Deliverable 3. Lasso estimation for the Boston housing data

Marc Valenti, Daniel Fuentes

27/2/2018

For the Boston House-price corrected dataset use ridge regression to fit the regression model where the response is MEDV and the explanatory variables are the remaining 13 variables in the previous list. Sólo hay que ajustar Lasso y ridge regression con glmnet usando los datos de Boston.

```
library(dplyr)
```

```
library(ggplot2)
library(reshape)
```

```
library(magrittr)
library(knitr)
library(mlbench)
library(MASS)
```

```
library(glmnet)
```

```
library(mlbench)
library(MASS)

multiplot <- function(..., plotlist=NULL, file, cols=1, layout=NULL, title="") {
  require(grid)

  # Make a list from the ... arguments and plotlist
  plots <- c(list(...), plotlist)

  numPlots = length(plots)

  # If layout is NULL, then use 'cols' to determine layout
  if (is.null(layout)) {
    # Make the panel
    # ncol: Number of columns of plots
    # nrow: Number of rows needed, calculated from # of cols
    layout <- matrix(seq(1, cols * ceiling(numPlots/cols)),
                     ncol = cols, nrow = ceiling(numPlots/cols))
  }

  if (nchar(title)>0){
    layout<-rbind(rep(0, ncol(layout)), layout)
  }

  if (numPlots==1) {
    print(plots[[1]])

  } else {
    # Set up the page
    grid.newpage()
```

```
    pushViewport(viewport(layout = grid.layout(nrow(layout), ncol(layout), heights =if(nchar(title)>0){u

    # Make each plot, in the correct location
    if (nchar(title)>0){
      grid.text(title, vp = viewport(layout.pos.row = 1, layout.pos.col = 1:ncol(layout)))
    }

    for (i in 1:numPlots) {
      # Get the i,j matrix positions of the regions that contain this subplot
      matchidx <- as.data.frame(which(layout == i, arr.ind = TRUE))

      print(plots[[i]], vp = viewport(layout.pos.row = matchidx$row,
                                      layout.pos.col = matchidx$col))
    }
  }
}


get_plot_cv<-function(PMSE,title){
  par(mfrow=c(1,2))
  aux<-cbind(PMSE,df.v,lambda.v) %>%
    as.data.frame()
  aux_min<-aux %>%
    slice(which.min(PMSE))

  p1<-ggplot() +
    geom_point(data=aux,
               aes(y=PMSE.mean,
               x=log(1+lambda.v))) +
    geom_errorbar(data=aux,
                  aes(x=log(1+lambda.v),
                      ymin=PMSE.mean - PMSE.sd,
                      ymax=PMSE.mean + PMSE.sd)) +
    geom_vline(xintercept = log(1+aux_min$lambda.v),
               linetype='dashed',
               color='red') +
    labs(x=expression(paste('log (1+',
                  lambda,
                  ')')))

  p2<-ggplot() +
    geom_point(data=aux,
               aes(y=PMSE.mean,
               x=df.v)) +
    geom_errorbar(data=aux,
                  aes(x=df.v,
                      ymin=PMSE.mean - PMSE.sd,
                      ymax=PMSE.mean + PMSE.sd)) +
    geom_vline(xintercept = aux_min$df.v,
               linetype='dashed',
               color='red') +
    labs(x=expression(paste('df(',
                  lambda,
```

```
                  ')')))
  multiplot(p1,p2,cols=2,title=title)

}

get_k_cv_metrics<-function(X,Y,lambda.v,k){
  n <- dim(X)[1] #N?mero files
  p <- dim(X)[2]
  XtX <- t(X)%*%X
  d2 <- eigen(XtX,symmetric = TRUE, only.values = TRUE)$values #Autovalors XtX
  n.lambdas<-length(lambda.v)

  PMSE.kCV <- matrix(-1,
                     ncol=n.lambdas,
                     nrow=k)

  #create random numbers from 1 to k
random_idx<-sample(1:k,n,replace=TRUE)
#if we are interested in the leave-one-out validation,
#where k = n (the number of observations)
#then we have to make sure that the index is not a random one
if(k==n){
  random_idx<-seq(from=1,
                  to=k,
                  by=1)
  print("Warning! Doint Leave-One-Out CV")
  name<-"LOO-CV"
}
#create vector of length of results
for (l in 1:n.lambdas){
  #work with the indexed lambda
  lambda <- lambda.v[l]
  l<-l
  for (i in 1:k){
    m.Y.i <- 0
    #fold that used as a trainingset
    X.i <- X[!random_idx==i,]
    Y.i <- Y[!random_idx==i]-m.Y.i
    #fold that used as a validationset (those which k equals their index)
    Xi <- X[random_idx==i,]
    Yi <- Y[random_idx==i]
    beta.i <- solve(t(X.i)%*%X.i + lambda*diag(1,p), tol = 1e-20) %*% t(X.i) %*% Y.i
    hat.Yi <- Xi %*% beta.i + m.Y.i
    #sum of the squared residuals
    PMSE.kCV[i,l] <- (sum(hat.Yi-Yi)^2)
  }
}
PMSE.kCV<-PMSE.kCV/n
PMSE.mean<-apply(PMSE.kCV,2,mean) #mean of the columns
PMSE.sd<-apply(PMSE.kCV,2,sd) #sd of the columns
return(cbind(PMSE.mean,PMSE.sd))
}
```

```r
data(BostonHousing2)
BostonHousing2$chas<-as.numeric(BostonHousing2$chas==1)

Y <- scale(BostonHousing2$cmedv, center=TRUE, scale=FALSE)
X <- scale(BostonHousing2[,7:19], center=TRUE, scale=TRUE)

mean.Y <- mean(Y)
mean.X <- apply(BostonHousing2[,7:18],2,mean)
sd.X <- apply(BostonHousing2[,7:18],2,sd)


XtX <- t(X)%*%X
d2 <- eigen(XtX,symmetric = TRUE, only.values = TRUE)$values #Autovalors XtX


lambda.max <- 1e5
n.lambdas <- 25
lambda.v <- exp(seq(0,log(lambda.max+1),length=n.lambdas))-1

df.v <- numeric(n.lambdas)
for (l in 1:n.lambdas){
  lambda <- lambda.v[l]
  df.v[l] <- sum(d2/(d2+lambda))
}

#cal definir una n. fixo folds a 10
n<-10
lasso.2 <- glmnet(X,Y, standardize=FALSE, intercept=FALSE, alpha=1)
cv.lasso.2 <- cv.glmnet(X,Y, standardize=FALSE, intercept=FALSE,nfolds=n)

# intercept: from the fitted model centering and scaling
mean.Y - sum((mean.X/sd.X) * coef(lasso.2,s=cv.lasso.2$lambda.1se)[-1])
```

```
## [1] -7.897793
```

```r
#coefficients: from the fitted model centering and scaling
coef(lasso.2,s=cv.lasso.2$lambda.1se)[-1] / sd.X
```

```
##  [1] -0.027089769  0.002152789  0.000000000  2.092922785 -5.770861997
##  [6]  4.243443141  0.000000000 -0.491181158  0.000000000  0.000000000
## [11] -0.793033593  0.006824735 -0.435935348
```

```r
op <- par(mfrow=c(2,2))
plot(cv.lasso.2)
plot(lasso.2,xvar="lambda")
abline(v=log(cv.lasso.2$lambda.min),col=2,lty=2)
abline(v=log(cv.lasso.2$lambda.1se),col=2,lty=2)
print(coef(lasso.2,s=cv.lasso.2$lambda.min))
```

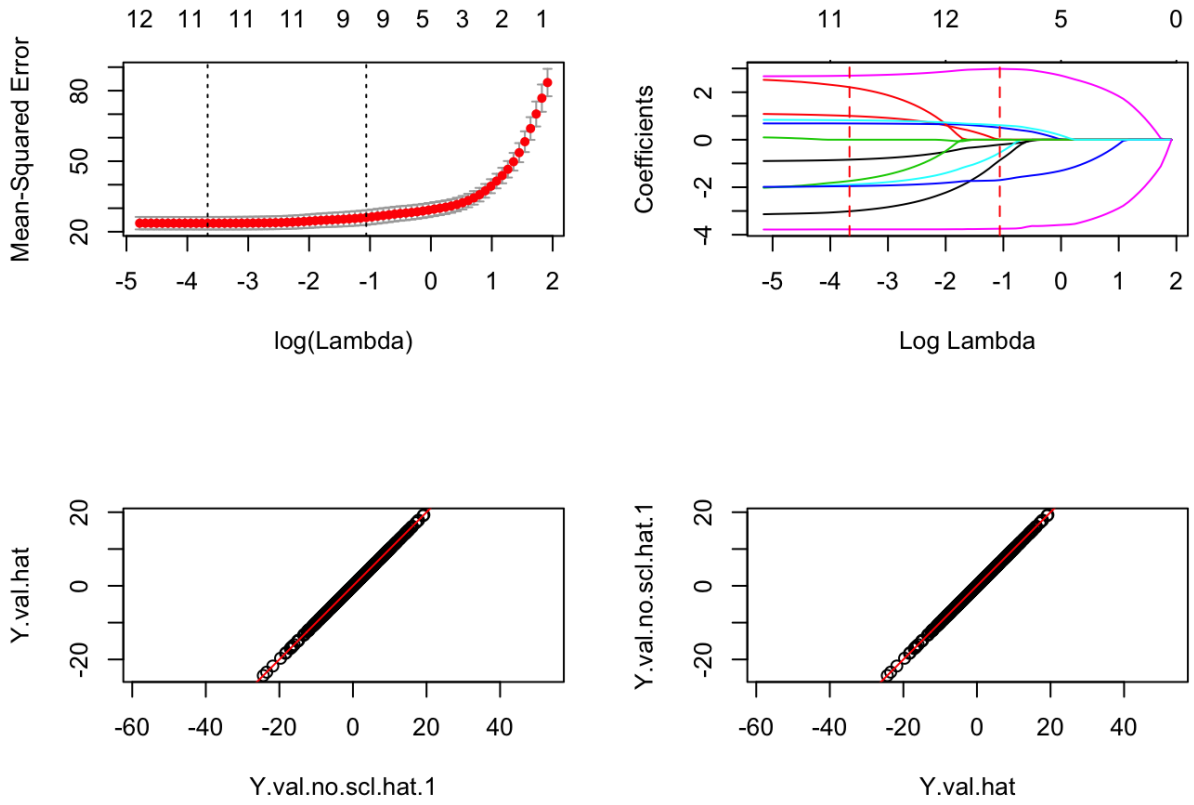
```
## 14 x 1 sparse Matrix of class "dgCMatrix"
##                        1
## (Intercept)  .
## crim        -0.88691444
## zn           1.06757542
## indus        0.07032071
```

```
## chas         0.68623721
## nox         -1.98146326
## rm           2.67474162
## age          .
## dis         -3.11443325
## rad          2.46563556
## tax         -1.96398995
## ptratio     -1.97949403
## b            0.83347530
## lstat       -3.77996771
```

```
print(coef(lasso.2,s=cv.lasso.2$lambda.1se))
```

```
## 14 x 1 sparse Matrix of class "dgCMatrix"
##                       1
## (Intercept)  .
## crim        -0.23301387
## zn           0.05020833
## indus        .
## chas         0.53158992
## nox         -0.66871407
## rm           2.98151590
## age          .
## dis         -1.03428514
## rad          .
## tax          .
## ptratio     -1.71687453
## b            0.62306323
## lstat       -3.74971756
```

```
# prediction in the validation set:
# b) from the fitted model centering and scaling
Y.val.hat <- predict(lasso.2,newx=X,s=cv.lasso.2$lambda.1se)
Y.val.no.scl.hat.1 <- mean.Y + Y.val.hat
plot(Y.val.no.scl.hat.1,Y.val.hat,asp=1)
abline(a=0,b=1,col=2)
plot(Y.val.hat,Y.val.no.scl.hat.1,asp=1)
abline(a=0,b=1,col=2)
```




```
par(op)
```

La regressio pel metode Lasso ens permet no nomes regularitzar (i aixi evitar l'overfitting) sino tambe seleccionar variables per tal d'arribar a un balanç entre simplicitat(com menys regressors millor) i encaix (tants regressors com es necessitin). -Si s'escolleix el lambda amb menor error CV (lambda.min), dues variables (INDUS, AGE) prenen valor zero. Els regressors ZN, CHAS, RM,RAD,B es troben positivament relacionats amb CMEDV, essent les majors d'aquestes relaciones les de RM i RAD. Els regressors CRIM, NOX, DIS, TAX, PTRATIO i LSTAT es troben negativament relacionats amb CMEDV, la major d'aquestes relacions negatives s?n les de LSTAT i DIS.

-Si s'escolleix el lambda.1se (valor de lambda que no difereix de lambda.min m?s que 1 desviaci? estandard) s'obte un model m?s simplificat, 5 variables prenen valor zero (INDUS, AGE, ZN, RAD, TAX). Els regressors CHAS, RM,B es troben positivament relacionats amb CMEDV, essent les majors d'aquestes relaciones les de RM i b. Els regressors CRIM, NOX, DIS, PTRATIO i LSTAT es troben negativament relacionats amb CMEDV, la major d'aquestes relacions negatives s?n les de LSTAT i PTRATIO.

At?s el model:

-Si seleccionem lambda.min el nombre d'habitacions (RM) i l'accesibilitat a les autopistes (RAD) son les variables que mes incrementen el preu de l'habitatge a Boston, alternativament el percentatge de poblacio d'estrat social baix (LSTAT) i la dist?ncia als centres de treball (DIS) son les variables que mes disminueixen el preu dels habitatges a Boston

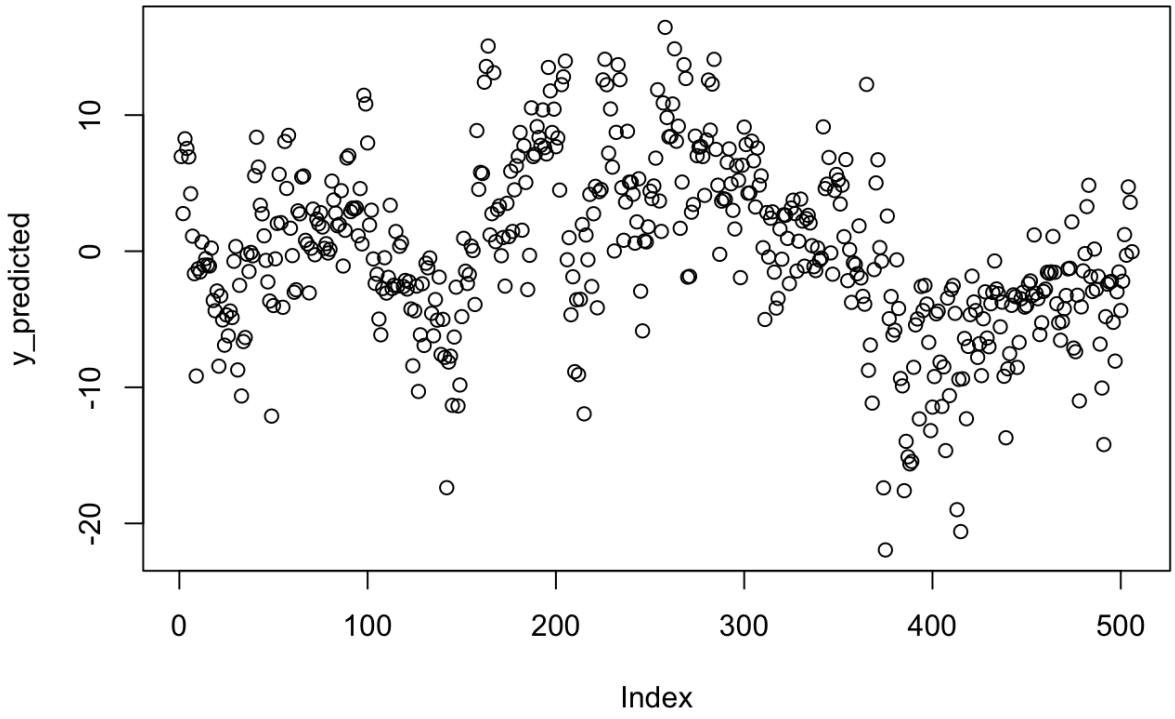
-Si seleccionem lambda.1se el nombre d'habitacions (RM) i un indicador de la quantitat de poblacio no negra (B)* son les variables que mes incrementen el preu de l'habitatge a Boston, alternativament el percentatge de poblaci? d'estrat social baix (LSTAT) i la quantitat d'alumnes per professor (PTRATIO) son les variables que mes disminueixen el preu dels habitatges a Boston.

*B es major com menor es el % de poblacio negra.

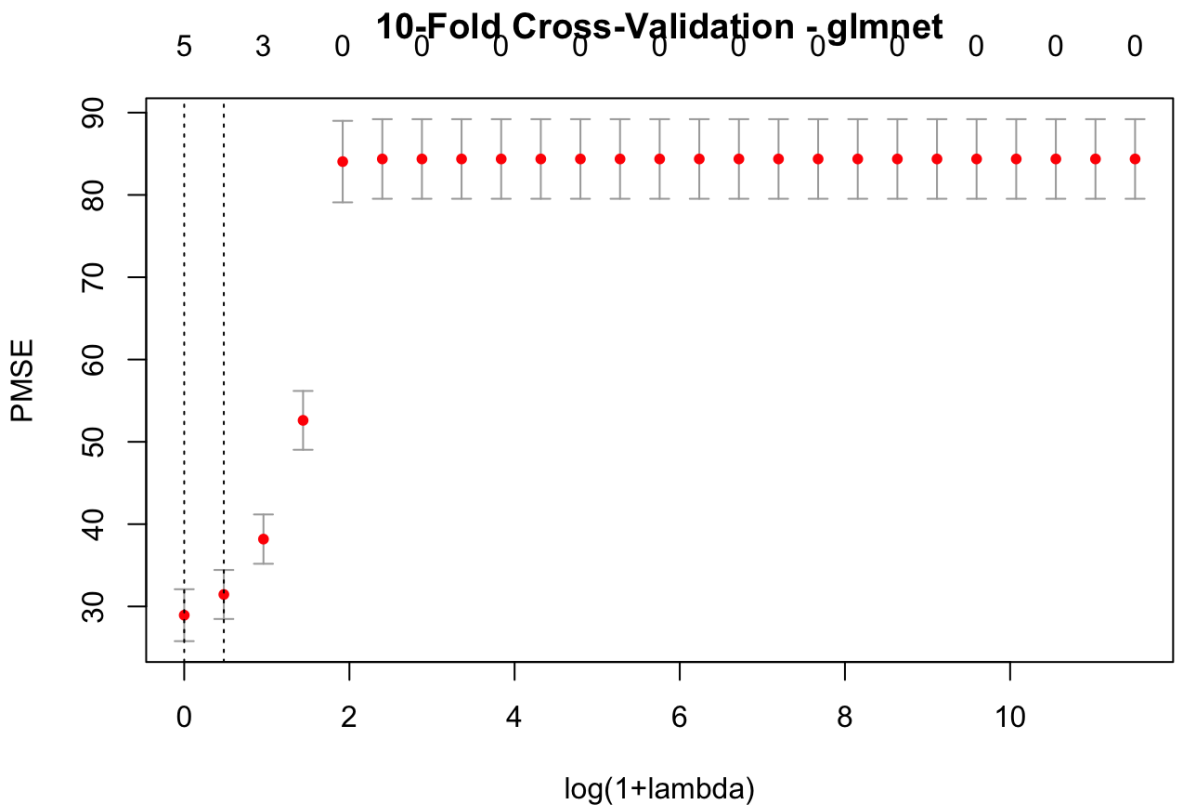
```
#EXERCICI 2
fit<-glmnet(X,Y,alpha=0,lambda=lambda.v)
cv_fit<-cv.glmnet(X,Y,nfolds=10,lambda=1+lambda.v)


opt_lambda<-cv_fit$lambda.min
opt_lambda #lambda.min = 0
```

```
## [1] 1
```

```
fit<-cv_fit$glmnet.fit
y_predicted<-predict(fit, s=opt_lambda, newx=X)
plot(y_predicted)
```

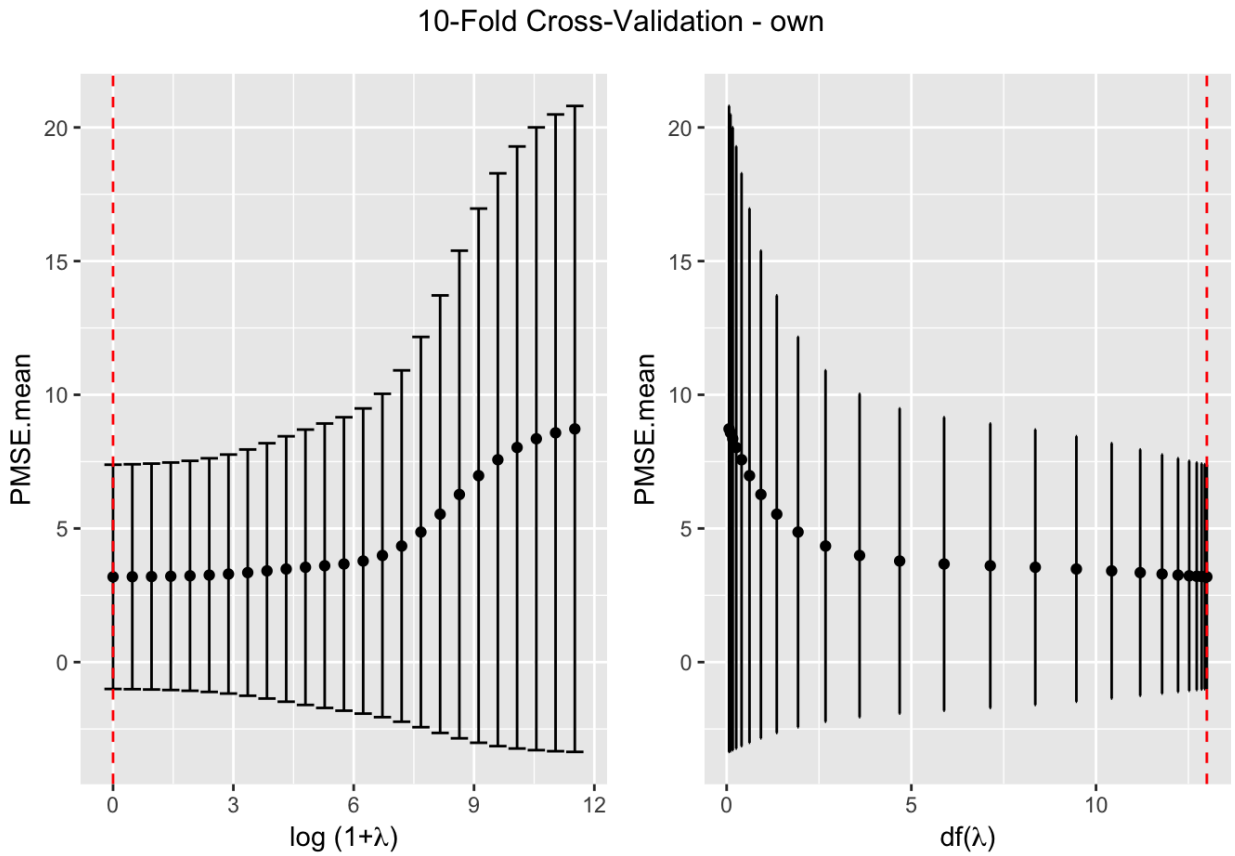



```
plot(cv_fit,main='10-Fold Cross-Validation - glmnet', xlab='log(1+lambda)', ylab="PMSE")
```




```
res_own_function<-get_k_cv_metrics(X,Y,lambda.v,10)
PMSE.10CV.own<-get_plot_cv(res_own_function,title='10-Fold Cross-Validation - own')
```

```
## Loading required package: grid
```



CV.glmnet genera un PMSE major que el resultant de la funci? generada per nosaltres mateixos.