Assignment 1: Ridge Regression

Wanchang Zhang, Lavinia Hriscu, Victor Jimenez

Contents

In this assignment we will fit our data to a multiple linear regression model by means of a ridge regression. Considering n pairs of observations (x_i, y_i), where $x_i \in \mathbb{R}^p$ and $y_i \in \mathbb{R}$, the linear model is the following:

$$y_i = \beta_0 + \sum_{j=1}^{p} x_{ij}\beta_j + \epsilon_i$$

where ϵ_i account for white noise and $\beta \in \mathbb{R}^{p+1}$ is the vector of coefficients of the model. By fitting the model, estimators for β and σ^2 will be obtained and the object variables y will be predicted from explanatory variables x as $\hat{y}_i = x_i^T \hat{\beta}$.

In general (OLS), the estimation of the coefficients is achieved by minimizing the square of the prediction error ϵ. It can be proven (Gauss-Markov) that this method provides the MVUE estimator of β. In the case of ridge regression, an extra penalization term is added in order to grant the numerical stability of the method by imposing a shrinkage of the parameter vector $\hat{\beta}$. For the cases where $p > n$, for example, the penalization term allows the minimization problem to reach an optimum value and provide the "shortest' vector possible. The estimation of the coefficients $\hat{\beta}_{ridge}$ is obtained as follows:

$$\hat{\beta}_{ridge} = \arg\min_{\beta} \left\{ \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right\}$$

The greater the value of the penalization parameter $\lambda \geq 0$, the greater the amount of shrinkage of the estimator $\hat{\beta}_{ridge}$. If we express it in matrix notation is easy to see that the problem has an explicit solution:

$$\hat{y}_{ridge} = X^T \hat{\beta}_{ridge} = X^T \left(X^T X + \lambda I_p \right)^{-1} X^T y = H_\lambda y$$

where H_λ is the hat matrix for ridge regression.

The main goal of this assignment is to implement different methods for the choice of λ, as lower values will yield a more complex model and viceversa. The complexity of various models can be compared with the number of degrees of freedom of the model, that in the case of ridge regression depends on λ according to the following expression:

$$df(\lambda) = Trace(H_\lambda) = \sum_{j=1}^{p} \frac{\Lambda_j(X)^2}{\Lambda_j(X)^2 + \lambda}$$

where $\Lambda(X)$ is the set of eigenvalues of the matrix X_T of the training set $\mathcal{Z}$. The function that provides $df(\lambda)$ is the following:

```
df.fun = function(X, lambda.v){
  n.lambdas = length(lambda.v)
  XtX <- t(X)%*%X # X'X
  XtX.vaps <- eigen(XtX,symmetric = TRUE, only.values = TRUE)$values
  df.lambda = matrix(0,nrow=n.lambdas, ncol=1)
  for (l in 1:n.lambdas){
    lambda = lambda.v[l]
    df.lambda[l,] = sum(XtX.vaps/(XtX.vaps+lambda))
  }
  return(df.lambda)
}
```

1. Choosing the penalization parameter *lambda*

As follows from the previous development, the λ chosen will be that which minimizes the predictive error of the model, i.e. the mean square error committed when providing an estimation of new observations. That can be defined as:

$$PMSE(\hat{\beta}_{ridge}) = \mathbb{E}_{\mathcal{Z},X_V}\left[(Y_V - X_V\hat{\beta}_{ridge})^2\right] \approx \frac{1}{n_V}\sum_{i=1}^{n_V}\left(y_{V_i} - x_{V_i}^T\hat{\beta}_{ridge}\right)^2$$

where X_V is the matrix of observations from the validation set $\mathcal{X}$, which is independent from the set of observations $\mathcal{Z}$ that have been used to fit the model, known as training set.

1.1 *lambda* choice with a validation set

In this first case our data will be divided into a training set $\mathcal{Z}$ and a validation set $\mathcal{X}$ according to the binary variable `train` of the dataset.

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

# Generate vector of lambdas
lambda.v = exp(seq(0,log(1e5+1),length=25))-1

# Divide into training and validation sets
training.sample = which(prostate$train==TRUE)
validation.sample = which(prostate$train==FALSE)

# Obtain observation matrices. X scaled and Y centered.
Y = scale(prostate$lpsa[training.sample], center=TRUE, scale=FALSE)
X = scale( as.matrix(prostate[training.sample,1:8]), center=TRUE, scale=TRUE)
Yv = scale(prostate$lpsa[validation.sample], center=TRUE, scale=FALSE)
Xv = scale( as.matrix(prostate[validation.sample,1:8]), center=TRUE, scale=TRUE)
```

The function that computes the predictive error $PMSE(\lambda)$ between the predicted $X_V\beta$ and real Y_v target values of the validation set $\mathcal{X}$ is the following. The error is computed by fitting observations in the training set $\mathcal{Z}$ into the model for each value of λ.

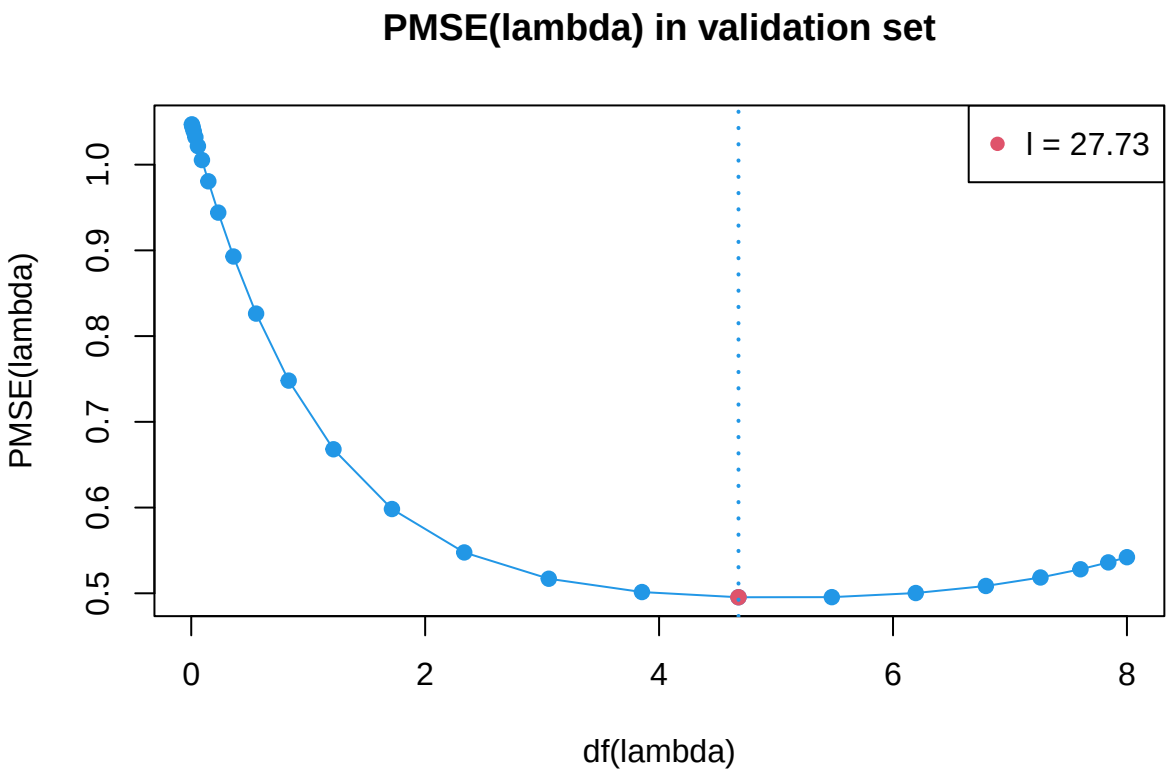
```
pmse_val <- function(X, Y, Xv, Yv, lambda.v) {
  nv <- dim(Xv)[1]
  p <- dim(X)[2]

  # Calculating the coefficients beta for each lambda
  pmse.lambda = matrix(0,nrow=length(lambda.v), ncol=1)
  for (l in 1:length(lambda.v)){
    beta.lambda = t(solve(t(X)%*%X + lambda.v[l]*diag(1,p))) %*% t(X) %*% Y

    # Now we can calculate the PMSE for lambda
    error.lambda = Yv - Xv %*% beta.lambda
    pmse.lambda[l,] = (1/nv) * t(error.lambda) %*% error.lambda
  }
  return(pmse.lambda)
}
```

We can plot $PMSE(\lambda)$ vs $df(\lambda)$ and identify the penalization coefficient that minimizes the error by calling the following function. When `bool_log` is `TRUE`, then the PMSE is presented against $log(1+\lambda)-1)$ instead.

```
plot.pmse.df = function(df.lambda, pmse.lambda, title, bool_log){
  if (bool_log == TRUE){
    df.lambda = log(1+lambda.v)-1
  }
  plot(range(df.lambda), range(pmse.lambda),type="n",xlab="df(lambda)",ylab="PMSE(lambda)",
       main=title)
  lines(df.lambda, pmse.lambda, col=4)
  points(df.lambda, pmse.lambda, pch=19,cex=1,col=4)
  posmin = which.min(pmse.lambda)
  points(df.lambda[posmin], pmse.lambda[posmin], pch=19,cex=1,col=2)
  abline(v=df.lambda[posmin],col=4,lty=3,lwd=2)

  lambda.text = paste("l = ", round(lambda.v[posmin],2),sep="")
  legend('topright', legend=c(lambda.text),
         pch = c(16), col = c(2))
}
```

Now we can obtain the plots associated with the implementation of `pmse_val` function into the `prostate` database.

```
df.lambda = df.fun(X, lambda.v)
pmse.lambda = pmse_val(X, Y, Xv, Yv, lambda.v)
plot.pmse.df(df.lambda, pmse.lambda, "PMSE(lambda) in validation set", FALSE)
```



1.2 *lambda* choice with k-fold cross-validation

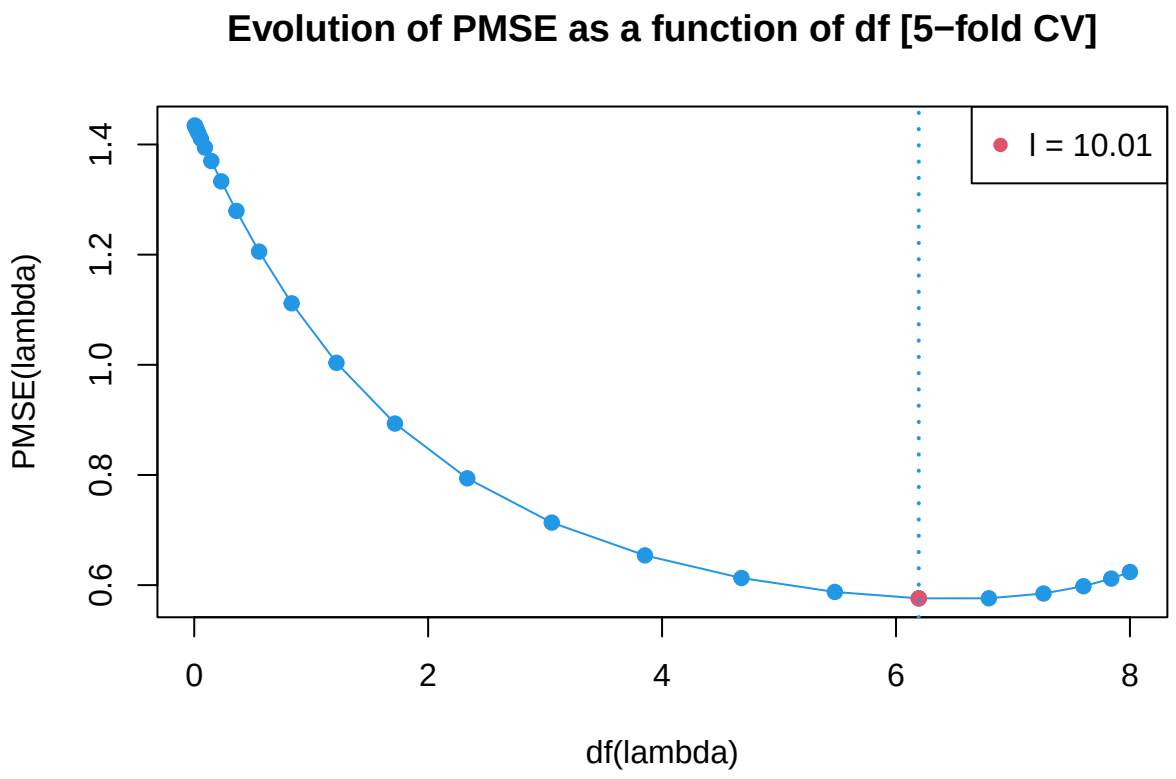
In some cases where the number of observations is small or in large scale data applications it is required that the whole dataset is used to fit the model (i.e. no validation set is obtained). In these cases, the computation of PMSE can be performed by cross-validation, by randomly dividing the data in k sets and fitting the model k times, each time leaving the i-th set out as validation set. This requires fitting the model k times for each λ.

```
pmse_kCV = function(X, Y, lambda.v, k){
  n.lambdas = length(lambda.v)
  p = dim(X)[2]

  # Split the data into k-folds
  set.seed(123) # for reproducibility
  folds = sample(rep(1:k, length.out = dim(X)[1])) # assign a set to each obs.
  pmse.lambda = matrix(0,nrow=n.lambdas, ncol=1)
  for (i in 1:k) {
    # Get the training and validation sets for this fold
    x_train <- X[folds != i,]
    y_train <- Y[folds != i]
    x_val <- X[folds == i,]
    y_val <- Y[folds == i]
    nv = nrow(x_val)

    # Now we fit the model for each lambda
    for (l in 1:n.lambdas){
      beta.lambda = t(solve(t(x_train)%*%x_train + lambda.v[l]*diag(1,p))) %*% t(x_train) %*% y_train
```

```
      # Now we can calculate the PMSE for lambda
      error.lambda = y_val - x_val %*% beta.lambda
      pmse.lambda[l] = pmse.lambda[l] + t(error.lambda)%*%error.lambda
    }
  }
  # Now we divide by n
  pmse.cv = pmse.lambda/dim(X)[1]
  return(pmse.cv)
}
```

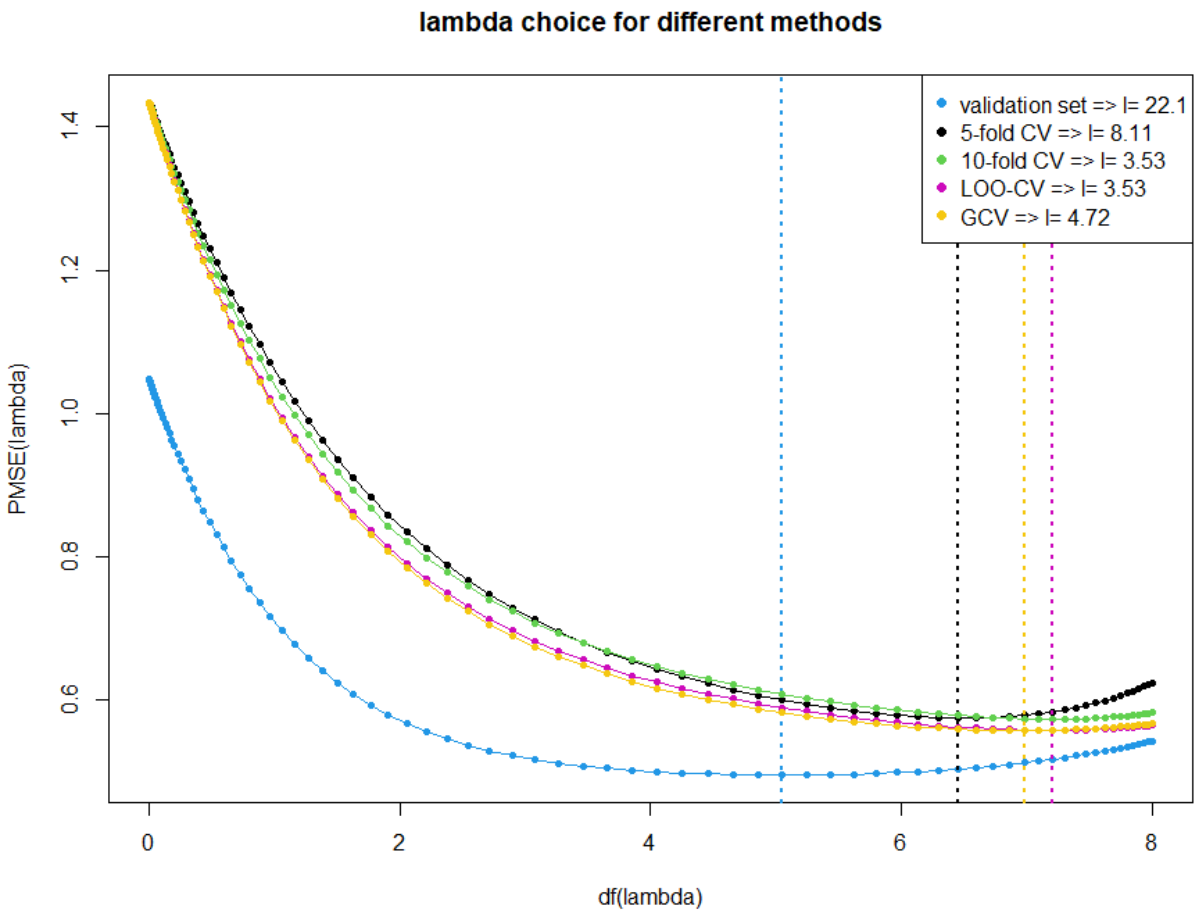
We can plot the results on the `prostate` database calling `plot.pmse.df` as well:

```
k = 5
pmse.cv = pmse_kCV(X, Y, lambda.v, k)

# Plot
title = paste('Evolution of PMSE as a function of df [', k, sep="")
title = paste(title, '-fold CV]', sep='')
plot.pmse.df(df.lambda, pmse.cv, title, FALSE)
```



1.3 Implementation of the routines to `prostate` dataset

We will implement the previous routines choose λ according to the behaviour of $PMSE(\lambda)$ in the validation set and also in 5-fold and 10-fold CV. The results will be compared with those obtained with LOO-CV and GCV. The code used to generate the plots can be found in the source code file `ridge_source.R`.

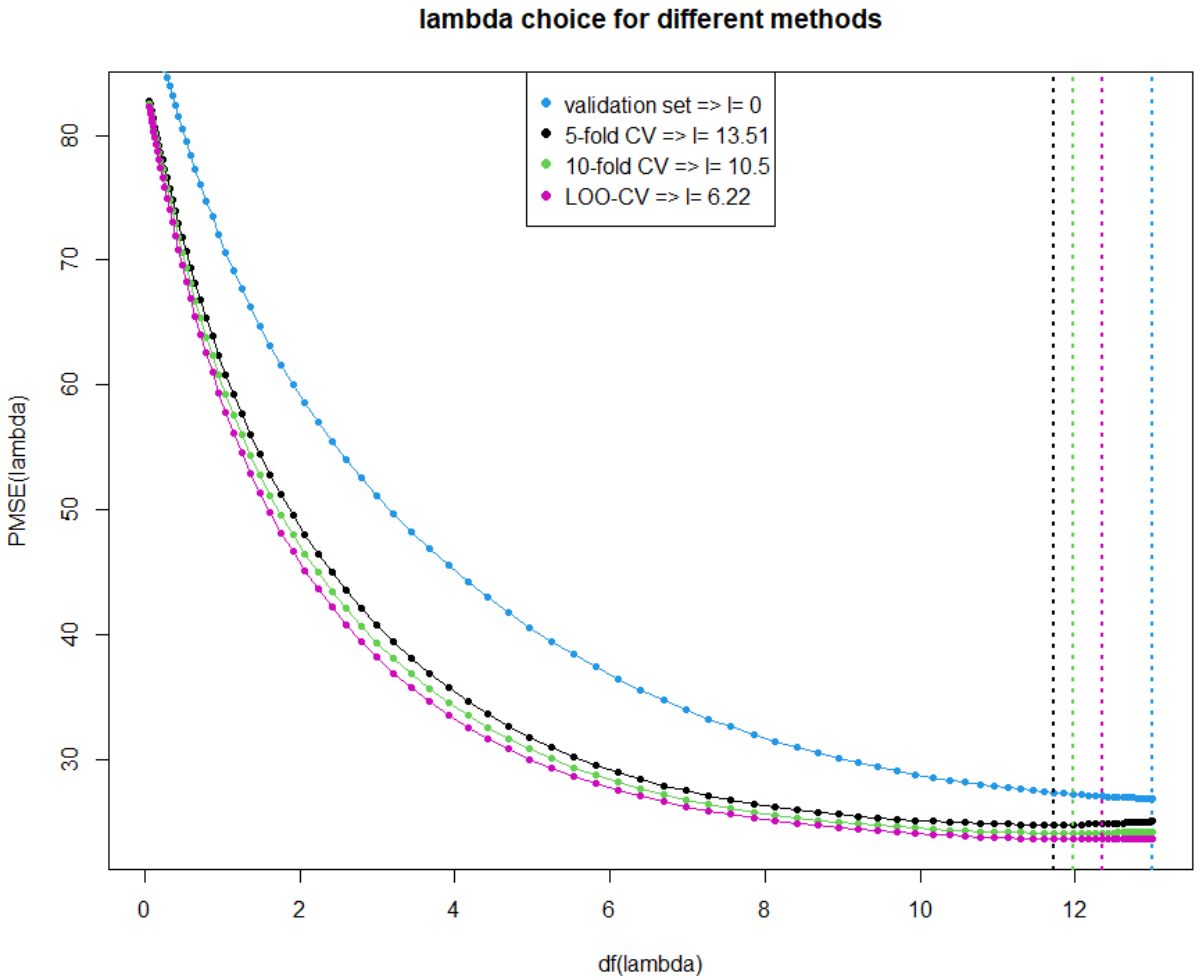


2. Ridge regression for `boston` housing data

The functions used in the first exercise (validation set, LOO cross-validation, K-fold cross validation) are now applied to the Boston dataset provided.

- CRIM: per capita crime rate by town ,
- ZN: proportion of residential land zoned for lots over 25,000 sq.ft. ,
- INDUS: proportion of non-retail business acres per town ,
- CHAS: Charles River dummy variable (= 1 if tract bounds river; 0 otherwise) ,
- NOX: nitric oxides concentration (parts per 10 million) ,
- RM: average number of rooms per dwelling ,
- AGE: proportion of owner-occupied units built prior to 1940 ,
- DIS: weighted distances to five Boston employment centres ,
- RAD: index of accessibility to radial highways ,
- TAX: full-value property-tax rate per $10,000 ,
- PTRATIO: pupil-teacher ratio by town ,
- B: 1000(Bk - 0.63)ˆ2 where Bk is the proportion of blacks by town ,
- LSTAT: % lower status of the population ,
- MEDV: Median value of owner-occupied homes in $1000's

The response variable of this dataset is MEDV. Notice that since CHAS is a factor (binary) variable, it has not been taken into account in the scale of the dataset, it has been removed and added again after the scaling of all the other explanatory variables.



From the plots, we observe that the lambda corresponding to the minimum error is for all of them around 12 degrees of freedom, therefore it's reasonable to approximate the values of β and λ for $df = 12$.

Variable	Beta value
CRIM	-0.017502171
ZN	0.016208176
INDUS	-0.021744495
NOX	-0.019158946
RM	0.031782174
AGE	-0.016876895
DIS	0.010983113
RAD	-0.017065703
TAX	-0.021056246
PTRATIO	-0.023091210
B	0.015059147
LSTAT	-0.033545094
CHAS	0.002061991

Focusing on the factors with higher (absolute) value, which are RM, INDUS, PTRATIO and LSTAT:

- LSTAT: it's negative and it measures the % of lower status of the population. If the value of LSTAT is high, it means the purchasing power is lower therefore the home price cost has to be smaller.
- RM: it's positive, this makes sense since it is the average number of rooms per dwelling, the higher the number, the more likely it is that the home cost is higher.
- PTRATIO: it's negative, this means that the higher the cost of the house, the less number of pupils per teacher, which is kind of a privilege.
- INDUS: it is referred to the number of non-retail business acres per town. If town has a large amount of non-retail areas, it's likely that the price cost is lower since the landscape of factories is not beautiful.

Another thing to notice is that there are some variables, as B, that at first attempt might be considered to be relevant in Boston, but it's not. We have found that the number of black people in Boston is around 30%, therefore that might be the reason why the value β associated to B is small thus not very significant.