Module 5: Recommended Exercises

Statistical Learning V2021

alexaoh

20 mai, 2021

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

a) Draw a figure

Done by hand. The figure shows how the data is divided into k folds. In each iteration, $k-1$ folds are used to train the model, while the last fold is used to test after training.

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

k iterations are run in the simulation. In the first iteration, e.g. the first fold is left as a testing set and the remaining $k-1$ folds are used to train the model. The MSE is computed on the left out fold for each of these iterations. In the second iteration, e.g. the second fold is left as a testing set and the remaining folds are used to train. Then, the MSE is calculated on the left out fold. In this manner the simulation is run until all the k folds have been left out. After this, the k-fold approximation of the MSE is calculated as

$$\text{CV}_{(k)} = \frac{1}{k}\sum_{i=1}^{k} \text{MSE}_i,$$

where MSE_i are the calculated MSE's in each of the iterations on the left out fold. This assumes that the size of each fold is of size n/k. The more general formula is

$$\text{CV}_(k) = \frac{1}{n}\sum_{j=1}^{k} n_j \cdot \text{MSE}_j,$$

where n_j is the size of the j^{th} fold.

For classification situations, the same algorithm can be followed, but instead of calculating the MSE, the ratio of misclassified points is calculated in each iteration and the total (average) k-fold misclassification rate is found in the end, i.e. the loss function is adapted depending on the situation.

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.

k-fold cross-validation can (e.g.) be used to investigate which degree of polynomial one should use for lowest prediction error when building a multiple linear regression model or to find the optimal value of 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.

k-fold cross-validation can (e.g.) be used to investigate which degree of polynomial one should use for lowest prediction error when building a logistic regression model or to find the optimal value of k in KNN-classification. It could also be used to choose between LDA and QDA.

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.

a) *Advantages* of k-fold compared to validation set: Less variability than in validation set error, since the dependency on which set of observations are used for training and testing (which is very prominent with the validation set approach) is lower. Moreover, the bias is lower, since a bigger part of the data is used while training each of the models. In other words, the validation set approact is the “worst of both worlds” (larger variance and larger bias compared to k-fold CV). *Disadvantages*: More computationally intensive, since more simulations are run.
b) *Advantages* of k-fold compared to LOOCV: Less variance between different data sets compared to LOOCV, because in LOOCV we are averaging over data which is positively correlated. Less computationally expensive then LOOCV, since the folds are bigger than one. *Disadvantages* of k-fold: Larger bias, since less of the data is used when training, compared to LOOCV. There is no randomness in the splits in LOOCV.
c) The recommended values for k are 5 or 10, since these give a nice balance between variance and bias, such that none of these are very dominating. Too small values of k give small variance but high bias (e.g. $k = 2$), and too large values of k give small bias but high variance (e.g. LOOCV).

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?

Steps of such an algorithm are: Make a simulation with a large amount of predictors and a small amount of data. This data should be generated from a known distribution, e.g. a Gaussian distribution, since then the correct test error when performing model validation can be calculated. The prediction on the data is split into two parts. Firstly, a small amount of the predictors are chosen, according to the ones that have the largest correlation with the class labels in the data. Secondly, a classifier is applied only to the 100 predictors.

When performing CV in the wrong way, one first chooses the predictors and then runs CV only on these chosen predictors when used in the given classifier. More specifically, only the second step is used when performing CV, i.e. CV is used to calculate the test error when using only the small subset, selected based on correlation with the class labels, of the predictors.

When performing CV in the correct way, CV is performed in both steps. More specifically, the data is split into folds, in which the subset of predictors is chosen and the test error is calculated across all the splits. This is the correct way of doing it since both parts mentioned above are a part of the training process, which must be done in the CV.

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 class labels (here only 2). This means that the "truth"" is that the misclassification rate can not get very small. What is the expected 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 run CV ($k = 5$) on either only the d (=wrong way), or on all $c + d$ (=right way) predictors.
- 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.

We start by generating data for $n = 50$ observations

```r
library(boot)
# GENERATE DATA; use a seed for reproducibility
set.seed(4268)
n = 50  #number of observations
p = 5000  #number of predictors
d = 25  #top correlated predictors chosen
# generating predictor data
xs = matrix(rnorm(n * p, 0, 4), ncol = p, nrow = n)  #simple way to make uncorrelated predictors
dim(xs)  # n times p
```

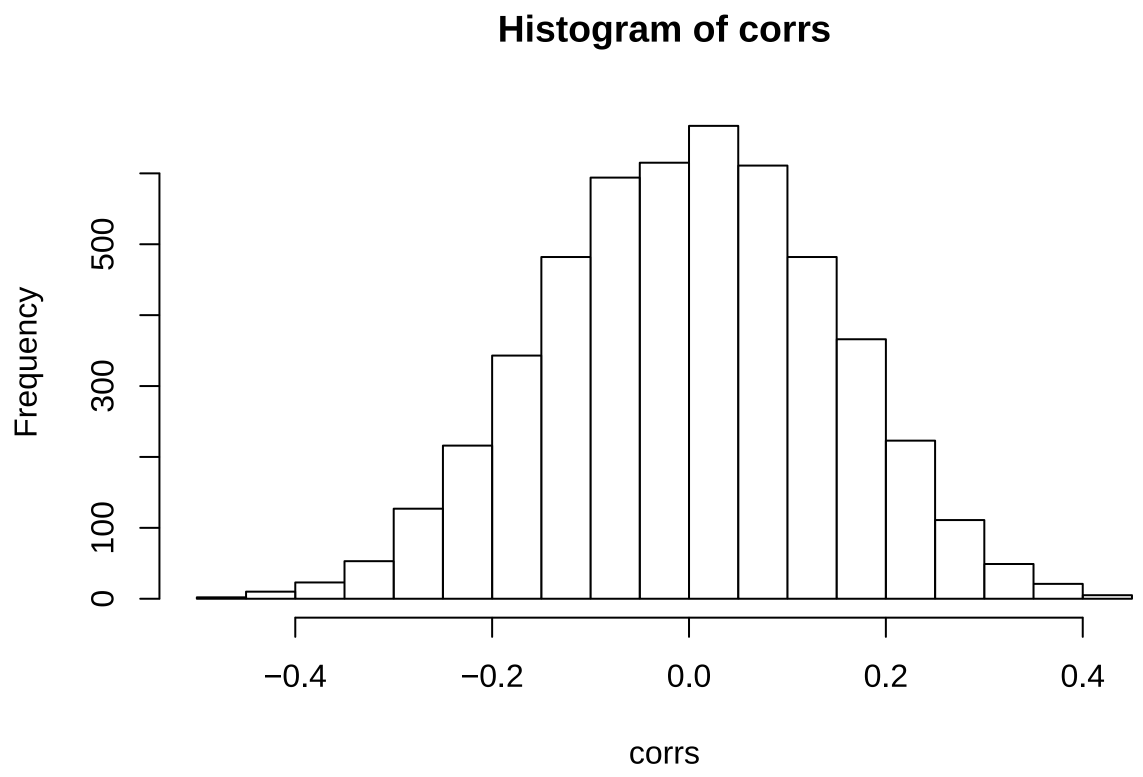
```
#> [1]   50 5000
```

```r
# 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)
```

```
#> ys
#>  0  1
#> 25 25
```

WRONG CV: Select the 25 most correlated predictors outside the CV.

```r
corrs = apply(xs, 2, cor, y = ys)
hist(corrs)
```



```r
selected = order(corrs^2, decreasing = TRUE)[1:d]  #top d correlated selected
data = data.frame(ys, xs[, selected])
```

Then run CV around the fitting of the classifier - use logistic regression and built in `cv.glm()` function

```r
logfit = glm(ys ~ ., family = "binomial", data = data)
cost <- function(r, pi = 0) mean(abs(r - pi) > 0.5)
kfold = 10
cvres = cv.glm(data = data, cost = cost, glmfit = logfit, K = kfold)
cvres$delta
```

```
#> [1] 0 0
```

Observe a zero misclassification rate!

CORRECT CV: Do not pre-select predictors outside the CV, but as part of the CV. We need to code this ourselves:

```r
reorder = sample(1:n, replace = FALSE)
validclass = NULL
validclass2 = NULL  # I tried a different way to predict from logistic regression (below)
# Leads to the same result!!
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]))
    pred2 = predict(trainlogfit, type = "response", newdata = validdata[,
        selected])
    validclass = c(validclass, ifelse(pred > 0.5, 1, 0))
    validclass2 = c(validclass2, ifelse(pred2 > 0.5, 1, 0))
}
table(ys[reorder], validclass)
```

```
#>    validclass
#>      0  1
#>   0  9 16
#>   1 12 13
```

```r
table(ys[reorder], validclass2)
```

```
#>    validclass2
#>      0  1
#>   0  9 16
#>   1 12 13
```

```r
1 - sum(diag(table(ys[reorder], validclass)))/n
```

```
#> [1] 0.56
```

```r
1 - sum(diag(table(ys[reorder], validclass2)))/n
```

```
#> [1] 0.56
```

Problem 4: 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.

Our sample size is n.

a. We draw one observation from our sample. What is the probability of drawing observation i (i.e., x_i)? And of not drawing observation i?

The probability of drawing observation x_i from our sample of size n is $\frac{1}{n}$. The probability of not drawing sample i is $\frac{n-1}{n} = 1 - \frac{1}{n}$.

b. We make n independent drawing (with replacement). What is the probability of not drawing observation i in any of the n drawings? What is then the probability that data point i is in our bootstrap sample (that is, more than 0 times)?

The probability of not drawing observation i in any of the n drawings is $(1 - \frac{1}{n})^n$. Then, the probability that the data point i is in our bootstrap sample is $1 - (1 - \frac{1}{n})^n$.

c. When n is large $(1 - \frac{1}{n})^n \approx \frac{1}{e}$. Use this to give a numerical value for the probability that a specific observation i is in our bootstrap sample.

A numerical value for the probability that a specific observation i is in our bootstrap sample is $1 - (1 - \frac{1}{n})^n \approx 1 - e^{-1} \approx 0.632$.

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 good is the approximation as a function of n?

```
n = 100
B = 10000
j = 1
res = rep(NA, B)
for (b in 1:B) res[b] = (sum(sample(1:n, replace = TRUE) == j) > 0)
mean(res)
```

```
#> [1] 0.6375
```

Problem 5: 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.

To estimate the standard deviation with the bootstrap I would make $B >> 0$ samples, with replacement, from the observations of the population that I have. For each of these samples, I would fit a multiple linear regression to find the given parameter in that case. This value would be stored for later. This exact algorithm should be done many times. In the end, finding the standard error of all the values of the parameter in question could work as an estimate of the standard deviation. The 95% confidence interval could be calculated similarly: by finding the lower 2.75% and 97.5% quantiles of the saved parameter estimates, we could attain a 95% confidence interval of the parameter.

The standard error can be found by the formula

$$\hat{SD}(\hat{\beta}) = \sqrt{\frac{1}{B-1}\sum_{b=1}^{B}(\hat{\beta}_b - \frac{1}{B}\sum_{b=1}^{B}\hat{\beta}_b)^2}.$$

Problem 6: Implement problem 5

Implement your algorithm from 5 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 = lm(wages ~ ., data = SLID)
summary(SLID.lm)$coeff["age", ]
```

```
#>      Estimate    Std. Error       t value      Pr(>|t|)
#>  2.551368e-01  8.714144e-03  2.927847e+01 7.822688e-171
```

Now go ahead and use bootstrap to estimate the 95% CI.

```
set.seed(4268)  # For reproducibility.

# Manually
B <- 1000
ages <- rep(NA, B)
for (b in 1:B) {
    index <- sample(x = nrow(SLID), size = nrow(SLID), replace = T)
    fit <- lm(wages ~ ., data = SLID, subset = index)
```

```r
    ages[b] <- coef(fit)["age"]
}

mean(ages)
```

```
#> [1] 0.2544882
```

```r
sd(ages)
```

```
#> [1] 0.009239405
```

```r
quantile(ages, c(0.025, 0.975))
```

```
#>      2.5%     97.5%
#> 0.2372161 0.2722875
```

```r
# Since the parameter estimates for Beta look approx normally
# distributed (below):
c(mean(ages) - 1.96 * sd(ages), mean(ages) + 1.96 * sd(ages))
```

```
#> [1] 0.2363789 0.2725974
```

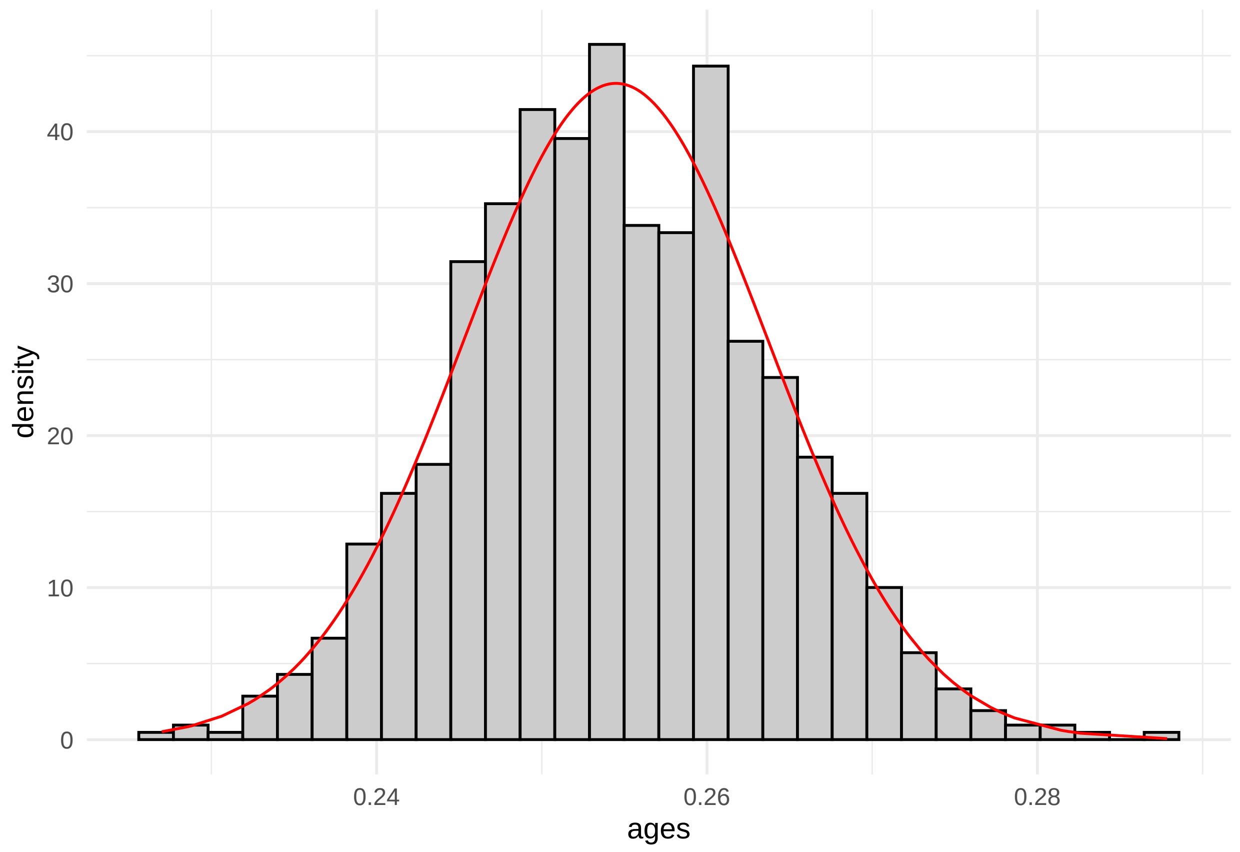
```r
# Using boot-function.
boot.fn <- function(data, index) {
    return(coef(lm(wages ~ ., data = data, subset = index))["age"])
}

boot(SLID, boot.fn, B)
```

```
#>
#> ORDINARY NONPARAMETRIC BOOTSTRAP
#>
#>
#> Call:
#> boot(data = SLID, statistic = boot.fn, R = B)
#>
#>
#> Bootstrap Statistics :
#>      original       bias    std. error
#> t1* 0.2551368 0.0003940973 0.008948562
```

```r
confint(SLID.lm)  # Compare to what R gives us when fitting the model.
```

```
#>                      2.5 %     97.5 %
#> (Intercept)    -9.0891576 -6.6884008
#> education       0.8484610  0.9847670
#> age             0.2380522  0.2722214
#> sexMale         3.0452719  3.8655493
#> languageFrench -0.8518577  0.8214111
#> languageOther  -0.4946904  0.7798996
```

```r
library(ggplot2)
data = data.frame(beta_hat = ages, norm_den = dnorm(ages, mean(ages),
    sd(ages)))
ggplot(data) + geom_histogram(aes(x = ages, y = ..density..), fill = "grey80",
    color = "black") + geom_line(aes(x = ages, y = norm_den), color = "red") +
    theme_minimal()
```


density
40
30
20
10
0
0.24
0.26
0.28
ages