Etude des propriétés empiriques du lasso par simulations

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. Les questions notées $*$ sont à faire à la maison.

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 plannification 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

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 $J_0 = \{j \in \{1, \dots, p\}, \beta_j^* \neq 0\}$. 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
b0      = 5
beta0   = c(rep(b0,p0),rep(0,p-p0))
X       = sapply(1:p, FUN=function(x){rnorm(n,0,1)})
vareps  = rnorm(n,0,sigma)
Y       = X%*%beta0 + vareps
```

Estimation des paramètres avec glmnet

A l'aide du package `glmnet` estimez les paramètres du modèle que vous avez simulé. On choisira le paramètre λ par validation croisée à l'aide de la fonction `cv.glmnet`. Choisissez la valeur de λ qui minimise l'erreur de validation croisée, et obtenez la valeur $\widehat{\beta}_{\widehat{\lambda}}$. Créer une fonction `getlasso(X,Y)` permettant d'estimer les paramètres de la régression pénalisée par lasso.

```
library(glmnet)
getlasso <- function(X,Y){
    p         = dim(X)[2]
    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)}
    return(bh)
}
```

Question:

Calculez l'estimateur des moindre-carrés et l'oracle sur les mêmes données (l'estimateur que l'on obtiendrait si on connaissait les positions des coefficients non nuls). L'estimateur des moindre carrés ordinaires, n'étant pas unique en grande dimension (X^TX non inversible), on utilisera un estimateur avec une faible pénalité ridge pour régulariser l'estimateur MCO (en utilisant l'option `alpha` dans `glmnet` et `lambda=(1e-2)/n`). Calculez un estimateur du bias et de la variance des estimateurs obtenus

```
### estimateur Oracle
betah     = c(as.vector(glmnet(X[,1:p0],Y,family = "gaussian",intercept=F, lambda=0)$beta),
              rep(0,p-p0))
### estimateur MCO
if (p>=n){
 betah     = glmnet(X,Y,family = "gaussian",intercept=F, lambda=(1e-2)/n, alpha=0)$beta
} else {
 betah     = glmnet(X,Y,family = "gaussian",intercept=F, lambda=0)$beta
}
```

Critères de performance

Question

Définissez les critères suivants à partir de J_0 et $\widehat{J}_0$: accuracy en terme de support, spécificité, sensibilité. Donnez leur interprétation. Ces critères permettront d'étudier les performances des méthodes de sélection de variable. Utilisez les fonctions suivantes pour leur implémentation.

```
getACC<-function(vtrue, vest){
  p   = length(vtrue)
  tp  = sum(vtrue!=0 & vest!=0)
  tn  = sum(vtrue==0 & vest ==0)
  ACC = (tp+tn)/p
  return(ACC)
}
getsens <- function(vtrue,vest){
    tp  = sum(vtrue!=0 & vest!=0)
    fn  = sum(vtrue!=0 & vest==0)
    tp/(fn+tp)
}
getspe <- function(vtrue,vest){
    tn  = sum(vtrue==0 & vest ==0)
    fp  = sum(vtrue==0 & vest!=0)
    tn/(tn+fp)
}
```

Question

Calculez les indicateurs de performance sur `nbsimul` simulations pour obtenir des indicateurs moyens (ainsi que leurs variations). Faites varier les valeurs de β^* (choisir 5 valeurs). Formatez vos sorties sous la forme d'une `data.frame` comportant en colonne les paramètres des simulations, ainsi que les variables étudiées, une ligne correspondant à une simulation (donc `nbsimul` lignes).

```
n       = 100
sigma   = 1
nbsimul = 2
res     = NULL
p0      = 10
b0      = 1
beta0  = c(rep(b0,p0),rep(0,p-p0))
for (i in 1:nbsimul){
  X      = sapply(1:p, FUN=function(x){rnorm(n,0,1)})
  Y      = X%*%beta0 + rnorm(n,0,sigma)
  betah  = getlasso(X,Y)
  res    = rbind(res, c(n = n, p = p, sigma= sigma, b0 = b0,
                        bias = sum(beta0-betah)/p,
                        mse  = sum( (beta0-betah)^2 )/p,
                        acc  = getACC(beta0,betah),
                        sens = getsens(beta0,betah),
                        spe  = getspe(beta0,betah),
                        sh   = sum(betah!=0),
                        method = "lasso")
                 )

  }
```

Plannification des simulations

A partir d'ici les éléments sont en place pour étudier les performances du lasso par 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. On peut donc raisonnablement fixer une valeur pour n par exemple $n = 100$ et faire varier p de telle sorte qu'on se place en petite ou grande dimension: $p \in \{50, 500, 1000\}$. On peut aussi fixer la valeur de p_0 en imposant $p_0 = 10$.
- 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:

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

Question

- 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 ?
- proposez des valeurs pour le SNR et interprétez les.
- Implémentez les simulations pour ces valeurs de paramètres.
- Représentez les boxplots des indicateurs de performance en fonction de n/p et de la valeur de β_0^*. On utilisera le package `ggplot2` à l'aide du code ci-dessous.
- Quels sont les effets de la grande dimension sur les performances des estimateurs (en estimation, en sélection) ?

```
res      = as.data.frame(res)
res$bias = as.numeric(as.character(res$bias))
res$mse  = as.numeric(as.character(res$mse))
res$acc  = as.numeric(as.character(res$acc))
res$spe  = as.numeric(as.character(res$spe))
res$sens = as.numeric(as.character(res$sens))
res$sh   = as.numeric(as.character(res$sh))
res$p    = factor(res$p,levels=levels(res$p)[c(2,3,1)])
res$b0   = factor(res$b0,levels=levels(res$b0)[c(1,2,4,3)])
#library(ggplot2)
#ggplot(data=res, aes(x=p,y=acc,color=b0)) + geom_boxplot() + facet_grid(method~.)
```

Question*.

Calculez l'erreur de prédiction du lasso, du lasso adaptatif (cf. section suivante) et de l'oracle avec une procédure de 10-fold validation croisée, et étudiez ses variations en fonction du plan de simulations précédent. On pourra utiliser la commande `split(sample(n),seq(1,n,by=K))` pour échantillonner des sous ensembles des données en K groupes.

```
getKfoldPredError <- function(X,Y,K){
    n = length(Y)
    out = sapply(split(sample(n),seq(1,n,by=K)), FUN=function(ii){
        betah    = getlasso(X[-ii,],Y[-ii])
        mse      = sum( (Y[ii]- X[ii,]%*%betah)^2 )
```

```
        beta.ada = getadalasso(X[-ii,],Y[-ii],betah)
        mse.ada  = sum( (Y[ii]-X[ii,]%*%beta.ada)^2 )
        beta.ora = c(as.vector(glmnet(X[-ii,1:p0],Y[-ii],family = "gaussian",intercept=F,
                                      lambda=0)$beta), rep(0,p-p0))
        mse.ora  = sum( (Y[ii]-X[ii,]%*%beta.ora)^2 )
        return(c(mse,mse.ada,mse.ora))
    })
    return(apply(out,1,sum)/K)
}
```

Lasso adaptatif

Une version modifiée du lasso a été proposée pour palier (notamment) au problème de biais, c'est le lasso adaptatif. L'idée du lasso adaptatif est de procéder en deux temps. Dans un premier temps, on estime des paramètres $\widehat{\beta}_{\text{init}}$ à l'aide du lasso. Ce premier estimateur va être utilisé comme poids pour une deuxième étape de lasso, en fixant $w_j = \widehat{\beta}_{j,\text{init}}$ tel que:

$$\widehat{\beta}_\lambda = \text{Argmin}_\beta \left(\frac{1}{n}\|Y - X\beta\|_2^2 + \lambda \sum_{j=1}^{p} \frac{|\beta_j|}{|w_j|} \right).$$

L'idée du lasso adaptatif est que si $\widehat{\beta}_{j,\text{init}} = 0$ alors $\widehat{\beta}_{j,\text{ada}} = 0$ de telle sorte que la première étape de lasso sert de pré-sélection. De plus, si $\widehat{\beta}_{j,\text{init}}$ est grand, le lasso adaptatif utilisera une pénalisation plus petite, donc un shrinkage plus petit pour le coefficient j, ce qui est sensé diminuer le bias pour ce coefficient.

Question.

- Implémentez le lasso adaptatif à l'aide de l'option `penalty.factor` dans `glmnet` qui permet d'introduire des λs différents pour chaque élement de β. Créer une fonction `getadalasso(X,Y,beta.init)` qui calcule l'estimateur lasso adaptatif à partir d'un premier vecteur $\widehat{\beta}_{\text{init}}$.
- Comparez les performances du lasso adaptatif à celles du lasso (en estimation, en sélection, et en prédiction*).

```
getadalasso <- function(X,Y,beta.init){
    p          = dim(X)[2]
    n          = dim(X)[1]
    nonnullpos = which(beta.init!=0)
    betah.ada  = rep(0,p)
    if (length(nonnullpos)>2){
        w             = abs(beta.init[nonnullpos])
        lambda.cv     = cv.glmnet(matrix(X[,nonnullpos],nrow=n),Y,
                                  family = "gaussian",
                                  intercept=F,penalty.factor=1/w)$lambda.min
        betah.ada[nonnullpos] = as.vector(glmnet(X[,nonnullpos],Y,
                                                 family = "gaussian",
                                                 intercept=F,lambda=lambda.cv,
                                                 penalty.factor=1/w)$beta)
    } else {
        betah.ada = beta.init
    }
    return(betah.ada)
}
```

Pour aller plus loin

Dans une dernière partie, nous nous interrogerons sur les conditions sur n, p et p_0 pour que le lasso détecte bien les entrées nulles et non-nulles de β^*. Dans un article publié en 2009 (IEEE Transactions on Information Theory, 55:2183–2202, May 2009), M. Wainwright propose des conditions nécessaires et suffisantes pour que le lasso soit consistant en sélection pour le support signé. On notera $\mathbb{S}_{\pm}(\beta)$ le vector de signes de β défini tel que:

$$\mathbb{S}_{\pm}(\beta_i) = \begin{cases} +1 & \text{si } \beta_i > 0 \\ -1 & \text{si } \beta_i < 0 \\ 0 & \text{si } \beta_i = 0 \end{cases}$$

Dans son article M. Wainwright démontre l'existence de deux constantes dépendant de $\Sigma = \mathbb{V}(X)$, $0 < \theta_\ell(\Sigma) \leq \theta_u(\Sigma) < \infty$ telles que pour une valeur

$$\lambda_n = \sqrt{\frac{2\sigma^2 \log(p_0) \log(p - p_0)}{n}}$$

du paramètre de régularisation du lasso,

- si $n/(2p_0(\log(p - p_0))) > \theta_u(\Sigma)$ alors il est toujours possible de trouver une valeur du paramètre de régularisation λ telle que le lasso a une solution unique $\widehat{\beta}$ telle que $\mathbb{P}\{\mathbb{S}_{\pm}(\beta^*) = \mathbb{S}_{\pm}(\widehat{\beta})\}$ tend vers 1.
- si $n/(2p_0(\log(p - p_0))) < \theta_\ell(\Sigma)$, alors quelle que soit la valeur du paramètre de régularisation $\lambda > 0$, aucune des solutions du lasso ne spécifie correctement le support signé de β^*, $\mathbb{P}\{\mathbb{S}_{\pm}(\beta^*) = \mathbb{S}_{\pm}(\widehat{\beta})\}$ tend vers 0.

Dans son article, M. Wainwright propose d'appeler la quantité $n/(2p_0(\log(p - p_0)))$ "taille d'échantillon normalisée". C'est un indicateur qui combine les informations nécessaires à la consistance du lasso. Dans la suite, nous nous placerons dans le cas $\Sigma = I$, avec $\theta_\ell(I) = \theta_u(I) = 1$.

Question

- Pour les paramètres suivants, étudiez l'évolution de l'accuracy en terme de support en fonction de la taille d'échantillon normalisée, pour le lasso avec la valeur théorique de λ proposée ci-dessus. $p \in \{128, 256, 512\}$, $n \in \{100, \ldots, 1000\}$, $p_0 = \lceil 0.4 \times p \rceil$, $\beta_0^* = 0.5$, $\sigma = 0.5$.
- Comparer ces performances avec celles du lasso utilisant un λ calibré par validation croisée, et celles du lasso adaptatif (également avec λ calibré par validation croisée). Discutez les différences de comportement de l'accuracy.