Assignment 1 Statistical Learning

Miguel Benítez, Iker Caballero, Alejandro Peraza

2023-02-23

Exercise 1. Choosing the penalization parameter λ

Function that implements $MSPE_{val}(\lambda)$

```
MSPE_val <- function(X, Y, X_val, Y_val, lambdas, plot=TRUE) {
  # Creating empty vector for MSPE and empty matrix for betas
  MSPE.val <- numeric(length(lambdas))
  beta_path <- matrix(0, nrow=length(lambdas), ncol=ncol(X))
  # Computing Ridge Regression betas for each value lambda
  tX <- t(X)
  XtX <- tX%*%X
  for (l in seq_along(lambdas)){
    lambda <- lambdas[l]
    H_lambda_aux <- solve(XtX + lambda * diag(1,ncol(X))) %*% tX
    beta_path[l,] <-  H_lambda_aux %*% Y
  }
  # Computing MSPE for each lambda value
  for (l in seq_along(lambdas)) {
    lambda <- lambdas[l]
    Y_hat_val <- X_val %*% beta_path[l,]
    MSPE.val[l] <- sum((Y_val - Y_hat_val) ^ 2) / nrow(X_val)
  }
  # Plotting the MSE for each lambda and returning it
  if (plot) {
    plot(lambdas, MSPE.val)
    plot(log(1+lambdas) - 1, MSPE.val)
  }
  return(MSPE.val)
}
```

Function that implements $MSPE_{k-CV}(\lambda)$

```
#' Warning -> It is necessary that nrows(X)%%k = 0
MSPE_kCV = function(X,Y,k,lambdas,seed){
  set.seed(seed)
  # Splitting the data into 10 groups
  indexes = matrix(0,nrow=dim(X)[1]/k,ncol=k) #Each column is one sub-sample
  numbers = 1:dim(X)[1]
  for (i in 1:k){
    ind = sample(numbers,dim(X)[1]/k,replace = FALSE)
    indexes[,i] = ind
    numbers = setdiff(numbers,ind)
```

```
  }
  # Creating a matrix of MSPE
  MSPE_matrix = matrix(0,nrow=length(lambdas),ncol=k)
  #Each column is the lambdas-MSPE vector associated to one of the k repetitions
  for (i in 1:k){
    training_X = X[indexes[,-i],]
    test_X = X[indexes[,-i],]
    training_Y = Y[indexes[,-i]]
    test_Y = Y[indexes[,-i]]
    MSPE_matrix[,i]=MSPE_val(training_X,training_Y,test_X,test_Y,lambdas=lambdas, plot=FALSE)
  }
  # Computing the MSPE associated to each lambda and plotting it against lambdas
  lambdas_MSPE = apply(MSPE_matrix, MARGIN = 1, FUN = mean)
  plot(log(1+lambdas), lambdas_MSPE)
  return(lambdas_MSPE)
}
```

Use the prostate data and compare your results.

We create two functions, one computes the values of MSPE according to Leave-One-Out Cross Validation method, for each value of lambda. The other does the same but using the Generalized Cross Validation MSPE formula presented in the course slides. (Code in Annex 1)

Using the formulas presented in this section and the preceding one, we compute which one of the possible lambda values results in a lower MSPE, according to each method. (For the sake of conciseness, we will not print the plots)

```
max_lambda <- 1e5
n_lambdas <- 25
lambdas <- exp(seq(0,log(max_lambda + 1), length=n_lambdas)) - 1
```

```
# ARBITRARY TRAINING SET RESULT
lambdas[which.min(MSPE_val(prostate.X_train, prostate.Y_train,
                           prostate.X_val, prostate.Y_val,
                           lambdas))]
```

```
## [1] 27.72993
```

```
# 5-FOLD CROSS VALIDATION RESULT
lambdas[which.min(MSPE_kCV(prostate.X, prostate.Y, 5, lambdas, seed = 467))]
```

```
## [1] 0
```

```
# 10-FOLD CROSS VALIDATION RESULT
lambdas[which.min(MSPE_kCV(prostate.X, prostate.Y, 10, lambdas, seed = 467))]
```

```
## [1] 0
```

```
# LEAVE ONE OUT CROSS VALIDATION
lambdas[which.min(MSPE_LOCV(prostate.X, prostate.Y, lambdas, seed = 467))]
```

```
## [1] 5.812932
```

```
# GVC
lambdas[which.min(MSPE_GCV(prostate.X, prostate.Y, lambdas))]
```

```
## [1] 5.812932
```

In the case of the arbitrary training result, the optimal lambda is 27.72993. For the 5 and 10-fold Cross

validations, we got a lambda of 0. Using Leave-One-Out Cross Validation and Generalized Cross Validation, we got 5.812932 as the optimal lambda value. The optimal lambda for each method may differ, but we cannot be certain that one method works better than others.

Ridge regresssion for the Boston Housing data

Use the Boston Housing Data

Given that this exercise asks us to estimate a ridge regression using the Boston Housing data and, then, interpreting the parameters, we need to implement the same methods as before regarding estimation and parameter selection. The first step is to do some data handling in order to divide the data into training and validation sets and to create the X input matrix (with the 13 variables) and the Y vector. Hence, we first import the data and get the relevant data from the data frame.

Now we can divide the data:

```
# We create a vector of possible lambdas
num_lambdas <- 15
vec_lambdas <- exp(seq(0,log(1e7 + 1),length=num_lambdas))-1
```

We then create the X^tX matrix and a matrix for both the betas estimated for each lambda and the lambdas.

We use the function defined earlier "MSPE_val" in order to choose λ:

```
vec_lambdas[which.min(MSPE_val(x_train,y_train,x_val,y_val,vec_lambdas))]
```

```
## [1] 30.62278
```

In this case, $\lambda^* = 30.62278$ is the best penalty parameter for this data, so the final model we will deal has the following parameters:

```
final_betas <- beta[4,]
mat_betas <- matrix(c(final_betas),ncol=1,nrow=13)
rownames(mat_betas) <- colnames(x_train)
print(t(round(mat_betas,3)))
```

```
##        CRIM    ZN  INDUS    NOX    RM  AGE    DIS   RAD    TAX PTRATIO     B
## [1,] -0.861 0.845 -0.489 -1.035 2.413 -0.1 -2.431 1.384 -0.855    -1.8 0.646
##       LSTAT  CHAS
## [1,] -3.705 1.836
```

As expected, most of the coefficients obtained by the ridge regression are lower in absolute value than the ones that would have been obtained using OLS. However, one can clearly see that the direction or sign of the coefficients is not changed, so the direction of the effect the explanatory variables have on Y is still the same.

We can interpret the coefficients through their sign, as the data has been scaled and the magnitude of the coefficients is not fully interpretable as a marginal effect. Here, we can see that variables that represent negative factors such as CRIM (crime) or NOX (nitric-oxide concentrarion) have a negative effect on the median value of owner-occupied homes in Boston. Other factors such as INDUS, AGE, DIS, TAX, PTRATIO or LSTAT have also negative effects even though these are not negative factors per se. High distance from employment areas, prevalent old population, high taxes on properties, low teachers per student ratio and presence of low income households is not attractive for new buyers, so this reduces the median value. In contrast, ZN, RM and RAD have a positive effect because they are related with purchase opportunities, space in properties and access to other zones, while B and CHAR are also positive but there is not a clear-cut explanation of why (there are many possible).

Even though the magnitude is not fully interpretable, we can say something about the absolute value of coefficients. For example, it seems that LSTAT and DIS are the most relevant factors that can reduce the price of houses, while rooms in property (RM) and access (RAD) seem the most important factors that increase the price.

Annex 1: code

L-O-O and GCV

```
# MSPE Leave One Out Cross Validation
MSPE_LOCV = function(X,Y,lambdas,seed){
  set.seed(seed)
  # Creating a matrix of MSPE
  MSPE_matrix = matrix(0,nrow=length(lambdas),ncol=dim(X)[1])
  for (i in 1:dim(X)[1]){
    training_X = X[-i,]
    test_X = X[i,]
    training_Y = Y[-i]
    test_Y = Y[i]
    for (j in 1:length(lambdas)){
      beta = solve(t(training_X)%*%training_X+lambdas[j]*diag(1,dim(training_X)[2]))%*%t(training_X)%*%
      ypred = test_X%*%beta
      MSPE_matrix[j,i] = (ypred - test_Y)**2
    }
  }
  # Computing the MSPE associated to each lambda and plotting it against lambdas
  lambdas_MSPE = apply(MSPE_matrix, MARGIN = 1,FUN = mean)
  return(lambdas_MSPE)
}
# MSPE GVC
MSPE_GCV = function(X,Y,lambdas){
  # I create an empty vector for the MSPE for each lambda
  Mspe.gvc = numeric(length(lambdas))
  # I compute the value for each lambda
  for (i in 1:length(lambdas)){
    hatm = X %*% solve(t(X)%*%X+lambdas[i]*diag(1,dim(X)[2])) %*% t(X)
    nu = sum(diag(hatm))
    Mspe.gvc[i] = mean(((Y-hatm%*%Y)/(1-nu/dim(X)[1]))**2)
  }
  return(Mspe.gvc)
}
```

Prostate data

```
prostate <- read.table("prostate_data.txt", header=TRUE, row.names = 1)

prostate.X <- scale(as.matrix(prostate[, 1:8]), center=TRUE, scale=TRUE)
prostate.Y <- scale(prostate$lpsa, center=TRUE, scale=FALSE)
# We create the training and validation sets
train_sample <- which(prostate$train == TRUE)
validation_sample <- which(prostate$train == FALSE)

prostate.X_train <- scale(as.matrix(prostate[train_sample, 1:8]), center=TRUE, scale=TRUE)
prostate.Y_train <- scale(prostate$lpsa[train_sample], center=TRUE, scale=FALSE)

prostate.X_val <- scale(as.matrix(prostate[validation_sample, 1:8]), center=TRUE, scale=TRUE)
prostate.Y_val <- scale(prostate$lpsa[validation_sample], center=TRUE, scale=FALSE)
```

Boston Housing Data

```
load("boston.Rdata")

# From the boston.c data frame, we just use the 14 variables we will work with
data <- cbind(boston.c[,6],boston.c[,8:20])
colnames(data) <- c("MEDV",names(boston.c[,8:20]))

# We now show that the data iss correctly extracted
head(data,5)

# As there is no indicator for training and validation, we extract a sample of 70% of total observation
set.seed(123)
train_ind <- sample(nrow(data),nrow(data)*0.7)
train_s <- data[train_ind,]
val_s <- data[-train_ind,]

# We then create Y and X for both sets of data
y_train <- scale(train_s[,"MEDV"],center=T,scale=F)
y_val <- scale(val_s[,"MEDV"],center=T,scale=F)

# We keep the factor out from the scaling procedure and then we add it again
x_train <- scale(as.matrix(train_s[,-c(1,5)],center=T,scale=T))
x_train <- cbind(x_train,train_s[,5])
colnames(x_train)[13] <- c("CHAS")

x_val <- scale(as.matrix(val_s[,-c(1,5)],center=T,scale=T))
x_val <- cbind(x_val,val_s[,5])
colnames(x_val)[13] <- c("CHAS")

# Because the factor variable levels have changed (from 0 & 1 to 1 & 2) while reincorporating the facto
x_train[,13][x_train[,13]==1] <- 0
x_train[,13][x_train[,13]==2] <- 1

x_val[,13][x_val[,13]==1] <- 0
x_val[,13][x_val[,13]==2] <- 1

# We save some parameters into variables so we can use them later
n <- dim(x_train)[1]
p <- dim(x_train)[2]
XX_t <- t(x_train)%*%x_train
beta <- matrix(0,nrow=num_lambdas, ncol=p)
diag_H_lambdas <- matrix(0,nrow=num_lambdas, ncol=n)
for (i in 1:num_lambdas){
  lambda <- vec_lambdas[i]
  H_prev <- t(solve(XX_t+lambda*diag(1,p)))%*%t(x_train)
  beta[i,] <- H_prev%*%y_train
  H_lambda <- x_train%*%H_prev
  diag_H_lambdas[i,] <- diag(H_lambda)
}
```