Ridge Regression - Tarea 2

David Cardoner - Arnau Mercader

19 de febrero de 2018

Objetivo

El siguiente documento pretende ilustrar el comportamiento del modelo ridge usando distintos métodos de validación cruzada.

Train - Validation approach

En este primer apartado, se crea una función para escoger el parámetro de penalización λ que hace mínima la métrica **mean square error** en un conjunto de datos de validación. La idea es entrenar un conjunto de datos que llamaremos **train**, y ver su comportamiento en otro conjunto no entrenado. Dicho de otra manera, que no influye en la estimación de nuestro modelo. A este conjunto de datos le llamaremos **validate**. Se estudia la función con el dataset Boston con test de validación igual al 20%.

```
## Ridge cv-validate set

load("D:/Users/str_aml/Desktop/pers/MESIO - UPC/Statistical learning/Ridge - Lasso - Notes/boston.Rdata

## variables a estudiar
matches <- c("CRIM","ZN","INDUS","CHAS","NOX","RM","AGE","DIS","RAD","TAX","PTRATIO",
             "B","LSTAT","MEDV")

data <- boston.c[,matches]
head(data)
```

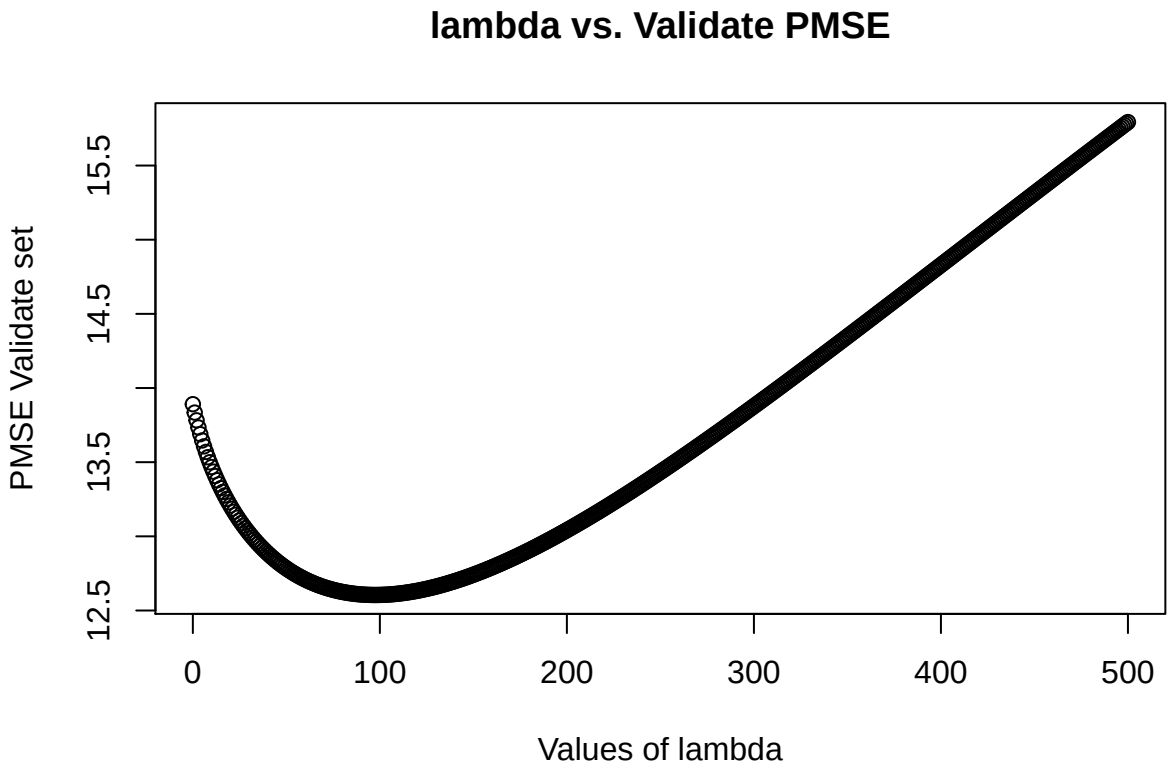
```
##      CRIM ZN INDUS CHAS   NOX    RM  AGE    DIS RAD TAX PTRATIO      B
## 1 0.00632 18  2.31    0 0.538 6.575 65.2 4.0900   1 296    15.3 396.90
## 2 0.02731  0  7.07    0 0.469 6.421 78.9 4.9671   2 242    17.8 396.90
## 3 0.02729  0  7.07    0 0.469 7.185 61.1 4.9671   2 242    17.8 392.83
## 4 0.03237  0  2.18    0 0.458 6.998 45.8 6.0622   3 222    18.7 394.63
## 5 0.06905  0  2.18    0 0.458 7.147 54.2 6.0622   3 222    18.7 396.90
## 6 0.02985  0  2.18    0 0.458 6.430 58.7 6.0622   3 222    18.7 394.12
##   LSTAT MEDV
## 1  4.98 24.0
## 2  9.14 21.6
## 3  4.03 34.7
## 4  2.94 33.4
## 5  5.33 36.2
## 6  5.21 28.7
```

```
set.seed(1002)

idx <- sample(x = nrow(data),size= 0.8*nrow(data))

x_train <- data[idx,-14]
y_train <- data[idx,14]

x_val <- data[-idx,-14]
y_val <- data[-idx,14]
```

```r
lambda.search <- seq(0,500,by=1)


CV_val_ridge <- function(x_train,y_train,x_val,y_val,lambda.v) {

# as.numeric all covariables X
x_train <- sapply(x_train,as.numeric)
x_val <- sapply(x_val,as.numeric)

# scale data & input data to -1
x_train[is.na(x_train)] <- -1
mean_train <- apply(x_train,2,mean)
sd_train <- apply(x_train,2,sd)
y_mean_train <- mean(y_train)

x_train <- as.matrix(scale(x_train, center=mean_train, scale=sd_train))
y_train <- as.matrix(scale(y_train, center=y_mean_train, scale=FALSE))

x_val <- as.matrix(scale(x_val, center=mean_train, scale=sd_train))
y_val <- as.matrix(scale(y_val, center=y_mean_train, scale=FALSE))

n_train <- dim(x_train)[1]
p_train <- dim(x_train)[2]

n_val <- dim(x_val)[1]



XtX <- t(x_train) %*% x_train
# d2 <- eigen(XtX,symmetric = TRUE, only.values = TRUE)$values
# lambda.v <- seq(0,5,by=1)

n.lambdas <- length(lambda.v)
beta.path <- matrix(0,nrow=n.lambdas, ncol=p_train)
# diag.H.lambda <- matrix(0,nrow=n.lambdas, ncol=n_train)

# To perform CV on validate set
PMSE.CV.validate.H.lambda <- numeric(n.lambdas)

for (l in 1:n.lambdas){
  lambda <- lambda.v[l]
  H.lambda.aux <- t(solve(XtX + lambda*diag(1,p_train))) %*% t(x_train)
  beta.path[l,] <-  H.lambda.aux %*% y_train
  # H.lambda <- X_train %*% H.lambda.aux
  # diag.H.lambda[l,] <- diag(H.lambda)
  hat.Y_val <- x_val %*% beta.path[l,]
  PMSE.CV.validate.H.lambda[l] <- sum((as.vector(y_val)-as.vector(hat.Y_val))^2)/n_val
}


min_lambda <- lambda.v[which.min(PMSE.CV.validate.H.lambda)]
cat("Resultados: \n")
cat("\n")
```

```r
cat("El lambda que hace mínimo el PMSE es: ",min_lambda,"\n")
cat("\n")
cat("El PMSE es: ",min(PMSE.CV.validate.H.lambda),"\n")
cat("El PRMSE es: ",sqrt(min(PMSE.CV.validate.H.lambda)),"\n")

plot(lambda.v,PMSE.CV.validate.H.lambda,main="lambda vs. Validate PMSE",xlab="Values of lambda",ylab="PM
}



CV_val_ridge(x_train=x_train,y_train=y_train,x_val=x_val,y_val=y_val,
             lambda.v=lambda.search)
```

```
## Resultados:
##
## El lambda que hace mínimo el PMSE es:  97
##
## El PMSE es:  12.60467
## El PRMSE es:  3.550306
```



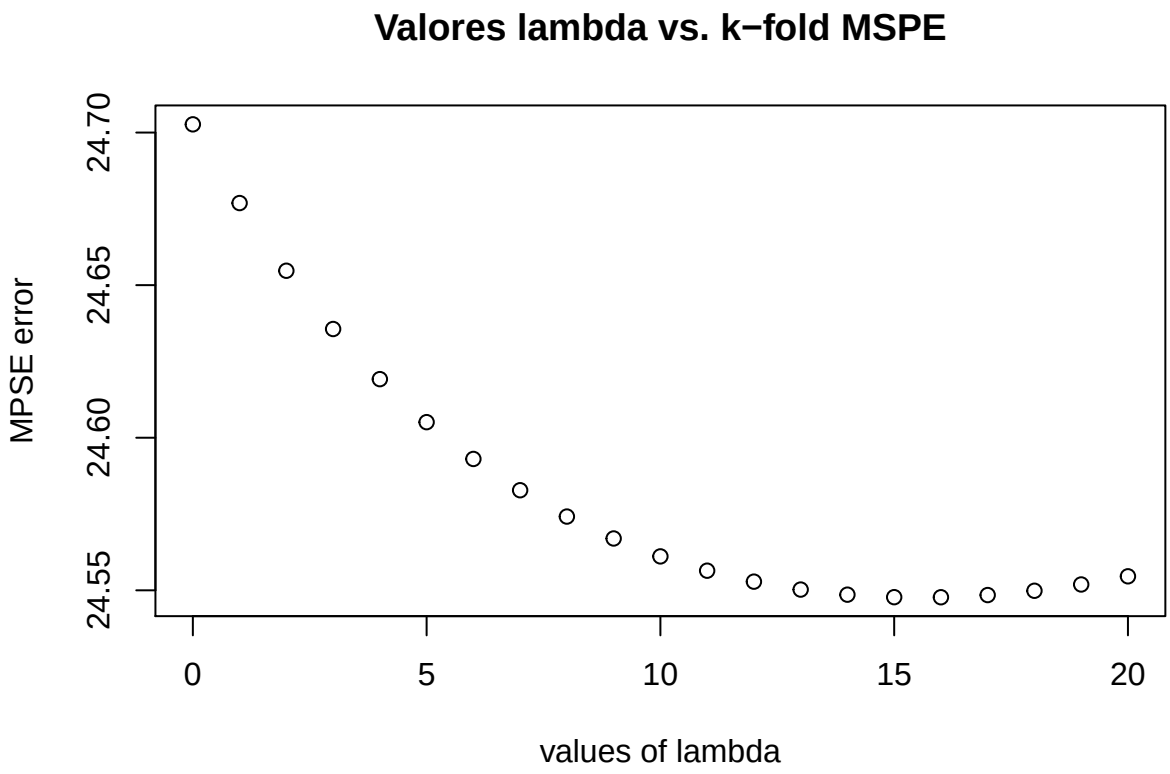
k fold validation

En este apartado, implementamos una función para encontrar el parámetro λ que minimiza la **k-fold cv**. El modelo se estima k veces, dejando k-1 partes para entrenar y una para validar el modelo. De esta manera, haciendo el promedio de todas las permutaciones de k obtenemos una idea de 'como es de bueno' el modelo en general. Se estudia la función con el dataset Boston.

```r
CV_kfolds_ridge <- function(X,y,folds,lambda.v,seed_value=NULL) {

  # as.numeric all covariables X
  X <- sapply(X,as.numeric)

  # scale data & input data to -1
  X[is.na(X)] <- -1
  mean_X <- apply(X,2,mean)
  sd_X <- apply(X,2,sd)
  y_mean <- mean(y)

  X <- as.matrix(scale(X, center=mean_X, scale=sd_X))
  y <- as.matrix(scale(y, center=y_mean, scale=FALSE))

  data <- cbind(y,X)

  if(is.null(seed_value)){1234}
  set.seed(seed_value)
  # permut data and make k folds
  data<-data[sample(nrow(data)),]


  # Create k size folds
  kfolds <- cut(seq(1,nrow(data)),breaks=folds,labels=FALSE)


  # result for lambda l
  cv_kfolds_mpse_lambda <- numeric(length(lambda.v))


  for (l in 1:length(lambda.v)){

    lambda <- lambda.v[l]
    cv_kfolds_mpse <- numeric(folds)

    for (i in 1:folds){

    idx <- which(kfolds==i)
    train <- data[-idx, ]
    test <- data[idx, ]
    n_train <- dim(train)[1]
    p_train <- dim(train)[2]-1
    n_test <- dim(test)[1]
    p_test <- dim(test)[2]-1


    beta.path <- matrix(0,nrow=1, ncol=p_train)


    XtX <- t(train[,-1]) %*% train[,-1]
    H.lambda.aux <- t(solve(XtX + lambda*diag(1,p_train))) %*% t(train[,-1])
    beta.path <-  H.lambda.aux %*% as.matrix(train[,1])
    hat.Y_val <- test[,-1] %*% beta.path
```

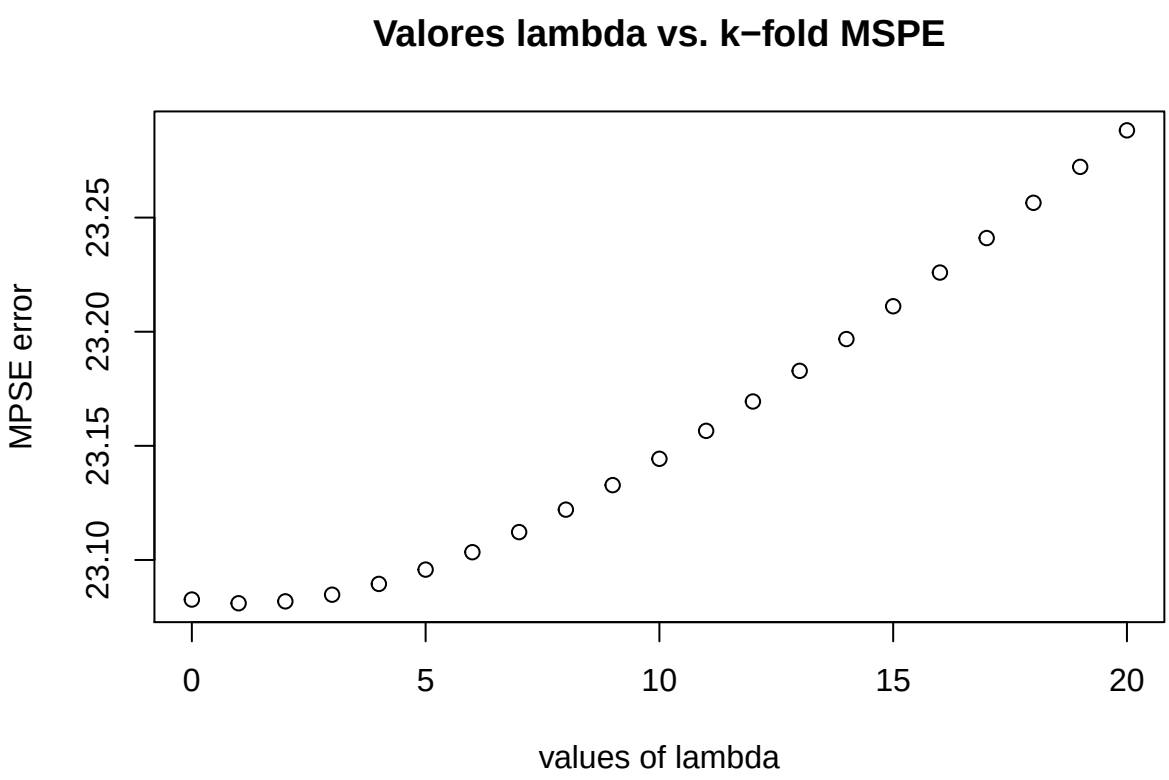
```
    cv_kfolds_mpse[i] <- sum((test[,1]-hat.Y_val)^2)/n_test
    }

    cv_kfolds_mpse_lambda[l] <- mean(cv_kfolds_mpse)
    cv_kfolds_mpse <- NULL
  }


  min_lambda <- lambda.v[which.min(cv_kfolds_mpse_lambda)]
  cat("Resultados considerando k=",folds,"folds: \n")
  cat("\n")
  cat("El lambda que hace mínimo el PMSE es: ",min_lambda,"\n")
  cat("\n")
  cat("El PMSE es: ",min(cv_kfolds_mpse_lambda) )

  plot(lambda.v,cv_kfolds_mpse_lambda,main = 'Valores lambda vs. k-fold MSPE',
       ylab='MPSE error',xlab="values of lambda")

}

load("D:/Users/str_aml/Desktop/pers/MESIO - UPC/Statistical learning/Ridge - Lasso - Notes/boston.Rdata")

## variables a estudiar

matches <- c("CRIM","ZN","INDUS","CHAS","NOX","RM","AGE","DIS","RAD","TAX","PTRATIO",
             "B","LSTAT","MEDV")

data <- boston.c[,matches]

lambda.search <- seq(0,20,by=1)

CV_kfolds_ridge(X=data[,1:13],y=data[,14],folds=5,lambda.v=lambda.search,seed_value = 12345)
```

```
## Resultados considerando k= 5 folds:
##
## El lambda que hace mínimo el PMSE es:  16
##
## El PMSE es:  24.54773
```




```
CV_kfolds_ridge(X=data[,1:13],y=data[,14],folds=10,lambda.v=lambda.search,seed_value = 100)
```

```
## Resultados considerando k= 10 folds:
##
## El lambda que hace mínimo el PMSE es:  1
##
## El PMSE es:  23.08104
```



Resultados

La estimación mediante validación es mejor, sin embargo escogemos solo un conjunto de validación, por lo que es un método mucho más optimista que la métrica que obtenemos mediante k-fold cv. Lo que no acabamos de entender, es los valores λ, ya que entre ambos enfoques dista mucho el **grid** usado. Quizás tenemos algun error en la implementación o calculo de la métrica.

Pruebas en prostate dataset

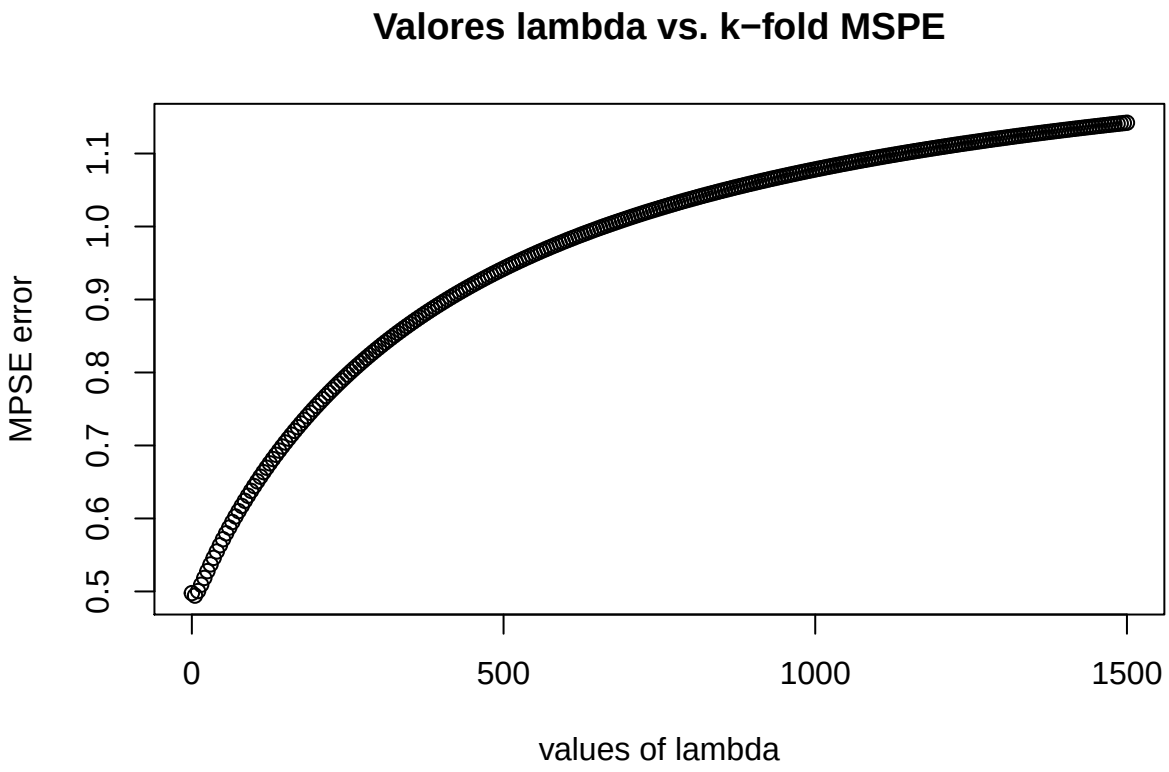
En este apartado, usaremos las 4 funciones de validación cruzada para encontrar el parámetro λ que minimza la métrica **mean square error**.

```
##       lcavol  lweight age      lbph svi       lcp gleason pgg45       lpsa
## 1 -0.5798185 2.769459  50 -1.386294   0 -1.386294       6     0 -0.4307829
## 2 -0.9942523 3.319626  58 -1.386294   0 -1.386294       6     0 -0.1625189
## 3 -0.5108256 2.691243  74 -1.386294   0 -1.386294       7    20 -0.1625189
## 4 -1.2039728 3.282789  58 -1.386294   0 -1.386294       6     0 -0.1625189
## 5  0.7514161 3.432373  62 -1.386294   0 -1.386294       6     0  0.3715636
## 6 -1.0498221 3.228826  50 -1.386294   0 -1.386294       6     0  0.7654678
##   train
## 1  TRUE
## 2  TRUE
## 3  TRUE
## 4  TRUE
## 5  TRUE
## 6  TRUE
```

k=5 fold validation

```
prostate$train <- NULL
data <- prostate
lambda.search <- seq(0,1500,by=5)
CV_kfolds_ridge(X=data[,1:8],y=data[,9],folds=5,lambda.v=lambda.search,seed_value = 12345)
```

```
## Resultados considerando k= 5 folds:
##
## El lambda que hace mínimo el PMSE es:  5
##
## El PMSE es:  0.494274
```



k=10 fold validation

```
prostate$train <- NULL
data <- prostate
lambda.search <- seq(0,1500,by=5)
CV_kfolds_ridge(X=data[,1:8],y=data[,9],folds=10,lambda.v=lambda.search,seed_value = 12345)
```

```
## Resultados considerando k= 10 folds:
##
## El lambda que hace mínimo el PMSE es:  5
##
## El PMSE es:  0.5041859
```