Etude des propriétés empiriques du lasso par simulation. Partie 1

Franck Picard

Janvier 2017

Contents

L'objectif de ce TP est d'étudier les propriétés empiriques du LASSO et de ses variantes à partir de données simulées. Un deuxième objectif est d'apprendre une méthodologie de simulations numériques, qui sont souvent utilisées dans les travaux de recherche, soit pour étudier les propriétés d'une méthode, soit pour comparer les performances de plusieurs méthodes.

1 Méthodologie de simulation

La simulation de données est un passage souvent obligé dans un travail de recherche, qu'il soit théorique ou appliqué. Au cours de ce TP, nous allons tenter de définir une méthodologie permettant de mettre en place, et ensuite d'analyser (et de rédiger) des résultats de simulation. Dans le TP nous considérerons les simulations comme un moyen d'étudier les propriétés de méthodes statistiques sur des données parfaitement contrôlées. A cet égard, il est nécessaire de définir "les propriétés des méthodes statistiques", et ce qu'on entend par "données contrôlées". Le premier terme dépend évidemment de la méthode étudiée. Dans notre cas, nous étudions une méthode de sélection de variables, et les propriétés d'intérêt sont : la capacité à retrouver les variables pertinentes quand elles existent, la capacité à prédire une nouvelle observation, la précision d'estimation. Le deuxième terme "données contrôlées" se réfère aux hypothèses qui sont faites lors de la définition du modèle statistique (distribution, dimensions par exemple). Un intérêt important des simulations numériques est de se placer dans le "bon cas" (*ie* celui prévu par la théorie), pour étudier les propriétés de la méthode dans ce cadre: c'est une sorte de contrôle positif (on s'assure que les performances sont bonnes dans le cadre prévu par la théorie). Ensuite, la simulation permet de considérer des cas qui s'éloignent de la théorie, et d'étudier la sensibilité de la méthode à des écarts aux hypothèses.

Le cadre méthodologique qui permet de construire les simulations et de les analyser est celui du modèle linéaire et de la planification expérimentale. Dans ce cadre, les priopriétés du modèle à étudier sont considérées comme des variables réponse dont il s'agit d'expliquer les variations par des facteurs contrôlés. Un exemple ici serait d'étudier les variations des performances du lasso en fonction du nombre d'observations n et du nombre de variables p. Cependant, on comprend bien que considérer n et p ou n/p n'est pas suffisant, car les performances du lasso en terme de sélection par exemple, dépendront du ratio signal/bruit présent dans les données. Par conséquent, un élément essentiel des études par simulation est la mise en place d'un plan d'expérience permettant

- d'identifier la variable réponse que l'on souhaite étudier
- d'identifier les facteurs de variations potentiels expliquant les variations de cette réponse
- le contrôle des croisements de facteurs
- l'analyse statistique des résultats de simulation
- La rédaction et la présentation concise des résultats

1.1 Simulation des observations

On considère le cas simple de la sélection de variables avec le lasso dans le cas des modèles linéaires gaussiens:

$$Y_i = x_i^T \beta^* + \varepsilon_i,\ \varepsilon_i \sim \mathcal{N}(0, \sigma^2),$$

avec Y un vecteur de $\mathbb{R}^n$, et β^* un vecteur de $\mathbb{R}^p$ dont p_0 éléments sont non-nuls. Dans la suite, on note $J_0 = \{j \in \{1, ..., p\}, \beta_j^* \neq 0\}$. On utilisera le code ci-dessous pour simuler des observations. On considèrera que les covariables sont indépendantes, gaussiennes centrées réduites dans un premier temps. On considèrera également que toutes les positions non nulles de β^* sont égales.

```
p       = 10
p0      = 5
n       = 5*p
sigma   = 1
sigmaX  = 1
b0      = 5
beta0   = c(rep(b0,p0),rep(0,p-p0))
X       = sapply(1:p, FUN=function(x){rnorm(n,0,sigmaX)})
vareps  = rnorm(n,0,sigma)
Y       = X%*%beta0 + vareps
plot(Y,X%*%beta0)
```

Pour la suite, on utilisera la fonction `setsimul(p,p0,n,sigma,sigmaX,b0)` pour générer les données:

```
source("lassofun.R")
sim = setsimul(p,p0,n,sigma,sigmaX,b0)
plot(sim$Y,sim$X%*%sim$beta0)
```

1.2 Valeur des paramètres de simulation

Concernant les facteurs modifiant les performances du lasso, on peut identifier n (pour étudier le cadre asymptotique), p (pour étudier le cadre de la grande dimension), β^* (pour étudier l'influence de la force du signal), σ (pour étudier l'influence de la force du bruit). On voit tout de suite qu'envisager un plan d'expérience qui croise toutes les modalités de tous ces facteurs n'est pas envisageable (et surtout peu informatif). On va donc cibler les facteurs qui nous intéressent le plus:

- n/p: c'est en fait le ratio qui nous intéresse pour répondre à la question : quelles sont les performances du lasso en "petite" ou en "grande" dimension.
- p_0, le nombre de composantes non nulles du vecteur β^*
- SNR^2 c'est le ratio signal sur bruit, défini par la puissance du signal divisée par la puissance du bruit. Il est défini par:

$$\text{SNR}^2 = \frac{\mathbb{E}_X\left(\mathbb{E}(Y_i|x_i)^2\right)}{\mathbb{E}_X\left(\mathbb{V}(Y_i|x_i)\right)}$$

- On peut également utiliser le coefficient de détermination du modèle linéaire R^2.
- Calculez le ratio signal sur bruit dans le cas du modèle linéaire gaussien avec régresseurs indépendants centrés réduits. Parmi les composantes de β^*, p_0 sont non nulles et $p - p_0$ sont nulles. Pour simplifier encore les simulations, on choisira la même valeur pour toutes les composantes non-nulles notée β_0^*. De quel(s) paramètres dépendent ce ratio ?
- Faites le même calcul pour le coefficient de détermination (R^2).

2 Performance du lasso en petite dimension

2.1 Implémentation du lasso avec `glmnet`

On estime les paramètres du modèle à l'aide du package `glmnet`. Le paramètre λ est choisi par validation croisée à l'aide de la fonction `cv.glmnet`:

```
library(glmnet)
lambda.cv = cv.glmnet(X,Y, family = "gaussian",intercept=F)$lambda.1se
bh        = glmnet(X,Y,family = "gaussian",intercept=F, lambda=lambda.cv)$beta
if ( sum(abs(bh))==0 ) {bh = rep(0,p)}
```

Dans la suite on utilisera la fonction `getlasso(X,Y)` permettant d'estimer les paramètres de la régression pénalisée par lasso.

2.2 Biais / Variance du lasso

A l'aide du code ci-dessous estimez le bias et la variance de l'estimateur obtenu par le lasso sur `nbsimul` simulations, pour une valeur de β_0^* fixée. Evaluez également le temps moyen nécessaire à l'estimation par le lasso. Formatez vos sorties sous la forme d'une `data.frame` comportant en colonne les paramètres des simulations, ainsi que les variables étudiées.

```
n         = 100
sigma     = 1
sigmaX    = 1
p0        = 5
b0        = 1
beta0     = c(rep(b0,p0),rep(0,p-p0))
nbsimul   = 10
res.lasso = NULL

for (i in 1:nbsimul){
  sim   = setsimul(p,p0,n,sigma,sigmaX,b0)
  ptm   = proc.time() ;
  betah = getlasso(sim$X,sim$Y) ;
  ptm   = proc.time()-ptm ;
  res.lasso = rbind(res.lasso, c(n = n, p = p, sigma= sigma, b0 = b0,
                        bias = sum(beta0-betah)/p,
                        mse  = sum( (beta0-betah)^2 )/p,
                        ptm[3],
                        method  = "lasso")
             )
}
res         = as.data.frame(res.lasso)
res$bias    = as.numeric(as.character(res$bias))
res$mse     = as.numeric(as.character(res$mse))
res$elapsed = as.numeric(as.character(res$elapsed))
library(ggplot2)
ggplot(data=res, aes(x=p,y=bias,color=b0)) + geom_boxplot()
```

2.3 Performances en sélection

Définissez les critères suivants à partir de J_0 et $\widehat{J_0}$: accuracy en terme de support, spécificité, sensibilité. Donnez leur interprétation et leurs valeurs limites en fonction de p et p_0. Ces critères permettront d'étudier les performances des méthodes de sélection de variables. Utilisez les fonctions `getACC(vtrue, vest)`, `getsens(vtrue,vest)`, `getspe(vtrue,vest)` pour leur implémentation. Rajoutez ces indicateurs de performance dans votre code qui contiendra donc les indicateurs de performance en terme d'estimation et de sélection.

```
res.lasso = rbind(res.lasso, c(n = n, p = p, sigma= sigma, b0 = b0,
                          bias = sum(beta0-betah)/p,
                          mse  = sum( (beta0-betah)^2 )/p,
                          acc  = ... ,
                          spe  = ... ,
                          sens = ... ,
                          ptm[3],
                          method  = "lasso")
           )
```

2.4 Propriétés empiriques du lasso en petite dimension

Nous allons étudier les propriétés empiriques de l'estimateur du lasso en petite dimension. On fixera les paramètres : $n = 100$, $p_0 = 5$, et $p \in \{10, 50, 100\}$. Dans la suite, on choisira trois valeurs pour le $\text{SNR}^2 \in \{0.02, 1, 20\}$ qui correspondent aux trois situations typiques : pas de signal (cas défavorable), autant de signal que de bruit (cas limite), plus de signal que de bruit (cas favorable).

- Calculez les valeurs de β_0^* pour atteindre ces trois valeurs de SNR.
- Utilisez le modèle de code ci-dessous pour étudier l'evolution du biais et de la variance de l'estimateur du lasso
- Présentez une analyse synthétique de vos résultats.

```
for (snr2 in c(0.02,1,20)){
  #b0 = ...
  #beta0 =
  for (i in 1:nbsimul){
    sim   = setsimul(p,p0,n,sigma,sigmaX,b0)
    # code précédent
  }
}
library(ggplot2)
ggplot(data=res, aes(x=p,y=bias,color=b0)) + geom_boxplot()
ggplot(data=res, aes(x=p,y=acc,color=b0)) + geom_boxplot()
```

3 Comparaison avec des méthodes alternatives

3.1 par la méthode de sélection pas-à-pas (stepwise)

La sélection de variables se fait classiquement par des méthodes pas-à-pas implémentées dans la fonction `step()` de `R`. Plusieurs options sont possibles, notamment pour sélectionner le nombre de variables (AIC ou BIC), ou pour la méthode d'inclusion des variables pas-à-pas. L'objet de ce TP n'étant pas de coder ces

méthodes, on utilisera la fonction `stepW(X,Y, k=log(n), dir="forward")` qui inclut les variables par la méthode `forward` en choisissant le nombre de variables par le critère du BIC.

Dans la suite, on considérera que la sélection step-wise était une des méthodes les plus utilisées avant le développement du lasso.

3.2 avec les moindre-carrés ordinaires

L'estimateur des moindre-carrés n'étant pas unique en grande dimension ($X^T X$ non inversible), on utilise un estimateur avec une faible pénalité ridge pour régulariser l'estimateur MCO (en utilisant l'option `alpha` dans `glmnet` et `lambda=(1e-2)/n`). Dans la suite on utilisera la fonction `getols(X,Y)`.

```
if (p>=n){
  betah     = glmnet(X,Y,family = "gaussian",intercept=F, lambda=(1e-2)/n, alpha=0)$beta
} else {
  betah     = glmnet(X,Y,family = "gaussian",intercept=F, lambda=0)$beta
}
```

3.3 calcul de l'oracle

L'oracle est calculé comme suit (dans la suite on utilisera la fonction `getoracle(X,Y)`):

```
X = sim$X
Y = sim$Y
betah = c(as.vector(glmnet(X[,1:p0],Y,family = "gaussian",intercept=F,lambda=0)$beta),
          rep(0,p-p0))
```

4 Comparaison des méthodes de sélection de variables en petite dimension

Comparez les performances des différentes méthodes en estimation et en sélection. On considèrera un nombre de variables qui varie entre p_0 (le nombre de variables non nulles), et n (cas limite de la grande dimension). Vous vous inspirerez du code ci-dessous. Si vous aviez un jeu de données à analyser dans ce contexte, quelle(s) méthode(s) recommanderiez-vous ? Pourquoi ?

```r
res.step = res.lasso = res.ols = res.oracle = NULL
for (p in c(p0,50,100)){
  for (snr2 in c(0.02,1,20)){
    #b0 = ...
    #beta0 = ...
    for (i in 1:nbsimul){
      sim   = setsimul(p,p0,n,sigma,sigmaX,b0)
      ptm   = proc.time() ;
      betah = getstep(sim$X,sim$Y) ;
      ptm   = proc.time()-ptm ;
      res.step = rbind(res.step, c(n = n, p = p, sigma= sigma, b0 = b0,
                        bias = sum(beta0-betah)/p,
                        mse  = sum( (beta0-betah)^2 )/p,
                        acc  = ...,
                        spe  = ...,
                        sens = ...,
                        ptm[3],
                        method  = "step")
               )
      # idem avec res.lasso
      # ...
    }
  }
}
res = rbind(res.step,res.lasso,res.ols,res.oracle)
res = as.data.frame(res)
# ... formatage de la data.frame
ggplot(data=res, aes(x=p,y=bias,color=b0)) + geom_boxplot() + facet_grid(method ~.)
```