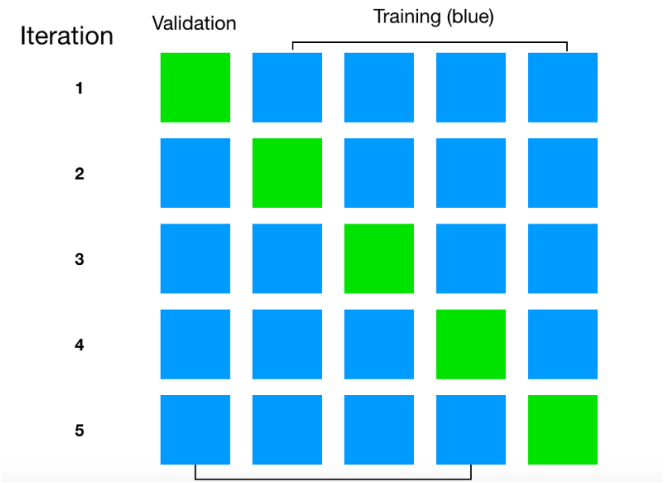
R-course:
Machine Learning using R

Cross-validation

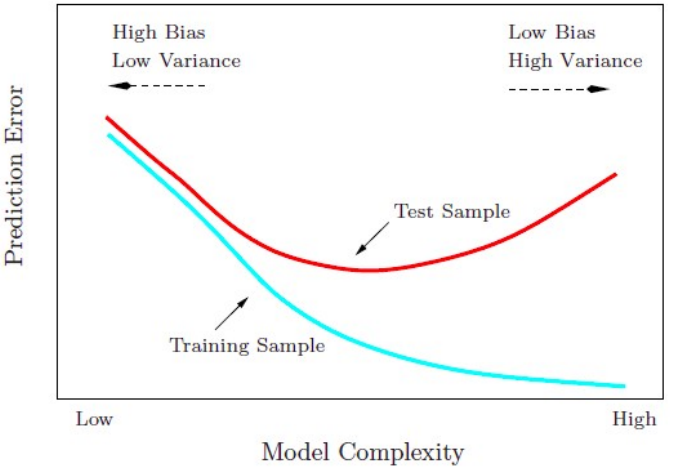


Yannick Rothacher

Zürich, 2021

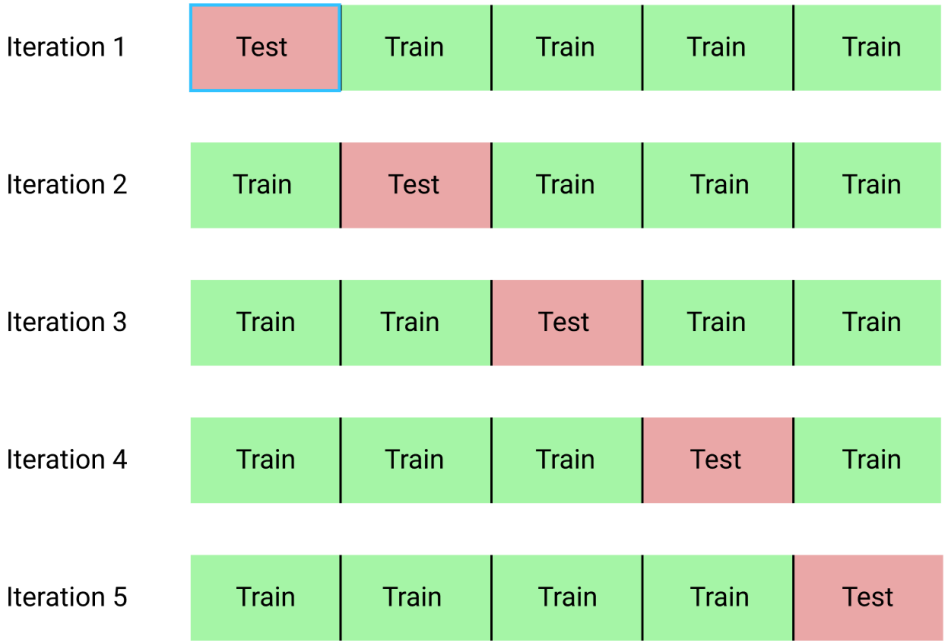
What is cross-validation?

- Recap last lecture: To test how well a classifier is performing we wish to test its predictions on **new "test" data**
  - Often an additional test data set is not available
- We want a method to estimate the test-error using only one data set
  - Cross-validation is exactly doing this
- Cross-validation works by **splitting** the data into multiple test- and training-sets and calculates the test-error for each test-set



What is cross-validation?

- K-fold cross-validation (e.g. 5-fold below)

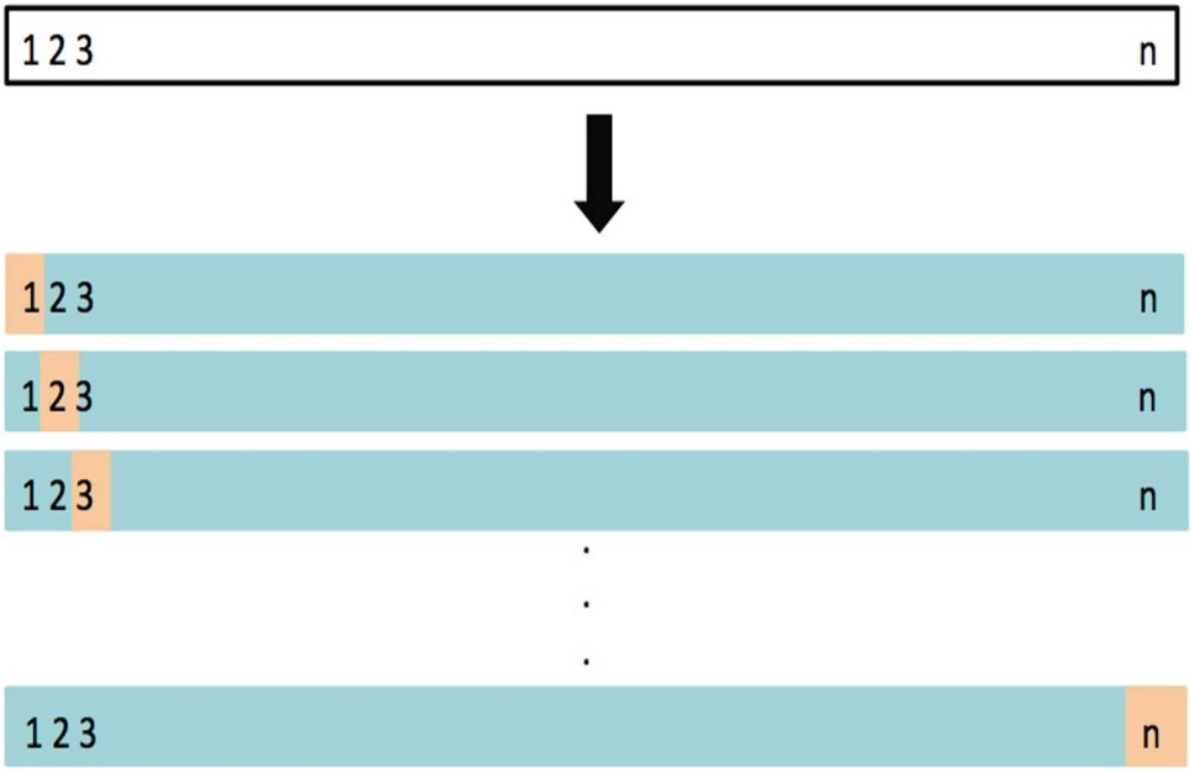


Also referred to as “validation sets”

- For each iteration use one k^{th} of the data as a test-set (fitting the classifier to the remaining data)
  - In the end **each observation** has been part of a test-set once
- Calculate the final test error based on all iterations

Leave-One-Out Cross-validation

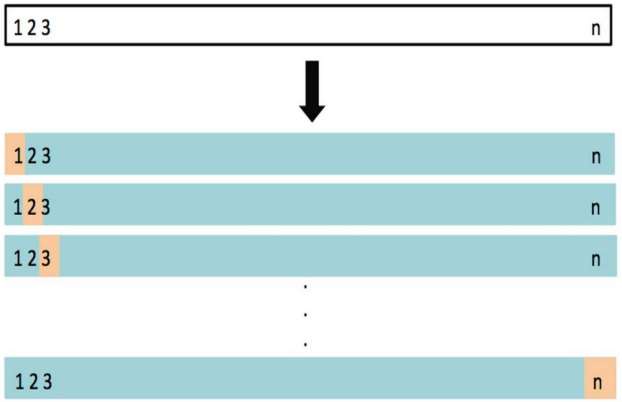
- Same as k-fold cross-validation but letting each observation be an independent test-set
- Each iteration: Use blue data as training-set to predict the orange observation.



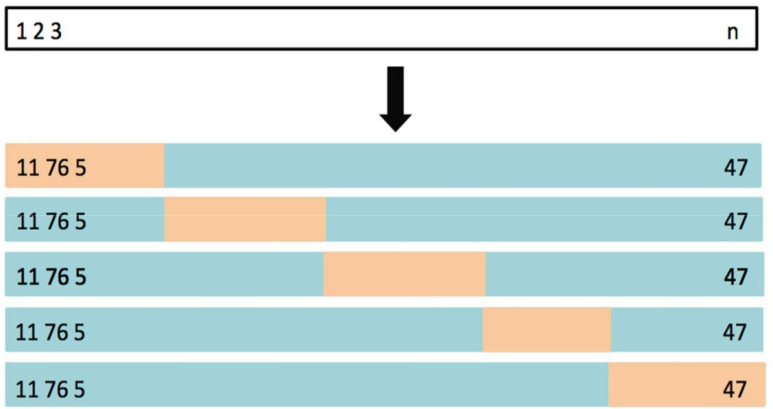
How to choose fold-size (k)?

- The fold size (k) can be chosen between **2 and n** (k=n results in leave-one-out CV)
- What value is best for k?
  - Leave-one-out CV has lowest possible bias but higher variance. Can sometimes be slow.
  - K-fold CV is usually faster and has lower variance. But has higher bias and is random.
- Standard today is to use **k=10 fold CV**

k=n



k=5



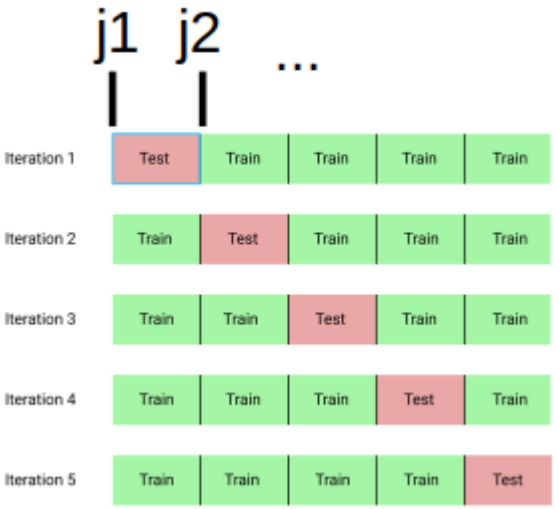
Cross-validation in R

- Since cross-validation follows a rather simple procedure, one can easily write a CV function
- Original plan: Exercise of writing a CV function
  - Here: Will go through complete CV function in R together
- Then will look at an R-package for cross-validation

- Steps in CV function (for KNN):
  - 1) Divide the data into k training- and test-sets
  - 2) Fit the KNN classifier to each training-set and measure its performance on the corresponding test-set
  - 3) Collect the results

Complete cross-validation function

```
KNN_crossVal <- function(data, label, k_fold=10, KNN_k=1){
  stopifnot(nrow(data)==length(label), is.factor(label),
            (1<k_fold & k_fold<=nrow(data)), all(apply(data, 2, is.numeric)),
            KNN_k<=nrow(data))
  # Create k sub-selections
  n <- nrow(data)
  ind_s <- sample(1:n)
  ind.L <- list()
  j1 <- 1
  for (i in 1:k_fold){
    j2 <- (i*n) %/% k_fold
    ind.L[[i]] <- ind_s[j1:j2]
    j1 <- j2+1
  }
  # Now run KNN on each selection (and collect results):
  confMat <- matrix(0,nrow=nlevels(label), ncol=nlevels(label))
  for(fold in 1:k_fold){
    ind_fold <- ind.L[[fold]]
    testDat <- data[ind_fold,]
    trainDat <- data[-ind_fold,]
    test.solu <- label[ind_fold]
    train.solu <- label[-ind_fold]
    knn.pred <- class::knn(train = trainDat, test = testDat,
                    cl = train.solu, k = KNN_k)
    confMat.fold <-  table(knn.pred, test.solu)
    confMat <- confMat + confMat.fold
  }
  # Calculate estimated test-error
  missMat <- confMat
  diag(missMat) <- 0
  missCount <- sum(missMat)
  test.err <- missCount/n
  L.res <- list(k_fold=k_fold, KNN_k=KNN_k, Indices=ind.L,
                confMatrix=confMat, errorRate=test.err)
  return(L.res)
}
# Try out function:
dat <- iris[,-5]
label <- iris$Species
KNN_crossVal(data = dat, label = label, k_fold = 10, KNN_k = 5)
```



→ Show in R...

R-Packages for cross-validation

- There are different R-packages with implemented CV procedures
- The **Caret** package is one example
- Caret is a package which can generally be used to train machine learning models
  - We will use it tomorrow for that purpose
- Caret has different validation procedures implemented, k-fold-CV is just one possibility

Crossvalidation with caret

- Using caret usually involves three steps:
  - 1) Defining a "tuning grid"
  - 2) Defining a validation procedure
  - 3) Training the model
- Example of 10-fold CV:

```
library(caret)
# Define tune grid:
t_grid <- data.frame(k=c(1, 10, 50))   # Define k for knn
# Define validation procedure:
train_crt <- trainControl(method = "cv", number = 10)
# "Train" the model:
set.seed(9823)
model <- train(Species ~., data = iris,
               method = "knn", tuneGrid=t_grid,
               trControl = train_crt)
# Look at output:
model
```

```
k-Nearest Neighbors

150 samples
  4 predictor
  3 classes: 'setosa', 'versicolor', 'virginica'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 135, 135, 135, 135, ...
Resampling results across tuning parameters:

  k   Accuracy   Kappa
   1  0.9600000  0.94
  10  0.9666667  0.95
  50  0.9200000  0.88

Accuracy was used to select the optimal
model using the largest value.The final value used
for the model was k = 10.
```

Crossvalidation with caret

- For classification caret returns two performance measures
  - "**Accuracy**" which is the rate of correct predictions
  - "**Kappa**" which is a similar measure but "normalized to a baseline of random success" (especially important for imbalanced data)
- We can also run a CV for regression problems
- For regression there are also different performance measures
  - "**Root Mean Squared Error**" (RMSE) which is the mean of the squared differences between predicted and true values

$$RMSE = \sqrt{\frac{1}{n}\sum_{i}^{n}(y_i - \hat{y}_i)^2}$$

  - "**R-squared**" which is the proportion of variation in the target variable explained by the model

→ Show in R...