Module 5: Recommended Exercises

TMA4268 Statistical Learning V2020

Stefanie Muff, Department of Mathematical Sciences, NTNU

February xx, 2020

Contents

Plan for interactive lecture

You may work alone, in pairs or larger groups - lecturer and TA will supervise.

14:15-14:35ish: Introduction to R Markdown (by lecturer) and the template to be used for Compulsory exercise 1 (approx 20 minutes). Bring your laptop, download https://www.math.ntnu.no/emner/TMA4268/2019v/CompEx1mal.Rmd and also install the packages listed here: https://www.math.ntnu.no/emner/TMA4268/2019v/Compulsory1.html#r_packages

14:35: Introduction to problems on cross-validation - work with problems 1-3: Recommended exercises on cross-validation, and then work with the problems

15:00-15:15: Break with light refreshments

15:15-15:25: Finish up the CV-problems, and lecturer summarizes

15:25: Introduction to the bootstrap problems - then work with problem 1: Recommended exercises on bootstrapping

15:45: Summing up problems and start Team Kahoot on Module 5 (and indirectly 1-4).

16:00: End - and you may stay for supervision of RecEx and CompEx until 18.

Recommended exercises on cross-validation

Problem 1: Explain how k-fold cross-validation is implemented

a) Draw a figure

b) Specify algorithmically what is done, and in particular how the "results" from each fold are aggregated

c) Relate to one example from regression. Ideas are the complexity w.r.t. polynomials of increasing degree in multiple linear regression, or K in KNN-regression.

d) Relate to one example from classification. Ideas are the complexity w.r.t. polynomials of increasing degree in logistic regression, or K in KNN-classification?)

Hint: the words "loss function", "fold", "training", "validation" are central.

Problem 2: Advantages and disadvantages of k-fold Cross-Validation

What are the advantages and disadvantages of k-fold cross-validation relative to

a) The validation set approach
b) Leave one out cross-validation (LOOCV)
c) What are recommended values for k, and why?

Hint: the words "bias", "variance" and "computational complexity" should be included.

Problem 3: Selection bias and the "wrong way to do CV".

The task here is to devise an algorithm to "prove" that the wrong way is wrong and that the right way is right.

a) What are the steps of such an algorithm? Write down a suggestion. Hint: How do you generate data for predictors and class labels, how do you do the classification task, where is the CV in the correct way and wrong way inserted into your algorithm? Can you make a schematic drawing of the right and the wrong way? Hint: ISL book slides, page 20+21 - but you can do better?

b) We are now doing a simulation to illustrate the selection bias problem in CV, when it is applied the wrong way. Here is what we are (conceptually) going to do:

Generate data

- Simulate high dimensional data ($p = 5000$ predictors) from independent or correlated normal variables, but with few samples ($n = 50$).
- Randomly assign a class labels (here only 2) - so the truth is that the misclassification rate can not get very small. What is the theoretical misclassification rate (for this random set)?

Classification task:

- We choose a few ($d = 25$) of the predictors (how? we just select those with the highest correlation to the outcome).
- Perform a classification rule (here: logistic empirical Bayes) on these predictors.
- Then, we do CV ($k = 5$) on only d =wrong way, or $c + d$=right way.
- Report misclassification errors for both situations.

One possible version of this is presented in the R-code below. Go through the code and explain what is done in each step, then run the code and observe if the results are in agreement with what you expected. Make changes to the R-code if you want to test out different strategies.

```
library(boot)
# GENERATE DATA reproducible
set.seed(4268)
n = 50  #number of observations
p = 5000  #number of predictors
d = 25  #top correlated predictors chosen
kfold = 10
# generating predictor data
xs = matrix(rnorm(n * p, 0, 4), ncol = p, nrow = n)  #simple way to to uncorrelated predictors
dim(xs)  # n times p
# generate class labels independent of predictors - so if all
# classifies as class 1 we expect 50% errors in general
ys = c(rep(0, n/2), rep(1, n/2))  #now really 50% of each
table(ys)

# WRONG CV - using cv.glm here select the most correlated predictors
# outside the CV
corrs = apply(xs, 2, cor, y = ys)
hist(corrs)
selected = order(corrs^2, decreasing = TRUE)[1:d]  #top d correlated selected
data = data.frame(ys, xs[, selected])
# apply(xs[,selected],2,cor,y=ys) yes, ave the most correlated then
# cv around the fitting of the classifier - use logistic regression
# and built in cv.glm function
logfit = glm(ys ~ ., family = "binomial", data = data)
cost <- function(r, pi = 0) mean(abs(r - pi) > 0.5)
cvres = cv.glm(data = data, cost = cost, glmfit = logfit, K = kfold)
cvres$delta
# observe - near 0 misclassification rate

# WRONG without using cv.glm - should be similar (just added to see
# the similarity to the RIGHT version)
reorder = sample(1:n, replace = FALSE)
validclass = NULL
for (i in 1:kfold) {
    neach = n/kfold
    trainids = setdiff(1:n, (((i - 1) * neach + 1):(i * neach)))
    traindata = data.frame(xs[reorder[trainids], ], ys[reorder[trainids]])
    validdata = data.frame(xs[reorder[-trainids], ], ys[reorder[-trainids]])
    colnames(traindata) = colnames(validdata) = c(paste("X", 1:p), "y")
    data = traindata[, c(selected, p + 1)]
    trainlogfit = glm(y ~ ., family = "binomial", data = data)
    pred = plogis(predict.glm(trainlogfit, newdata = validdata[, selected]))
    print(pred)
    validclass = c(validclass, ifelse(pred > 0.5, 1, 0))
}
table(ys[reorder], validclass)
1 - sum(diag(table(ys[reorder], validclass)))/n

# CORRECT CV
```

```r
reorder = sample(1:n, replace = FALSE)
validclass = NULL
for (i in 1:kfold) {
    neach = n/kfold
    trainids = setdiff(1:n, (((i - 1) * neach + 1):(i * neach)))
    traindata = data.frame(xs[reorder[trainids], ], ys[reorder[trainids]])
    validdata = data.frame(xs[reorder[-trainids], ], ys[reorder[-trainids]])
    colnames(traindata) = colnames(validdata) = c(paste("X", 1:p), "y")
    foldcorrs = apply(traindata[, 1:p], 2, cor, y = traindata[, p + 1])
    selected = order(foldcorrs^2, decreasing = TRUE)[1:d]  #top d correlated selected
    data = traindata[, c(selected, p + 1)]
    trainlogfit = glm(y ~ ., family = "binomial", data = data)
    pred = plogis(predict.glm(trainlogfit, newdata = validdata[, selected]))
    validclass = c(validclass, ifelse(pred > 0.5, 1, 0))
}
table(ys[reorder], validclass)
1 - sum(diag(table(ys[reorder], validclass)))/n
```

Recommended exercises on bootstrapping

Problem 1: Probability of being part of a bootstrap sample

We will calculate the probability that a given observation in our original sample is part of a bootstrap sample. This is useful for us to know in Module 8, and was also given on the exam in 2018.

Our sample size is n.

a. We draw one observation from our sample. What is the probability of drawing observation x_i? And of not drawing observation x_i?
b. We make n independent drawing (with replacement). What is the probability of not drawing observation x_i in any of the n drawings? What is then the probability that x_i is in our bootstrap sample (that is, more than 0 times)?
c. When n is large $(1 - \frac{1}{n})^n = \frac{1}{\exp(1)}$. Use this to give a numerical value for the probability that a specific observation x_i is in our bootstrap sample.
d. Write a short R code chunk to check your result. (Hint: An example on how to this is on page 198 in our ISLR book.) You may also study the result in c. - how fast this happens as a function of n.

Problem 2: Estimate standard deviation and confidence intervals with bootstrapping

Explain with words and an algorithm how you would proceed to use bootstrapping to estimate the standard deviation and the 95% confidence interval of one of the regression parameters in multiple linear regression. Comment on which assumptions you make for your regression model.

Problem 3: Implement problem 2

Implement your algorithm from 2 both using for-loop and using the `boot` function. Hint: see page 195 of our ISLR book. Use our SLID data set and provide standard errors for the coefficient for age. Compare with the theoretical value $(\mathbf{X}^T\mathbf{X})^{-1}\hat{\sigma}^2$ that you find in the output from the regression model.

```
library(car)
library(boot)
SLID = na.omit(SLID)
n = dim(SLID)[1]
SLID.lm = glm(wages ~ ., data = SLID)
summary(SLID.lm)$coeff[3, 2]
```

Todo: Bootstrap the CI, and compare to

```
confint(SLID.lm)
```

Exam 2018, Problem 1: K-nearest neighbour regression

Q1: Write down the formula for the K-nearest neighbour regression curve estimate at a covariate value x_0, and explain your notation.

$$\hat{f}(x_0) = \frac{1}{K} \sum_{i \in \mathcal{N}_0} y_i$$

Given an integer K and a test observation x_0, the KNN regression first identifies the K points in the training data that are closest (Euclidean distance) to x_0, represented by $\mathcal{N}_0$. It then estimates the regression curve at x_0 as the average of the response values for the training observations in $\mathcal{N}_0$.

Common mistakes in the exam papers: many answered how to do KNN-classification. In addition some wrote either x_i or $f(x_i)$ in place of y_i in the formula above.

Q3: Explain how 5-fold is performed, and specify which error measure you would use. Your answer should include a formula to specify how the validation error is calculated. A drawing would also be appreciated.

We look at a set of possible values for K in the KNN, for example $K = (1, 2, 3, 5, 7, 9, 15, 25, 50, 100)$. First we devide the data into a training set and a test set - and lock away the test set for model evaluation.

We work now with the training set.

5-fold CV: we divide the training data randomly into 5 folds of size $n/5$ each and call the folds $j = 1$, to $j = 5$.

For each value of K we do the following.

For $j = 1, \ldots, 5$:

- use the $4n/5$ observations from the folds except fold j to define the K-neighbourhood $\mathcal{N}_0$ for each of the observations in the jth fold
- the observations in the jth fold is left out and is the validation set, there are $n/5$ observations - and we denote them (x_{0jl}, y_{0jl}),
- we then estimate $\hat{f}(x_{0jl}) = \frac{1}{K} \sum_{i \in \mathcal{N}_0} y_i$.

- We calculate the error in the jth fold of the validation set as $\sum_l (y_{0jl} - \hat{f}(x_{0jl}))^2$ where the j is for the validation fold

The total error on the validation set is thus the validation MSE=$\frac{1}{n}\sum_{j=1}^{5}\sum_l (y_{0jl} - \hat{f}(x_{0jl}))^2$.

So, for each value of K we get an estimate of the validation MSE. Finally, we choose the value of K that gives the lowest validation MSE.

Common mistakes: many forget to shuffle data. Some forget to explain how this is related to choosing the number of neighbours in KNN (only focus on the 5-fold CV process) - that is, relate this to the setting of the problem. A few mention the error measure for classification.

Q4: In your opinion, would you prefer the leave-one-out cross-validation or the 5-fold cross-validation method? Justify your answer.

The LOOCV would give a less biased estimate of the MSE on a test set, since the sample size used $(n-1)$ is close to the sample size to be used in the real world situation (n), but the LOOCV will be variable (the variance of the validation MSE is high).

On the other hand the 5-fold CV would give a biased estimate of the MSE on a test set since the sample size used is 4/5 of the sample size that would be used in the real world situation. However, the variance will be lower than for the LOOCV.

When it comes to computational complexity we have in general that with LOOCV we need to fit n models but with 5-fold only 5. However, for KNN-regression there is really no model fitting, we only choose the K closest neighbours in the training part of the data. This means that LOOCV should not be more time consuming than 5-fold for KNN.

There is no "true" answer here – and arguments for both solutions can be given.

Common mistakes: many did not talk about bias, variance and computational complexity, a few made mistakes on the comparison of LOOCV and 5-fold CV with respect to bias and variance.

Summing up

Take home messages

- Use $k = 5$ or 10 fold cross-validation for model selection or assessment.
- Use bootstrapping to estimate the standard deviation of an estimator, and understand how it is performed before module 8 on trees.

Further reading

- Videoes on YouTube by the authors of ISL, Chapter 5, and corresponding slides
- Solutions to exercises in the book, chapter 5

R packages

```
# packages to install before knitting this R Markdown file to knit
# the Rmd
install.packages("knitr")
install.packages("rmarkdown")
# cool layout for the Rmd
install.packages("prettydoc")  # alternative to github
# plotting
install.packages("ggplot2")  # cool plotting
install.packages("ggpubr")  # for many ggplots
# datasets
install.packages("ElemStatLearn")
install.packages("ISLR")
# cross-validation and bootstrapping
install.packages("boot")
```