Numerical Calibration of the Lasso

ENS Advanced Math, Non parametrics

Franck Picard, Fall 2020

Preliminaries

Groups of students are asked to send an `R markdown` report generated via `R studio` to franck.picard@ens-lyon.fr at the end of the tutorial. You will need `Rstudio`, LaTeX and packages for markdown:

```
library(knitr)
library(rmarkdown)

library(glmnet)
```

This report should answer the questions by commentaries and codes generating appropriate graphical outputs. A cheat sheet of the markdown syntax can be found here.

1 Regression model

This project aims at studying the empirical properties of the LASSO based on simulated data. The statistical framework is the linear regression model, such that

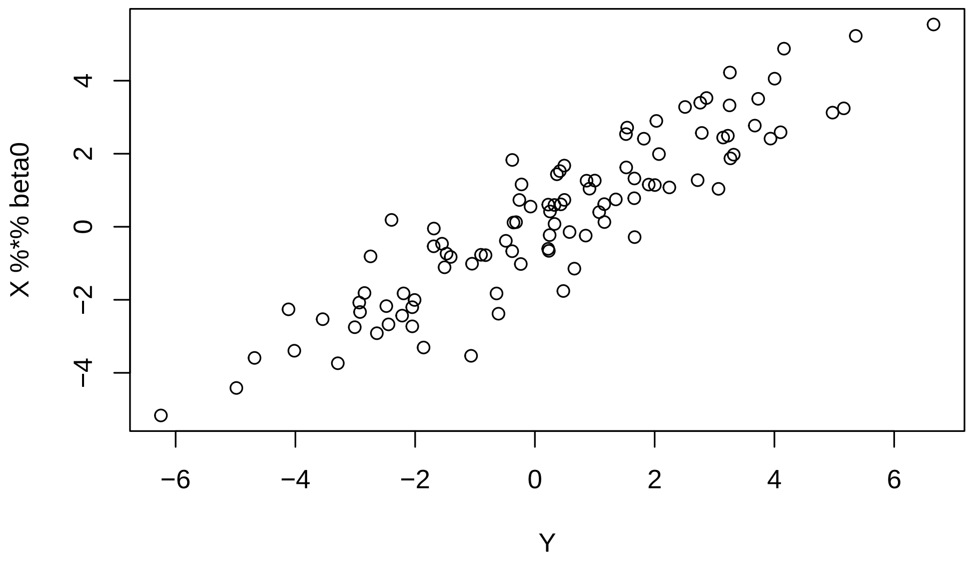
$$Y_i = x_i^T \beta^* + \varepsilon_i, \ \ \varepsilon_i \sim \mathcal{N}(0, \sigma^2),$$

with Y a vector in $\mathbb{R}^n$, and β^* a vector in $\mathbb{R}^p$ with p_0 non-null elements. In the following $J_0 = \{j \in \{1, ..., p\}, \beta_j^* \neq 0\}$. For simplification, we will consider that there is only one distinct non null value in β^*: $\beta^* = \beta_0^* \times (1, ..., 1, 0, ...0)$.

2 Simulation of observations

```
n        = 100
p        = 10
p0       = 5
sigma    = 1
sigmaX   = 1
b0       = 1
beta0    = c(rep(b0,p0),rep(0,p-p0))
X        = sapply(1:p, FUN=function(x){rnorm(n,0,sigmaX)})
Y        = X%*%beta0 + rnorm(n,0,sigma)
```

```
plot(Y,X%*%beta0)
```



3 Lasso and the `glmnet` R-package

In practice, model parameters are estimated using the `glmnet` R-package to compute the lasso estimator

$$\widehat{\beta}_\lambda = \min_{\beta_0,\beta} \frac{1}{N}\sum_{i=1}^{N} w_i l(y_i, \beta_0 + \beta^T x_i) + \lambda \left[(1-\alpha)||\beta||_2^2/2 + \alpha||\beta||_1\right],$$

with $\alpha = 1$ for the Lasso, $\alpha = 0$ for Ridge Regression and $\alpha \in]0,1[$ for Elastic Net. In a first step you are invited to check the online documentation of the package that is very complete. Then the purpose of calibration is to determine the value of the hyperparameter λ based on the observations.

```
library(glmnet)
n       = 100
p       = 10
p0      = 5
sigma   = 1
sigmaX  = 1
b0      = 1
beta0   = c(rep(b0,p0),rep(0,p-p0))
X       = sapply(1:p, FUN=function(x){rnorm(n,0,sigmaX)})
Y       = X%*%beta0 + rnorm(n,0,sigma)

fit = glmnet(X, Y)
```

```
plot(fit,label=TRUE)
names(fit)
print(fit)
coef(fit)
```

4 Numerical Calibration in practice

Parameter λ is chosen by cross validation using `cv.glmnet` such that:

```
library(glmnet)
n        = 100
p        = 10
p0       = 5
sigma    = 1
sigmaX   = 1
b0       = 1
beta0    = c(rep(b0,p0),rep(0,p-p0))
X        = sapply(1:p, FUN=function(x){rnorm(n,0,sigmaX)})
Y        = X%*%beta0 + rnorm(n,0,sigma)

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)}
bh         = as.vector(bh)
```

The first part of your projet will be to re-implement the cross validation procedure, and to verify that your implementation is correct based on the `cv.glmnet` function that will be used to check your results. You will also implement the calibration of λ based on the AIC and on the BIC.

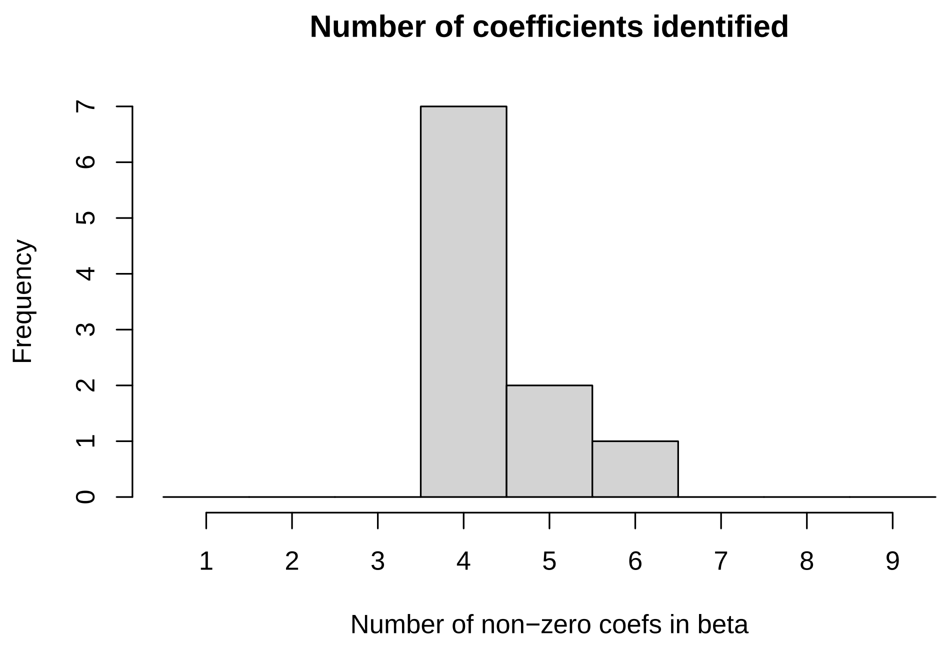
5 Simulations setting

The performance of the lasso depends on different factors, and numerical simulations are used to study the impact of these factors on the capacity of the lasso to select the dimension of the model. Among those factors, we can identify n (number of observations), p (dimensionality), p_0 and β^* (strength of the signal), σ (strength of noise). Studying the impact of all factors would not be realistic, so we focus on:

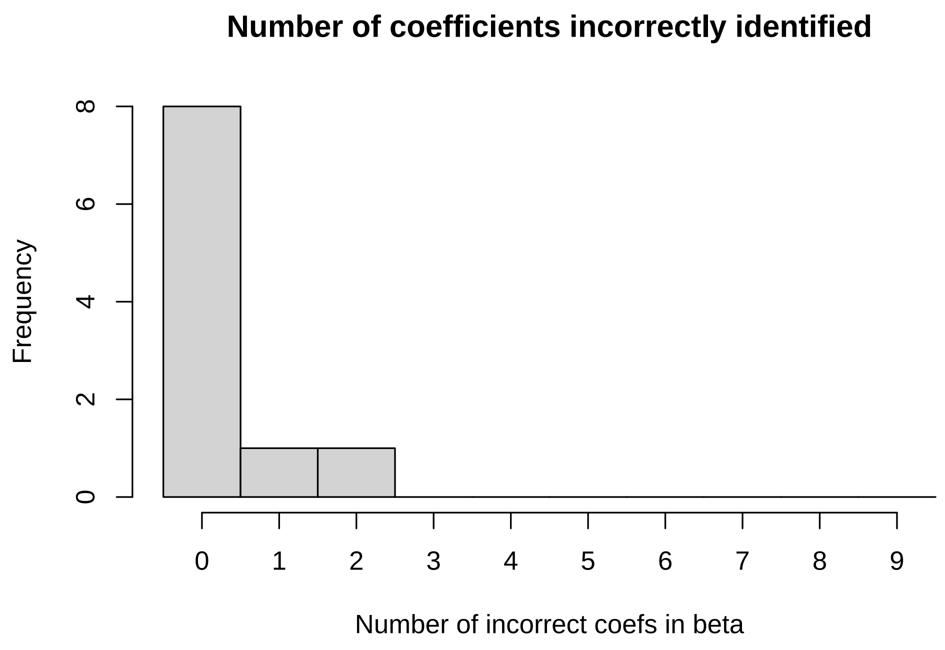
- n/p will be chosen so that we study the "low-dimensional regime" as well as the "high-dimensional regime".
- p_0 will be fixed at 5
- $\beta_0^* = 1$
- σ will vary.

As an indicator of performance, we will study only the dimension of the selected model, ie $\hat{s} = \sum_{j=1}^{p} 1_{\hat{\beta}_j \neq 0}$. In order to conduct your simulations, you will start from the following example code:

```r
n      = 100
p      = 10
p0     = 5  #Number of nonzero coefficients
sigma  = 1
sigmaX = 1
b0     = 1
beta0  = c(rep(b0,p0),rep(0,p-p0))
B      = 10 #Number of simulations
res    = matrix(NA,ncol=5,nrow=B) #Stores n,p, sigma, number of nonzero
                  #coefficients recovered, number of nz incorrectly identified

betah  = matrix(NA, ncol = p, nrow = B)

# fixed design setting
X = sapply(1:p, FUN=function(x){rnorm(n,0,sigmaX)})

for (b in 1:B){
  Y     = X%*%beta0 + rnorm(n,0,sigma)
  # estimate betah using the calibrated lasso
  # betah =
  lambda.cv = cv.glmnet(X,Y, family = "gaussian",intercept=F)$lambda.1se
  temp = glmnet(X,Y,family = "gaussian",intercept=F, lambda=lambda.cv)$beta
  ##the value returned by glmnet(...)$beta is a sparse matrix
  #In a sparse matrix : @i (non zero indices); @x non zero values

  indices_identified = c(rep(0, p)) #Fill indices_identified with 1 at
                                    #indices in temp@i, set others to 0
  for (index in temp@i) {
    indices_identified[index] = 1
  }

  betah[b,]  = c(temp@i, rep(NA, p-length(temp@i))) #We store in betah[b,]
    #the nonzero indices and we fill with NA to get a vector of the proper size
  res[b,] = c(n, p, sigma, length(temp@i), sum(abs(indices_identified - beta0)))
}
res            = as.data.frame(res)
colnames(res) = c("n","p","sigma","nz", "nzcorrect")


hist1<-hist(res$nz, breaks = 1:10, axes = FALSE,
            xlab="Number of non-zero coefs in beta",
            main = "Number of coefficients identified")
axis(side=1,at=hist1$mids,labels=1:9) #So that breakpoints are centered
axis(side=2)          #Because of previous trick, we had to hide both axis,
```



```
                       #so we display again the Y-axis


hist2 <- hist(res$nzcorrect, breaks = 0:10, axes = FALSE,
              xlab="Number of incorrect coefs in beta",
              main = "Number of coefficients incorrectly identified")
axis(side=1,at=hist2$mids,labels=0:9)
axis(side=2)
```



5.1 AIC

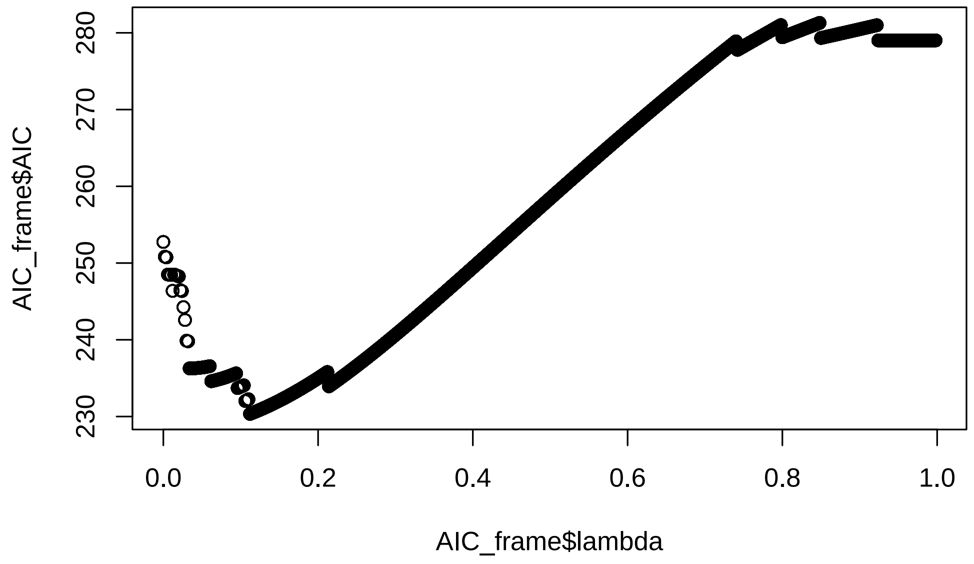
In this section, we implement the Akaike Information Criterion (AIC):

$$AIC = 2k - 2ln(\hat{L})$$

where $\hat{L}$ is the maximum value of the likelihood function for the model under consideration. The procedure consists of computing the Lasso with parameters $\lambda_1, \cdots, \lambda_N$.

```
likelihood <- function(betal, Yl, sigma2 = 1.0){
  S<-0.0
  S <- S + t(Yl-betal)%*%(Yl-betal)
  n<- length(Yl)

  return (1/sqrt(2 * pi* sigma2))**(n/2)*exp(-S/(2*sigma2))
}


n       = 200
p       = 20
p0      = 5  #Number of nonzero coefficients
sigma   = 1
sigmaX  = 1
b0      = 1
beta0   = c(rep(b0,p0),rep(0,p-p0))
betah   = matrix(NA, ncol = p, nrow = B)
X = sapply(1:p, FUN=function(x){rnorm(n,0,sigmaX)})
```

```r
Y      = X%*%beta0 + rnorm(n,0,sigma)



N <- 500 #Number of lambda tested
lambda_step <- 0.002 #Difference between two consecutive lambda
lambda_init <- 0.0 #First value used by lambda

lambda <- lambda_init
AIC_frame <- data.frame(
  lambda = c(),
  betah = c(),
  k = c(),
  loglikelihood = c(),
  AIC = c()
)

for (b in 1:N){
  temp = glmnet(X,Y,family = "gaussian",intercept=F, lambda=lambda)$beta
  ##the value returned by glmnet(...)$beta is a sparse matrix
  #In a sparse matrix : @i (non zero indices); @x non zero values

  betah <- c(rep(0,p))
  j<-0
  for(i in temp@i){
    betah[j+1] <- temp@x[j+1]
    j<-j+1
  }

  #llik = log(((1/sqrt(2 * pi))**(p/2))*exp(-t(Y-betah)%*%(Y-betah)/(2)))
  llik = p*log(t(Y-X%*%betah)%*%(Y-X%*%betah)) #Simpler expression for log likelihood
  new_row_df <- data.frame(
    lambda = lambda,
    betah = betah,
    k = j,
    negloglikelihood = -llik,
    AIC = 2*j + 2*llik
  )
  AIC_frame <- rbind(AIC_frame, new_row_df)

  lambda <- lambda+lambda_step

}
plot( x = AIC_frame$lambda, y = AIC_frame$AIC)
```



We then define the best value for λ with respect to AIC as:

$$\lambda_{AIC} = argmin_{\lambda_i} AIC.$$

Hence, for each i we need to compute the likelihood of the model with parameter β_{λ_i} and k as the number of nonzero coefficients of β_{λ_i}. The likelihood, with Gaussianity assumption and covariance matrix σI, is given by

$$\hat{L_{\lambda_i}} = \Pi_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{1}{2\sigma^2}(Y_i - \beta)^t(Y_i - \beta)$$

. In our setting, we compute $S = \sum_i (Y_i - \beta_\lambda)^2$ and then

$$L_Y(\lambda) = (2\pi\sigma^2)^{-\frac{n}{2}} \exp(-\frac{1}{2\sigma^2}S)$$

```
likelihood <- function(betal, Yl, sigma2 = 1.0){

  S <- t(Yl-betal)%*%(Yl-betal)
  n<- length(Yl)

  return (1/sqrt(2 * pi* sigma2))**(n/2)*exp(-S/(2*sigma2))
}
```

6 Overview of the project

You will first implement the cross validation and the BIC to calibrate the Lasso numerically in the regression setting. Then you will study the statistical properties of the lasso in terms of model selection in different simulation setting (high signal / high noise, low dimensional / high dimensional). You will compete CV, the AIC, and the BIC in order to determine if one procedure is more accurate than the other.

7 Reference

Trevor Hastie, Robert Tibshirani, Jerome Friedman, The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Second Edition, Springer, 2009