Business Analytics - ETC3250 2017 - Lab 6

Model selection

Souhaib Ben Taieb

29 August 2017

Note: the online version of "An Introduction to Statistical Learning with Applications in R" (ISLR) is availalble at http://www-bcf.usc.edu/~gareth/ISL/.

Exercise 1

Understand all the steps in the proof of the leave-one-out cross-validation (LOOCV) statistic for linear models available at https://github.com/bsouhaib/BA2017/blob/master/slides/6/6-loocv.pdf

Exercise 2

We will build a regression model using the `Boston` data set from the `MASS` package. The aim is to predict `medv` (median value of owner-occupied homes in each town) using characteristics of each town.

Find the best model you can using the available predictors. Consider dropping predictors, using transformations of predictors, adding interaction terms, using splines, etc.

You will determine which model is "best" based on the LOOCV statistic. For linear models, it can be computed using the following function.

```
CV <- function(object)
{
  cv <- mean((residuals(object)/(1 - hatvalues(object)))^2, na.rm = TRUE)
  return(cv)
}
```

You can apply this function to the output from the `lm()` function.

Hints:

- Regression plots can help you decide if a variable has a nonlinear effect on the response variable.
- P-values can suggest potential variables to drop, but remember that p-values are misleading when predictors are correlated.

Try to work systematically, and keep track of the various models you have tried to avoid wasting time by re-fitting old models.

Assignment - Question

In the previous exercise, we used a function `CV()` that computed the LOOCV statistic using the neat computational trick available for linear models based on the hat-matrix (see Exercise 2).

Now you will write your own function `myCV()` that will compute the same statistic, but do it the long way by fitting models to different sets of training data.

myCV() should work for linear regression models and take two arguments:

```
myCV <- function(formula, data)
{

}
```

You can apply the model `formula` to the data set leaving out the ith observation like this:

```
fit <- lm(formula, data=data[-i,])
```

To get the prediction of the omitted observation, use the predict function:

```
pred <- predict(fit, newdata=data[i,])
```

The response value for the omitted observation can be found as follows.

```
responsevar <- as.character(formula(fit))[2]
data[i,responsevar]
```

You will need to use a `for` loop to cycle through all the values of `i`.

To test you have done it correctly, check that `myCV()` and `CV()` return the same value.

```
library(MASS)

myCV <- function(formula, data)
{
    n <- nrow(data)
    err <- numeric(n)
    for(i in 1:n)
    {
        fit <- lm(formula, data = data[-i, ])
        pred <- predict(fit, newdata = data[i,])
        responsevar <- as.character(formula(fit))[2]
        actual <- data[i, responsevar]
        err[i] <- (actual - pred)^2
    }
    mean(err)
}

formula <- "ptratio + dis + lstat + zn + chas + nox + rm + dis"
my.fit <- lm(as.formula(paste("medv ~", formula)), data = Boston)
CV(my.fit)
# [1] 24.98204
myCV(my.fit, Boston)
# [1] 24.98204
```

Now add a new argument to your `myCV()` function to allow it to do k-fold cross-validation.

```
myCV <- function(formula, data, k)
{

}
```

The `k` argument can take values between 2 and `n` where `n=nrow(data)`. For example `myCV(formula, data, k=5)` will do a 5-fold cross-validation. When `k=n`, the function should do leave-one-out cross-validation.

You should shuffle your data frame before doing the division of data into folds.

You can check that your updated function works correctly by comparing the results against `CV()` for different values of `k`. When `k=n`, the results should be identical. For other values of `k`, the results should be close but not identical.

```
# As example, here is a solution from one student
myCV <- function(formula, data, k) {
  data <- data[sample(nrow(data)),]
  j<- floor(nrow(data)/k)
  err <- vector(mode ="numeric", nrow(data))
  for(i in 1:(k-1)) {
    ex <- (((i-1)*j+1):(i*j))
    fit <- lm(formula, data=data[-ex,])
    pred <- predict(fit, newdata=data[ex,])
    responsevar <- as.character(formula(fit))[2]
    actual <- data[ex,responsevar]
    err[ex] <- (actual - pred)
  }
  ex <- (((k-1)*j+1):nrow(data))
  fit <- lm(formula, data=data[-ex,])
  pred <- predict(fit, newdata=data[ex,])
  responsevar <- as.character(formula(fit))[2]
  actual <- data[ex,responsevar]
  err[ex] <- (actual - pred)
  errs <- err^2

  return(mean(errs))
}

myCV(as.formula(paste("medv ~", formula)),Boston, nrow(Boston))
# [1] 24.98204
```

TURN IN

- Your `.Rmd` file (which should knit without errors and without assuming any packages have been pre-loaded)
- Your Word (or pdf) file that results from knitting the Rmd.
- DUE: September 3, 11:55pm (late submissions not allowed), loaded into moodle