SCR Week 10

Cross-Validation Exercises

R-team

21 November, 2019

1. Your first cross-validation

We are going to work with the `mtcars` dataset in the `datasets` package. The data set contains measurements on 10 aspects of some (very old) automobiles (1973-1974 models). You can look at the helpfile of this dataset to find out more about this data set. You can load the data yourself using the following code:

```
data(mtcars)
head(mtcars)
```

```
##                    mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
## Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

We'll try to predict the efficiency of each cars in terms of fuel consumption based on some simple characteristics of the car. We'll first make a sensible selection of the variables we think are reasonable to select in this regard.

Some simple physics: having a heavier car leads to higher consumption per mile, because a heavier car will need more energy to speed up. The amount of energy required to keep it moving at the same speed is (about) the same. Each stop and start (for instance at a stopping light) will mean that a heavier car will consume more fuel. So `wt` is a sensible variable to include.

A variable such as `cyl` (the number of cylinders in the car) may be meaningless: if one car has two 500cc cylinders, and another four 250cc cylinders, the amount of fuel displacement given the same rotations per minute is equal.

Another variable which *may* be meaningless is the gross horsepower (`hp`). Higher horsepower is nice when the car is either heavy, a race car, or both. Just because a car is *able* to have a higher horsepower throughput, does not necessarily mean that this is also used during driving. This is where psychology comes in. Whether horsepower matters or not thus depends on the way in which these cars were tested to determine miles/gallon. A likely scenario: a car with higher horse power will usually be used to accelerate faster, which will lead to a higher amount of accumulated wind resistance at top speeds as cars with more horsepower are driving in topspeed longer (and windresistance is not linear, so driving 80 miles per hour means more than twice as much windresistance as driving at 40 miles per hour). A similar point can be made for the transmission type of the car (variable `am`): in practice automatic cars are more fuel efficient because the machinery is better at shifting gears at the right times than (average) humans are (also they are usually set-up to be fuel efficient).

For now we'll use physics and pop psychology to stick with the three variables `wt`, `hp` and `am` to predict `mpg`. Note that this is an *unsupervised* decision: we have not looked at the relation between these variables and the outcome to *decide* whether to include these or not!

1.1 The folds

The first step is to define the sets of objects that we split into K folds. There are 32 rows in our data set. If we want to do a 5-fold cross-validation, this means we cannot split our data into neat, equal parts. Too bad. We're going to do 5-fold cross-validation anyway. We will *not* put some objects into more than 1 fold (such that we have 5 folds, and each fold contains observations). We *will* create 5 folds that will contain (unfortunately) an unequal number of objects.

Note, this is just one way to create the indices, other ways are possible of course!

a

Create a vector that contains the numbers 1 through 5, some repeated 7 times, others repeated only 6 times, such that the length of this vector with numbers is equal to 32.

Answer:

```r
index <- rep(1:K, floor(nrow(mtcars)/K)+1)[1:nrow(mtcars)]
```

b

The numbers you've created can be seen as indicators to which fold each object belongs. We have indicators, and objects/rows in our dataset. Shuffle these indices around so that the 'fold' assigned to a particular object, is a random one.

Answer:

```r
set.seed(23112016)
fold.index <- sample(index)
# or
# fold.index <- cut(sample(1:nrow(mtcars)), breaks=K, labels=F)
```

c

Use the shuffled set of indices to select all objects in the data that belong to the 3th fold.

Answer:

```r
mtcars[fold.index==(K-2), ]
```

d

Use the shuffled set of indices to select all objects that *do not* belong to the 3th fold.

Answer:

```r
mtcars[fold.index!=(K-2), ]
```

1.2 Fitting and predicting

We will now use `lm` to fit a regression model on the first 4 folds, and use the fitted model to predict the outcomes in the 5th fold.

a

Select from the dataset the objects belonging to folds 1 through 4. Save the observations for these objects into a new data set called `training`.

Answer:

```
training <- mtcars[fold.index!=K, ]
```

b

Select from the dataset the objects belonging to fold 5. Save the observations for these objects into a new data set called `validation`.

Answer:

```
validation <- mtcars[fold.index==K, ]
```

c

Fit a linear model on the training set, and predict the outcomes of the validation set.

Answer:

```
training.fit <- lm(mpg~wt+hp+am, data=training)
summary(training.fit)
```

```
##
## Call:
## lm(formula = mpg ~ wt + hp + am, data = training)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.2479 -1.4318 -0.1094  1.2040  4.9556
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 32.80686    3.01960  10.865 2.61e-10 ***
## wt          -2.15856    1.06765  -2.022 0.055528 .
## hp          -0.04449    0.01047  -4.249 0.000329 ***
## am           2.99036    1.61211   1.855 0.077057 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.544 on 22 degrees of freedom
## Multiple R-squared:  0.8551, Adjusted R-squared:  0.8353
## F-statistic: 43.28 on 3 and 22 DF,  p-value: 2.13e-09
```

```
validation.predict <- predict(training.fit, newdata=validation, type='response')
```

d

Calculate the percentage of variance explained in the outcomes of the validation sample by your prediction.

Answer:

```
cor(validation$mpg, validation.predict)^2
```

```
## [1] 0.8985355
```

Note that this would be the variance explained, given that we regress the predictions on the outcome. It is not useful as a measure of correct prediction.

1.3 Cross-validation

We'll now use cross-validation to estimate the expected prediction error of the model we proposed in part one of this exercise.

a: physics/pop psych model

Write a for-loop, that repeats the above procedure for all folds. Save the percentage of variance explained in each of the folds. Calculate the average amount of variance explained (think about if you need to weight the results from your folds differently).

Answer:

```
Loss <- function(x, y){
  sum((x-y)^2)/length(x)
}

loss <- numeric(K)

for (k in 1:K){
  training <- mtcars[fold.index!=k, ]
  validation <- mtcars[fold.index==k, ]

  training.fit <- lm(mpg~wt+hp+am, data=training)
  validation.predict <- predict(training.fit, newdata=validation, type='response')

  loss[k] <- Loss(validation$mpg, validation.predict)
}

#average, with weights equal to the number of objects used to calculate the loss at each fold:
mean(loss)
```

```
## [1] 7.868768
```

b: full model

Write another for loop, that does cross-validation to estimate the prediction error, but instead of using just the variables `hp`, `wt` and `am` use **all** variables (other than `mpg`) in the `mtcars` dataset.

Answer:

```r
loss <- numeric(K)

for (k in 1:K){
  training <- mtcars[fold.index!=k, ]
  validation <- mtcars[fold.index==k, ]

  training.fit <- lm(mpg~., data=training)
  validation.predict <- predict(training.fit, newdata=validation, type='response')

  loss[k] <- Loss(validation$mpg, validation.predict)
}

mean(loss)
```

```
## [1] 17.15302
```

c: subset selection

Write another for loop, that does cross-validation, to estimate the prediction error. But instead using the variables `hp`, `wt` and `am` or all variables, select in each fold the three best univariate predictors (using correlation as a measure) and fit a model using these three variables as explantory variables. Save the names of the variables selected in each fold.

What is the average estimated prediction error? Are the same variables selected in each fold?

Answer:

```r
loss <- numeric(K)

selected.var.names <- matrix("", nrow=K, ncol=3)

for (k in 1:K){
  training <- mtcars[fold.index!=k, ]
  validation <- mtcars[fold.index==k, ]

  univar.cor <- apply(training[, 2:ncol(training)], 2, function(var, mpg=training$mpg) cor(var, mpg))

  var.names <- tail(names(sort(abs(univar.cor))), 3)
  selected.var.names[k, ] <- var.names

  training.fit <- lm(mpg~., data=training[, c("mpg", var.names)])
  validation.predict <- predict(training.fit, newdata=validation, type='response')

  loss[k] <- Loss(validation$mpg, validation.predict)
}

mean(loss)
```

```
## [1] 8.184914
```

The same variables are selected each time. Although true in this case, this would not necessarily always be the case, different variables may be selected in each fold.

d

In the previous exercise, did we calculate the estimated prediction error of the model 'mpg = cyl + disp + wt'? Or of something else?

Answer: We calculated the estimated prediction error of a procedure where we selected the best three univariate predictors, not of the model 'mpg = cyl + disp + wt'.

1.5 A little more difficult

A simulation study

In this exercise we are going to do our own simulation study to investigate the characteristics of K-fold cross-validation for a 'realistic' setting using simple linear regression as our example.

a

Although this is quite a lot of stuff to do, we'll leave it to you this time to nicely break this up into smaller more manageable pieces.

The study design is a two-way design. The first experimental variable is the number of folds, the second is the number of times we are going to repeat each procedure. We will use 2-fold, 10-fold, 30-fold and 50-fold cross-validation. We will repeat each procedure 5, 25 and 100 times. Each of the 5, 25, 100 repeats will have objects randomly assigned to the K folds. The results of the experiment that we are interested in are:

- The average of the estimated prediction error for each procedure, over these repetitions
- The variance of the estimated prediction error for each procedure, over these repetitions

For example, in the case where we do 2-fold cross-validation and 5 repeats: repeat 1 is going to give us two prediction errors, one for each fold. Average these. This is the outcome of repeat 1. Do this again for repeat number 2. This will give you another outcome. Continue until you have done 5 repeats. You now have 5 outcomes. Calculate the average and the variance of these 5 outcomes. These are the outcomes of the first experimental setting where we do 2-fold cross-validation using 5 repeats.

In the end you should have a 4 by 3 table with the results for the average estimated expected prediction error and a 4 by 3 table containing the results for the variance of the estimated expected prediction errors. Interpret these results. What are the expected differences between the outcomes?

We generate the data as follows:

```r
set.seed(25112016)
N <- 150
x <- rnorm(N)
y <- x + rnorm(N, sd = 1.5)
my_data <- data.frame(x, y)
```

Answer:

```r
#Set up matrix to write results
nfold.options <- c(2, 10, 30, 50)
repeat.options <- c(5, 25, 100)
```

```r
results.mean <- matrix(0, ncol=length(nfold.options), nrow=length(repeat.options))
results.var <- matrix(0, ncol=length(nfold.options), nrow=length(repeat.options))
colnames(results.mean) <- colnames(results.var) <- nfold.options
rownames(results.mean) <- rownames(results.var) <- repeat.options

for (i in 1:length(repeat.options)){
  for (j in 1:length(nfold.options)){

    repeats <- repeat.options[i]
    K <- nfold.options[j]

    study.cel.results <- matrix(0, nrow=repeats, ncol=K)

    for (h in 1:repeats){
      fold.index <- sample(rep(c(1:K), each=N/K))

      var.expl <- numeric(K)

      for (k in 1:K){
        training <- my_data[fold.index!=k, ]
        test <- my_data[fold.index==k, ]

        training.fit <- lm(y~x, data=training)
        test.predict <- predict(training.fit, newdata=test, type='response')

        study.cel.results[h, k] <- Loss(test.predict, test$y)
      }
    }

    results.mean[i, j] <- mean(study.cel.results)
    results.var[i, j] <- var(colMeans(study.cel.results))

  }
}

library(knitr)
kable(results.mean)
```

	2	10	30	50
5	2.216555	2.136568	2.137456	2.136673
25	2.145787	2.137127	2.132795	2.133892
100	2.162533	2.137320	2.134358	2.132833

```r
kable(results.var)
```

	2	10	30	50
5	0.0053845	0.0740772	0.2951544	0.5433539
25	0.0006914	0.0137343	0.0578274	0.1042731
100	0.0011416	0.0068871	0.0197981	0.0394721

b

Can you think of a way to estimate the actual expected prediction error conditional on this training set?

Answer:

The ‘ideal’ would be to simulate a new data set, exactly like our original set, of infinite size. Then use our model to predict these new outcomes, this would give us an ‘estimate’ of the actual prediction error on new data with infinite precision.

c

Can you think of a way to estimate the actual expected prediction error (unconditional on the training set)?

Answer:

Repeat the study for many data sets (of equal size) and predict outcomes on data sets similar to the ones discussed in **b**. Since in this particular case we know the data generation process this is very easy.

1.6 Cross-validation and logistic regression

This will be an exercise that you can revisit once logistic regression has been discussed in the Linear Models course.

```
set.seed(25112016)
N <- 100
x <- rnorm(N)
y <- 0.25+x+rnorm(N, sd=0.5)
p <- pnorm(y)
outcome <- round(p)
my_data <- data.frame(x, outcome)
write.csv(my_data, "0_data/ex_2.csv", row.names=F)
```

Read in `ex_2.csv` (see the `0_data` folder). Use leave-1-out cross-validation to decide between logistic regression using the probit and the logit link function.

Answer:

```
N <- length(outcome)

l.predict <- numeric(N)
p.predict <- numeric(N)

for (i in 1:N){
  training <- my_data[-i, ]
  test <- my_data[i, ]
  l.predict[i] <- round(predict(glm(outcome~x, binomial(link="logit"), data=training), newdata=test, ty
  p.predict[i] <- round(predict(glm(outcome~x, binomial(link="probit"), data=training), newdata=test, t
}

mean(my_data$outcome!=l.predict)
```

```
## [1] 0.12
```

```
mean(my_data$outcome!=p.predict)
```

```
## [1] 0.14
```

The estimated prediction error is lower for logistic regression using the probit link.