("functions.R")
> load("prostate.rda")
> x <- scale(as.matrix(x))
> n <- sum(set)
> n.test <- sum(!set)
> x.test <- x[!set, ]
> y.test <- y[!set]
> x <- x[set, ]
> y <- y[set]
```

Pour calculer les estimateurs, nous chargeons les packages suivants[2] :

```
> library(glmnet)
> library(grplasso)

----------------------------------------------------------------------
Please note that this is an early test release of package 'grplasso'.
It should only be used for experimental reasons. Use at your own risk!
----------------------------------------------------------------------
```

La package `glmnet` a été conçu pour calculer la solution de l'elastic-net, mais il permet également de calculer la solution du Lasso. Le package `grplasso` est utile pour le group-Lasso. Un point fort de ces packages est de gérer les modèles de régressions linéaires généralisés (ainsi la régression logistique) : lorsque la réponse y a expliquer n'est pas une variable continue, le modèle linéaire simple n'est pas toujours approprié, puisqu'il suppose que y est gaussien. Ainsi, dans le cas d'une variable catégorielle (comme dans le cas des données sport – y représente le sport pratiqué), il peut être judicieux d'utiliser un modèle logistique. Ces packages permettent de le faire[3].

Ici la variable explicative est continue, donc le modèle linéaire usuel est bien adapté.

3.1 Estimation

3.1.1 Lasso

Par défaut, `glmnet` calcule la solution du Lasso pour 100 valeur de λ depuis une valeur annulant tous les coefficients jusqu'à une valeur proche de zéro, où le vecteur estimé est proche des moindres carrés. L'algorithme ne descend pas nécessairement jusqu'à de pénalités $\lambda = 0$ pour des problèmes de stabilité. Nous calculons l'erreur au passage :

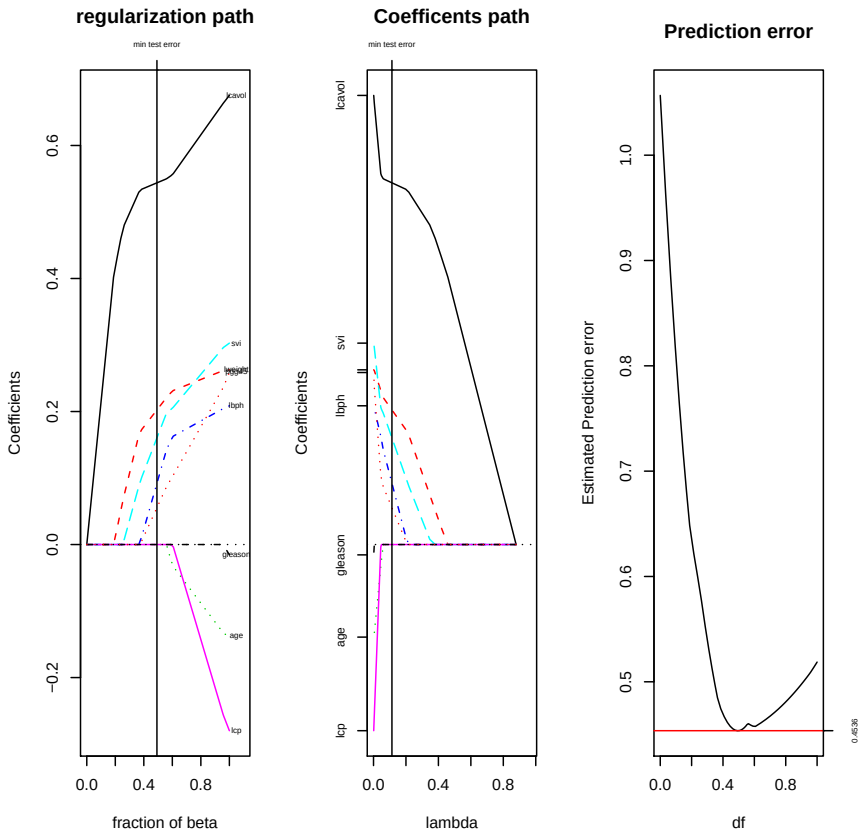
```
> out.lasso <- glmnet(x, y)
> err.lasso <- 1/n.test * colSums((y.test - out.lasso$a0[1] - x.test %*%
+     out.lasso$beta)^2)
```

[2] N'ayez pas peur du message d'avertissement du package `grplasso`.

[3] à bon entendeur...

Dans l'objet `out.lasso` se trouvent `out.lasso$beta`, valeurs estimées des coefficients en fonction des valeurs `out.lasso$lambda` de la pénalité. Voici le tracé des résultats (chemin de régularisation, coefficients et erreur de prédiction) à l'aide des fonctions contenues dans le fichier `functions.R`.

```
> par(mfrow = c(1, 3))
> frac.lasso <- plot.path(t(out.lasso$beta), err = err.lasso)
> plot.coef(t(out.lasso$beta), out.lasso$lambda, err.lasso)
> plot.err(err.lasso, frac.lasso)
```

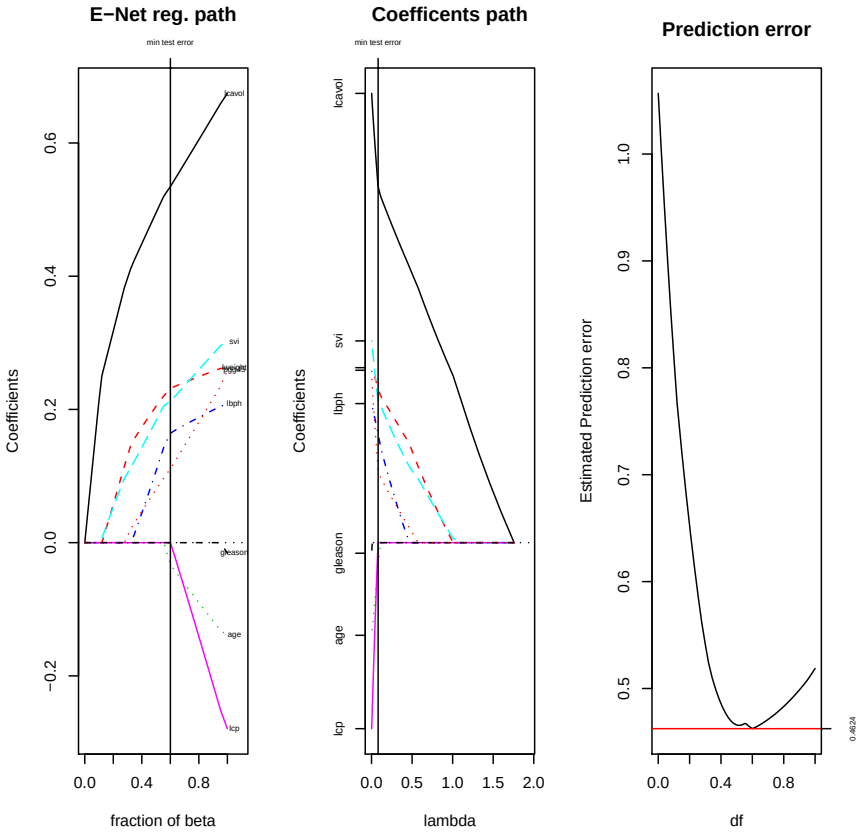


3.1.2 Elastic-Net

Pour l'estimateur de l'elastic-net, choisissons par exemple $\alpha = 0.5$, c'est-à-dire accordant autant d'importance à la pénalité en norme ℓ_1 qu'en norme ℓ_2 :

```
> out.elas <- glmnet(x, y, alpha = 0.5)
> err.elas <- 1/n.test * colSums((y.test - out.elas$a0[1] - x.test %*%
+     out.elas$beta)^2)
```

Graphiquement, les résultats sont :



3.1.3 Adaptive-Lasso

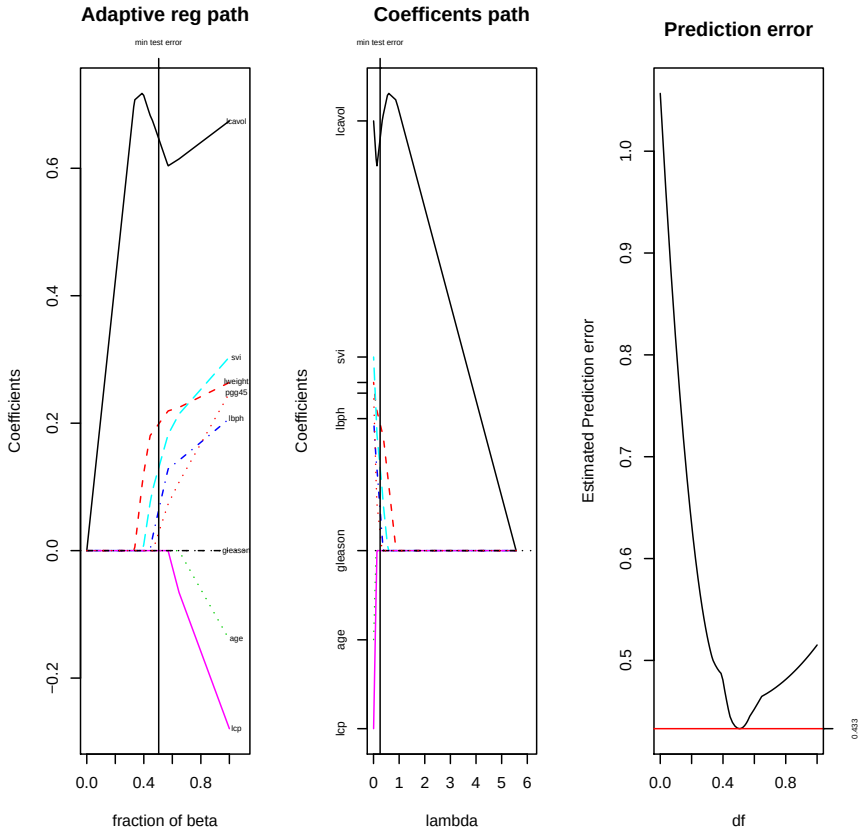
Pour l'adaptive-Lasso un choix usuelle pour les poids est le suivants :

$$\mathbf{w} = \max\left(1, 1/\boldsymbol{\beta}^{\text{ols}}\right),$$

d'où le code :

```
> beta.ols <- lm(y ~ x)$coefficients[-1]
> p.fact <- pmax(1, abs(beta.ols)^(-1))
> out.alasso <- glmnet(x, y, penalty.factor = p.fact)
> err.alasso <- 1/n.test * colSums((y.test - out.alasso$a0[1] -
+     x.test %*% out.alasso$beta)^2)
```

Les résultats sont :



3.1.4 Group-Lasso

Commençons par déterminer des groupes : au vu du chemin de régularisation du Lasso, on a envie d'activer simultanément les variables 2 (`lweight`) et 5 (`svi`), tout comme les variables 3 (`age`) et 6 (`scp`) ; de même pour les variables 4 (`lbph`) et 8 (`pgg45`). D'où le vecteur indiquant les groupes

```
> group <- c(1, 2, 3, 4, 2, 3, 5, 4)
```

La package `grplasso` est moins facile à utiliser que `glmnet`. Nous devons créer à la main un objet `R` contenant les valeurs de l'intercept et ainsi qu'un vecteur de pénalité. Nous prenons les mêmes valeurs de λ que pour le Lasso que l'on remet à échelle (effet de bord du package `grplasso`) :

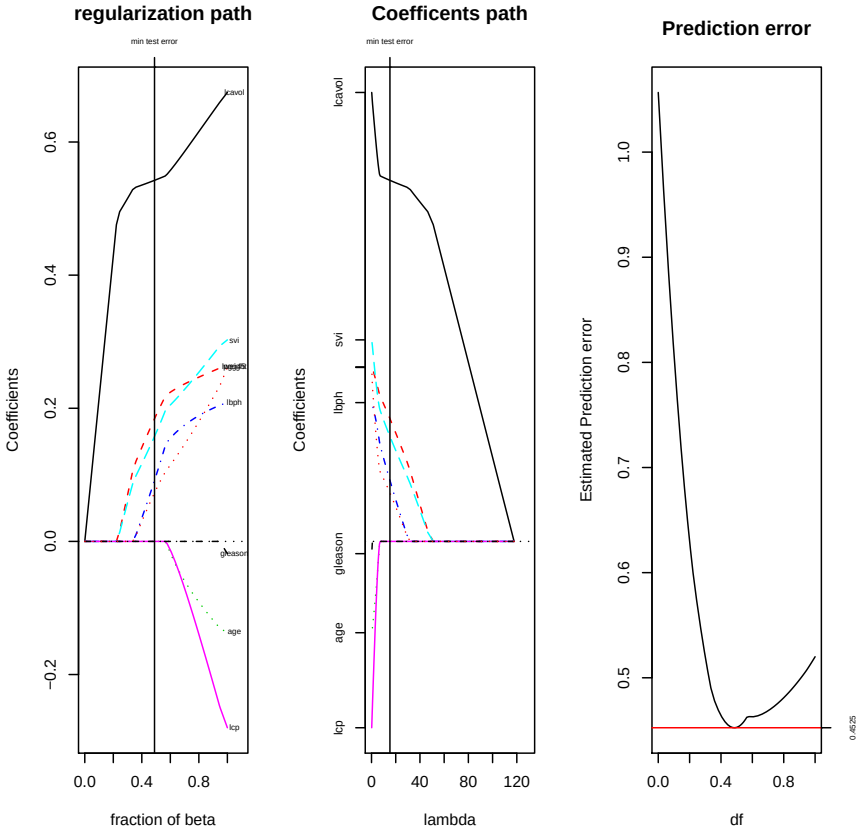
```
> out.group <- list()
> out.group$beta0 <- mean(y)
> out.group$lambda <- 2 * n * out.lasso$lambda
```

Nous somme prêt à faire appelle à la fonction `grplasso` : on notera le paramètre modèle (ici `LinReg()`, pouvant être `LogReg`). Le paramètre `grpl.control()` est utilisé ici pour éviter les messages intempestifs.

```
> out <- grplasso(x, y = y - mean(y), index = group, lambda = out.group$lambda,
+     model = LinReg(), control = grpl.control(trace = 0))
```

```
> out.group$beta <- rbind(t(out$coefficients))
> err.group <- 1/n.test * colSums((y.test - mean(y) - x.test %*%
+     t(out.group$beta))^2)
```

Les résultats donnent



Références

[Tib96] Tibshirani, R., Regression Shrinkage and Selection via the Lasso, *Journal of the Royal Statistical Society*, 1996.

[ZH05] Zou, H. and Hastie T., Regularization and variable selection via the elastic net, *Journal of the Royal Statistical Society*, 2005.

[Zou06] Zou, H. , The adaptive Lasso and its Oracle properties, *Journal of the American Statistical Society*, 2006.

[YL06] Yuan, M., Lin, Y., Model Selection and Estimation in Regression with Grouped Variables, *Journal of the Royal Statistical Society*, 2006.