Lab #3: Matrix algebra and cross-validation

Biostat 216: Machine Learning in R for the Biomedical Sciences

The purpose of this lab is to get you manipulating vectors and matrices and to apply validation and cross-validation methods in R.

Vectors and matrices:

We will play around with specifying, manipulating and performing arithmetic on vectors and matrices in R. We will also do a little matrix algebra by hand!

1. Create the following matrix in R and call it X

$$\begin{pmatrix} -3.2 & 4.1 & 1.7 \\ 2.0 & 2.8 & -6.4 \\ 1.8 & 1.7 & -2.8 \\ -7.8 & -9.3 & 4.6 \end{pmatrix}$$

   Take a look at X to make sure you specified it correctly. Then examine some of its properties and specific elements using the following commands:

   i. `dim(X)`
   ii. `nrow(X)`
   iii. `ncol(X)`
   iv. `X[1, ]`
   v. `X[ ,3]`
   vi. `X[3, 2]`
   vii. `X[1:2, 1]`
   viii. `X[ ,-1]`

   Now let's work on changing X through various manipulations. After each command type X to look at the matrix and see what (if anything) changed

   ix. `X < -X+1`
   x. `X <- t(X)`
   xi. `X <- diag(3) %*% X`
   xii. `X <- X %*% diag(4)`
   xiii. `X <- X %*% diag(3)`
   xiv. `X <- X[ ,1:3]`

   Finally, let's play around with X a little

   xv. `Xinv <- solve(X)`
   xvi. `X %*% Xinv`
   xvii. `Xinv %*% X`

2. Perform the following computations by hand and then confirm your results by performing the calculations in `R`

   a. $\begin{pmatrix} -3.1 & 2.4 \\ 0.2 & -1.7 \\ 0.8 & -0.7 \end{pmatrix} + \begin{pmatrix} 3.5 & 3.4 \\ -0.8 & -6.2 \\ -3.0 & 4.5 \end{pmatrix}$

   b. $\begin{pmatrix} -0.1 & 2.2 \\ 2.6 & 1.4 \end{pmatrix} - \begin{pmatrix} 1.0 & 3.5 \\ 2.1 & 4.2 \end{pmatrix}$

   c. $3\begin{pmatrix} -3.1 & 2.4 \\ 0.2 & -1.7 \\ 0.8 & -0.7 \end{pmatrix}$

   d. $\begin{pmatrix} 4.2 & 3.7 \\ 3.6 & 4.1 \\ 2.9 & 5.7 \end{pmatrix}\begin{pmatrix} 3.5 \\ 4.0 \end{pmatrix}$

   e. $\begin{pmatrix} 4 & 1 \\ 2 & 3 \end{pmatrix}\begin{pmatrix} 6 & 7 \\ 8 & 5 \end{pmatrix}$

   f. $\begin{pmatrix} 2 & 0.5 & -2 \\ -4 & 0 & 1 \\ 3 & 1 & 0 \end{pmatrix}\begin{pmatrix} 3 & 5 \\ 2 & -1 \\ -2 & 0 \end{pmatrix}$

   g. $\begin{pmatrix} 7 & 6 \\ 4 & 5 \end{pmatrix}^{\mathrm{T}}\begin{pmatrix} 6 & 5 \\ 5 & 6 \end{pmatrix}$

Validation and Cross-validation:

3. Let's try simulating an analysis and then perform training-test validation:
   a. `set.seed(5)`
   b. Generate a vector of predictors `X1` of length 50, normally distributed with mean 3 and SD 4
   c. Generate a second vector of predictors `X2` of length 50, uniformly distributed between 0 and 5 (see the `runif` command)
   d. Generate some random noise `epsilon` of length 50, normally distributed with mean 0 and SD 2
   e. Generate `trueY` as `X1+X2`
   f. Generate `Y` as `X1+X2+epsilon`
   g. `set.seed(3)`

h. Generate `set` as a factor with 2 levels:
```
set <- sample(c(rep("train",25),rep("test",25)))
```

i. Make a dataframe with `X1`, `X2`, `epsilon`, `trueY`, `Y`, and `set`

j. Fit a linear model to the training data with `Y` as outcome and `X1`, `X2`, `X1*X2`, $X1^2$ and $X2^2$ as predictors (clearly this has the potential to overfit)

k. Determine the mean square error (MSE) in the training data

l. Determine the MSE in the test data

m. Compare the training and test MSE – does this match your expectations?

n. Now try rerunning the process with `set.seed(7)` in step g. What do you notice now when comparing MSE in training vs. test data? Can you explain what might be going on?

4. Cross-validation example

a. `set.seed(5)`

b. Generate a vector of predictors `X1` of length 50, normally distributed with mean 3 and SD 4

c. Generate a second vector of predictors `X2` of length 50, uniformly distributed between 0 and 5 (see the `runif` command)

d. Generate some random noise `epsilon` of length 50, normally distributed with mean 0 and SD 2

e. Generate `trueY` as `X1+X2`

f. Generate `Y` as `X1+X2+epsilon`

g. `set.seed(3)`

h. Generate `set` as a factor with 5 levels:
```
set <- sample(rep(1:5, 10))
```

i. Make a dataframe with `X1`, `X2`, `epsilon`, `trueY`, `Y`, and `set`

j. Fit a linear model to four of the levels of `set` with `Y` as outcome and `X1`, `X2`, `X1*X2`, $X1^2$ and $X2^2$ as predictors

k. Calculate the MSE in the remaining level of `set`

l. Repeat the process in j and k until each level has been the hold out set (recording the MSE each of the 5 times)

m. Now combine the MSE estimates from the 5 test folds to produce a final MSE estimate