Module 5: RESAMPLING

TMA4268 Statistical Learning V2019

Mette Langaas, Department of Mathematical Sciences, NTNU

week 6 2019

Contents

Last changes: (05.02: added link to R Markdown intro, under Interactive lecture. 04.02. added class notes with improved drawing for two layers of CV)

Introduction

Learning material for this module

- James et al (2013): An Introduction to Statistical Learning. Chapter 5.
- Classnotes 04.02.2019

Additional material for the interested reader: Chapter 7 (in particular 7.10) in Friedman et al (2001): Elements of Statistical learning.

Move to

- Introduction
- Cross-validation and Recommended exercises on cross-validation
- Bootstrapping and Recommended exercises on bootstrapping
- Summing up
- Further reading
- R packages

What will you learn?

- What is model assessment and model selection?
- Ideal solution in a data rich situation.
- Cross-validation: validation set - LOOCV and k-fold CV - what is the best?
- Bootstrapping - how and why.
- Summing up
- The plan for the interactive lesson.

Generalization performance of learning method

- prediction capacity on independent test data
- inference and understanding

This is important both for

Model selection

estimate the *performance* of different models (often different order of complexity within one model class) to *choose the best model*.

Model assessment

having chosen a final model, estimating its performance (prediction error) on new data.

Example

We aim to do *model selection* in KNN-regression, where true curve is $f(x) = -x + x^2 + x^3$ with $x \in [-3, 3]$. $n = 61$ for the training data.

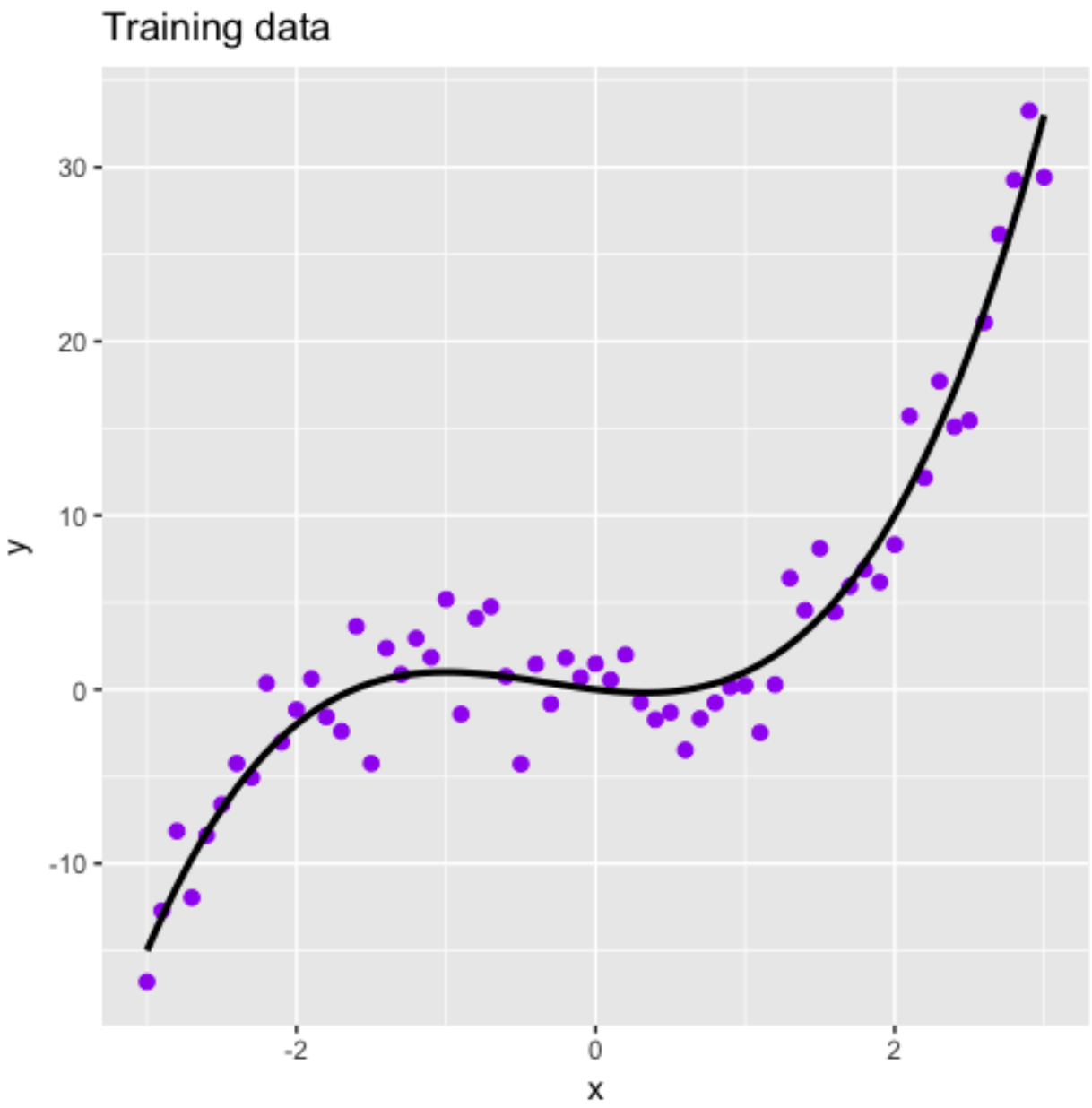


Figure 1: Regression problem: training data and true regression curve

KNN-regression

$n = 61$ both for training and for test data (using same x-grid).

K small: high complexity (left) and K large: low complexity (right).

Take home message: the training error rate is often very different from the test error rate - and looking at the training error to estimate the test error will dramatically underestimate the latter.

The bias-variance trade-off

Left (high complexity): low squared-bias (red) and high variance (green). Right (low complexity): high squared-bias (red) and low variance (green).

Loss functions

reminder - we will use

- Mean squared error (quadratic loss) for regression problems:

$$Y_i = f(\mathbf{x}_i) + \varepsilon_i \text{ for } i = 1, \ldots, n \text{ and MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(\mathbf{x}_i))^2$$

- 0/1 loss for classification problems:

$$P(Y = j \mid \mathbf{x}_0) \text{ for } j = 1, \ldots, K$$

  and classify to the class with the highest probability $\hat{y}_i$. Then

$$\frac{1}{n}\sum_{i=1}^{n} \mathrm{I}(y_i \neq \hat{y}_i)$$

  will give the misclassification rate.

The challenge

our example was based on simulated data, so I had unlimited access to data. Now, let us move to real data.

Data rich situation

If we had a large amount of data we could divide our data into three parts:

- training set: to fit the model
- validation set: to select the best model (aka model selection)
- test set: to assess how well the model fits on new independent data (aka model assessment)

Q: Before we had just training and test. Why do we need the additional validation set?

A: We have not discussed model selection before.

Q: Why can't we just use the training set for training, and then the test set both for model selection and for model evaluation?

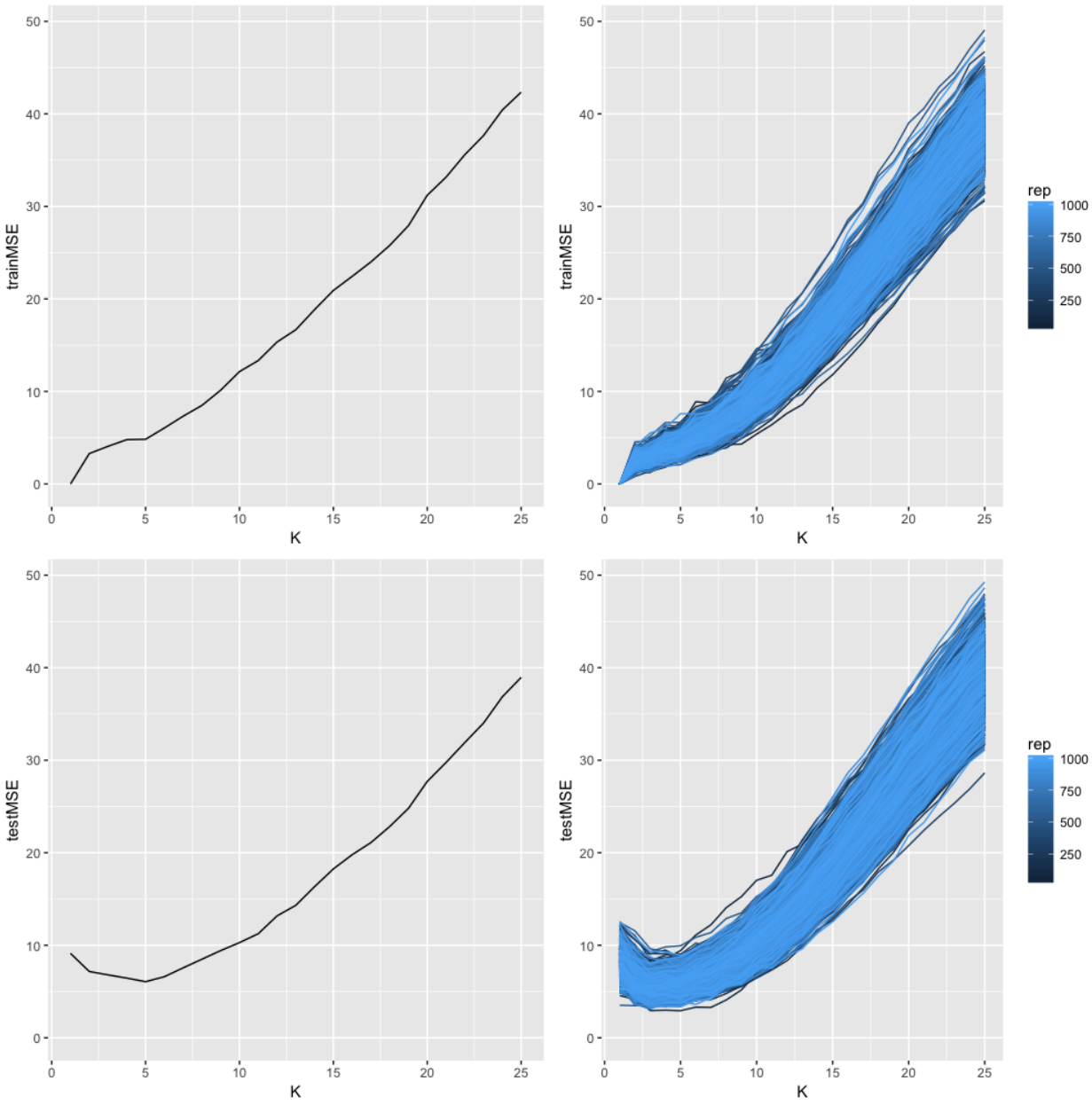


Figure 2: Regression problem: MSE for training and test set, M=1000 versions

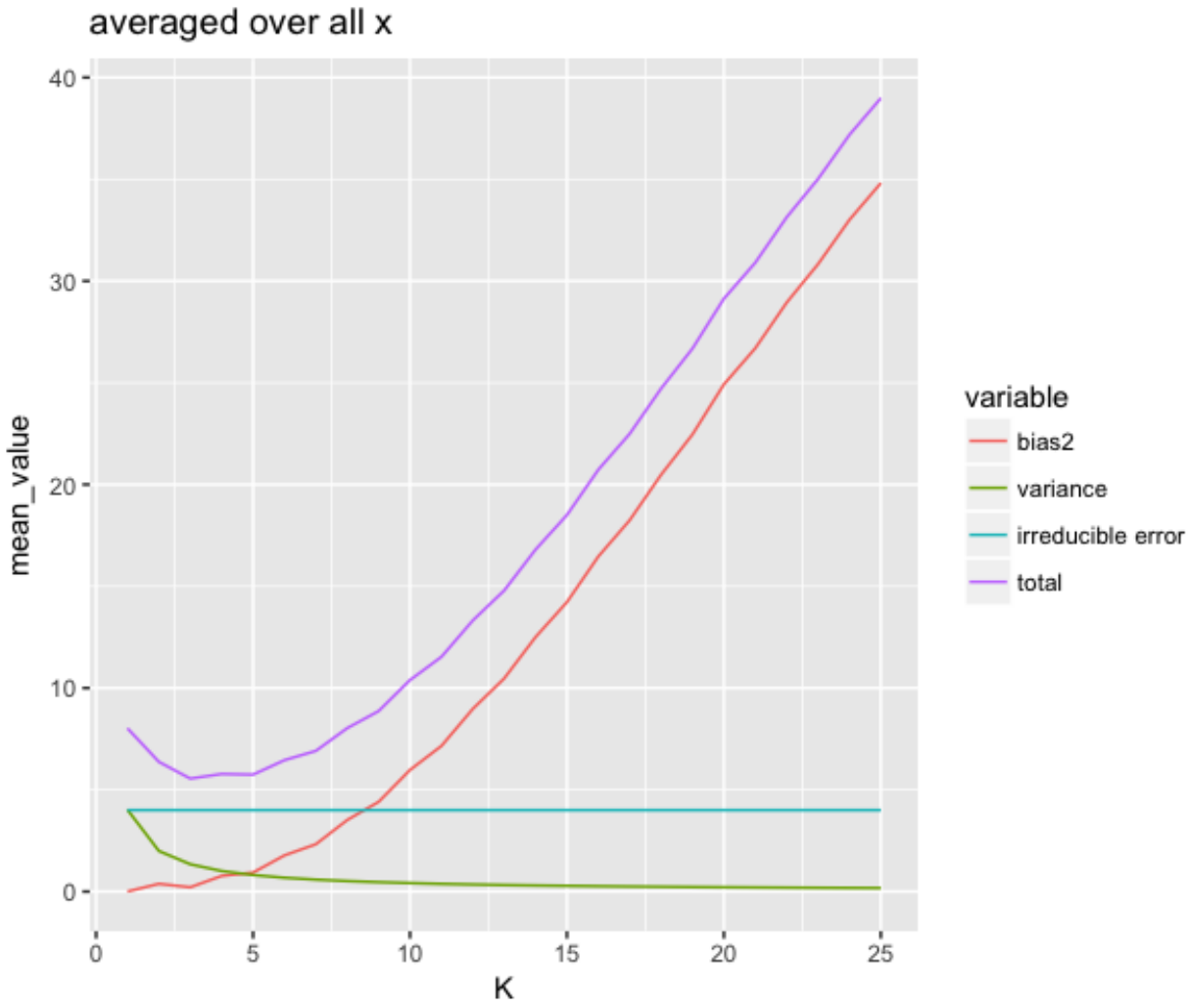


Figure 3: Regression problem: bias-variance traceoff

A: We will too optimistic if we report the error on the test set when we have already used the test set to choose the best model.

If this is the case - great - then you do not need Module 5. But, this is very seldom the case - so we will study other solutions based on efficient sample reuse with *resampling* data.

An alternative strategy for model selection (using methods penalizing model complexity, e.g. AIC or lasso) is covered in Module 6.

First we look at *crossvalidation*, then at *bootstrapping*.

Cross-validation (CV)

Consider the following "model selection" situation: We assume that test data is available (and has been put aside), and we want to use the rest of our data to both fit the data and to find the best model.

This can be done by:

- the validation set approach (what we have already looked at above)
- leave one out cross validation (LOOCV)
- 5 and 10 fold crossvalidation (CV)

We will also discuss that there is a "right and a wrong way" to to CV

- selection bias - all elements of a model selection strategy need to be within the CV-loop
- recommended exercises

The validation set approach

Consider the case when you have a data set consisting of n observations.

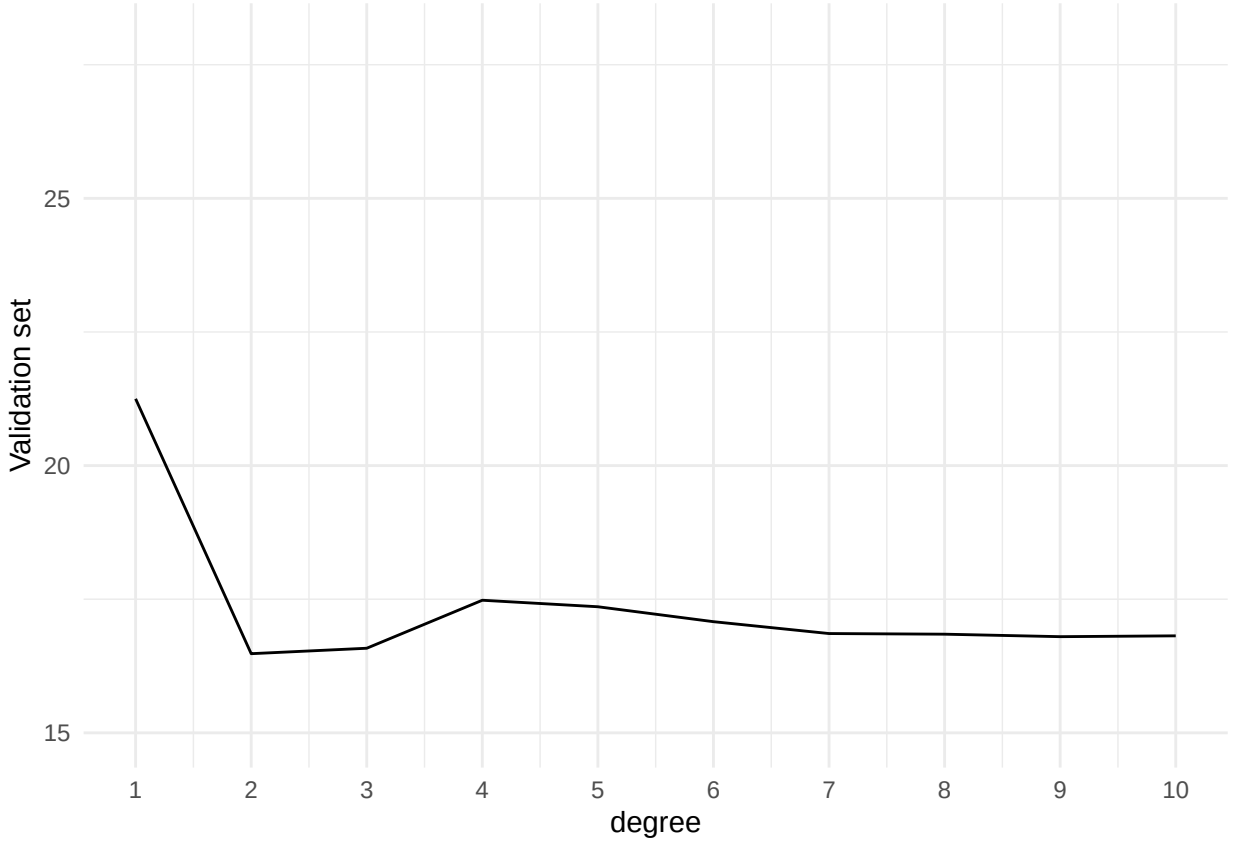
To fit a model and to evaluate its predictive performance you randomly divide the data set into two parts ($n/2$ sample size each):

- a *training set* (to fit the model) and
- a *validation set* (to make predictions of the response variable for the observations in the validation set)

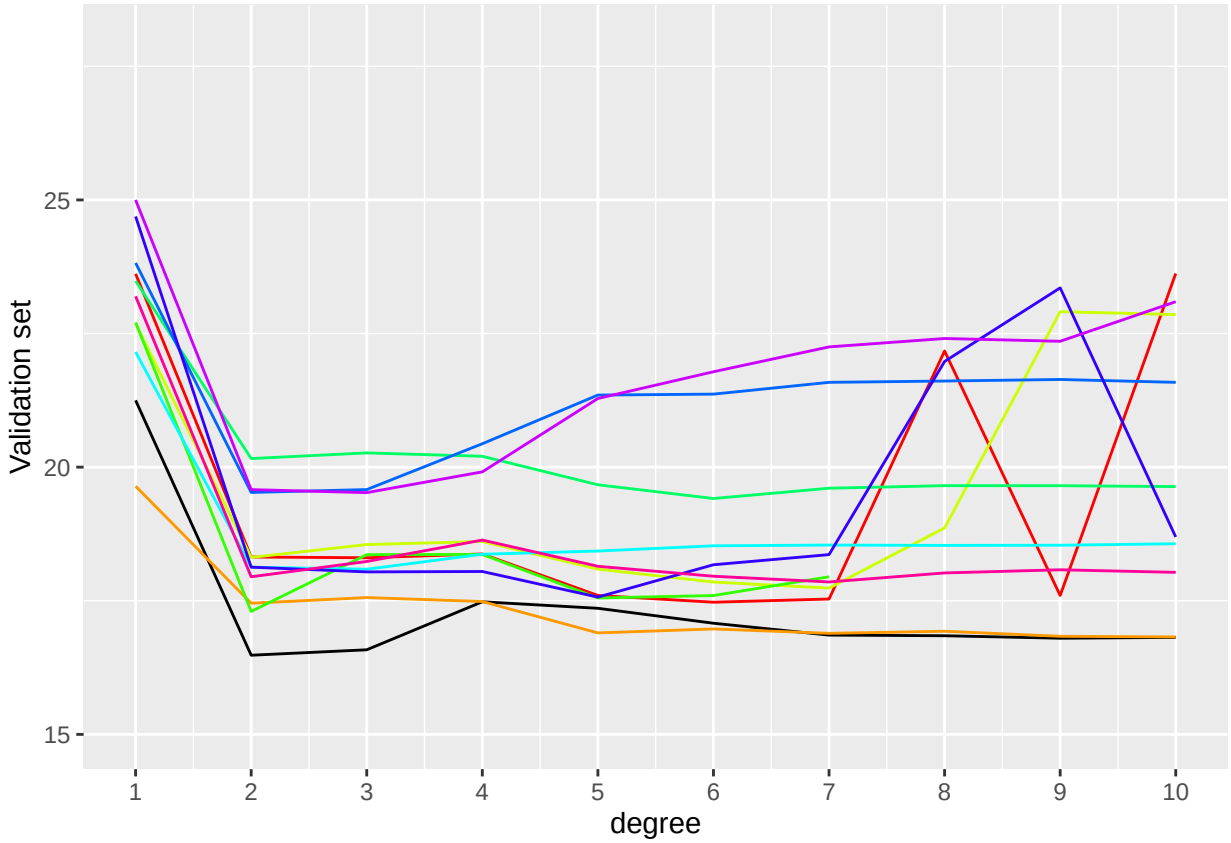
Remember: focus is on model selection (more in Module 6 - for example how to perform model selection with linear regression).)

Regression model selection example: validation set error

Auto data set (library `ISLR`): predict `mpg` (miles pr gallon) using polynomial function of `horsepower` (of engine), $n = 392$. What do you see?


25
20
15
Validation set
1
2
3
4
5
6
7
8
9
10
degree

Regression example: validation set error for many random divisions



Drawbacks with the validation set approach

- high variability of validation set error - since this is dependent on which observation are included in the training and validation set
- smaller sample size for model fit - since not all observations can be in the training set
- the validation set error may tend to overestimate the error rate on new observations for a model that is fit on the full data set (because - the more data the lower error, and here our training set is half of our data set).

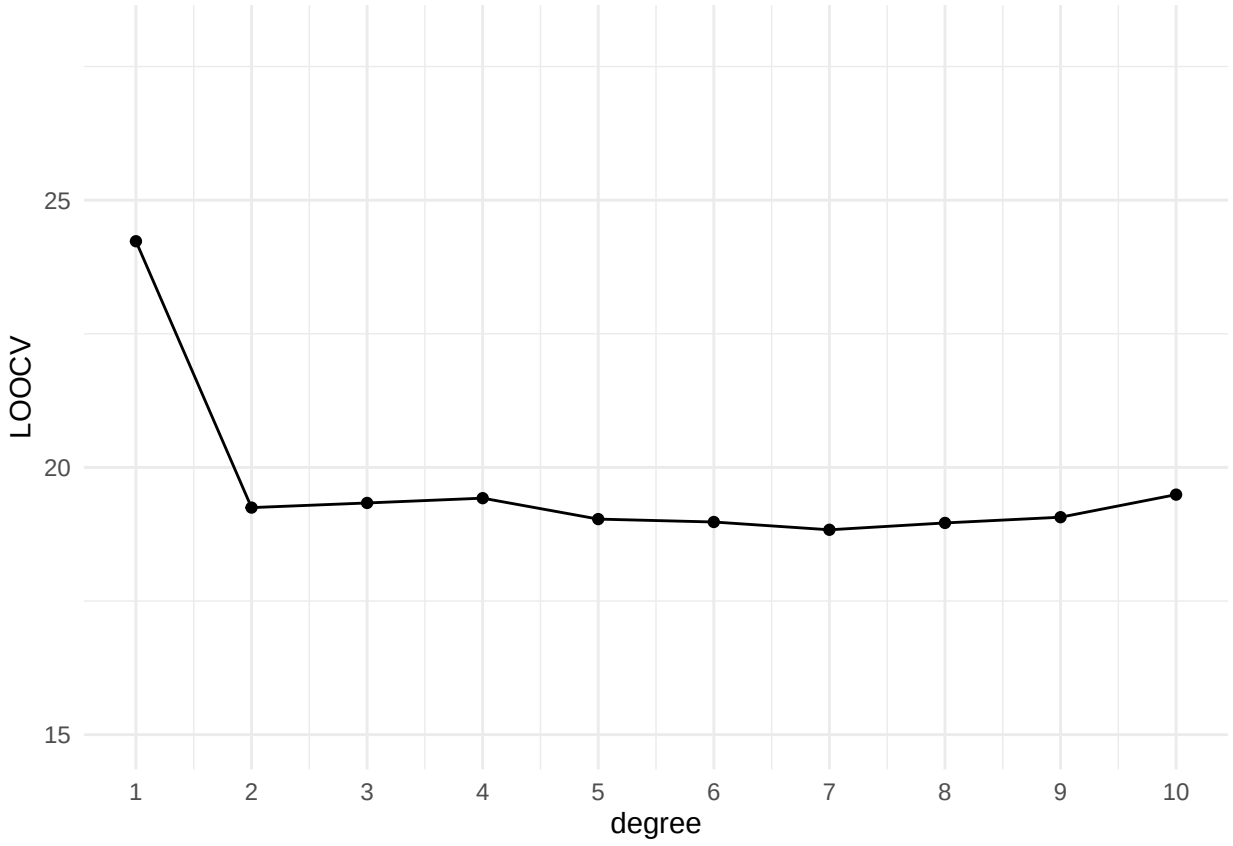
Leave-one-out cross-validation (LOOCV)

- If the data is very limited and the division of the data into two parts is unreasonable, leave-one-out cross-validation (LOOCV) can be used.
- In LOOCV one observation at a time is left out and makes up the new observations (test set).
- The remaining $n - 1$ observations make up the training set.
- The procedure of model fitting is repeated n times, such that each of the n observations is left out once.
- The total prediction error is the mean across these n models.

$$\text{MSE}_i = (y_i - \hat{y}_i)^2$$

$$CV_n = \frac{1}{n}\sum_{i=1}^{n} \text{MSE}_i$$

Regression example: LOOCV



```
library(ISLR)  #for Auto data set
library(boot)  #for cv.glm
library(ggplot2)  #for plotting
set.seed(123)
n = dim(Auto)[1]
testMSEvec = NULL
start = Sys.time()
for (polydeg in 1:10) {
    glm.fit = glm(mpg ~ poly(horsepower, polydeg), data = Auto)
    glm.cv1 = cv.glm(Auto, glm.fit, K = n)
    testMSEvec = c(testMSEvec, glm.cv1$delta[1])
}
stopp = Sys.time()
yrange = c(15, 28)
plotdf = data.frame(testMSE = testMSEvec, degree = 1:10)
g0 = ggplot(plotdf, aes(x = degree, y = testMSE)) + geom_line() + geom_point() +
    scale_y_continuous(limits = yrange) + scale_x_continuous(breaks = 1:10) +
    labs(y = "LOOCV")
g0 + theme_minimal()
```

Issues with leave-one-out cross-validation

- Good:
  - no randomness in training/validation splits!
  - little bias since nearly the whole data set used for training (compared to half for validation set approach)
- Bad:

– expensive to implement - need to fit n different models - however nice formula for linear model LOOCV - but not genereally so
– high variance since: two training sets only differ by one observation - which makes estimates from each fold highly correlated and this can lead to that their average can have high variance.

$$\mathrm{Var}(\sum_{i=1}^{n} a_i X_i + b) = \sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j \mathrm{Cov}(X_i, X_j)$$

$$= \sum_{i=1}^{n} a_i^2 \mathrm{Var}(X_i) + 2\sum_{i=2}^{n}\sum_{j=1}^{i-1} a_i a_j \mathrm{Cov}(X_i, X_j).$$

LOOCV for multiple linear regression

$$CV_n = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \hat{y}_i}{1 - h_{ii}}\right)^2$$

where h_i is the ith diagonal element (leverage) of the hat matrix $\mathbf{H} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$.

---

k-fold cross-validation

See class notes for drawing.

Formally

- Indices of observations - divided into k folds: $C_1, C_2, \ldots, C_k$.
- n_k elements in each fold, if n is a multiple of k then $n_k = n/k$.

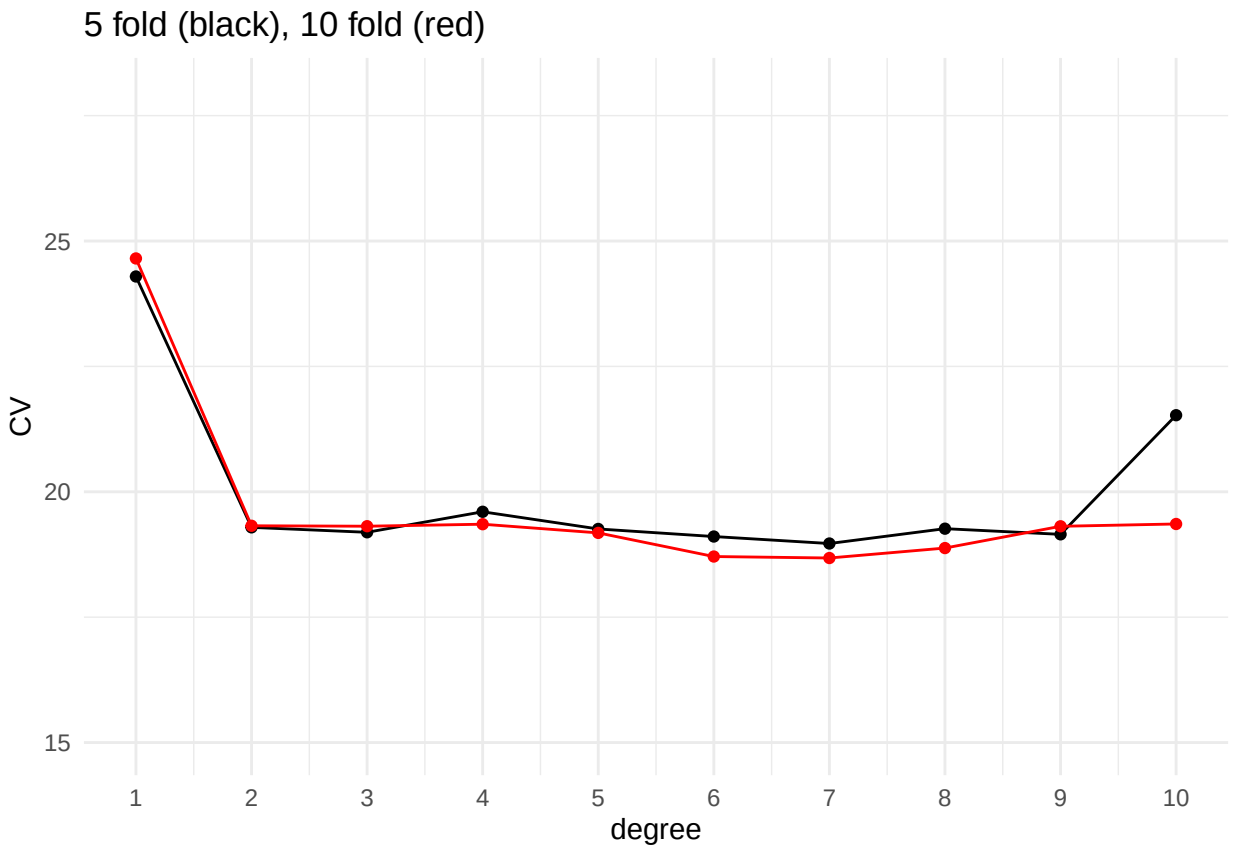
$$\mathrm{MSE}_k = \frac{1}{n_k}\sum_{i \in C_k}(y_i - \hat{y}_i)^2$$

where $\hat{y}_i$ is the fit for observation i obtained from the data with part k removed.

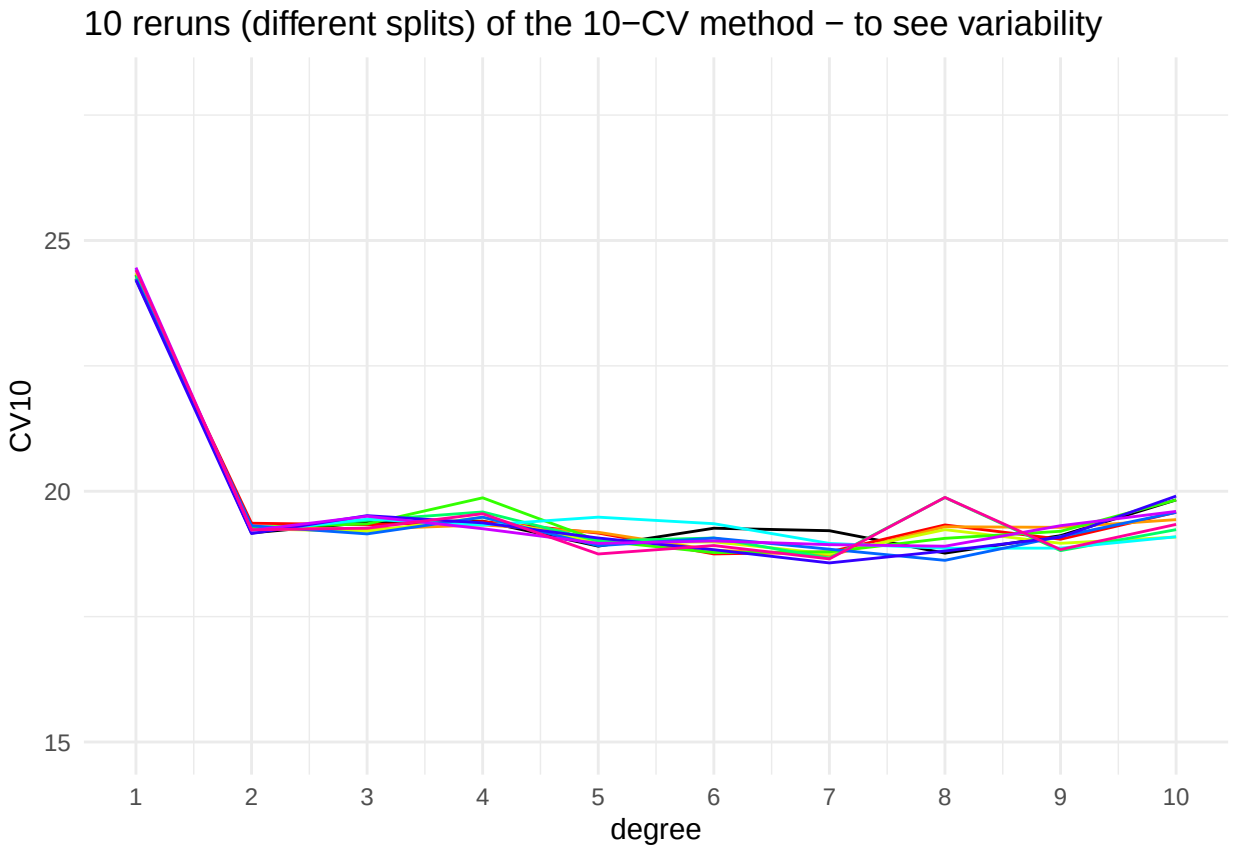
$$CV_k = \frac{1}{n}\sum_{j=1}^{k} n_j \mathrm{MSE}_j$$

Observe: setting $k = n$ gives LOOCV.

Regression example: 5 and 10-fold cross-validation




```
library(ISLR)
library(boot)
library(ggplot2)
set.seed(123)
n = dim(Auto)[1]
testMSEvec5 = NULL
testMSEvec10 = NULL
start = Sys.time()
for (polydeg in 1:10) {
    glm.fit = glm(mpg ~ poly(horsepower, polydeg), data = Auto)
    glm.cv5 = cv.glm(Auto, glm.fit, K = 5)
    glm.cv10 = cv.glm(Auto, glm.fit, K = 10)
    testMSEvec5 = c(testMSEvec5, glm.cv5$delta[1])
    testMSEvec10 = c(testMSEvec10, glm.cv10$delta[1])
}
stopp = Sys.time()
yrange = c(15, 28)
plotdf = data.frame(testMSE5 = testMSEvec5, degree = 1:10)
g0 = ggplot(plotdf, aes(x = degree, y = testMSE5)) + geom_line() + geom_point() +
    scale_y_continuous(limits = yrange) + scale_x_continuous(breaks = 1:10) +
    labs(y = "CV") + ggtitle("5 and 10 fold CV")
g0 + geom_line(aes(y = testMSEvec10), colour = "red") + geom_point(aes(y = testMSEvec10),
    colour = "red") + ggtitle("5 fold (black), 10 fold (red)") + theme_minimal()
```



Issues with k-fold cross-validation

1. As for the validation set, the result may vary according to how the folds are made, but the variation is in general lower than for the validation set approach.
2. Computational issues: less work with $k = 5$ or 10 than LOOCV.
3. The training set is $(k-1)/k$ of the original data set - this will give and estimate of the prediction error that is biased upwards.
4. This bias is the smalles when $k = n$ (LOOCV), but we know that LOOCV has high variance.
5. This is way often $k = 5$ or $k = 10$ is used as a compromise between 3 and 4.

Choosing the best model

Remember that we *randomly divide* the data into k folds and then perform the CV for all possible model that we want to choose between.

We have not directly indicated that there is a model parameter (maybe K in KNN), say θ, involved to calculate CV_j, $j = 1, \ldots, k$

Smallest CV-error:

- Based on the CV-plot we may choose the best model (θ) to be the model with the smallest CV_k.
- We then fit this model using the whole data set (not the test part, that is still kept away), and evaluate the performance on the test set.

One standard error rule:

Denote by $\text{MSE}_j(\theta)$, $j = 1, \ldots, k$ the k parts of the MSE that together give the CV_k.

We can compute the sample standard deviation of $\text{MSE}_j(\theta)$, $j = 1, \ldots, k$ for each value of the complexity parameter θ, say $\text{SD}(\theta)$ and then use $\text{SE}(\theta) = \text{SD}(\theta)/\sqrt{k}$ as the standard deviation of CV_k.

The *one standard error rule* is to choose the simplest (e.g. highest value of K in KNN or lowest polynomial degree in the polynomial example) θ, that is, we move θ in the direction of simplicity until it ceases to be true that

$$\text{CV}(\theta) \leq \text{CV}(\hat{\theta}) + \text{SE}(\hat{\theta})$$

This will be the simplest model whose error is within one standard error of the minimal error.

---

k-fold cross-validation in classification

- what do we need to change from our regression set-up?

For LOOCV $\hat{y}_i$ is the fit for observation i obtained from the data with observations i removed, and $\text{Err}_i = I(y_i \neq \hat{y}_i)$. LOOCV is then

$$CV_n = \frac{1}{n} \sum_{i=1}^{n} \text{Err}_i$$

The k-fold CV is defined analogously.

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.

Can we use CV for model assessment?

Assume that we have a method where we not need to perform model selection (maybe this is done using the methods from Module 6, that is AIC or lasso), but want to perform model assessment based on all our data.

Then we can use CV with all data (then the validation part is really the test part) and report on the model performance using the validation parts of the data as above.

Can we use CV both for model selection and model assessment?

Then you can use two layers of CV - also called *nested CV*. See drawing in class.

The right and the wrong way to do cross-validation

ISL book slides, page 17: model assessment.

- We have a two-class problem and would like to use a simple classification method, however,
- we have many possible predictors $p = 5000$ and not so big sample size $n = 50$.

We use this strategy to produce a classifier:

1. We calculate the correlation between the class label and each of the p predictors, and choose the $d = 25$ predictors that have the highest (absolute value) correlation with the class label. (We need to have $d < n$ to fit the logistic regression uniquely.)
2. Then we fit our classifier (here: logistic regression) using only the $d = 25$ predictors.

How can we use cross-validation to produce an estimate of the performance of this classifier?

Q: Can we apply cross-validation only to step 2? Why or why not?

Can we apply cross-validation only to step 2?

A: No, step 1 is part of the training procedure (the class labels are used) and must be part of the CV to give an honest estimate of the performance of the classifier.

- Wrong: Apply cross-validation in step 2.
- Right: Apply cross-validation to steps 1 and 2.

We will see in the Recommended Exercises that doing the wrong thing can give a misclassification error approximately 0 - even if the "true" rate is 50%.

Selection bias in gene extraction on the basis of microarray gene-expression data

Article by Christophe Ambroise and Geoffrey J. McLachlan, PNAS 2002: Direct quotation from the abstract of the article follows.

- In the context of cancer diagnosis and treatment, we consider the problem of constructing an accurate prediction rule on the basis of a relatively small number of tumor tissue samples of known type containing the expression data on very many (possibly thousands) genes.
- Recently, results have been presented in the literature suggesting that it is possible to construct a prediction rule from only a few genes such that it has a negligible prediction error rate.
- However, in these results the test error or the leave-one-out cross-validated error is calculated without allowance for the selection bias.

- There is no allowance because the rule is either tested on tissue samples that were used in the first instance to select the genes being used in the rule or because the cross-validation of the rule is not external to the selection process; that is, gene selection is not performed in training the rule at each stage of the crossvalidation process.
- We describe how in practice the selection bias can be assessed and corrected for by either performing a crossvalidation or applying the bootstrap external to the selection process.
- We recommend using 10-fold rather than leave-one-out cross-validation, and concerning the bootstrap, we suggest using the so-called .632 bootstrap error estimate designed to handle overfitted prediction rules.
- Using two published data sets, we demonstrate that when correction is made for the selection bias, the cross-validated error is no longer zero for a subset of only a few genes.

Recommended exercises on cross-validation

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

- draw a figure,
- specify algorithmically what is done,
- and in particular how the "results" from each fold are aggregated,
- relate to one example from regression (maybe is complexity wrt polynomials of increasing degree in multiple linear regression or K in KNN-regression?)
- relate to one example from classification (maybe is complexity wrt polynomials of increasing degree in logistic regression or K in KNN-classification?)

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

Problem 2: What are the advantages and disadvantages of k-fold cross-validation

relative to

- the validation set approach
- leave one out cross-validation (LOOCV)
- 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. 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.

```r
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)
```

```r
# 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
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
```

The Bootstrap

- flexible and powerful statistical tool that can be used to quantify *uncertainty* associated with an estimator or statistical learning method
- we will look at getting an estimate for the standard error of a sample median and of a regression coefficient
- in Module 8 - bootstrapping is the core of the *ensemble method* referred to at *bagging*=bootstrap aggregation,
- in TMA4300 Computation statistics - a lot more theory on the bootstrap.

The inventor: Bradley Efron in 1979 - see interview.

The name? *To pull oneself up by one's bootstraps* from "The Surprising Adventures of Baron Munchausen" by Rudolph Erich Raspe:

The Baron had fallen to the bottom of a deep lake. Just when it looked like all was lost, he thought to pick himself up by his own bootstraps.

Idea: Use the data itself to get more information about a statistic (an estimator).

Example: the standard deviation of the sample median?

Assume that we observe a random sample $X_1, X_2, \ldots, X_n$ from an unknown probability distribution f. We are interesting in saying something about the population median, and to do that we calculate the sample median $\tilde{X}$. But, how accurate is $\tilde{X}$ as an estimator?

The bootstrap was introduced as a computer-based method to estimate the standard deviation of an estimator, for example our estimator $\tilde{X}$.

But, before we look at the bootstrap method, first we assume that we know F and can sample from F, and use simulations to answer our question.

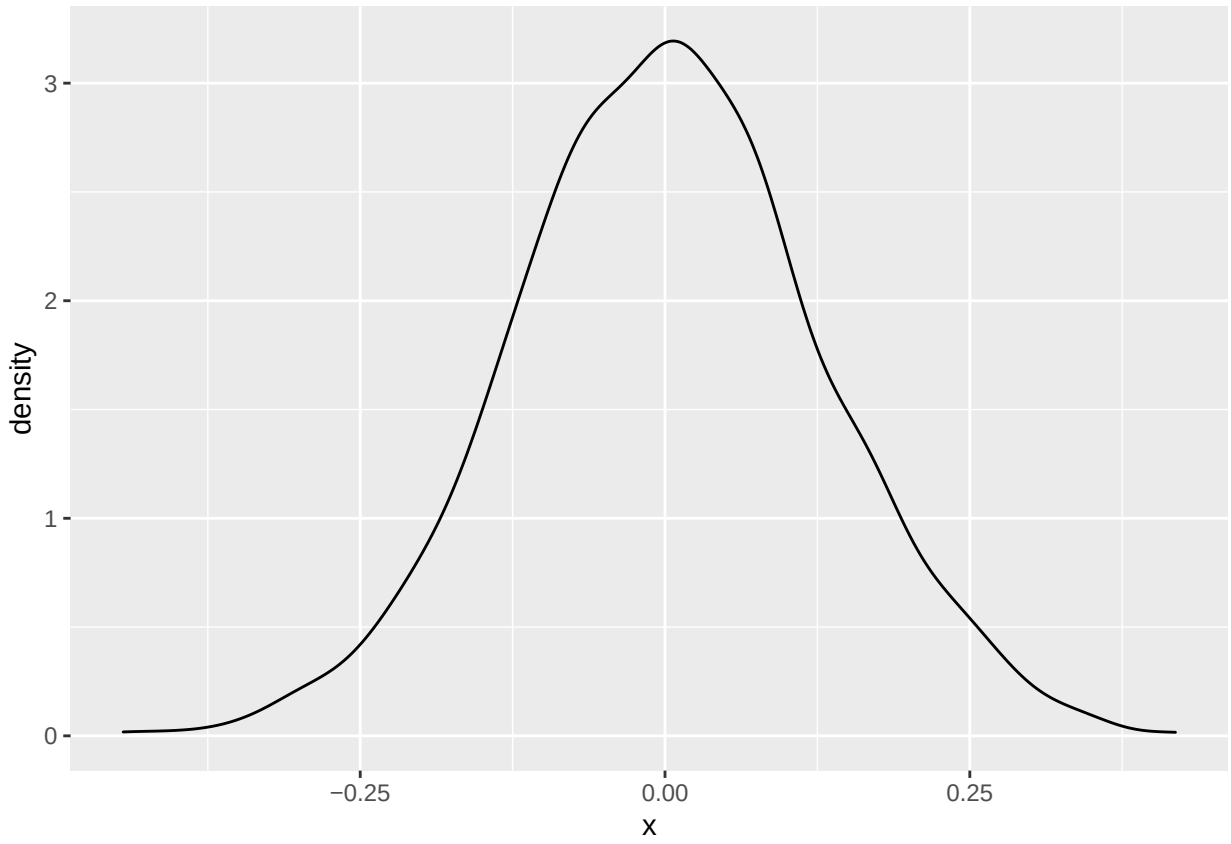
```
set.seed(123)
n = 101
B = 1000
estimator = rep(NA, B)
for (b in 1:B) {
    xs = rnorm(n)
    estimator[b] = median(xs)
}
sd(estimator)
```

```
## [1] 0.1259035
```

```
# approximation for large samples (sd of median of standard normal)
1.253 * 1/sqrt(n)
```

```
## [1] 0.1246782
```

```
ggplot(data = data.frame(x = estimator), aes(x = x)) + geom_density()
```



Moving from simulation to bootstrapping

The bootstrap method is using the observed data to estimate the *empirical distribution* $\hat{f}$, that is each observed value of x is given probability $1/n$.

A *bootstrap sample* $X_1^*, X_2^*, \ldots, X_n^*$ is a random sample drawn from $\hat{f}$.

A simple way to obtain the bootstrap sample is to *draw with replacement from* $X_1, X_2, \ldots, X_n$.

This means that our bootstrap sample consists of n members of $X_1, X_2, \ldots, X_n$ - some appearing 0 times, some 1, some 2, etc.

```
set.seed(123)
n = 101
original = rnorm(n)
median(original)
```

```
## [1] 0.05300423
```

```
boot1 = sample(x = original, size = n, replace = TRUE)
table(table(boot1))
```

```
##
##  1  2  3  4
## 34 22  5  2
```

```r
# how many observations have not been selected in our boot1 sample?
n - sum(table(table(boot1)))
```

```
## [1] 38
```

```r
median(boot1)
```

```
## [1] -0.02854676
```

The bootstrap algorithm for estimating standard errors

1. B bootstrap samples: drawn with replacement from the original data
2. Evaluate statistic: on each of the B bootstrap samples to get $\tilde{X}_b^*$ for the bth bootstrap sample.
3. Estimate squared standard error by:

$$\frac{1}{B-1}\sum_{b=1}^{B}(\tilde{X}_b^* - \frac{1}{B}\sum_{b=1}^{B}\tilde{X}_b^*)^2$$

---

with for-loop in R

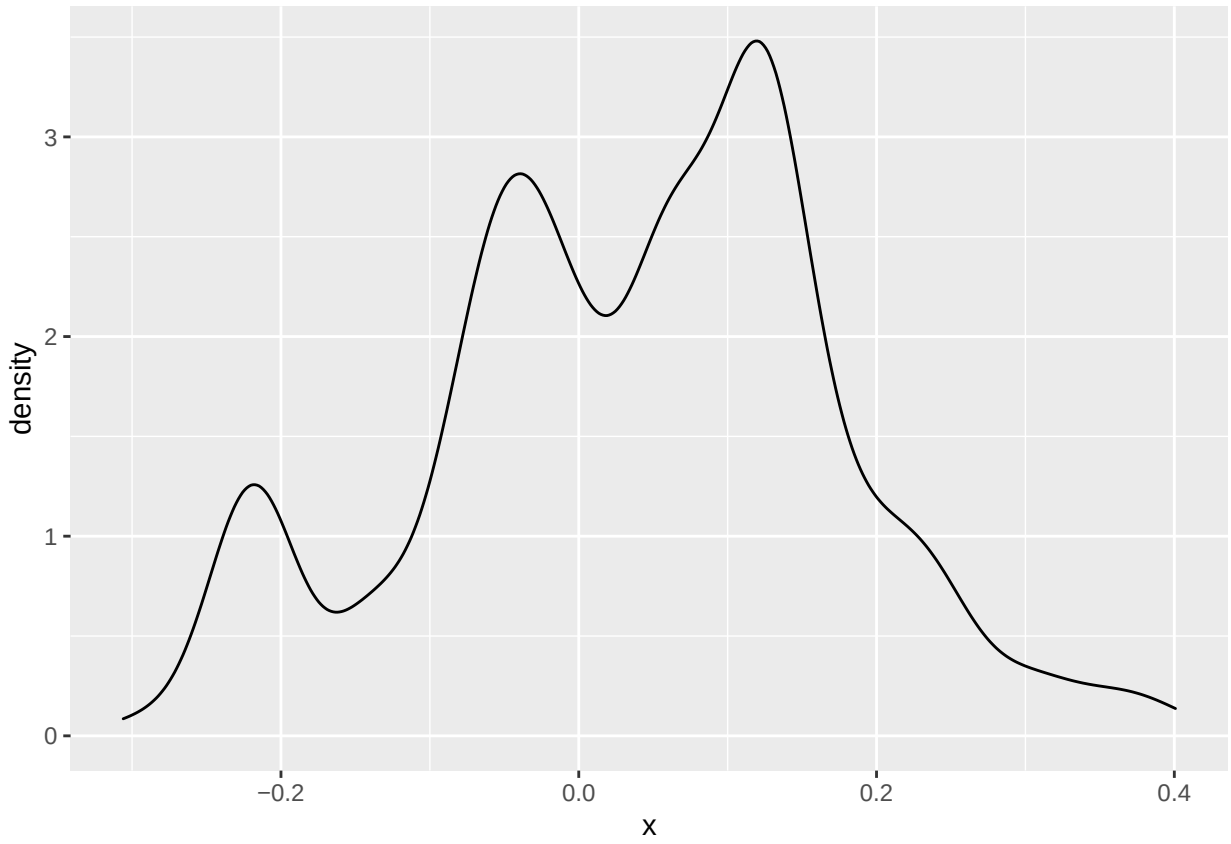
```r
set.seed(123)
n = 101
original = rnorm(n)
median(original)
```

```
## [1] 0.05300423
```

```r
B = 1000
estimator = rep(NA, B)
for (b in 1:B) {
    thisboot = sample(x = original, size = n, replace = TRUE)
    estimator[b] = median(thisboot)
}
sd(estimator)
```

```
## [1] 0.1365448
```

```r
ggplot(data = data.frame(x = estimator), aes(x = x)) + geom_density()
```



---

using built in `boot` function from library `boot`

```
library(boot)
set.seed(123)
n = 101
original = rnorm(n)
median(original)
```

```
## [1] 0.05300423
```

```
summary(original)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -2.30917 -0.50232  0.05300  0.08248  0.68864  2.18733
```

```
boot.median = function(data, index) return(median(data[index]))
B = 1000
boot(original, boot.median, R = B)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = original, statistic = boot.median, R = B)
```

```
##
##
## Bootstrap Statistics :
##       original      bias    std. error
## t1* 0.05300423 -0.01577692   0.1290482
```

With or without replacement?

In bootstrapping we sample *with replacement* from our observations.

Q: What if we instead sample *without replacement*?

A: Then we would always get the same sample - given that the order of the sample points is not important for our estimator.

(In permutation testing we sample without replacement to get samples under the null hypothesis - a separate field of research.)

Example: multiple linear regression

We assume, for observation i:

$$Y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + ... + \beta_p x_{ip} + \varepsilon_i,$$

where $i = 1, 2, ..., n$. The model can be written in matrix form:

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}.$$

The least squares estimator: $\hat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}$ has $\text{Cov}(\boldsymbol{\beta}) = \sigma^2(\mathbf{X}^T\mathbf{X})^{-1}$.

We will in the recommended exercises look at how to use bootstrapping to estimate the covariance of the estimator. Why is that "needed" if we already know the mathematical formula for the standard deviation? Answer: not needed - but OK to look at an example?

We will not do this here - but our bootstrap samples can also be used to make confidence intervals for the regression coefficients or prediction intervals for new observations. This means that we do not have to rely on assuming that the error terms are normally distributed!

Bagging

Bagging is a special case of *ensemble methods*.

In Module 8 we will look at bagging, which is built on bootstrapping the the fact that it is possible to reduce the variance of a prediction by taking the average of many model fits.

- Assume that we have B different predictors $X_1, X_2, \ldots, X_B$. We have built them on B different bootstrap samples.
- All are predictors for some parameter μ and that all have some unknown variance σ^2.
- We then decide that we want to use all the predictors together - equally weighted - and make $\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$, which we often use to predict μ.

We can therefore obtain a new model (our average of the individual models) that has a smaller variance than each of the individual model because

$$\text{Var}(\bar{X}) = \frac{\sigma^2}{n}$$

Since this averaged predictor has smaller variance than each of the predictors we would assum that this is a more accurate prediction. However, the interpretation of this bagged prediction might be harder then for the separate predictors.

Models that have poor prediction ability (as we may see can happen with regression and classification trees) might benefit greatly from bagging. More in Module 8.

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 with bootstrapping

Explain with words and an algorithm how you would proceed to use bootstrapping to estimate the standard deviation 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]
```

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.

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.

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")
```