

Lecture 6

Generalization

ISLR 5, ESL 7,8



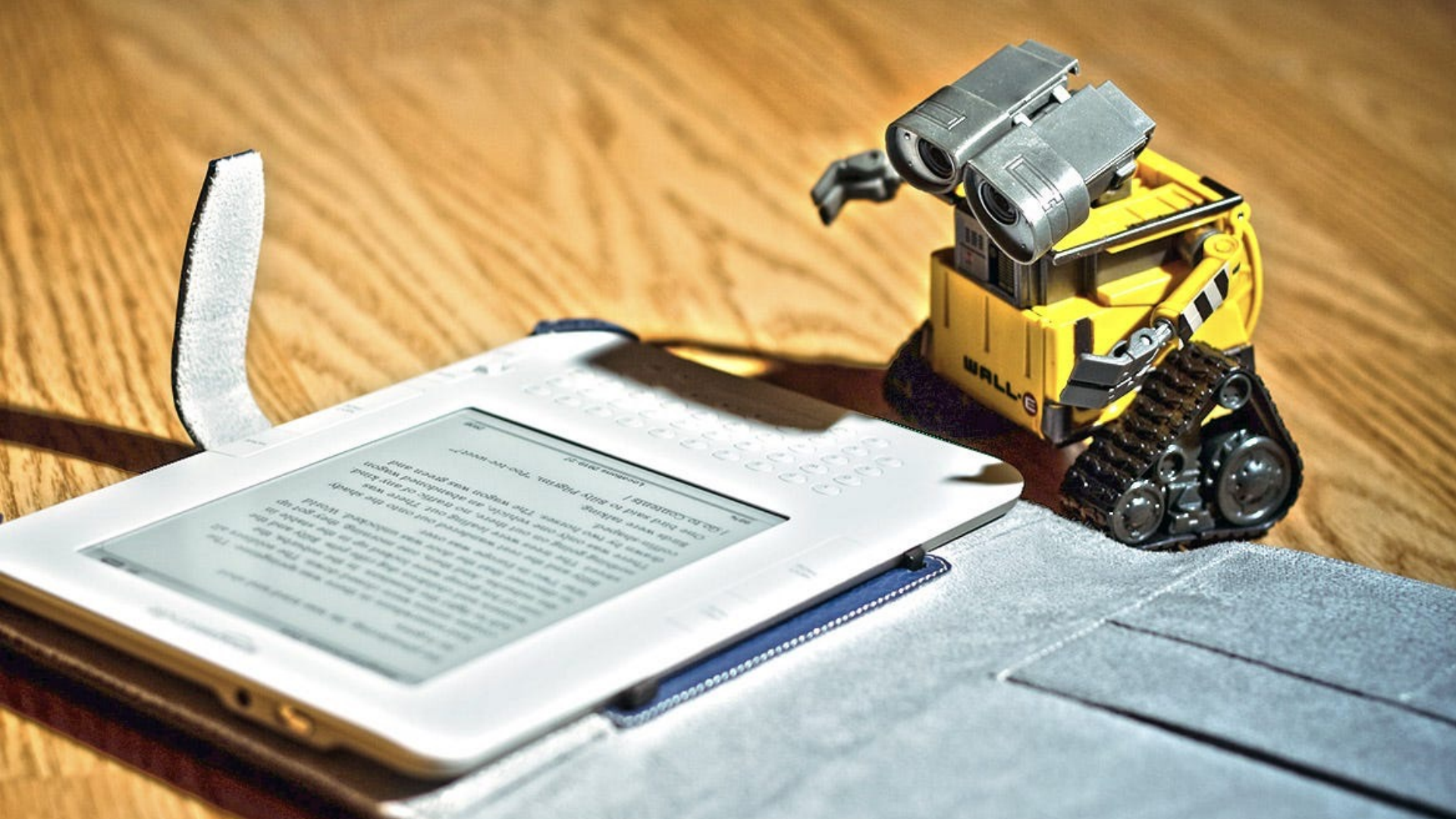
Krikamol Muandet
Jilles Vreeken



UNIVERSITÄT
DES
SAARLANDES



CISPA
HELMHOLTZ CENTER FOR
INFORMATION SECURITY

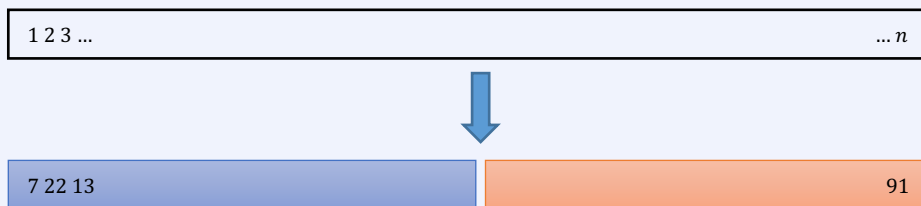


Validation

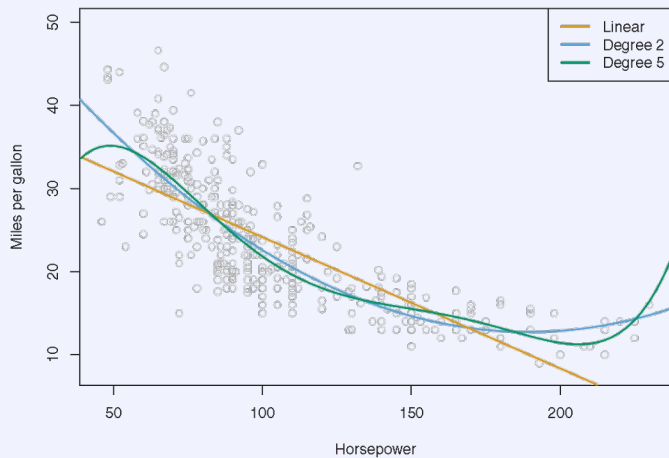
To estimate how well a model **generalizes**, we should test on **different data** than we trained on

The most basic approach is to **randomly** divide the data into two

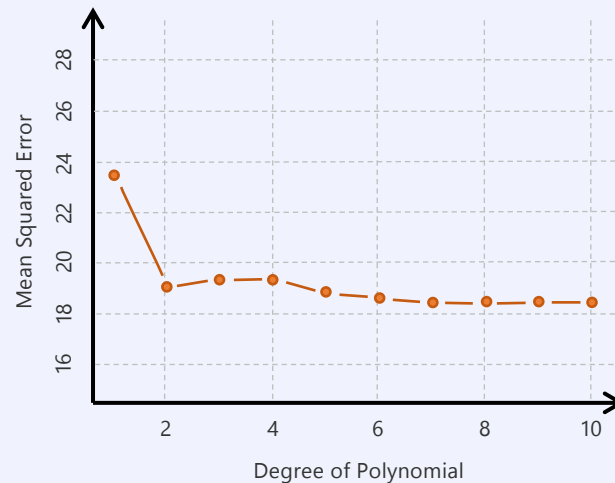
- we fit our models using the **training** data
- we estimate the **generalization error** (aka test error) on the **test** data (aka hold-out set)



Example Validation



$$\text{mpg} = \beta_0 + \beta_1 \times \text{horsepower} + \beta_2 \times \text{horsepower}^2 + \varepsilon$$



Validation error for one random 50-50 splits into training and validation set

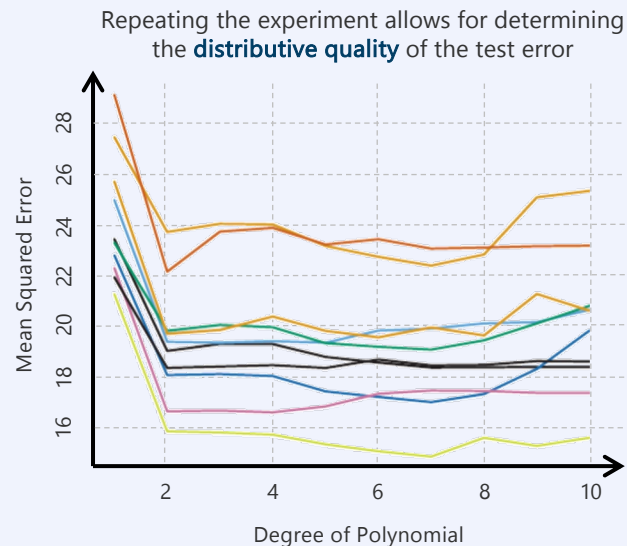
Validation Set Approach

Observations

1. test error can vary widely
2. quadratic term affords large improvement of test error
3. higher-order terms not of benefit

Problems

1. high variability of estimated test error
2. training set becomes smaller;
model may be suboptimal (undertrained);
test error may be overestimated



Validation error for one random 50-50 splits into training and validation set

Leave-one-out Cross Validation (LOOCV)

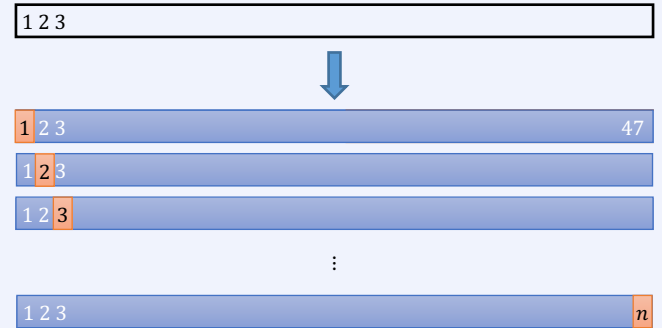
Key idea: set only **one data point aside** for testing

- training set is now as large as can be, so little bias
- but, only one point to test on, so high variance

Repeating for every data point averages out variance

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

- process is **deterministic**, repeating always gives same result



Leave-one-out Cross Validation (LOOCV)

Key idea: set only **one data point aside** for testing

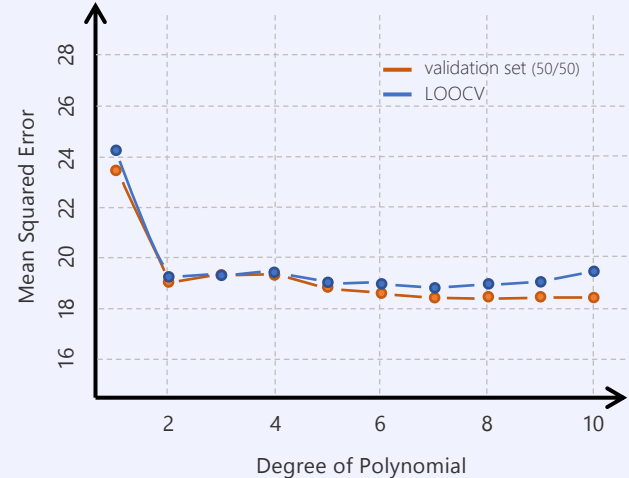
- training set is now as large as can be, so little bias
- but, only one point to test on, so high variance

Repeating for every data point averages out variance

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

- process is **deterministic**, repeating always gives same result
- for least-squares linear or polynomial regression we have

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



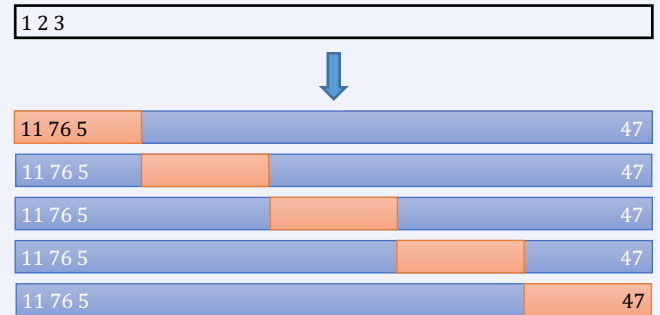
k -fold Cross Validation

Randomly divide the data into k folds

- train on $k-1$ folds, test on the remaining 1 fold
- repeat such that all folds have been tested on
- gives k estimates of the test error, the final estimate is

$$CV_{(k)} = \frac{1}{k} \sum_{i=1}^k MSE_i$$

- in practice, we use $k = 5$ or 10
- LOOCV is k -fold CV with $k = n-1$
- k -fold CV is more efficient but has higher bias than LOOCV



k -fold Cross Validation

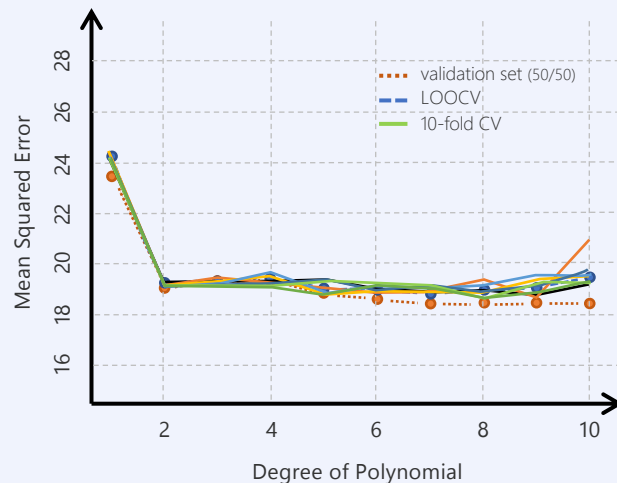
Randomly divide the data into k folds

- train on $k-1$ folds, test on the remaining 1 fold
- repeat such that all folds have been tested on
- gives k estimates of the test error, the final estimate is

$$CV_{(k)} = \frac{1}{k} \sum_{i=1}^k MSE_i$$

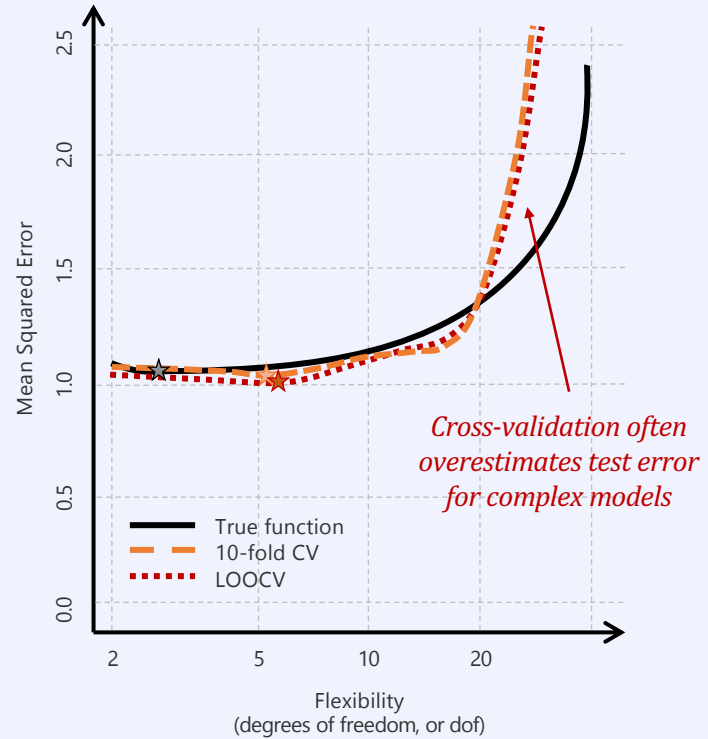
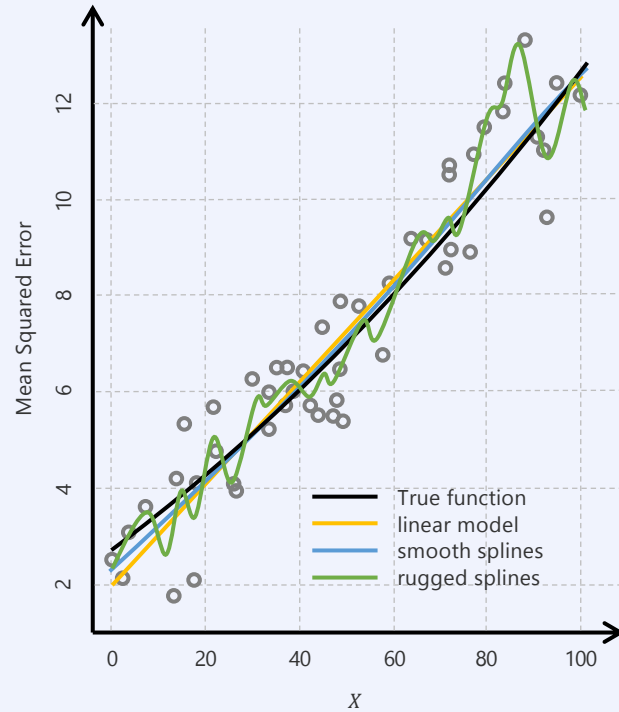
- in practice, we use $k = 5$ or 10
- LOOCV is k -fold CV with $k = n-1$
- k -fold CV is more efficient but has higher bias than LOOCV

We can reduce variance of the CV-estimated error by averaging the over different (e.g. 10) random splits

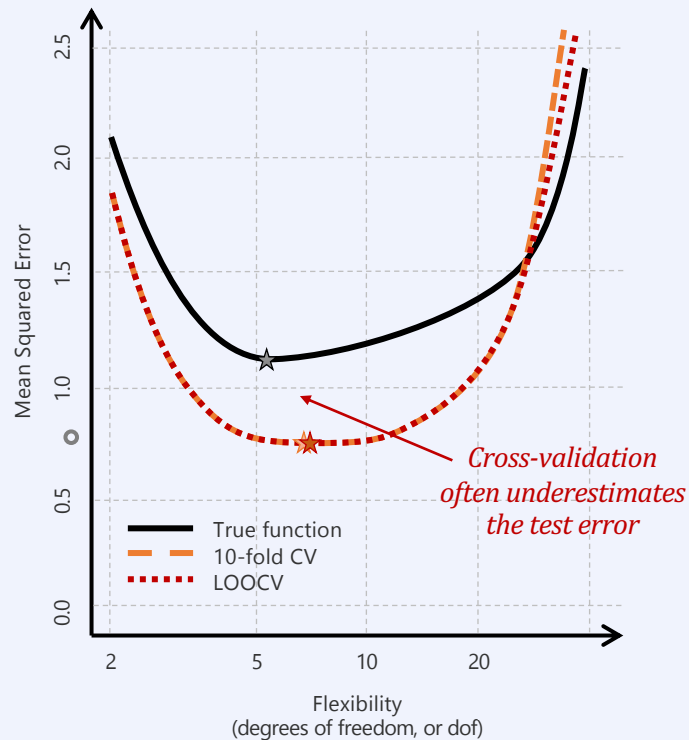
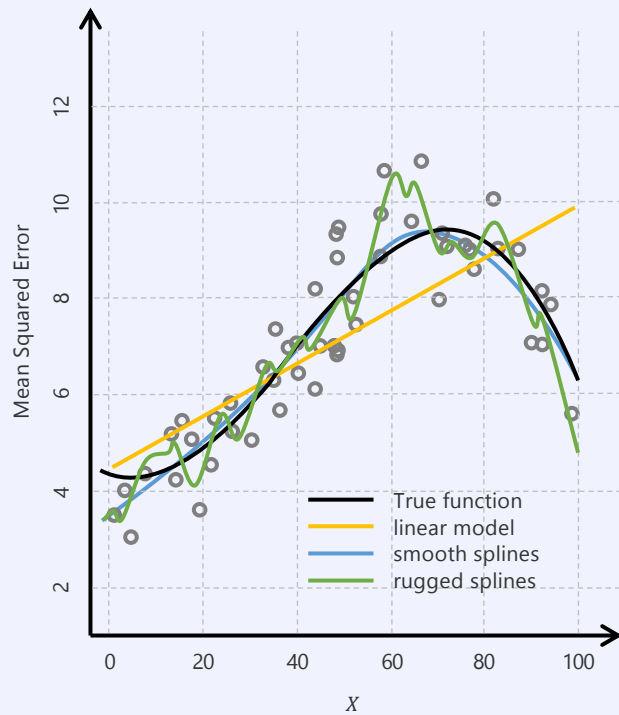


*10-fold cross-validation on
10 different random splits*

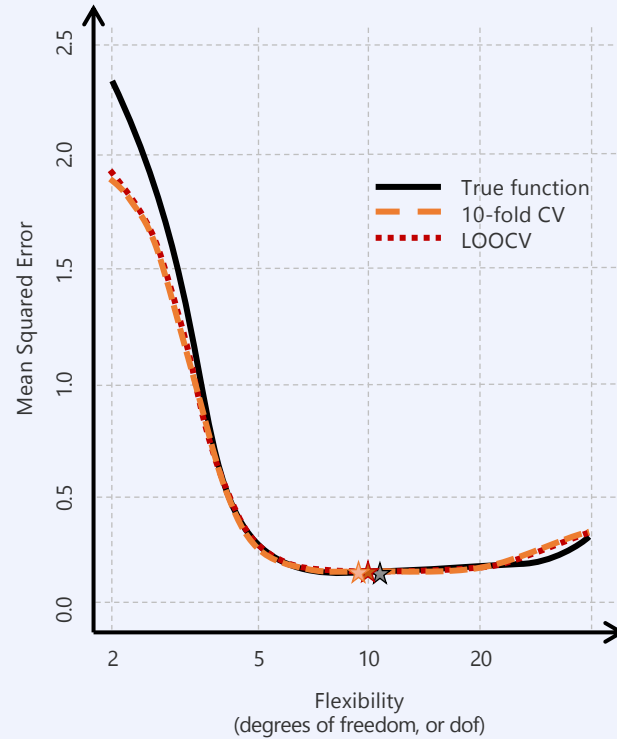
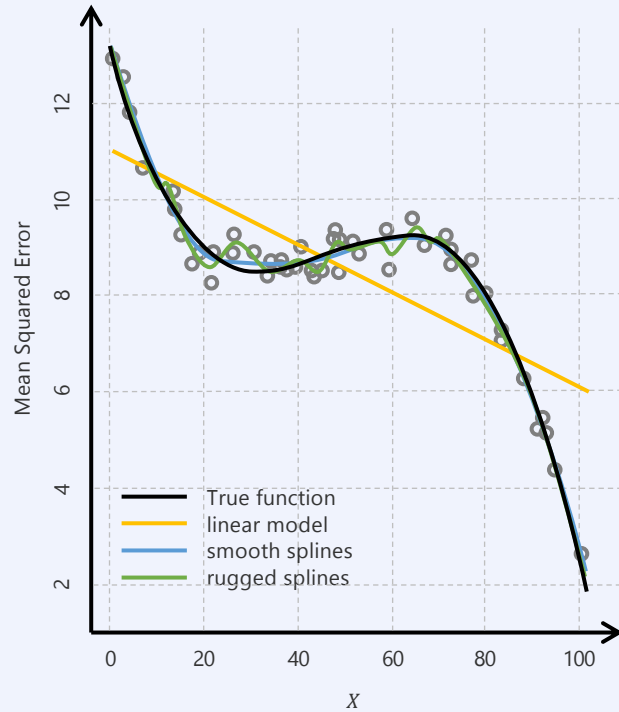
Example almost-linear data



Example moderately non-linear data



Example highly non-linear data



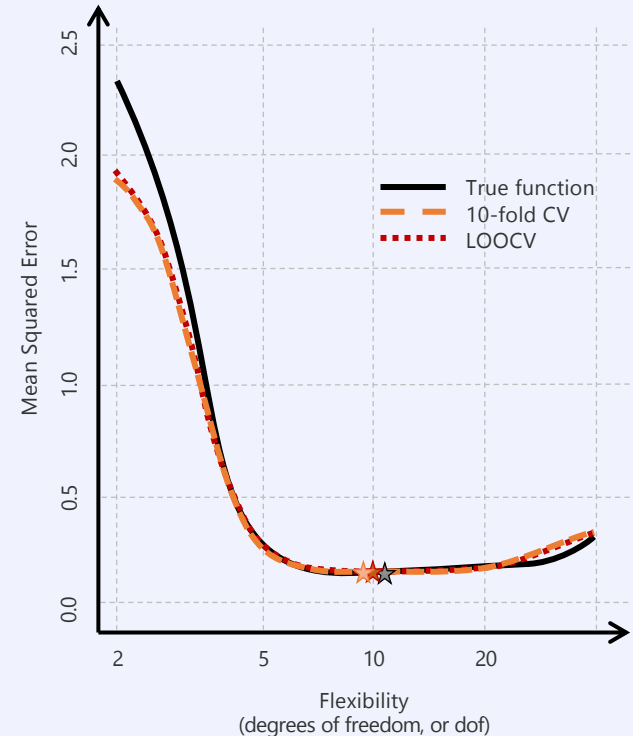
k -fold Cross Validation

We are often not only interested in the test error itself, but even more interested in **the location of the minimum**, as that shows **the complexity of the optimal model**

- test errors vary so minimum is difficult to identify
- use CV to estimate variance of the test error!
- many different models have almost the same error

General rule:

Choose the simplest model within 1 standard error of the optimum



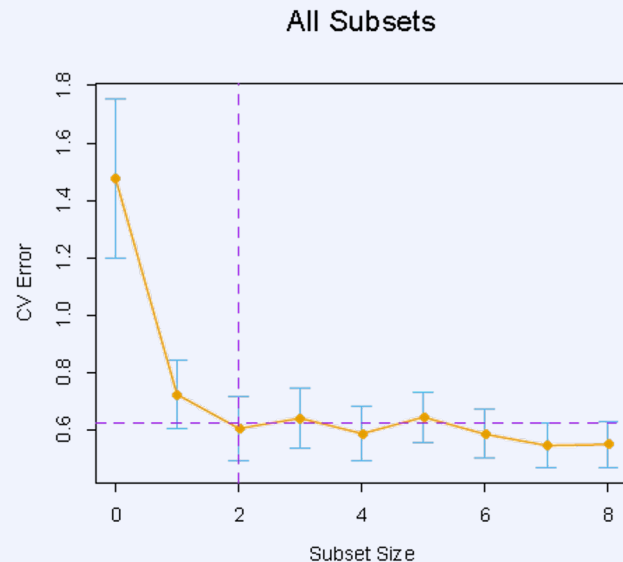
k -fold Cross Validation

We are often not only interested in the test error itself, but even more interested in **the location of the minimum**, as that shows **the complexity of the optimal model**

- test errors vary so minimum is difficult to identify
- use CV to estimate variance of the test error!
- many different models have almost the same error

General rule:

Choose the simplest model within 1 standard error of the optimum





Bias-Variance Tradeoff for k -fold CV

k -fold CV often gives **more accurate** error estimates than LOOCV

- this is due to the bias-variance tradeoff

Increasing the relative size of the training set **reduces bias** but **increases variance**

- LOOCV averages over models trained on **almost the same data**: all estimates are **highly correlated**
- k -fold CV has **less overlap** in training data, and hence **less correlated** estimates

error estimates



different folds



The variance of the average over the outcomes of k identically distributed events with correlation ρ , each with variance σ^2 is

correlation between folds

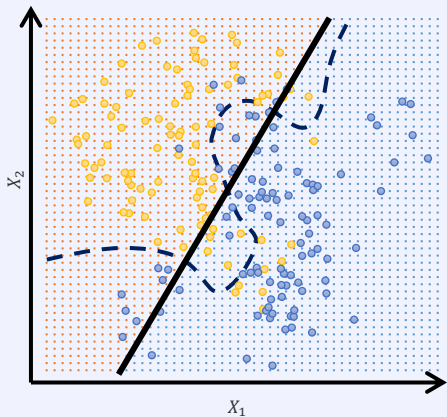


of error estimate in a fold



$$\rho\sigma^2 + \frac{1-\rho}{k}\sigma^2$$

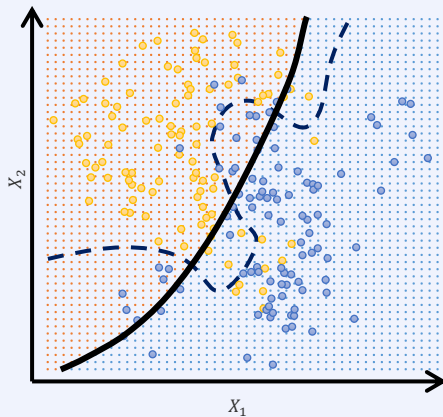
Example Cross Validation for Classification



Logistic Regression
degree = 1

$$\log \frac{p}{1-p} = \beta_0 + \beta_1 X_1 + \beta_2 X_2$$

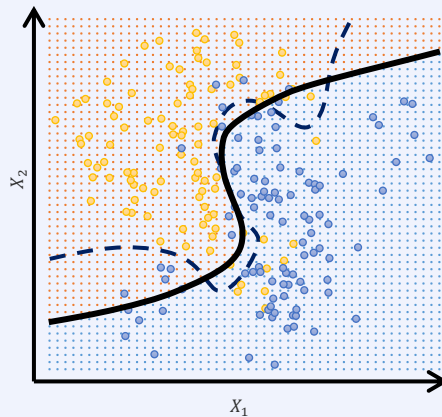
test error 0.201



Quadratic Logistic Regression
degree = 2

$$\log \frac{p}{1-p} = \beta_0 + \beta_1 X_1 + \beta_2 X_1^2 + \beta_3 X_2 + \beta_4 X_2^2$$

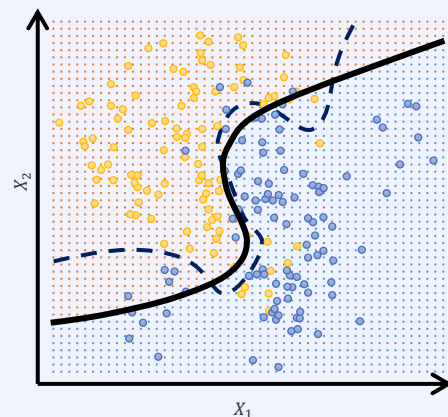
test error 0.197



Cubic Logistic Regression
degree = 3

$$\log \frac{p}{1-p} = \beta_0 + \beta_1 X_1 + \beta_2 X_1^2 + \beta_3 X_1^3 + \dots$$

test error 0.160



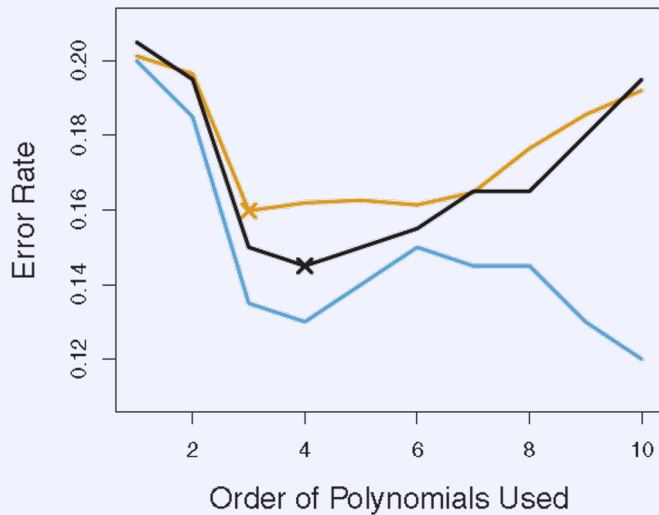
Quartic Logistic Regression
degree = 4

$$\log \frac{p}{1-p} = \beta_0 + \beta_1 X_1 + \beta_2 X_1^2 + \beta_3 X_1^3 + \beta_4 X_1^4 + \dots$$

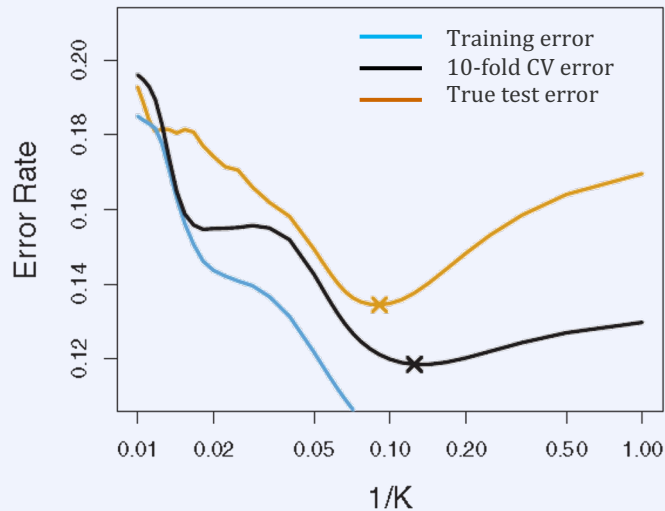
test error 0.162

*Test error (true) for Logistic Regression on synthetic data
(Bayes error 0.11)*

Example Cross Validation for Classification



*CV estimates of test error
logistic regression*



*CV estimates of test error
k-NN classification*



Doing Cross Validation ~~Right~~ Wrong

Assume a scenario with many predictors

How we typically build a model

1. select the predictors with strongest univariate correlation with the output
2. build a multivariate model on these predictors
3. use CV to select the model parameters

Now, let's assume we get

- $N = 50$ samples, two balanced classes
- $p = 5000$ continuous-valued Gaussian predictors, independent of class labels

The true test error of any classifier is 50%

Let's build that model!

1. select 100 predictors in step 1
2. use 1-NN classifier on these predictors
3. do CV in 50 simulations to estimate test error

Resulting average test error is 3%

- what? how? why? what is this? I don't even?
- the predictors we chose have an **unfair advantage**, as we chose them over **all** data, rather than over just the training data
- Always keep training and test set separate!

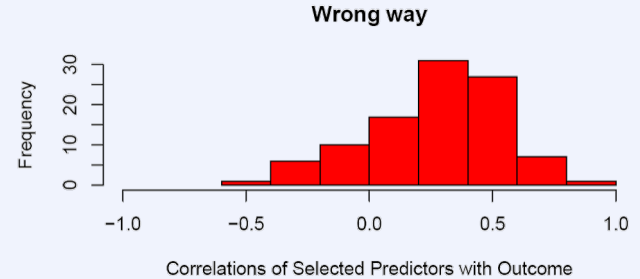


Doing Cross Validation Wrong

The Wrong Way

1. select the 100 predictors most correlated with the outcome variable using all n samples
2. choose a random set of 10 samples
3. compute correlations of selected predictors with the output over just these 10 samples

Mean correlation of selected predictors with outcome is 28%





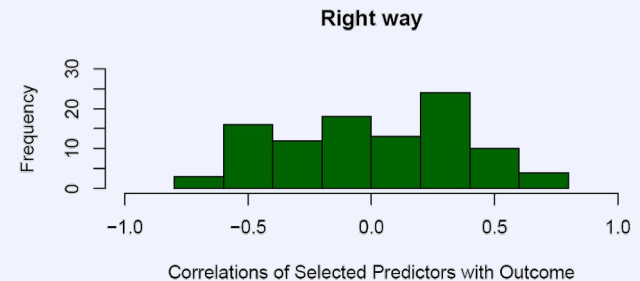
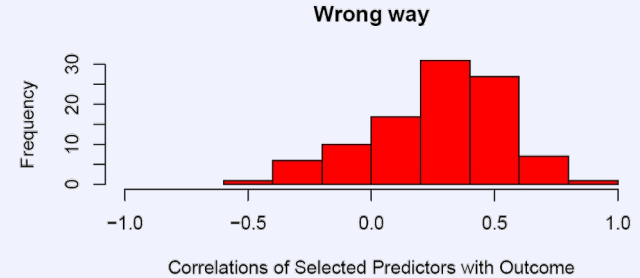
Doing Cross Validation Right

The Right Way

1. divide samples into K classes (K -fold CV)
2. for each $k=1, \dots, K$ do
 1. find that subset of predictors most correlated with the outcome on all samples except those in fold k
 2. use these predictors to build multivariate model
 3. use classifier to predict classes of samples in fold k
3. accumulate errors on all K folds to compute correlation

In a multistep procedure, we have to apply cross-validation to entire sequence of steps!

- initial clustering can be done before samples are left out (because it does not use class labels), e.g. select 50 predictors with highest variances across the training set

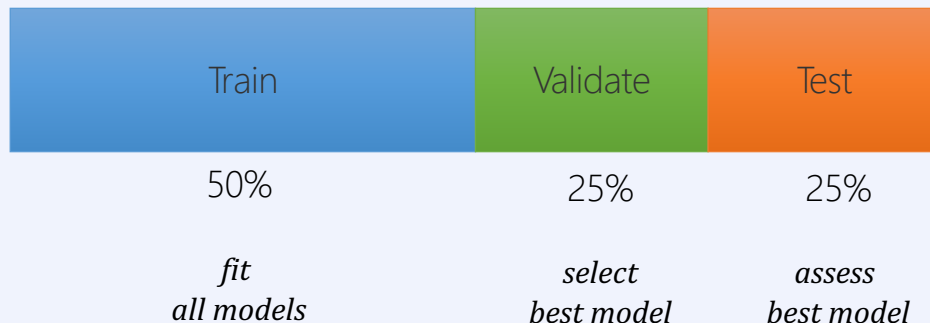




Train-Validation-Test Paradigm

More conservatively, we can divide the data **three-way**

1. **training set** for fitting models
2. **validation set** for comparatively assessing model performance in order to select a model
3. **test set** in order to assess the performance of the selected model





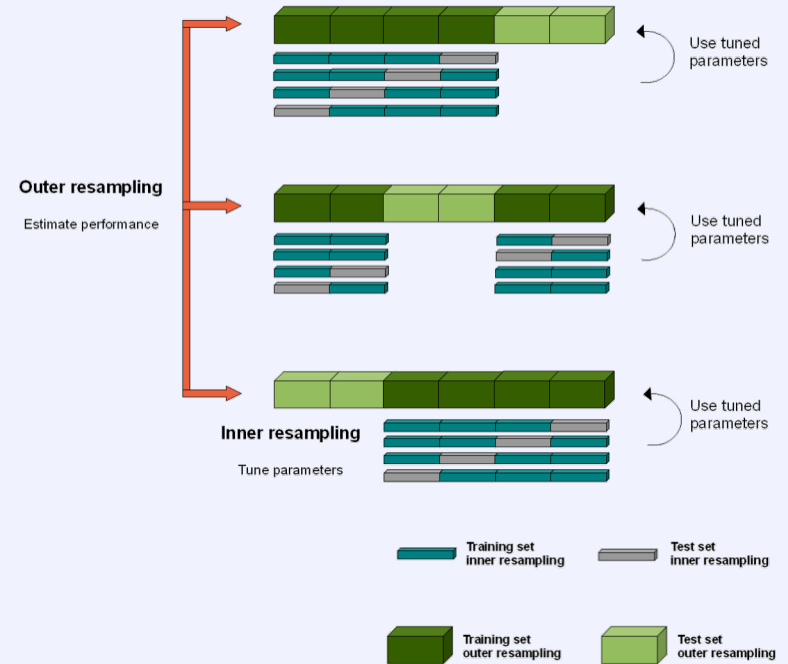
Nested Cross Validation

Purpose: optimize hyper-parameters first, then assess performance of the best model

- **inner cross validation** for tuning hyper-parameters and model selection
- **outer cross validation** for evaluating performance of the final model

Provides more conservative error estimates than normal cross-validation

- different test sets for selecting model parameters and assessing model performance



The Bootstrap

ISLR 5.2, ESL 7.11

The Bootstrap

Bootstrap is used to quantify the uncertainty of a given estimator

- for linear least-squares regression, we already saw theoretical methods (and software) for doing so
- the bootstrap is **applicable** to all kinds of methods for which **no theory exists**

Toy example

- we want to invest a fixed sum in two assets that yield returns X and Y , fraction α in X and $1 - \alpha$ in Y
- we want to choose α such that we minimize the total risk (variance) of the investment $\text{Var}(\alpha X + (1 - \alpha)Y)$

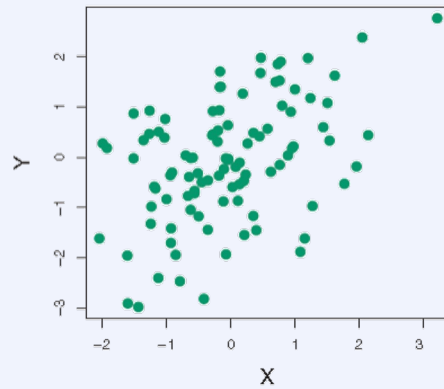
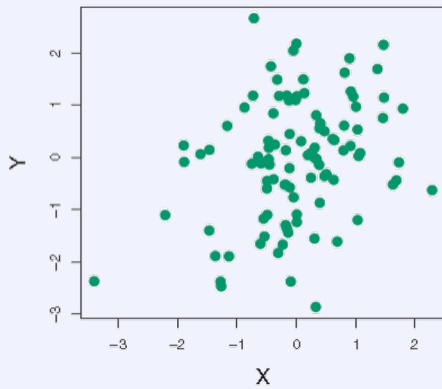
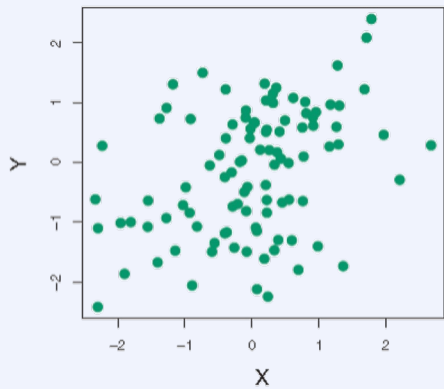
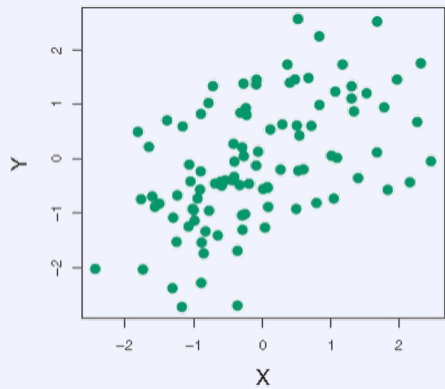
$$\alpha = \frac{\sigma_Y^2 - \sigma_{XY}}{\sigma_X^2 + \sigma_Y^2 - 2\sigma_{XY}}$$

$$\sigma_X^2 = \text{Var}(X) \quad \sigma_Y^2 = \text{Var}(Y) \quad \sigma_{XY} = \text{Cov}(X, Y)$$

- our estimate of α will hence depend on the sample estimates of the variances

$$\hat{\alpha} = \frac{\hat{\sigma}_Y^2 - \hat{\sigma}_{XY}}{\hat{\sigma}_X^2 + \hat{\sigma}_Y^2 - 2\hat{\sigma}_{XY}}$$

The Bootstrap



What are the estimates of α when we *repeatedly* draw 100 samples?

The Bootstrap

We repeat the process 1000 times, obtaining estimates $\alpha_1, \alpha_2, \dots, \alpha_{1000}$

$$\bar{\alpha} = \frac{1}{1,000} \sum_{r=1}^{1,000} \hat{\alpha}_r = 0.5996$$

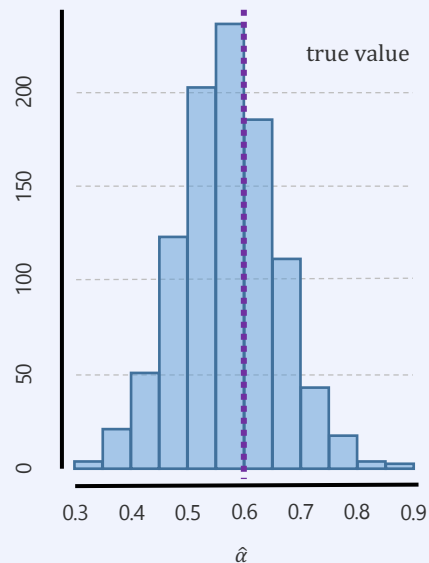
$$SE(\hat{\alpha}) = \sqrt{\frac{1}{1,000 - 1} \sum_{r=1}^{1,000} (\hat{\alpha}_r - \alpha)^2} = 0.083$$

In a real setting, we **cannot get new samples**

- but, we can re-sample from the training data
- we draw uniformly and with replacement $B = 1000$ samples Z^{*i} each of size n
- then

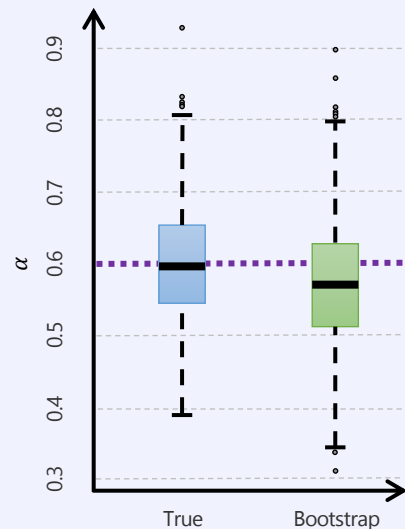
$$SE_{B(\hat{\alpha})} = \sqrt{\frac{1}{B-1} \sum_{r=1}^B \left(\hat{\alpha}^{*r} - \frac{1}{B} \sum_{r'=1}^B \hat{\alpha}^{*r'} \right)^2}$$

Thousand estimates of α

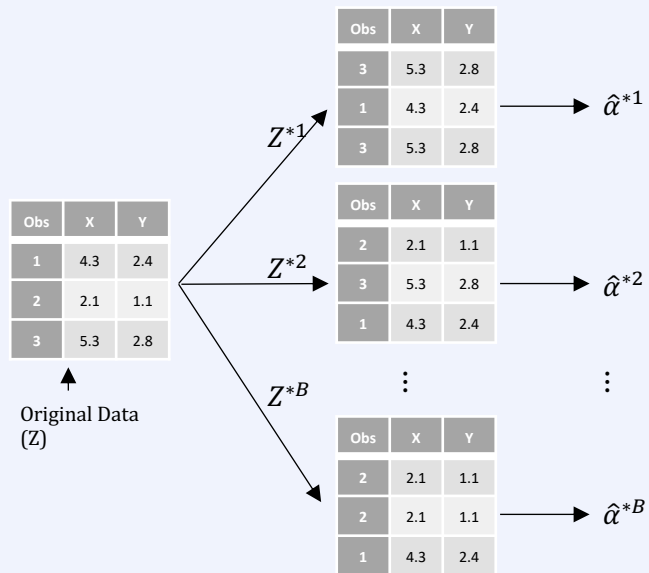


$$\begin{aligned} \sigma_X^2 &= 1 & \hat{\alpha} &= 0.5996 \\ \sigma_Y^2 &= 1.25 & SE(\hat{\alpha}) &= 0.083 \\ \sigma_{XY} &= 0.5 & & \\ \alpha &= 0.6 & & \end{aligned}$$

Simulated data vs. Bootstrap



Example The Bootstrap



Bootstrap on a data set of 3 rows



The Bootstrap

Bootstrap samples tend to contain only a **subset** of the training data

- some of the training observations will occur multiple times in a bootstrap set
- we have compute the probability of an observation being selected into a bootstrap set as

$$\Pr\{\text{observation } i \in \text{bootstrap sample } b\} = 1 - \left(1 - \frac{1}{n}\right)^n \approx 1 - e^{-1} = 0.632$$

On average, about a third of the data is not contained in a bootstrap sample

- we should **test** the model on these **out-of-bag** samples, mimicking 3-fold CV
- if we *do include* in-bag samples, training and test set overlap, and we will underestimate the test error

Bootstrap samples are **highly correlated**, which increases the variance of the error estimate

Conclusion for today

To estimate how well a model generalizes, we have to test on *other* data than it was trained on

Resampling or **subsampling** are methods for repeatedly drawing samples from training data to

- **learn** about the **variability** of the fitted models, as the training set changes (model variability)
- **assess** the test **error** which is hard to estimate (model assessment)
- **optimize** model **performance** in terms of test error (model selection)

Resampling methods can be **computationally expensive**

- repeated model fitting on a potentially large number of data sets
- today's computing power allows for this expense
- but not when your model is extremely expensive (looking at you, Deep Learning)