Advanced Statistical Modelling: Ridge Regression

Ricard Meyerhofer & Joel Cantero

29/10/2019, correction 20/11/2019

Choosing the penalization parameter λ

The objective of this exercise is to implement Ridge Regression with two different approaches: $MSPE_{val}(\lambda)$ and $MSPE_{k-CV}(\lambda)$. In both cases we are going to take as input data the following:

- Matrix x and vector y corresponding to the training sample.
- Matrix x_{val} and vector y_{val} corresponding to the validation set.
- Vector *lambda.v* of candidate values for λ.

We are going to output for each element λ in *lambda.v* and the value of the $MSPE_{val}(\lambda)$ / $MSPE_{k-CV}(\lambda)$. Furthermore, we are going to plot these values against $log(1+\lambda)-1$.

Once we have build these two functions, we are going to use the prostate data used in class. We are going to choose a λ according to:

- Behaviour in the validation set (30 validations not included in the training sample)
- 5-fold, 10-fold cross-validation.
- Compare our results with those obtained with leave-one-out and generalized cross-validation.

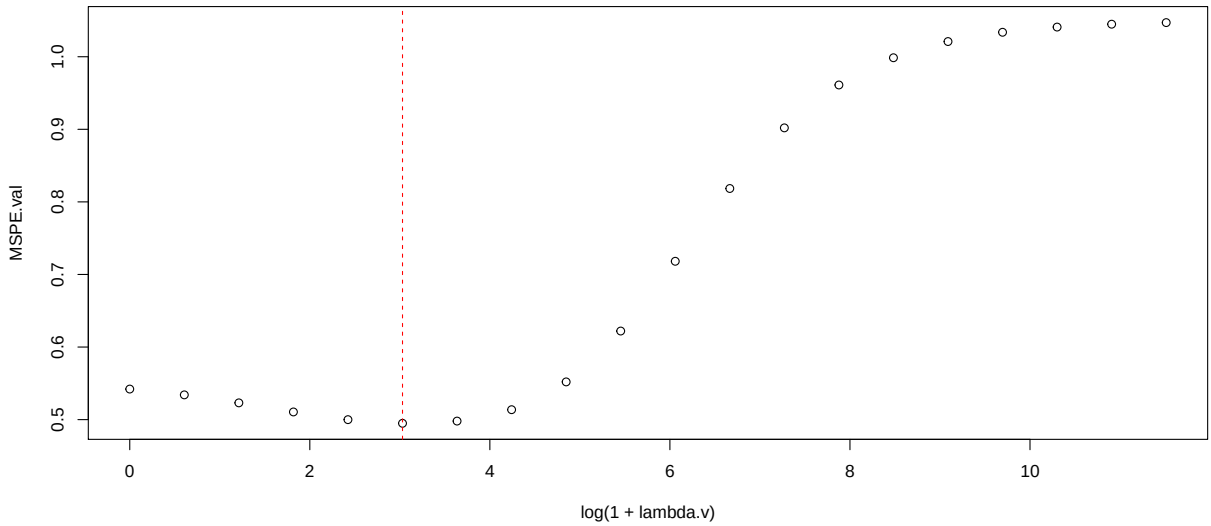
Ridge regression based on $MSPE_{val}(\lambda)$

In order to choose the penalization parameter λ, we are going to write a function implementing the ridge regression penalization parameter λ choice based on the minimization of the $MSPE_{val}(\lambda)$.

```
MSPEval <- function(X, Y, Xval, Yval, lambda.v, n.lambdas) {
  PMSE.VAL <- n.lambdas
  df <- rep(0,length(n.lambdas))
  p <- dim(X)[2]
  # Iterate through candidate values
  x.svd <- svd(x)
  for (l in 1:n.lambdas){
    lambda <- lambda.v[l]
    r <- dim(Xval)[1]
    PMSE.VAL[l] <- 0
    # Compute beta with the traning dataset
    beta.i <- solve(t(X)%*%X + lambda*diag(1,p)) %*% t(X) %*% Y
    for (i in 1:r){
      # Compute the errors with the validation dataset
      Xi <- Xval[i,]
      Yi <- Yval[i]
      y.hat <- Xi %*% beta.i
      PMSE.VAL[l] <-PMSE.VAL[l] + (y.hat - Yi)^2
      df <- sum(x.svd$d^2 / (x.svd$d^2 +lambda))
    }
    PMSE.VAL[l] <- PMSE.VAL[l]/r
  }
```

```
  return(PMSE.VAL)
}
```

Penalization parameter



Ridge regression based on $MSPE_{k-CV}(\lambda)$

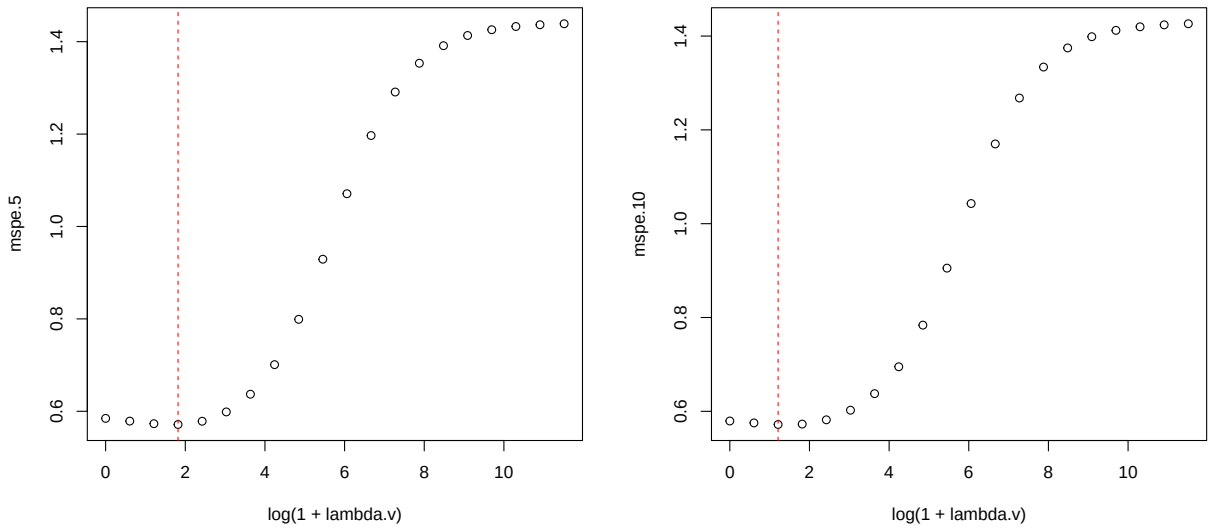
Now, we will write an R function implementing the ridge regression penalization parameterλ choice based on k-fold cross-validation $MSPE_{kCV}(\lambda)$.

```
MSPEkfold <- function(X, Y, K, n.lambdas, lambda.v, n) {
  n <- dim(X)[1]
  folds <- sample(rep(1:K, length=n), n, replace=FALSE)
  MPSE <- data.frame()
  # Iterate through K Folds
  for (k in 1:K){
    Xk <- as.matrix(X[folds != k,])
    Yk <- as.matrix(Y[folds != k])
    Xv <- as.matrix(X[folds == k,])
    Yv <- as.matrix(Y[folds == k])
    mspe.k.val = MSPEvalidation(Xk, Yk, Xv, Yv, lambda.v, n.lambdas)
    MPSE <- rbind(MPSE, mspe.k.val)
  }
  MPSE2 <- c()
  for (i in 1:n.lambdas){
    MPSE2 <- append(MPSE2, mean(MPSE[,i]))
  }
  return(MPSE2)
}

MSPEvalidation <- function(X, Y, Xval, Yval, lambda.v, n.lambdas) {
  PMSE.VAL <- n.lambdas
  p <- ncol(X)
```

```
  # Iterate through candidate values
  for (l in 1:n.lambdas){
    lambda <- lambda.v[l]
    sizeXval <- dim(Xval)[1]
    PMSE.VAL[l] <- 0
    # Compute beta with the traning dataset
    beta.i <- solve(t(X)%*%X + lambda*diag(1,p)) %*% t(X) %*% Y
    for (i in 1:sizeXval){
      # Compute the errors with the validation dataset
      Xi <- Xval[i,]; Yi <- Yval[i]
      y.hat <- Xi %*% beta.i
      PMSE.VAL[l] <-PMSE.VAL[l] + (y.hat - Yi)^2
    }
    PMSE.VAL[l] <- PMSE.VAL[l]/sizeXval
  }
  return(PMSE.VAL)
}
```

Penalization parameter 5-fold and 10-fold.

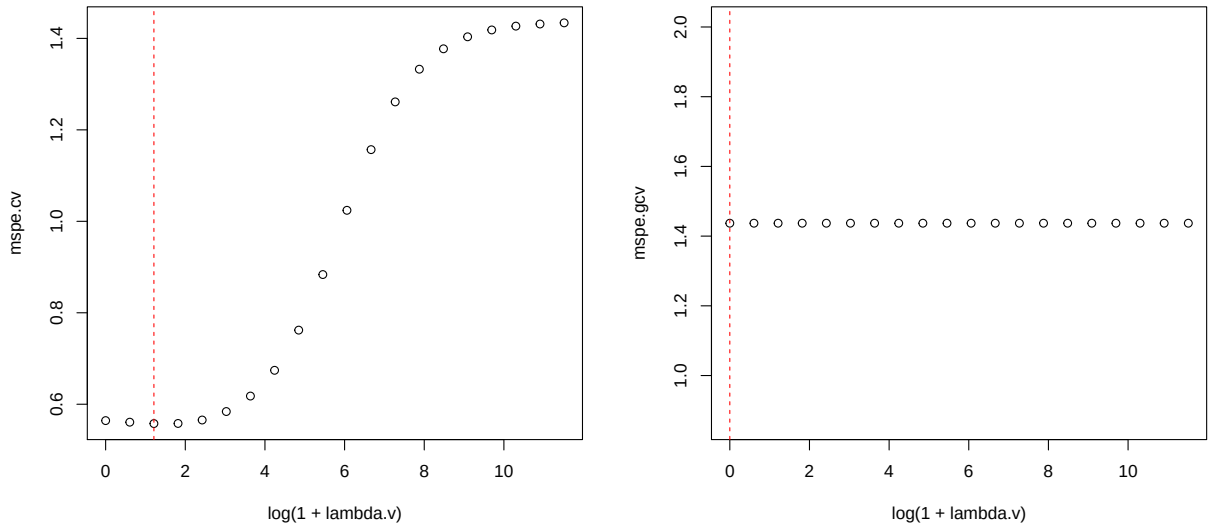


Comparation between penalization parameters of $MSPE_{val}(\lambda)$ and $MSPE_{k-CV}(\lambda)$ with other models

```
MSPEcv <- function(X, Y, n.lambdas, lambda.v, n) {
  PMSE.CV <- numeric(n.lambdas)
  p <- ncol(X)
  for (l in 1:n.lambdas){
    lambda <- lambda.v[l]
    PMSE.CV[l] <- 0
    for (i in 1:n){
      X.i <- X[-i,]
      Y.i <- Y[-i]
```

```
      Xi <- X[i,]
      Yi <- Y[i]
      beta.i <- solve(t(X.i)%*%X.i + lambda*diag(1,p)) %*% t(X.i) %*% Y.i
      y.hat <- Xi %*% beta.i
      PMSE.CV[l] <- PMSE.CV[l] + (y.hat-Yi)^2
    }
    PMSE.CV[l] <- PMSE.CV[l]/n
  }
  return(PMSE.CV)
}

#generalized cv
MSPEgcv <- function(X, Y, n.lambdas, lambda.v, beta.path, diag.H.lambda, n) {
    PMSE.GCV <- numeric(n.lambdas)
    for (l in 1:n.lambdas){
        hat.Y <- X %*% beta.path[l,]
        nu <- sum(diag.H.lambda[l,])
        PMSE.GCV[l] <- sum( ((Y-hat.Y)/(1-nu/n))^2 )/n
    }
    return(PMSE.GCV)
}
```



We can see that the Validation approach has the lowest optimal number of degrees of freedom. Furtheremore, we can find the degrees of freedom of each method and their MSPE. As we can see, LOO method has same degrees of freedom than 10 fold and 5-10 fold also very similar MSPE than validation.

MSPE Method	df.v	MSPE
Validation	5.231211	0.4949079
Leave-one-out (CV)	7.437145	0.5577224
5-Fold val.	6.903307	0.5714631
10-Fold val.	7.437145	0.5719459
Generalized C.V.	8.000000	1.4370365

Ridge regression for the Boston Housing data

The Boston Housing dataset is a classical dataset which containes the values of 506 suburbs of Boston corresponding to 1978. This dataset can be found in many places but we are going to use a version with some corrections that was provided to us, which additionally includes the UTM coordinates of the geographical centers of each neighborhood. Therefore, the variables are the following:

Variable	Description	Type
CRIM	per capita crime rate by town	Numeric
ZN	proportion of residential land zoned for lots over 25,000 sq.ft.	Numeric
INDUS	proportion of non-retail business acres per town	Numeric
CHAS	Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)	Factor
NOX	nitric oxides concentration (parts per 10 million)	Numeric
RM	average number of rooms per dwelling	Numeric
AGE	proportion of owner-occupied units built prior to 1940	Numeric
DIS	weighted distances to five Boston employment centres	Numeric
RAD	index of accessibility to radial highways	Numeric
TAX	full-value property-tax rate per $10,000	Numeric
PTRATIO	pupil-teacher ratio by town	Numeric
B	1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town	Numeric
LSTAT	% lower status of the population	Numeric
MEDV	Median value of owner-occupied homes in $1000's	Numeric

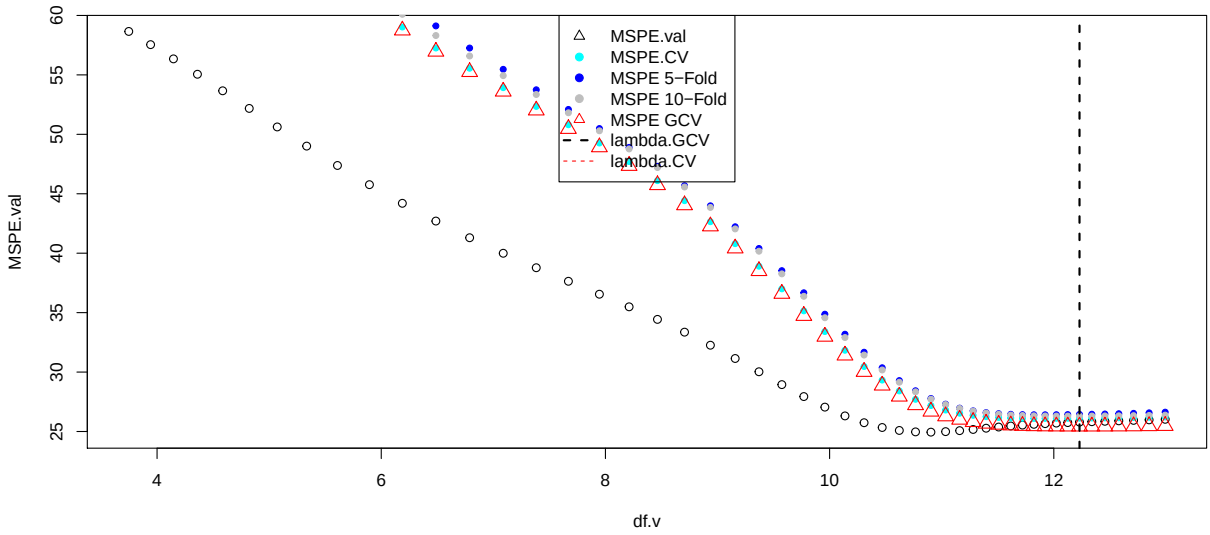
In this exercise we are going to use ridge regression on the Boston Housing dataset to fit the regression model where the response variable is $MEDV$ and the explanatory variables are the remaining 13 variables shown in the list. As we can see when loading the data, there are more variables than the ones listed ($TOWN$, $TOWNNO$, LON, LAT, $CMEDV$). We decided not to use them since the statement explicitly defines which to use.

```
load("boston.Rdata")
names(boston.c)
dataset <- boston.c[,-c(1:5)] #"TOWN"    "TOWNNO"  "TRACT"   "LON"     "LAT"
dataset$CMEDV <- NULL
```

Besides from eliminating variables, we divided the dataset between train and test. To do so, we have decided to do 3/4 training, 1/4 test.

```
trainIndex <- createDataPartition(dataset$MEDV, p =0.75, list = F)
train <- dataset[trainIndex,]
test <- dataset[-trainIndex,]
```

We are going to compare the different models and choose lambda with the one that performs best.



MSPE Method	df.v	MSPE
Validation	10.90382	24.94631
Leave-one-out (CV)	11.92268	26.10335
5-Fold val.	11.92268	26.40932
10-Fold val.	12.12439	26.29733
Generalized C.V.	12.22932	25.42651

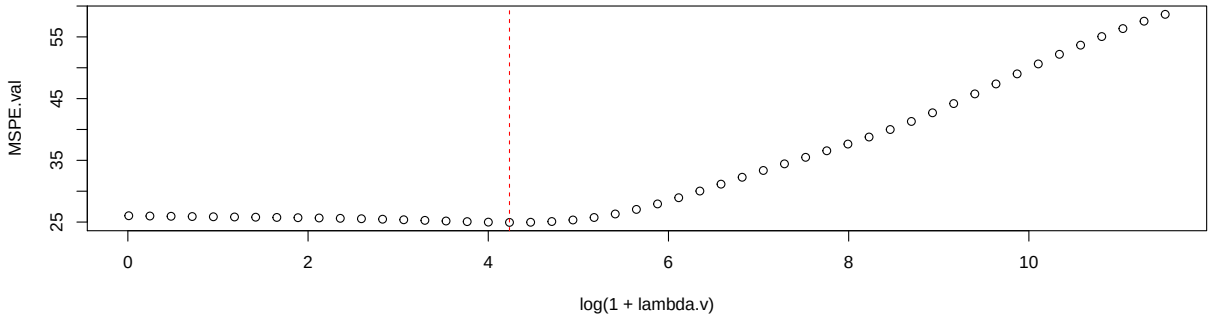
We can see from both the graphic and the table that the lowest values have been obtained again with the Validation method, with a selected degrees of freedom of 10.90 and a MSPE of 24.94.

The generalized cross validation method as well as the leave one out are very close to the K-Fold methods and their evolution over different is very similar.

Choosing λ

Now that we know which is the method that performs the best, we are going to see which λ is the best

```
plot(log(1+lambda.v), MSPE.val)
abline(v=log(1+lambda.val),col=2,lty=2)
```



We can see that the 19th lambda is the one with the lowest MSPE value with $\lambda_{19} = 68.10$.