Machine Learning Basics
Overfitting

HKUST MSBD 6000B

Instructor: Yu Zhang

Review: machine learning basics

Math formulation

- Given training data $\{(x_i, y_i): 1 \le i \le n\}$ i.i.d. from distribution D
- Find $y = f(x) \in \boldsymbol{\mathcal{H}}$ that minimizes $\hat{L}(f) = \frac{1}{n}\sum_{i=1}^{n} l(f, x_i, y_i)$
- s.t. the expected loss is small

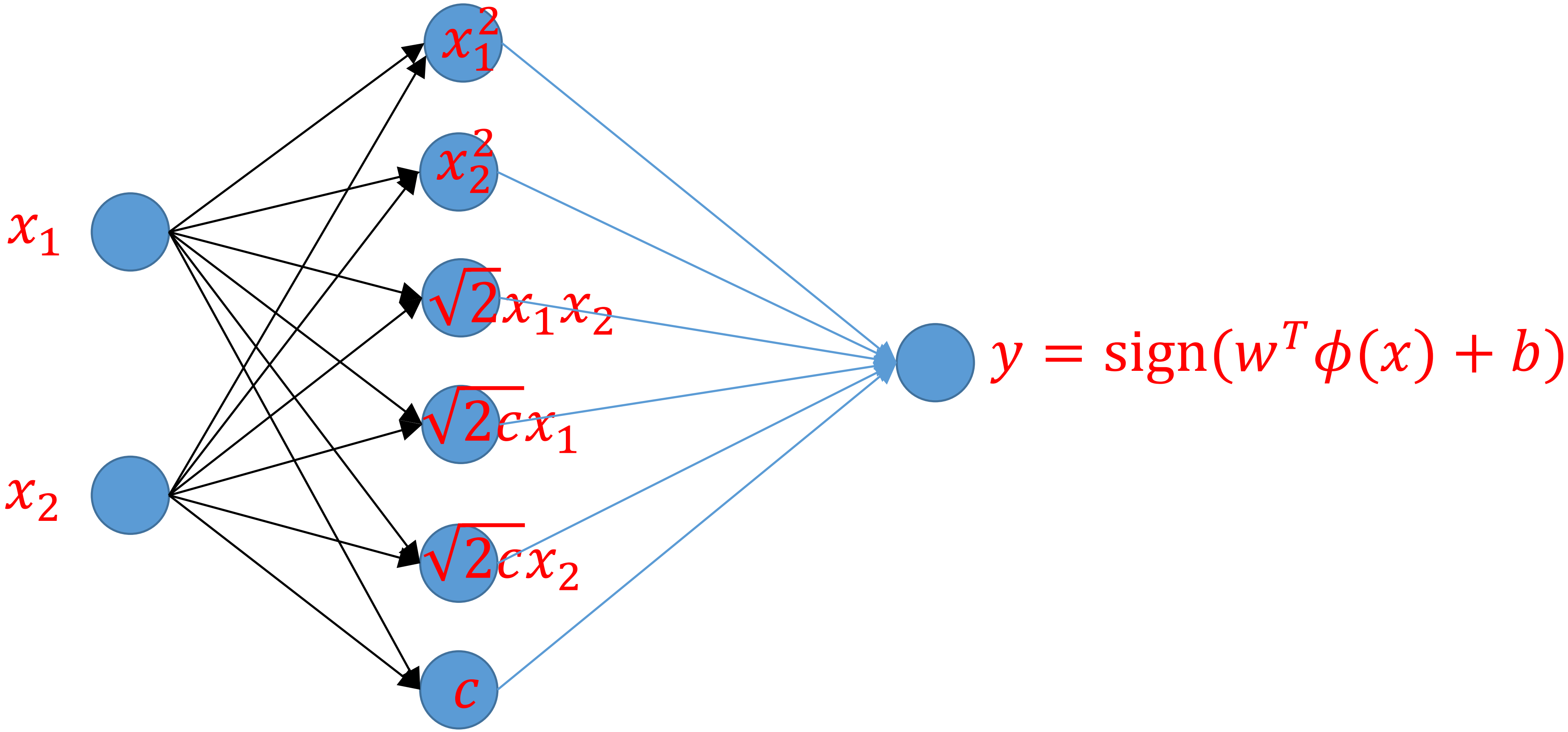
$$L(f) = \mathbb{E}_{(x,y)\sim D}[l(f, x, y)]$$

Machine learning

- Collect data and extract features
- Build model: choose hypothesis class $\mathcal{H}$ and loss function l
- Optimization: minimize the empirical loss

Overfitting

Linear vs nonlinear models



Polynomial kernel

Linear vs nonlinear models

- Linear model: $f(x) = a_0 + a_1 x$
- Nonlinear model: $f(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \ldots + a_M x^M$

- Linear model $\subseteq$ Nonlinear model (since can always set $a_i = 0 \ (i > 1)$)

- Looks like nonlinear model can always achieve same/smaller error
- Why one use Occam's razor (choose a smaller hypothesis class)?

Example: regression using polynomial curve

$$t = \sin(2\pi x) + \epsilon$$

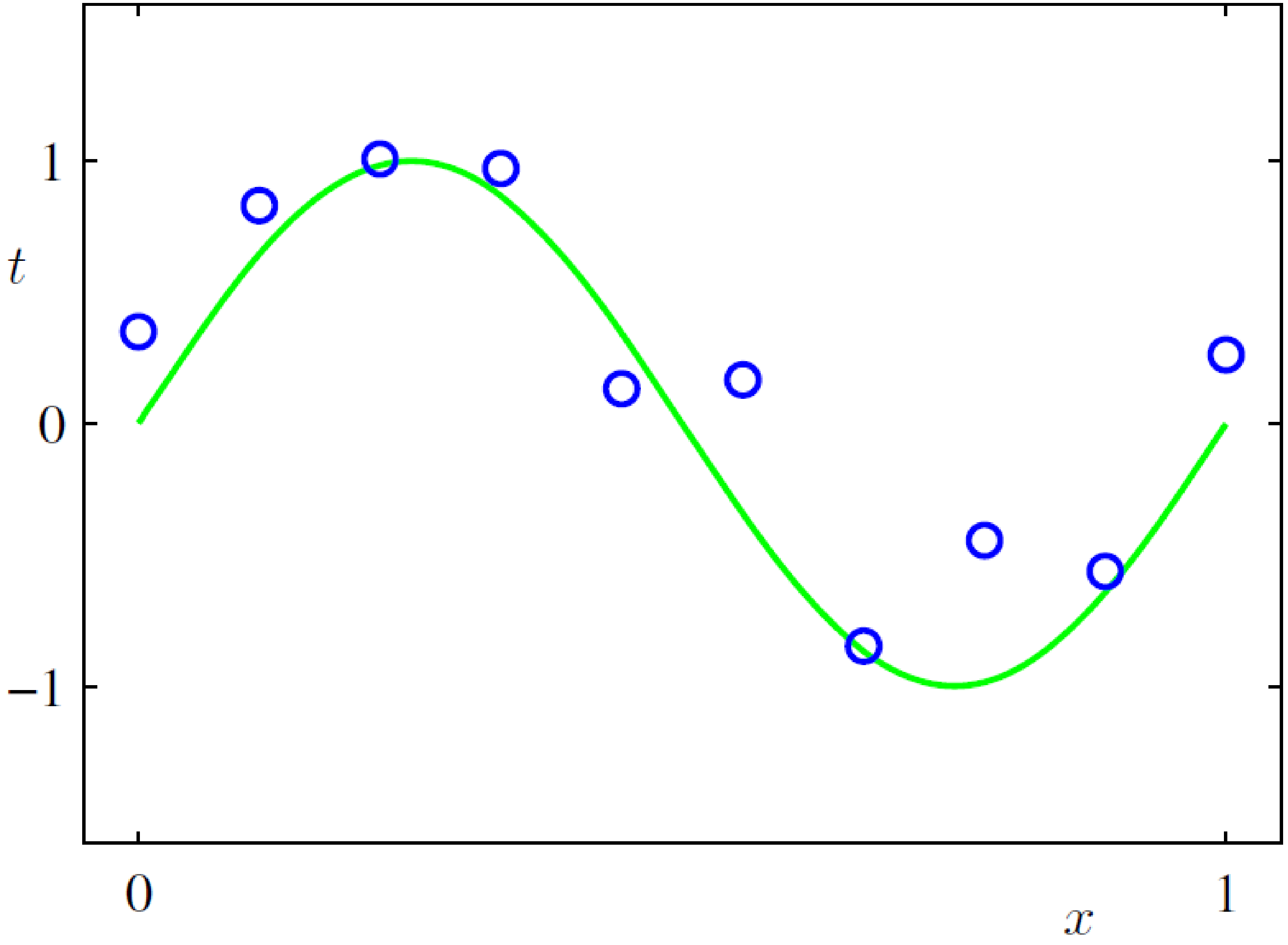
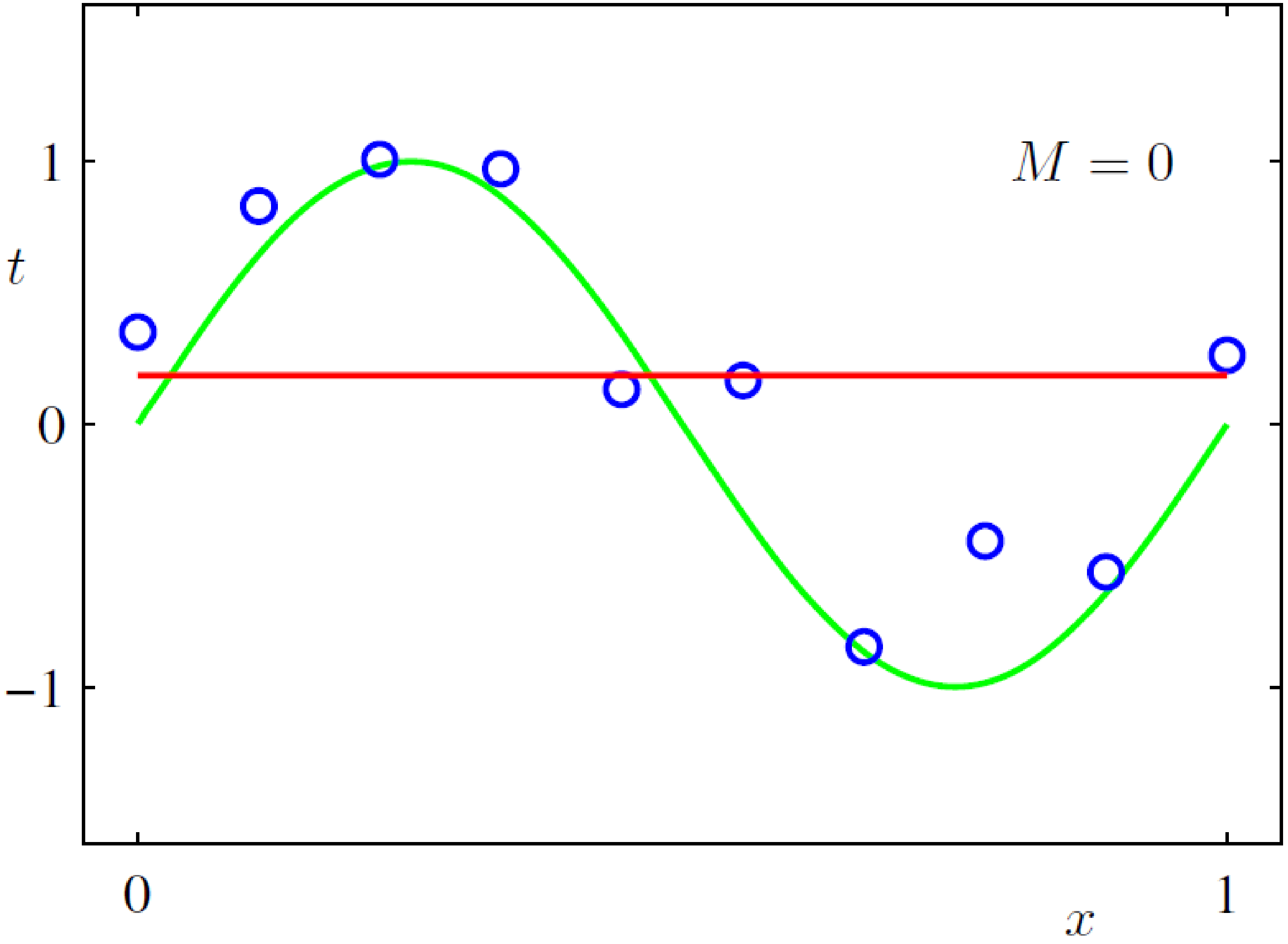


Figure from *Machine Learning and Pattern Recognition*, Bishop

Example: regression using polynomial curve

$$t = \sin(2\pi x) + \epsilon$$



Regression using polynomial of degree *M*

Figure from *Machine Learning and Pattern Recognition*, Bishop

Example: regression using polynomial curve

$$t = \sin(2\pi x) + \epsilon$$

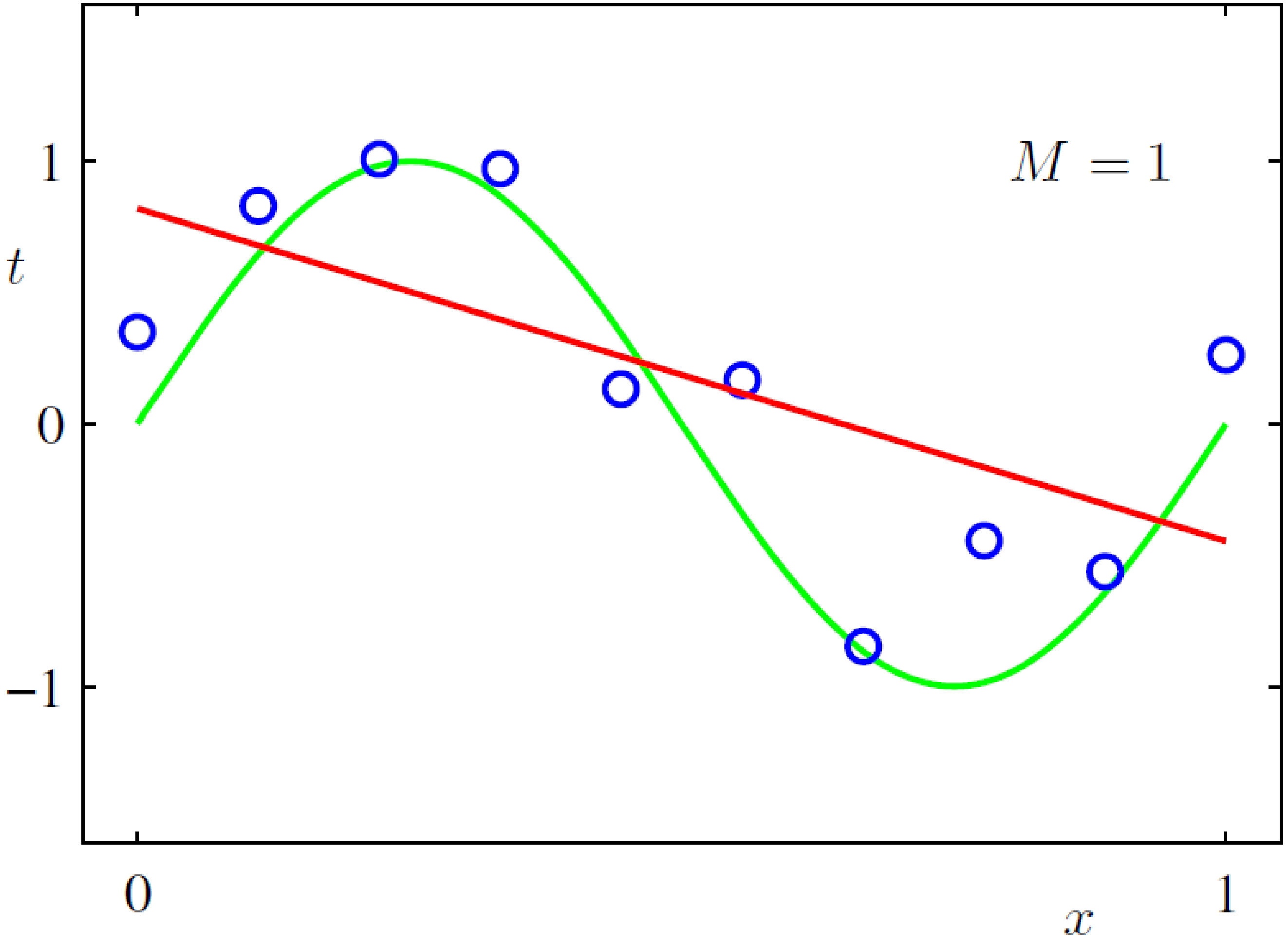


Figure from *Machine Learning and Pattern Recognition*, Bishop

Example: regression using polynomial curve

$$t = \sin(2\pi x) + \epsilon$$

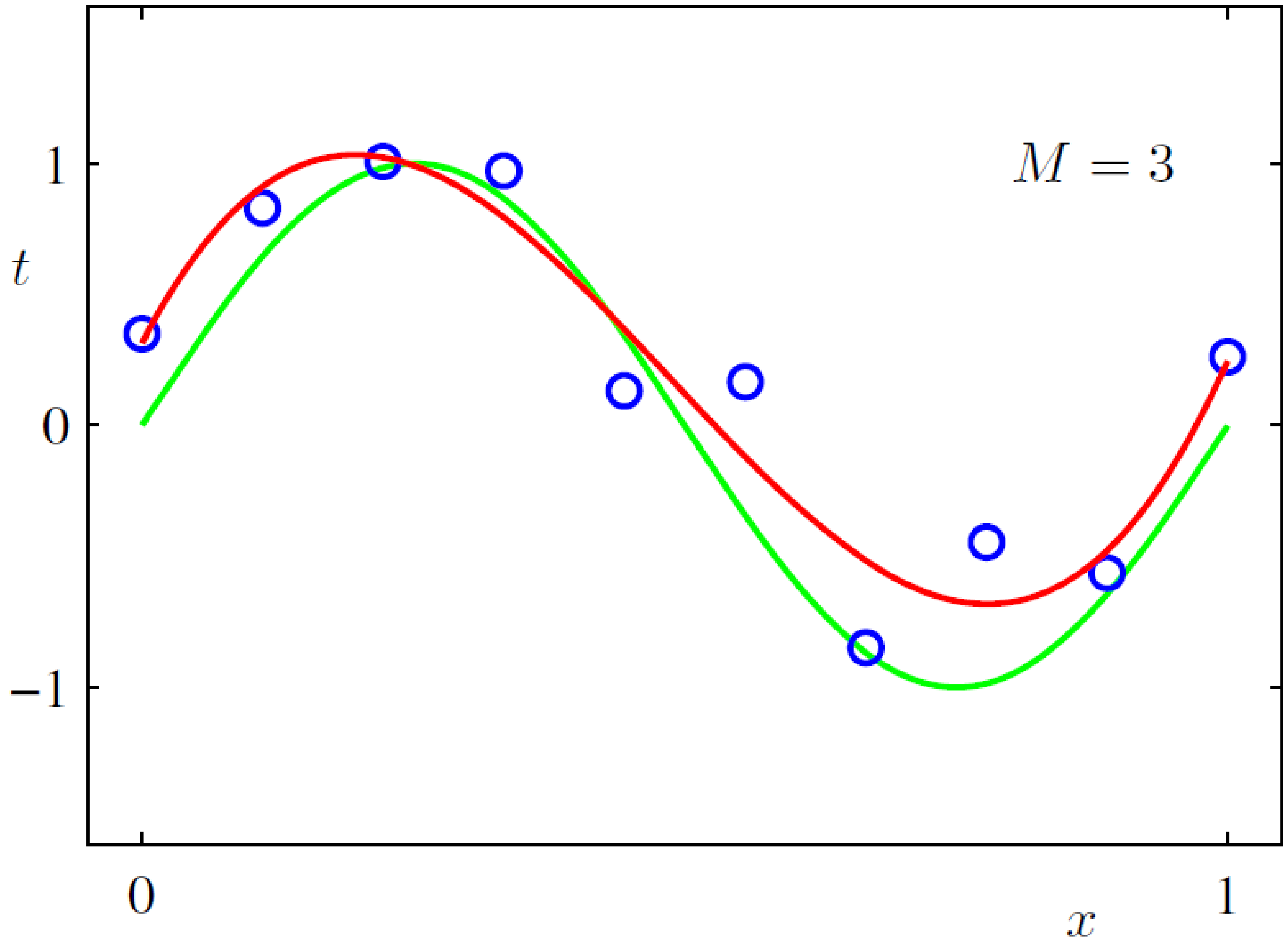


Figure from *Machine Learning and Pattern Recognition*, Bishop

Example: regression using polynomial curve

$$t = \sin(2\pi x) + \epsilon$$

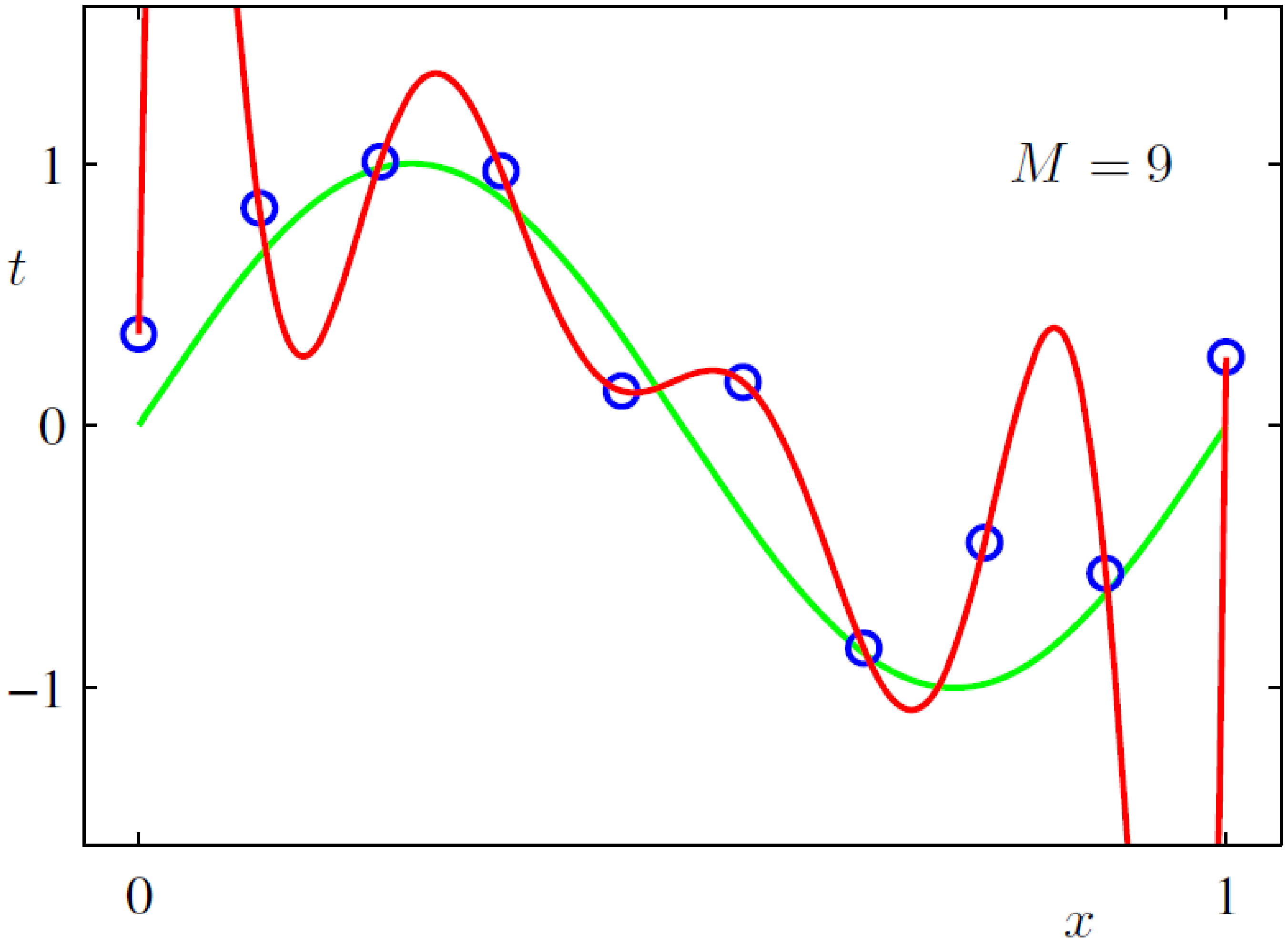


Figure from *Machine Learning and Pattern Recognition*, Bishop

Example: regression using polynomial curve

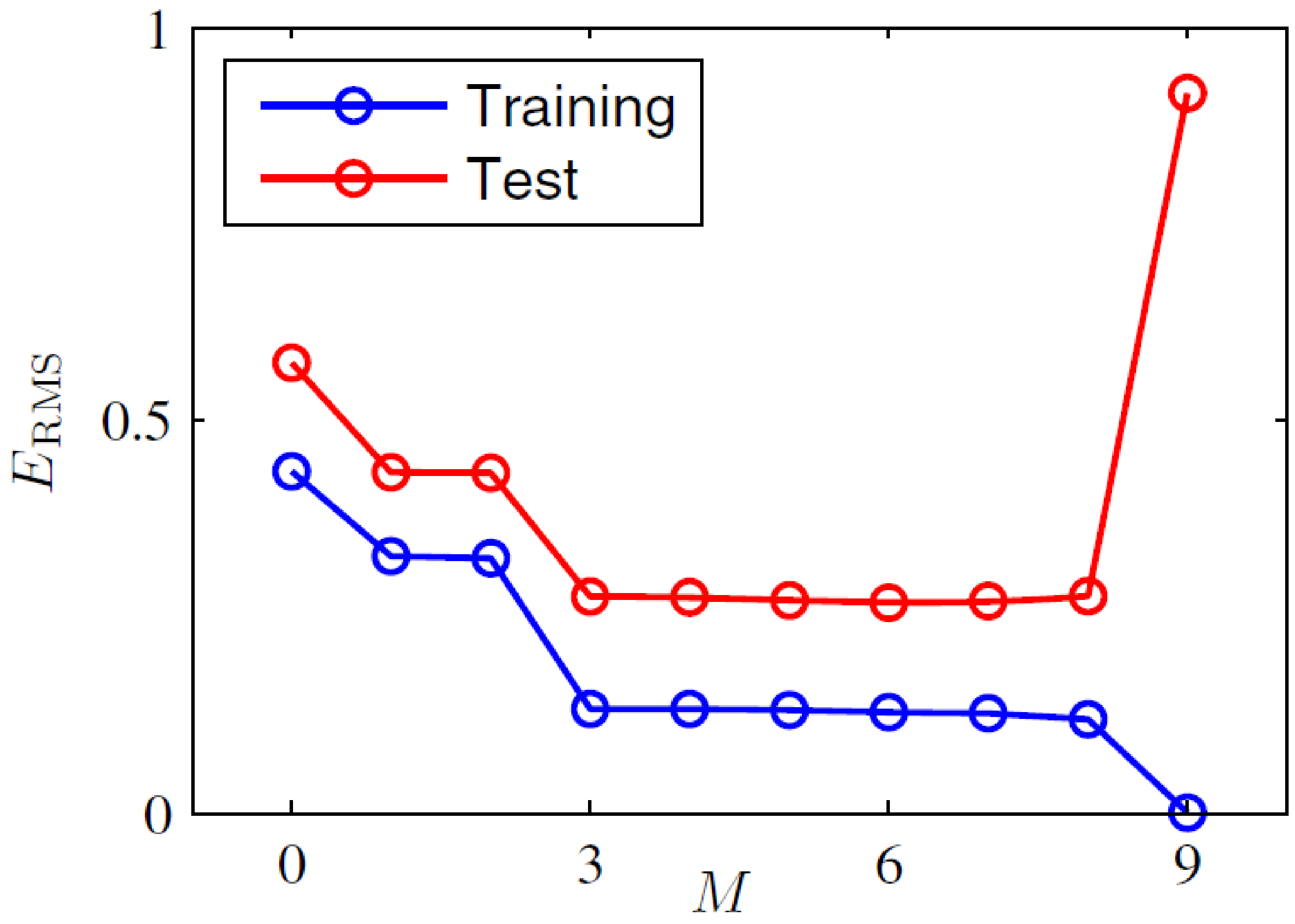


Figure from *Machine Learning and Pattern Recognition*, Bishop

Prevent overfitting

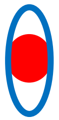
- Empirical loss and expected loss are different
  - Also called training error and test/generalization error
- Larger the data set, smaller the difference between the two
- Larger the hypothesis class, easier to find a hypothesis that fits the difference between the two
  - Thus has small training error but large test error (overfitting)

- Larger data set helps!
- Throwing away useless hypotheses also helps!

Prior v.s. data

Prior vs experience

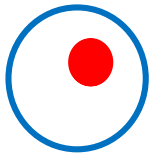
- Super strong prior knowledge: $\mathcal{H} = \{f^*\}$
- No data is needed!



f^*: the best function

Prior vs experience

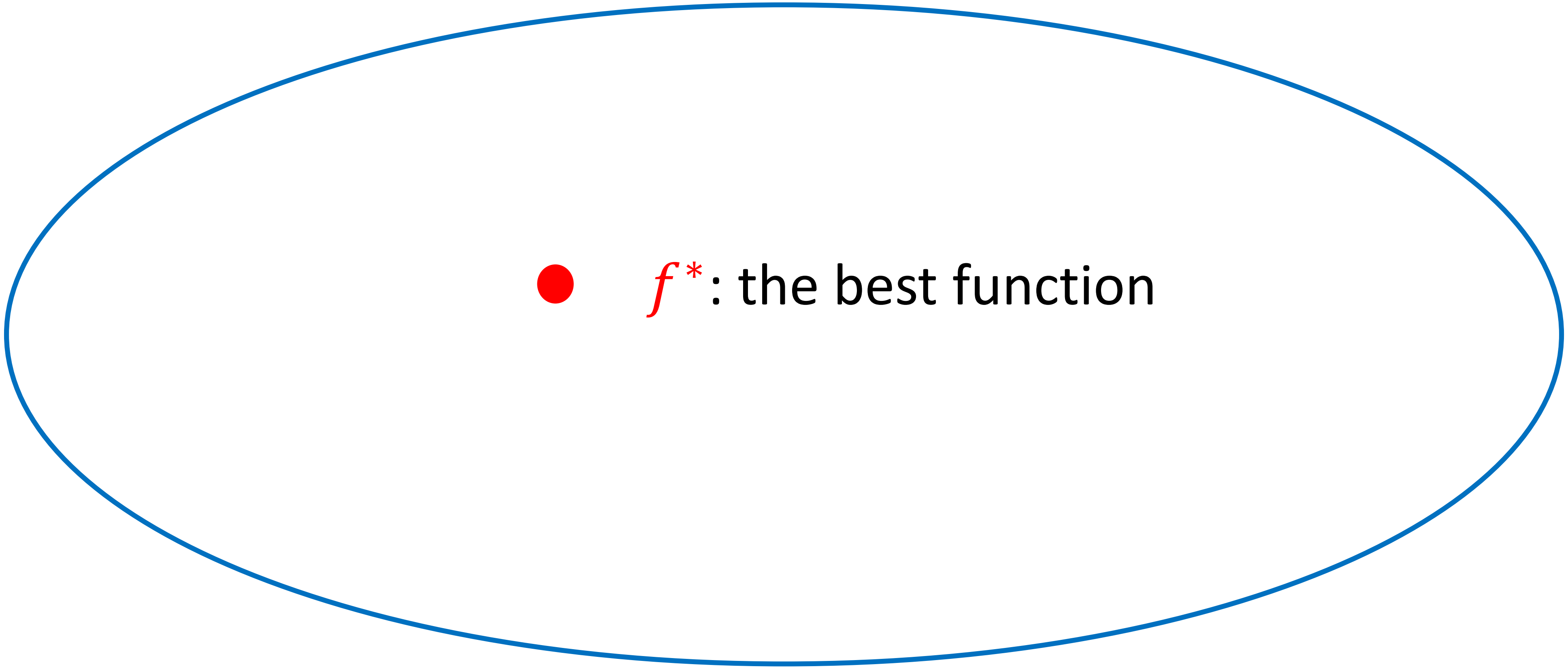
- Super strong prior knowledge: $\boldsymbol{\mathcal{H}} = \{f^*, f_1\}$
- A few data points suffices to detect f^*



f^*: the best function

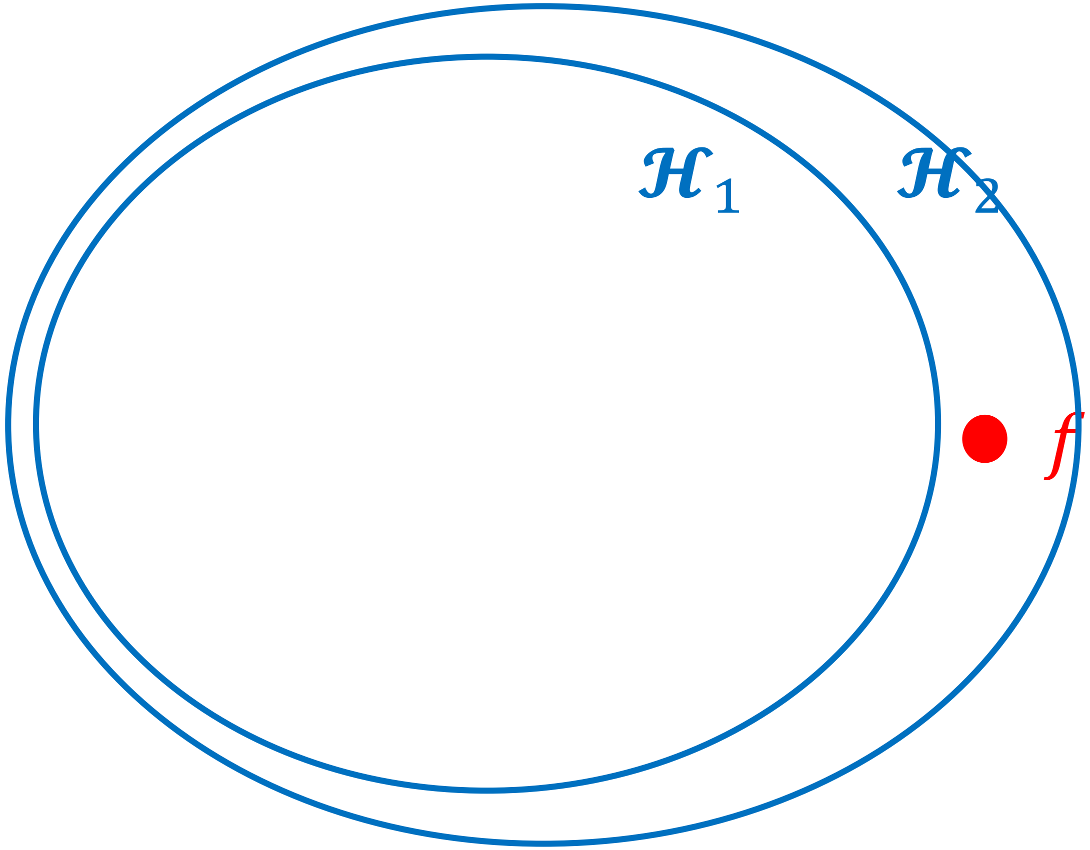
Prior vs experience

- Super larger data set: infinite data
- Hypothesis class $\mathcal{H}$ can be all functions!



Prior vs experience

- Practical scenarios: finite data, $\mathcal{H}$ of median capacity, f^* in/not in $\mathcal{H}$



f^*: the best function

Prior vs experience

- Practical scenarios lie between the two extreme cases

General Phenomenon

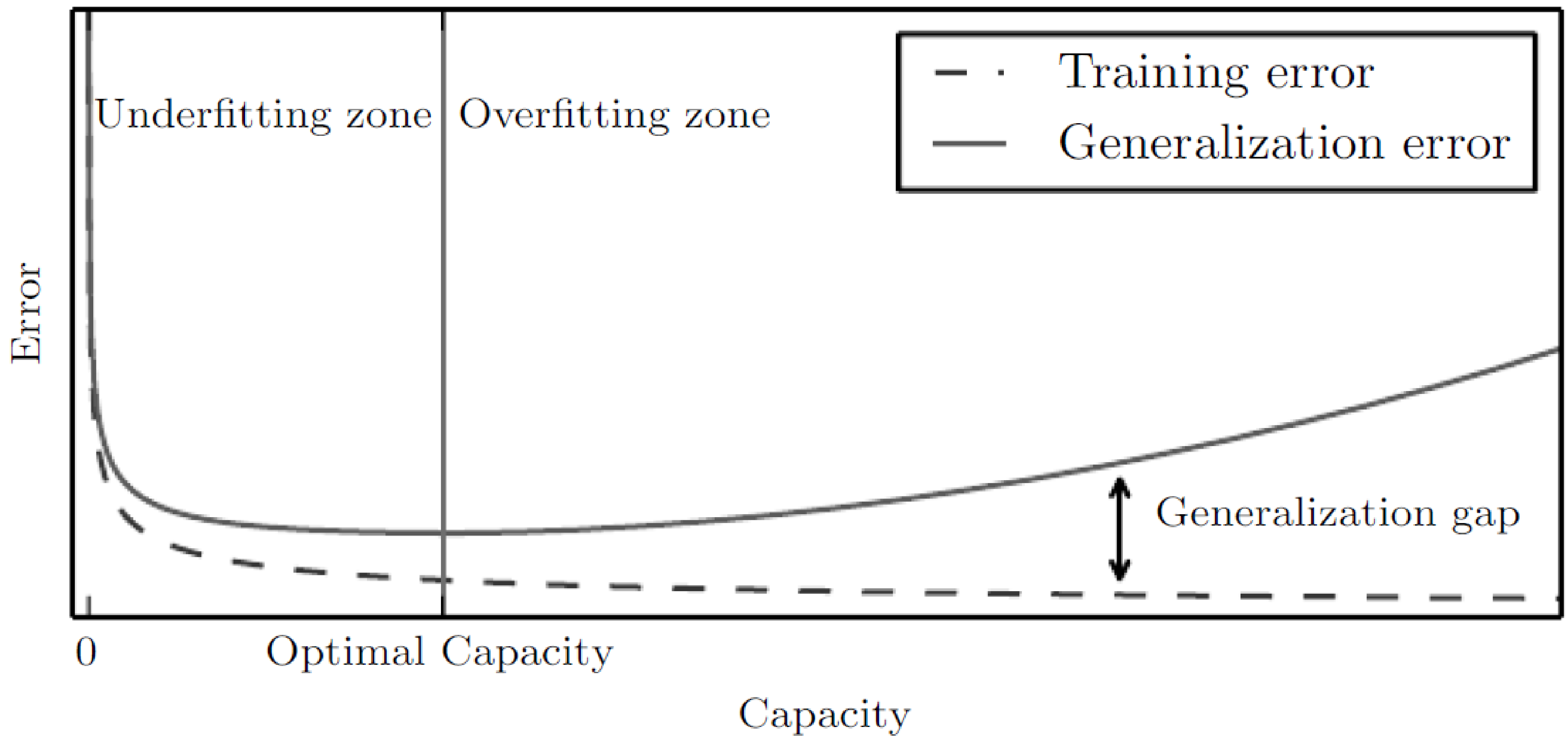


Figure from *Deep Learning*, Goodfellow, Bengio and Courville

Cross validation

Model selection

- How to choose the optimal capacity?
  - e.g., choose the best degree for polynomial curve fitting

- Cannot be done by training data alone
- Create held-out data to approx. the test error
  - Called validation data set

Model selection: cross validation

- Partition the training data into several groups
- Each time use one group as validation set

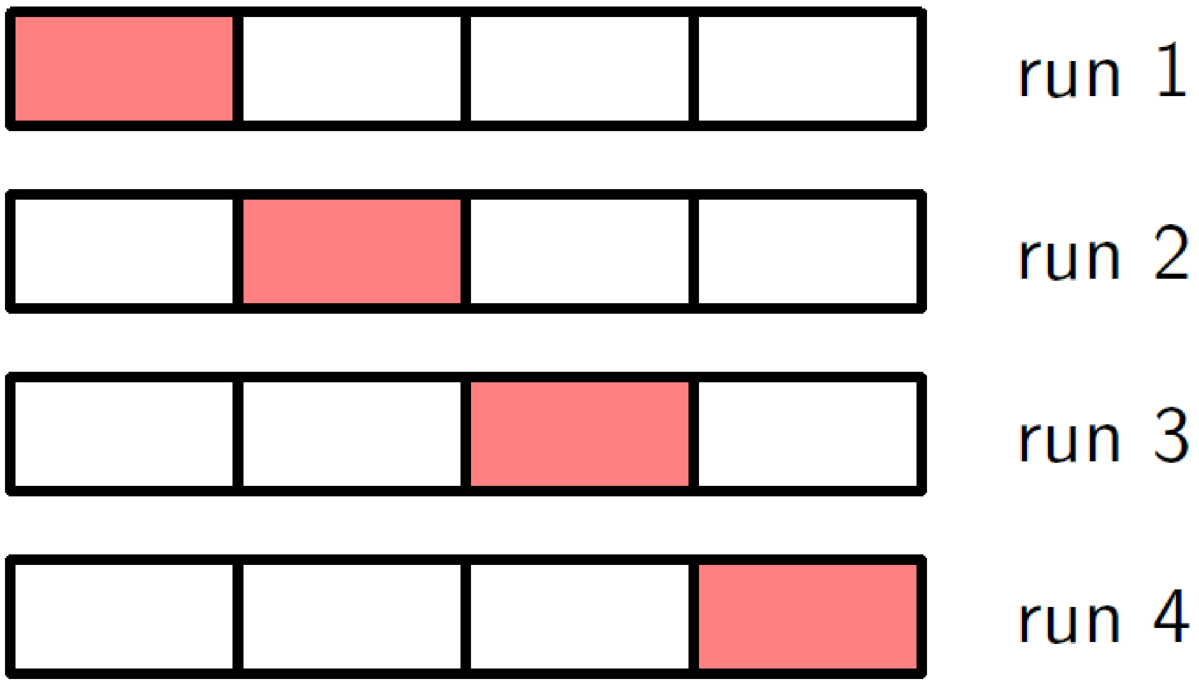


Figure from *Machine Learning and Pattern Recognition*, Bishop

Model selection: cross validation

- Also used for selecting other hyper-parameters for model/algorithm
  - E.g., learning rate, stopping criterion of SGD, etc.

- Pros: general, simple
- Cons: computationally expensive; even worse when there are more hyper-parameters