Machine Learning with Python

MTH786U/P 2022/23

Lecture 4: The model selection problem

Nicola Perra, Queen Mary University of London (QMUL)

n.perra@qmul.ac.uk

Model selection

Recall ridge regression:

Model selection

Recall ridge regression:

$$\mathbf{w}_\alpha = \arg\min_{\mathbf{w}} \left\{ \frac{1}{2s} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|^2 + \frac{\alpha}{2} \|\mathbf{w}\|^2 \right\}$$

Model selection

Recall ridge regression: $$\mathbf{w}_\alpha = \arg\min_{\mathbf{w}} \left\{ \frac{1}{2s} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|^2 + \frac{\alpha}{2} \|\mathbf{w}\|^2 \right\}$$

The regularisation parameter α is also referred to as *hyperparameter*

Model selection

Recall ridge regression: $$\mathbf{w}_\alpha = \arg\min_{\mathbf{w}} \left\{ \frac{1}{2s} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|^2 + \frac{\alpha}{2} \|\mathbf{w}\|^2 \right\}$$

The regularisation parameter α is also referred to as *hyperparameter*

Hyperparameters are parameters of prior distributions

Model selection

Recall ridge regression:

$$\mathbf{w}_\alpha = \arg\min_{\mathbf{w}} \left\{ \frac{1}{2s} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|^2 + \frac{\alpha}{2} \|\mathbf{w}\|^2 \right\}$$

The regularisation parameter α is also referred to as *hyperparameter*

Hyperparameters are parameters of prior distributions

The degree d in polynomial regression is also a hyperparameter

Model selection

Recall ridge regression: $$\mathbf{w}_\alpha = \arg\min_{\mathbf{w}} \left\{ \frac{1}{2s} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|^2 + \frac{\alpha}{2} \|\mathbf{w}\|^2 \right\}$$

The regularisation parameter α is also referred to as *hyperparameter*

Hyperparameters are parameters of prior distributions

The degree d in polynomial regression is also a hyperparameter

How do we choose hyperparameters?

Model selection

Recall ridge regression: $$\mathbf{w}_\alpha = \arg\min_{\mathbf{w}} \left\{ \frac{1}{2s} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|^2 + \frac{\alpha}{2} \|\mathbf{w}\|^2 \right\}$$

The regularisation parameter α is also referred to as *hyperparameter*

Hyperparameters are parameters of prior distributions

The degree d in polynomial regression is also a hyperparameter

How do we choose hyperparameters?

Selection of hyperparameters is known as the *model selection* problem

Probabilistic setup

Assume underlying distribution $\mathscr{D}$

and that we sample from this distribution:

$$S := \left\{ (\mathbf{x}_i, \mathbf{y}_i) \text{ iid } \sim \mathscr{D} \right\}_{i=1}^{s}$$

Probabilistic setup

Assume underlying distribution $\mathscr{D}$

and that we sample from this distribution:

$$S := \left\{ (\mathbf{x}_i, \mathbf{y}_i) \text{ iid } \sim \mathscr{D} \right\}_{i=1}^{s}$$

Based on these samples the ridge regression model computes the 'best' (linear) weight function for fixed regularisation parameter α

$$f_S(x) := \langle \mathbf{x}, \mathbf{w}_\alpha \rangle \quad \text{for} \quad \mathbf{w}_\alpha = \arg\min_{\mathbf{w}} \left\{ \frac{1}{2s} \sum_{i=1}^{s} \left| \langle \mathbf{x}_i, \mathbf{w} \rangle - \mathbf{y}_i \right|^2 + \frac{\alpha}{2} \|\mathbf{w}\|^2 \right\}$$

Probabilistic setup

Assume underlying distribution $\mathscr{D}$

and that we sample from this distribution:

$$S := \left\{(\mathbf{x}_i, \mathbf{y}_i) \text{ iid } \sim \mathscr{D}\right\}_{i=1}^{s}$$

Based on these samples the polynomial regression model computes the 'best' (linear) weight function for fixed degree d

$$f_S(\mathbf{x}) := \langle \mathbf{x}, \mathbf{w}_d \rangle \quad \text{for} \quad \mathbf{w}_d = \arg\min_{\mathbf{w}} \left\{ \frac{1}{2s} \sum_{i=1}^{s} \left| \sum_{n=0}^{d} \mathbf{x}_i^n w_n - \mathbf{y}_i \right|^2 \right\}$$

Training error vs. expected error

Given a prediction function f_S , how can we assess if it is any good?

Training error vs. expected error

Given a prediction function f_S , how can we assess if it is any good?

Assume we knew the distribution $\mathscr{D}$, then we could compute

$$E(f) = \mathbb{E}_{\mathbf{x},\mathbf{y}}\left[\ell(\mathbf{y}, f(\mathbf{x}))\right]$$

Training error vs. expected error

Given a prediction function f_S , how can we assess if it is any good?

Assume we knew the distribution $\mathscr{D}$, then we could compute

$$E(f) = \mathbb{E}_{\mathbf{x},\mathbf{y}}\left[\ell(\mathbf{y}, f(\mathbf{x}))\right]$$

for a given loss function

$$\ell(\mathbf{y}, f(\mathbf{x})) = \frac{1}{2} | \mathbf{y} - f(\mathbf{x}) |^2$$

Training error vs. expected error

Given a prediction function f_S , how can we assess if it is any good?

Assume we knew the distribution $\mathscr{D}$, then we could compute

$$E(f) = \mathbb{E}_{\mathbf{x},\mathbf{y}}\left[\ell(\mathbf{y}, f(\mathbf{x}))\right]$$

for a given loss function

$$\ell(\mathbf{y}, f(\mathbf{x})) = \frac{1}{2}|\mathbf{y} - f(\mathbf{x})|^2$$

and $$\mathbb{E}_{\mathbf{x},\mathbf{y}}\left[\ell(\mathbf{y}, f(\mathbf{x}))\right] = \int_{(\mathbf{x},\mathbf{y})\in\mathscr{D}} \ell(\mathbf{y}, f(\mathbf{x}))\,\rho(\mathbf{x}, \mathbf{y})\,dxdy$$

Training error vs. expected error

$E(f) = \mathbb{E}_{\mathbf{x},\mathbf{y}}\left[\ell(\mathbf{y}, f(\mathbf{x}))\right]$ is known as

Training error vs. expected error

$E(f) = \mathbb{E}_{\mathbf{x},\mathbf{y}} \left[\ell(\mathbf{y}, f(\mathbf{x})) \right]$ is known as

- *Population risk*
- *Expected risk*
- *Expected error*

Training error vs. expected error

$E(f) = \mathbb{E}_{\mathbf{x},\mathbf{y}}\left[\ell(\mathbf{y}, f(\mathbf{x}))\right]$ is known as

- *Population risk*
- *Expected risk*
- *Expected error*

This is the quantity that we are fundamentally interested in

Training error vs. expected error

$E(f) = \mathbb{E}_{\mathbf{x},\mathbf{y}}\left[\ell(\mathbf{y}, f(\mathbf{x}))\right]$ is known as

- *Population risk*
- *Expected risk*
- *Expected error*

This is the quantity that we are fundamentally interested in

but it is unknown as we do not know $\mathscr{D}$

(nor the probability density function ρ)

Training error vs. expected error

$E(f) = \mathbb{E}_{\mathbf{x},\mathbf{y}}\left[\ell(\mathbf{y}, f(\mathbf{x}))\right]$ is known as

- *Population risk*
- *Expected risk*
- *Expected error*

This is the quantity that we are fundamentally interested in

but it is unknown as we do not know $\mathscr{D}$

(nor the probability density function ρ)

Hence, we cannot compute $E(f_S)$!

Training error vs. expected error

What can we do instead?

Training error vs. expected error

What can we do instead? We are given the set of samples S

It is therefore natural to compute the empirical risk

$$L_S(f) = \frac{1}{|S|} \sum_{(\mathbf{x}_i, \mathbf{y}_i) \in S} \ell(\mathbf{y}_i, f(\mathbf{x}_i))$$

Training error vs. expected error

What can we do instead? We are given the set of samples S

It is therefore natural to compute the empirical risk

$$L_S(f) = \frac{1}{|S|} \sum_{(\mathbf{x}_i, \mathbf{y}_i) \in S} \ell(\mathbf{y}_i, f(\mathbf{x}_i))$$

The problem with this quantity is that f is usually a function of S itself:

$$L_S(f_S) = \frac{1}{|S|} \sum_{(\mathbf{x}_i, \mathbf{y}_i) \in S} \ell(\mathbf{y}_i, f_S(\mathbf{x}_i))$$

Training error vs. expected error

What can we do instead? We are given the set of samples S

It is therefore natural to compute the empirical risk

$$L_S(f) = \frac{1}{|S|} \sum_{(\mathbf{x}_i, \mathbf{y}_i) \in S} \ell(\mathbf{y}_i, f(\mathbf{x}_i))$$

The problem with this quantity is that f is usually a function of S itself:

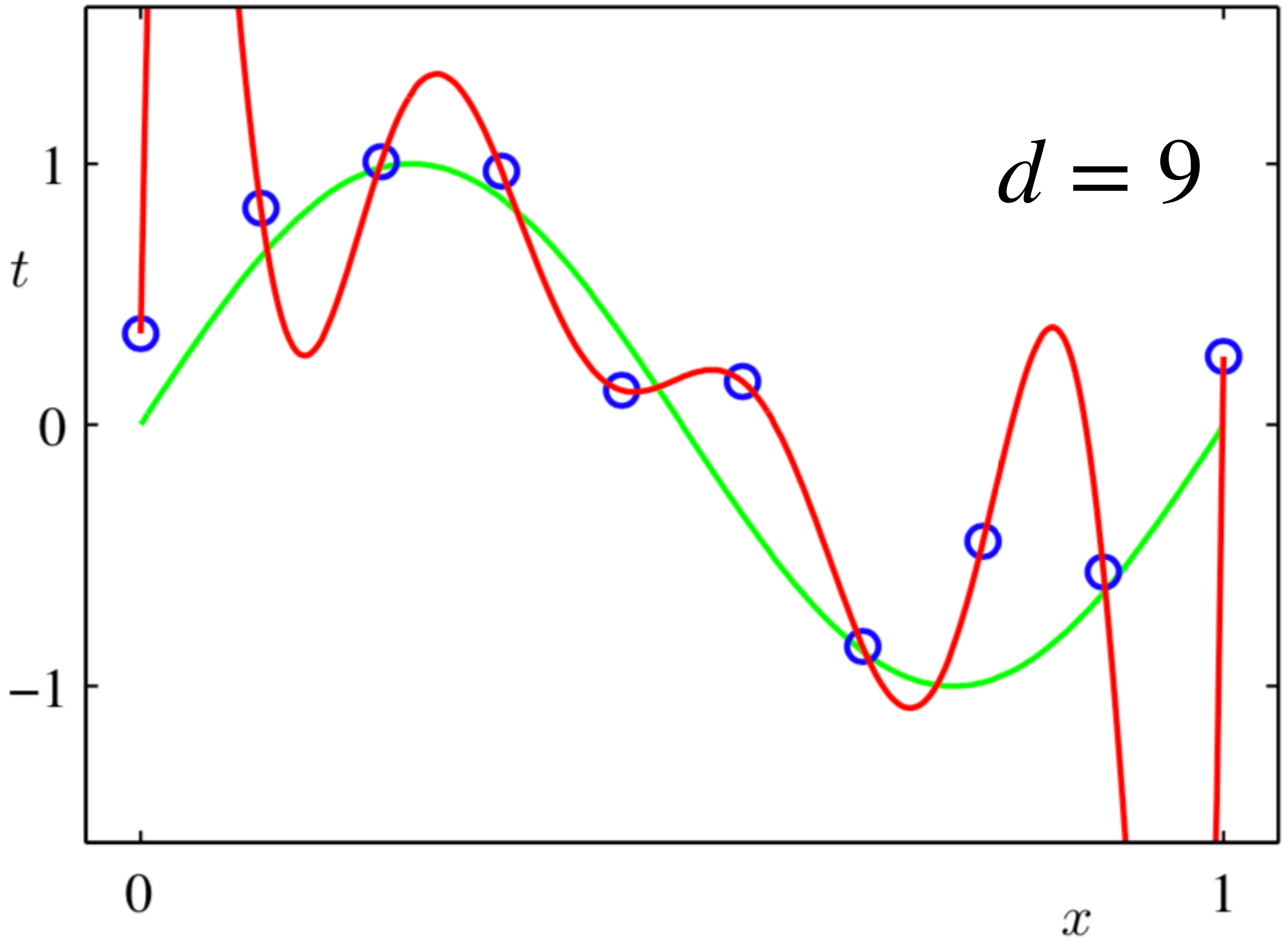
$$L_S(f_S) = \frac{1}{|S|} \sum_{(\mathbf{x}_i, \mathbf{y}_i) \in S} \ell(\mathbf{y}_i, f_S(\mathbf{x}_i))$$

This quantity is also known as the *training error*

Training error vs. expected error

$$L_S(f_S) = \frac{1}{|S|} \sum_{(\mathbf{x}_i,\mathbf{y}_i)\in S} \ell(\mathbf{y}_i, f_S(\mathbf{x}_i))$$

Training error is usually not representative for generalisation error, remember



From Bishop. Pattern Recognition & Machine Learning

Splitting the data

In order to avoid that we validate our model on the same data that we train it on, we can split the data:

Splitting the data

In order to avoid that we validate our model on the same data that we train it on, we can split the data:

Splitting the data

In order to avoid that we validate our model on the same data that we train it on, we can split the data:

Properties: $S = S_t \cup S_v$, and usually also $S_t \cap S_v = \emptyset$

Splitting the data

In order to avoid that we validate our model on the same data that we train it on, we can split the data:

Properties: $S = S_t \cup S_v$, and usually also $S_t \cap S_v = \emptyset$

Example: take original data and split into 80% training and 20 % validation data

Training error vs. validation error

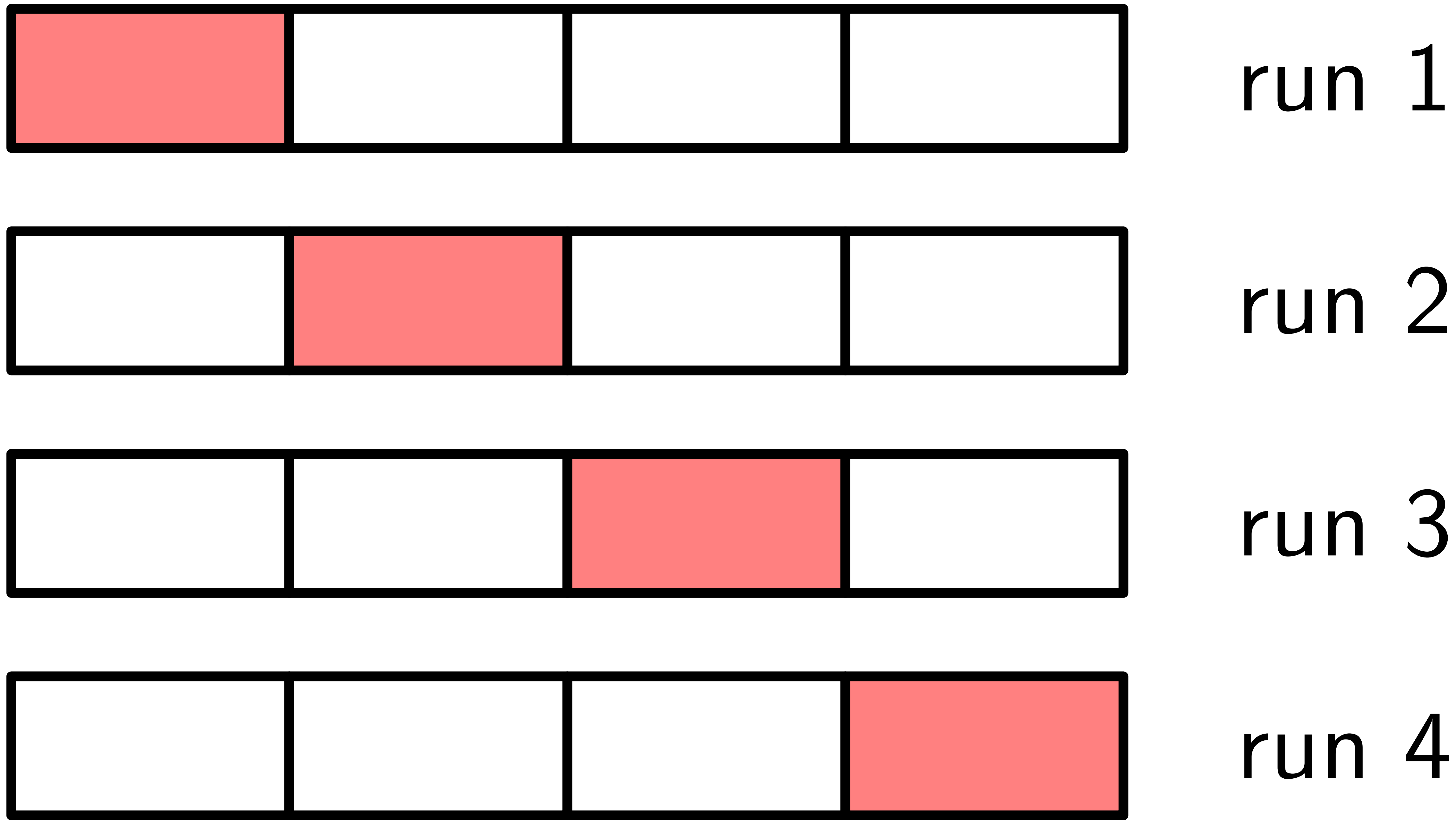
Training error: $$L_t(f_t) = \frac{1}{|S_t|} \sum_{(\mathbf{x}_i, \mathbf{y}_i) \in S_t} \ell(\mathbf{y}_i, f_t(\mathbf{x}_i))$$

Validation error: $$L_v(f_t) = \frac{1}{|S_v|} \sum_{(\mathbf{x}_i, \mathbf{y}_i) \in S_v} \ell(\mathbf{y}_i, f_t(\mathbf{x}_i))$$

where f_t is short-hand-notation for

$$f_t := f_{S_t}$$

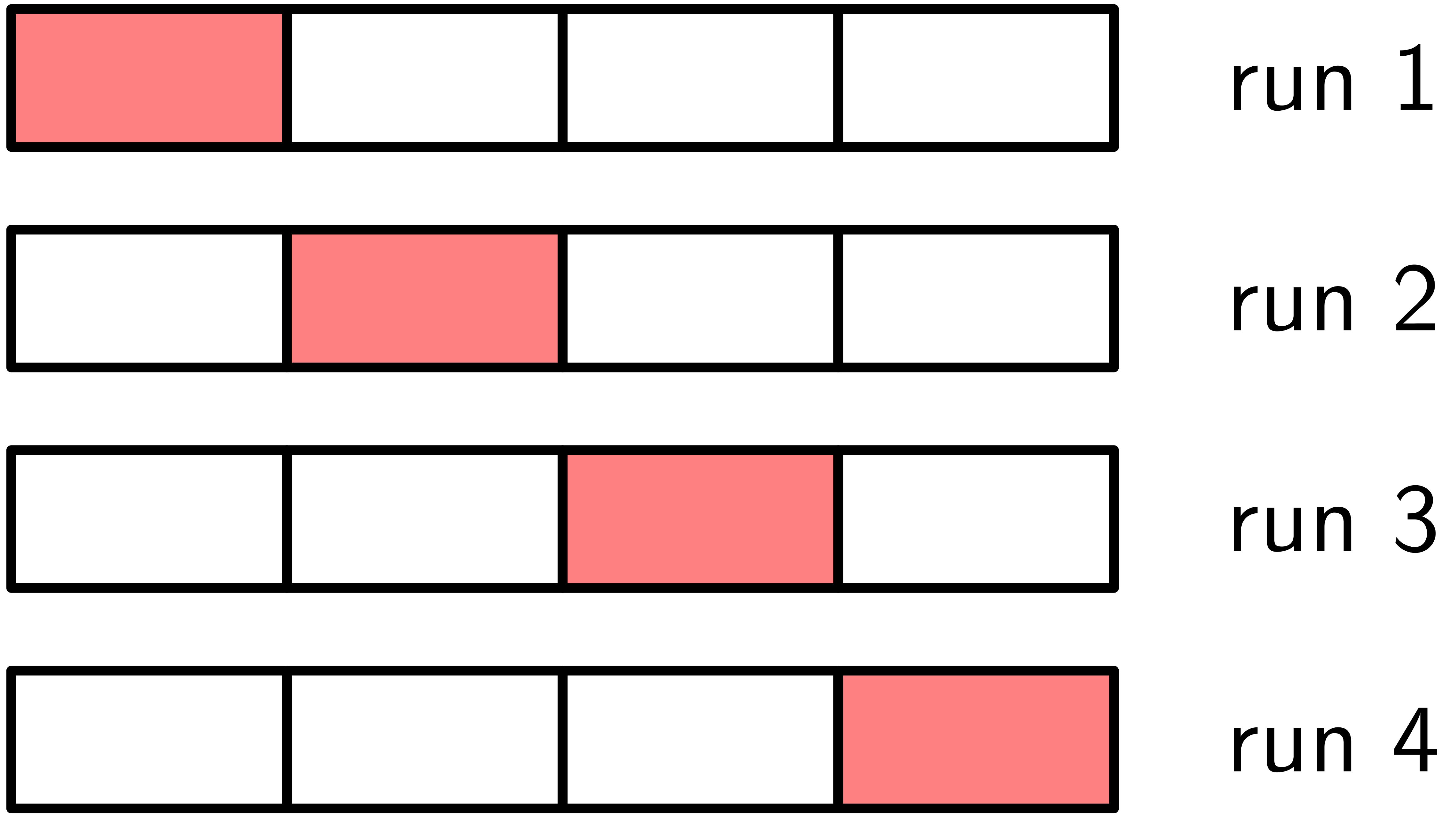
Cross-validation



K-fold cross-validation

From Bishop. Pattern Recognition & Machine Learning

Cross-validation

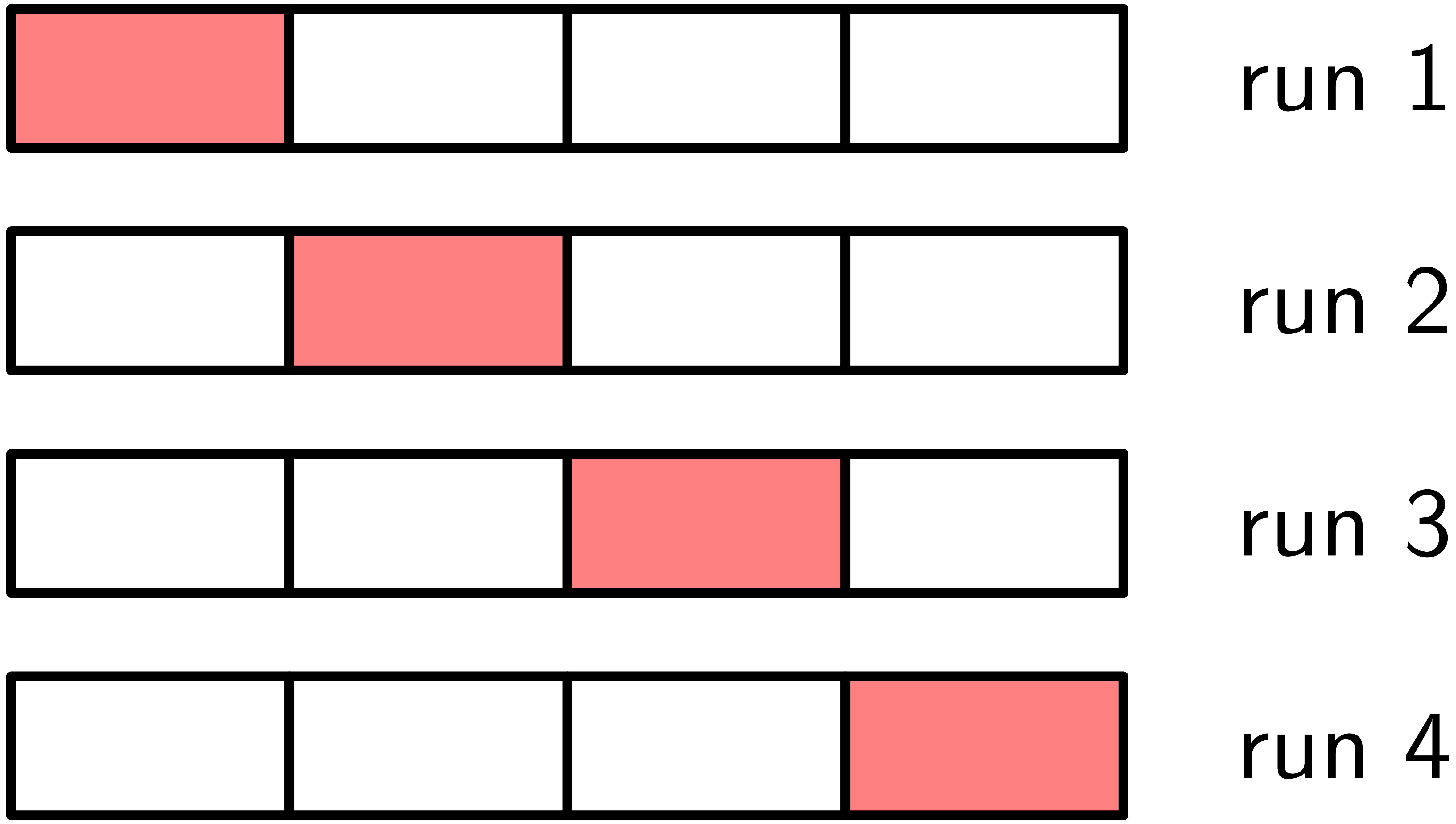


K-fold cross-validation

- Randomly partition data into K groups

From Bishop. Pattern Recognition & Machine Learning

Cross-validation

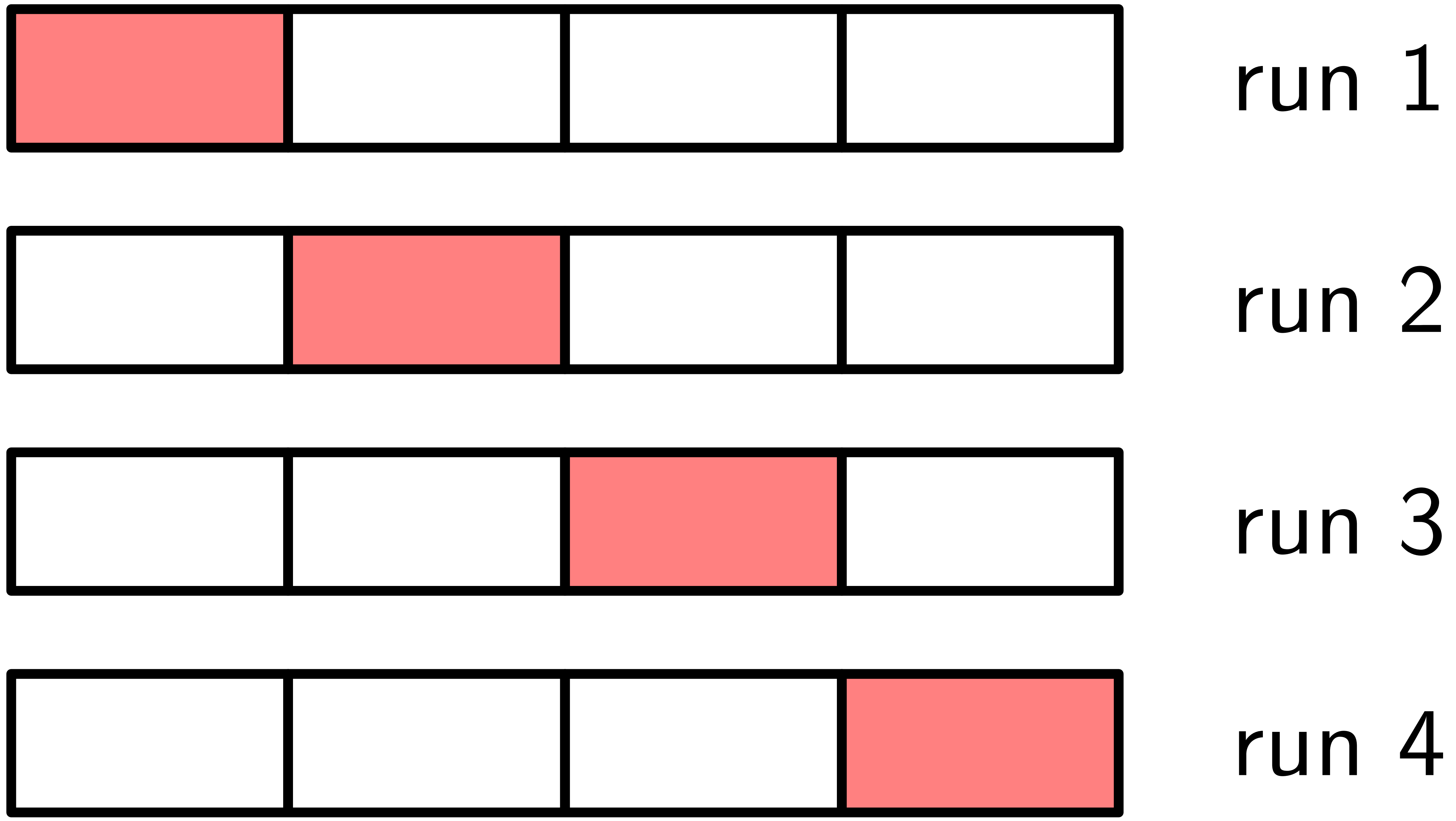


From Bishop. Pattern Recognition & Machine Learning

K-fold cross-validation

- Randomly partition data into K groups
- Train K times, each time leaving 1 group for testing and $K-1$ for training

Cross-validation



From Bishop. Pattern Recognition & Machine Learning

K-fold cross-validation

- Randomly partition data into K groups
- Train K times, each time leaving 1 group for testing and $K-1$ for training
- Average the K results

The validation error

Central question that we need to address:

How do we choose hyper parameters in

$$f_t(\mathbf{x}) = \langle \phi(\mathbf{x}), \mathbf{w}_\alpha \rangle \text{ for } \mathbf{w}_\alpha = \arg\min_{\mathbf{w}} \left\{ \frac{1}{2s} \sum_{(\mathbf{x}_i, \mathbf{y}_i) \in S_t} \left| \langle \phi(\mathbf{x}_i), \mathbf{w} \rangle - \mathbf{y}_i \right|^2 + \frac{\alpha}{2} \|\mathbf{w}\|^2 \right\}$$

such that we minimise

$$L_v(f_t) = \frac{1}{|S_v|} \sum_{(\mathbf{x}_i, \mathbf{y}_i) \in S_v} \ell(\mathbf{y}_i, f_t(\mathbf{x}_i)) \qquad ?$$

The validation error

This is a bi-level optimisation problem:

$$(\hat{\alpha}, \hat{d}) = \arg\min_{\alpha,d} \left\{ L_v(f_t) = \frac{1}{|S_v|} \sum_{(\mathbf{x}_i, \mathbf{y}_i) \in S_v} \ell(\mathbf{y}_i, f_t(\mathbf{x}_i)) \right\}$$

subject to

$$f_t(\mathbf{x}) = \langle \phi(\mathbf{x}), \mathbf{w}_\alpha \rangle \text{ for } \mathbf{w}_\alpha = \arg\min_{\mathbf{w}} \left\{ \frac{1}{2s} \sum_{(\mathbf{x}_i, \mathbf{y}_i) \in S_t} \left| \langle \phi(\mathbf{x}_i), \mathbf{w} \rangle - \mathbf{y}_i \right|^2 + \frac{\alpha}{2} \|\mathbf{w}\|^2 \right\}$$

The validation error

This is a bi-level optimisation problem:

$$(\hat{\alpha}, \hat{d}) = \arg\min_{\alpha,d} \left\{ L_v(f_t) = \frac{1}{|S_v|} \sum_{(\mathbf{x}_i, \mathbf{y}_i) \in S_v} \ell(\mathbf{y}_i, f_t(\mathbf{x}_i)) \right\}$$

Upper-level problem

subject to

$$f_t(\mathbf{x}) = \langle \phi(\mathbf{x}), \mathbf{w}_\alpha \rangle \text{ for } \mathbf{w}_\alpha = \arg\min_{\mathbf{w}} \left\{ \frac{1}{2s} \sum_{(\mathbf{x}_i, \mathbf{y}_i) \in S_t} \left| \langle \phi(\mathbf{x}_i), \mathbf{w} \rangle - \mathbf{y}_i \right|^2 + \frac{\alpha}{2} \|\mathbf{w}\|^2 \right\}$$

The validation error

This is a bi-level optimisation problem:

$$(\hat{\alpha}, \hat{d}) = \arg\min_{\alpha,d} \left\{ L_v(f_t) = \frac{1}{|S_v|} \sum_{(\mathbf{x}_i, \mathbf{y}_i) \in S_v} \ell(\mathbf{y}_i, f_t(\mathbf{x}_i)) \right\}$$

Upper-level problem

subject to

$$f_t(\mathbf{x}) = \langle \phi(\mathbf{x}), \mathbf{w}_\alpha \rangle \text{ for } \mathbf{w}_\alpha = \arg\min_{\mathbf{w}} \left\{ \frac{1}{2s} \sum_{(\mathbf{x}_i,\mathbf{y}_i)\in S_t} \left| \langle \phi(\mathbf{x}_i), \mathbf{w} \rangle - \mathbf{y}_i \right|^2 + \frac{\alpha}{2} \|\mathbf{w}\|^2 \right\}$$

Lower-level problem

The validation error

This is a bi-level optimisation problem:

$$(\hat{\alpha}, \hat{d}) = \arg\min_{\alpha,d} \left\{ L_v(f_t) = \frac{1}{|S_v|} \sum_{(\mathbf{x}_i, \mathbf{y}_i) \in S_v} \ell(\mathbf{y}_i, f_t(\mathbf{x}_i)) \right\}$$

Upper-level problem

subject to

$$f_t(\mathbf{x}) = \langle \phi(\mathbf{x}), \mathbf{w}_\alpha \rangle \text{ for } \mathbf{w}_\alpha = \arg\min_{\mathbf{w}} \left\{ \frac{1}{2s} \sum_{(\mathbf{x}_i, \mathbf{y}_i) \in S_t} \left| \langle \phi(\mathbf{x}_i), \mathbf{w} \rangle - \mathbf{y}_i \right|^2 + \frac{\alpha}{2} \|\mathbf{w}\|^2 \right\}$$

Lower-level problem

Important: given α and d we are guaranteed to find the best possible solution of the lower-level problem

How can we solve such a problem?

Grid search

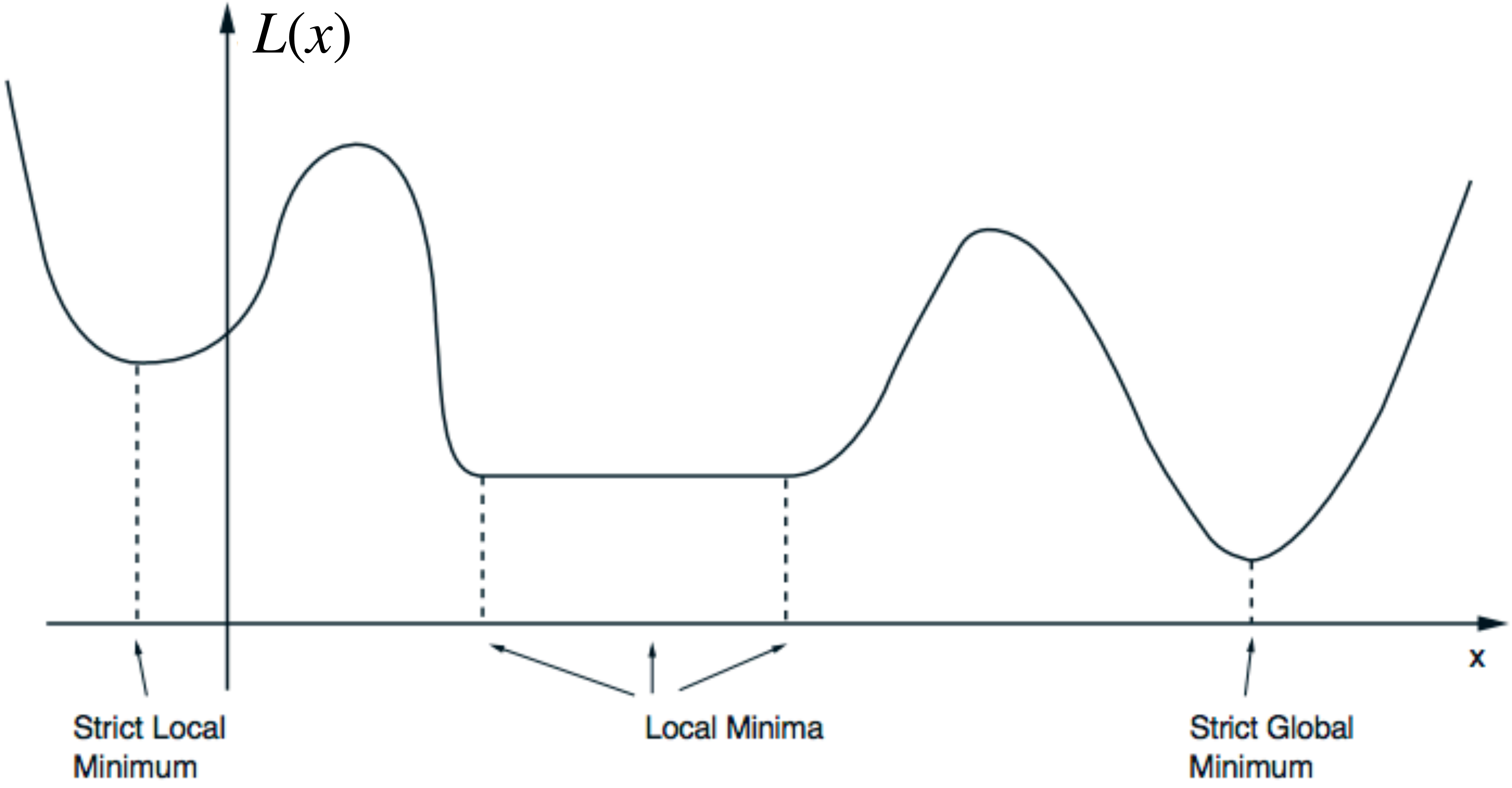
The easiest of all optimisation algorithms is grid search:

Grid search

The easiest of all optimisation algorithms is grid search:

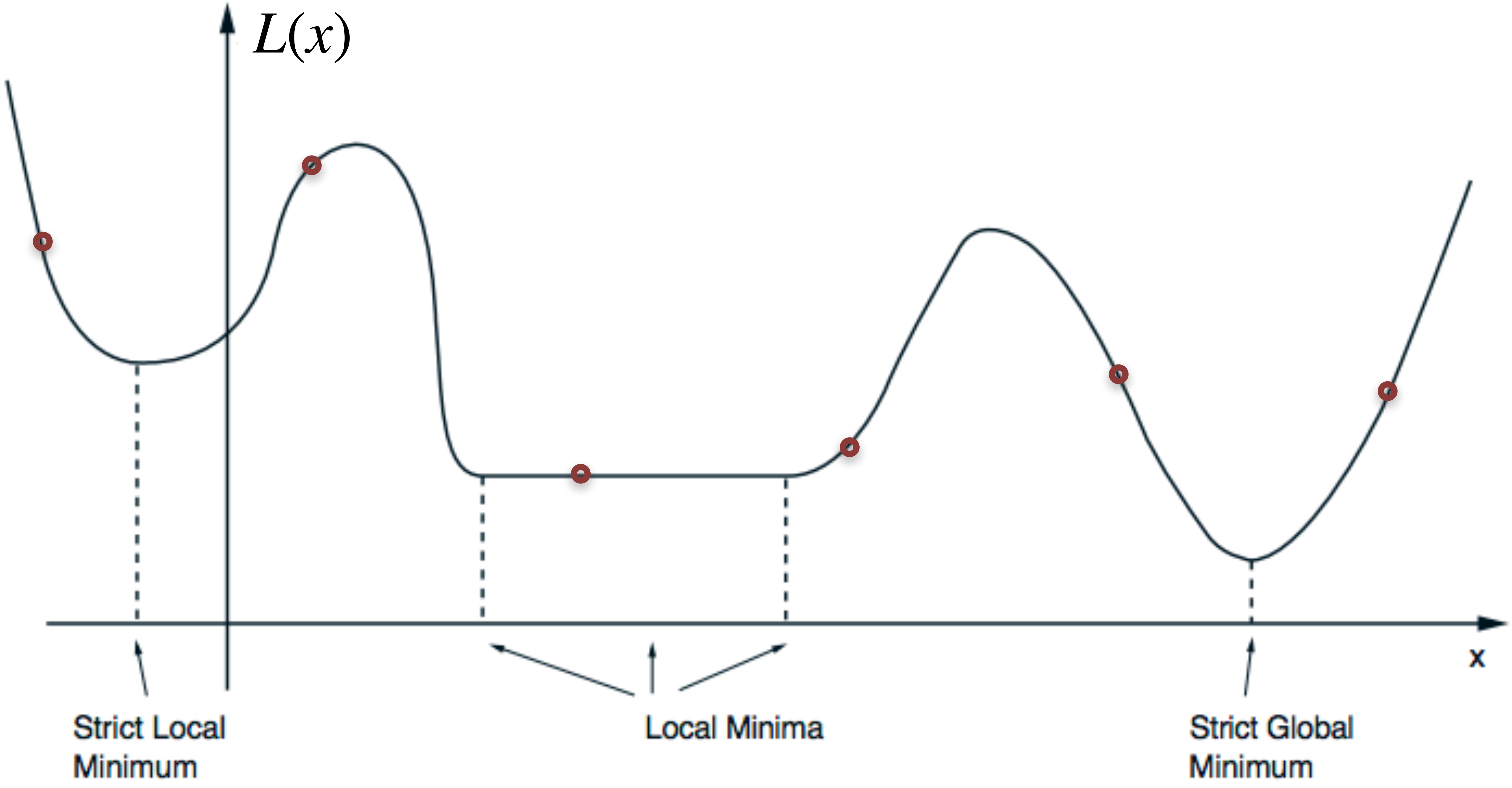
Evaluate a function L at points on a grid and record smallest value

Grid search



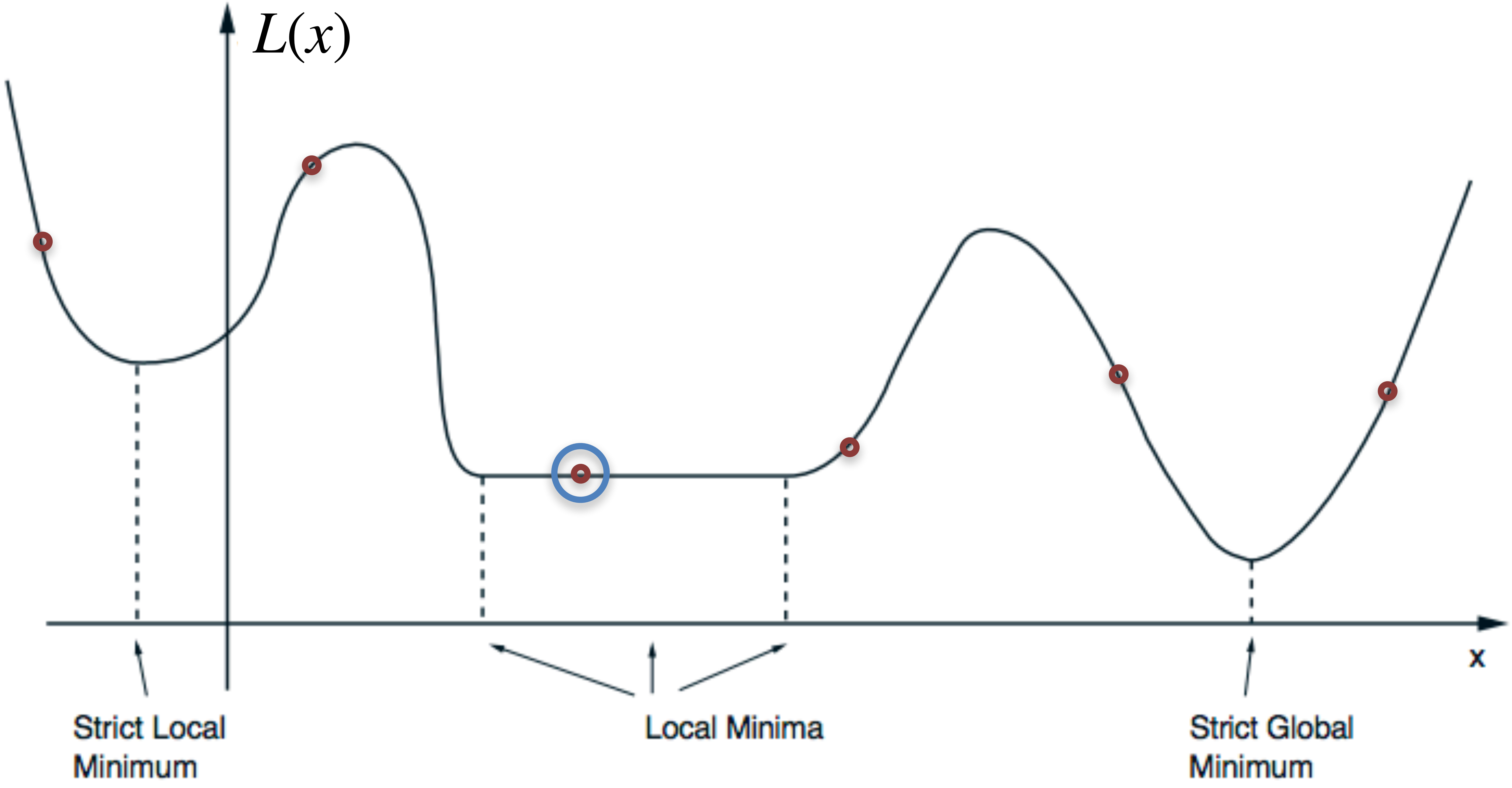
From Bertsekas, Nonlinear programming

Grid search



From Bertsekas, Nonlinear programming

Grid search



From Bertsekas, Nonlinear programming

Grid search

The easiest of all optimisation algorithms is grid search:

Evaluate a function L at points on a grid and record smallest value

Grid search

The easiest of all optimisation algorithms is grid search:

Evaluate a function L at points on a grid and record smallest value

Advantages:

Grid search

The easiest of all optimisation algorithms is grid search:

Evaluate a function L at points on a grid and record smallest value

Advantages:

- works for any kind of function!

Grid search

The easiest of all optimisation algorithms is grid search:

Evaluate a function L at points on a grid and record smallest value

Advantages:

- works for any kind of function!
- very easy to implement

Grid search

The easiest of all optimisation algorithms is grid search:

Evaluate a function L at points on a grid and record smallest value

Advantages:

- works for any kind of function!
- very easy to implement

Disadvantage: computationally infeasible for large no. of parameters

Grid search

The easiest of all optimisation algorithms is grid search:

Evaluate a function L at points on a grid and record smallest value

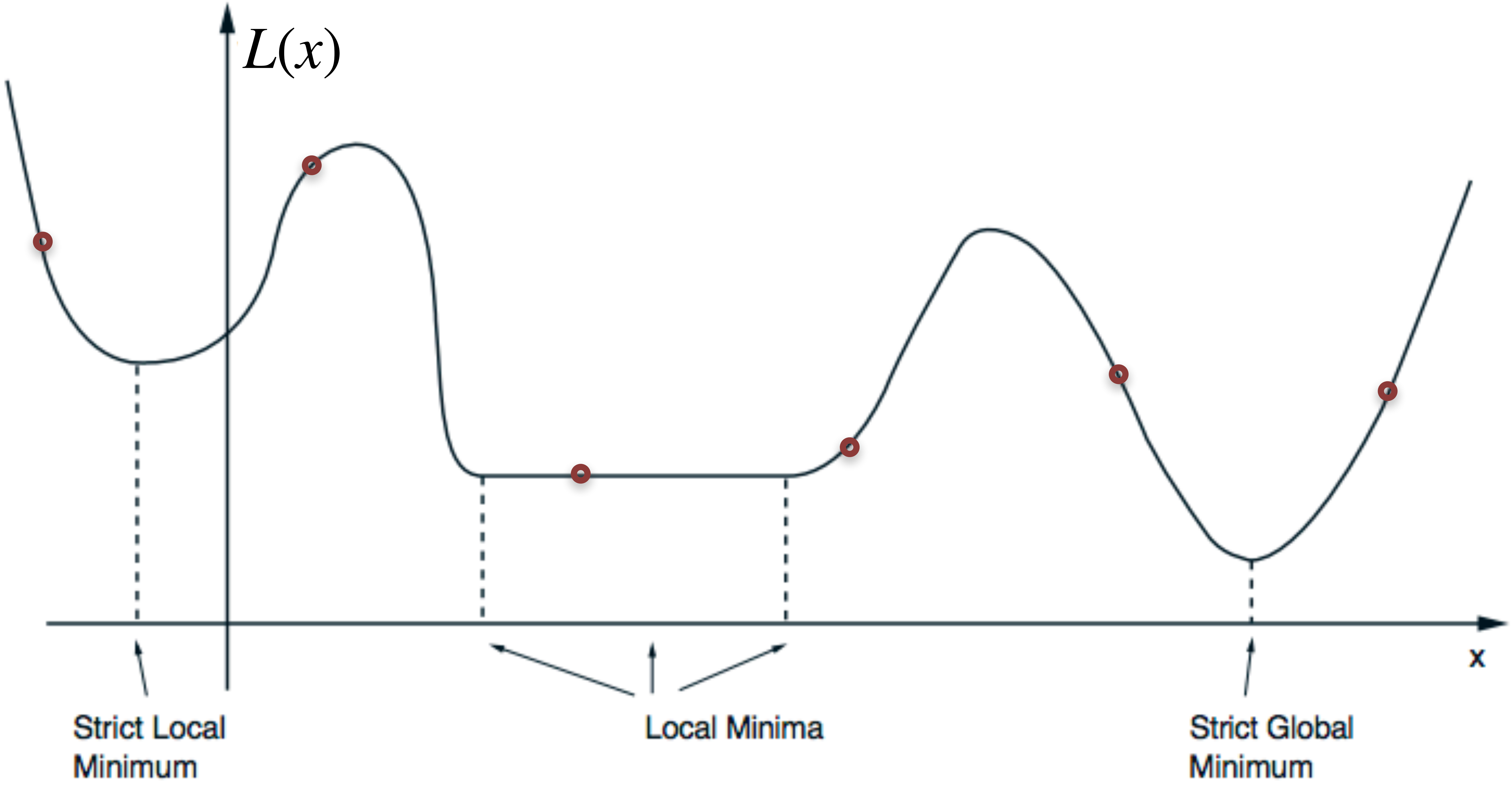
Advantages:

- works for any kind of function!
- very easy to implement

Disadvantage: computationally infeasible for large no. of parameters

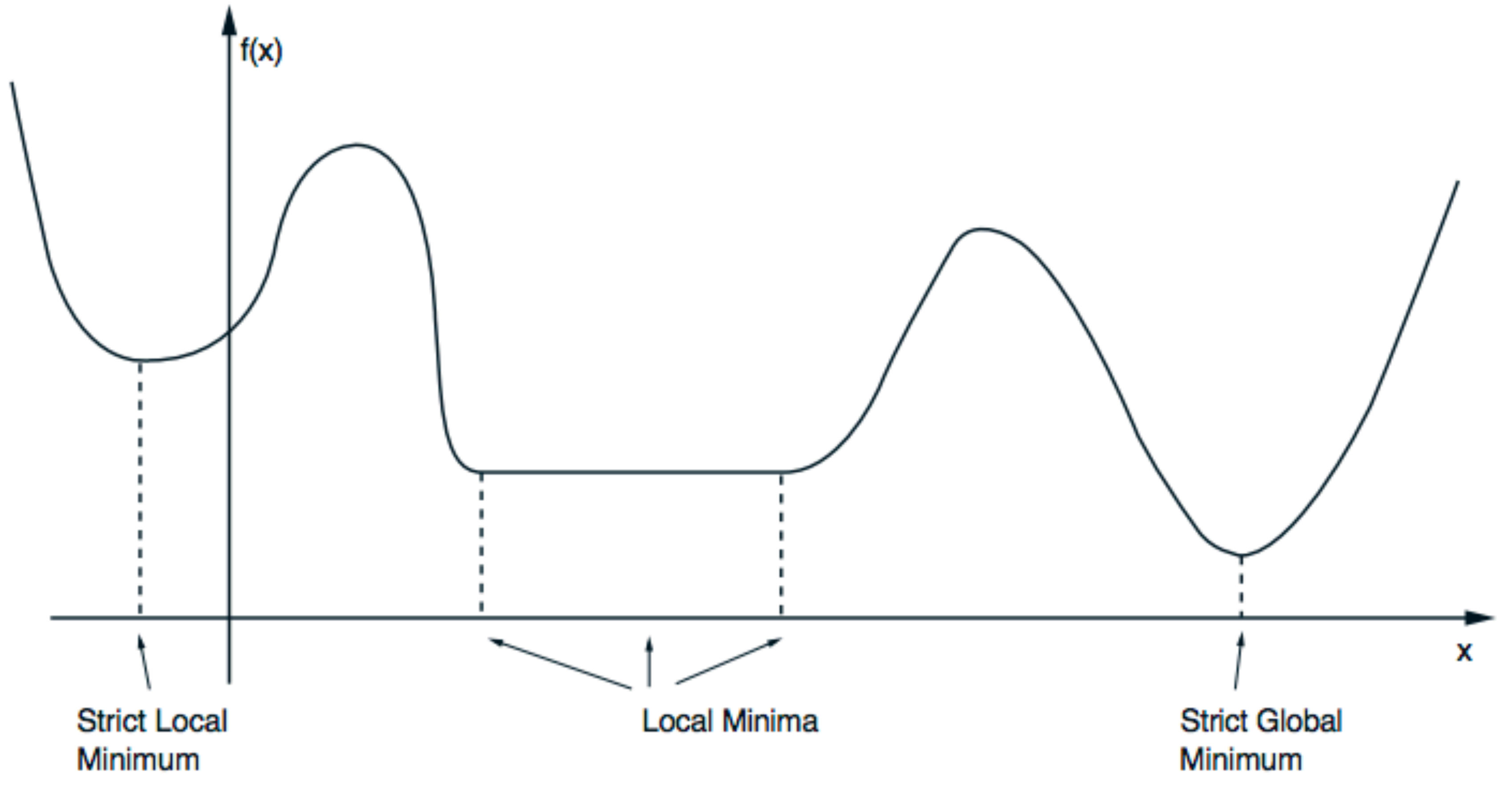
Sample L at n points $\mathbf{p} \in \mathbb{R}^m$ in each dimension $\implies$ m^n evaluations of L

Other disadvantage: no guarantee that we end up close to a minimum!



From Bertsekas, Nonlinear programming

Other disadvantage: no guarantee that we end up close to a minimum!



From Bertsekas, Nonlinear programming

An element is called a local minimum point if there exists $\hat{p}$ such that

$$L(\hat{\mathbf{p}}) \leq L(\mathbf{p}) \qquad \forall \mathbf{p} \text{ with } \|\mathbf{p} - \hat{\mathbf{p}}\| \leq \varepsilon$$

An element is called a global minimum point if there exists $\hat{\mathbf{p}}$ such that

$$L(\hat{\mathbf{p}}) \leq L(\mathbf{p}) \qquad \forall \mathbf{p} \in \mathbb{R}^m$$

Other noticeable approaches

Random search: grid search with random selection of parameter combinations

Gradient-based opt.: we will see in more details next

Bayesian opt.: builds a probabilistic model of function mapping hyperparameters to validation error

Evolutionary opt.: use evolutionary algorithms to search space of hyperparameters

Other noticeable approaches

Random search: grid search with random selection of parameter combinations

Gradient-based opt.: we will see in more details next

Bayesian opt.: builds a probabilistic model of function mapping hyperparameters to validation error

Evolutionary opt.: use evolutionary algorithms to search space of hyperparameters

In this module, grid search will usually be sufficient as we deal with relatively few hyperparameters

Bias-variance decomposition

Assume the following data generation model:

$$y = f(x) + \varepsilon$$

Bias-variance decomposition

Assume the following data generation model:

$$y = f(x) + \varepsilon$$

for

- some arbitrary and unknown function f

Bias-variance decomposition

Assume the following data generation model:

$$y = f(x) + \varepsilon$$

for
- some arbitrary and unknown function f
- additive iid noise ε with $\mathbb{E}_\varepsilon[\varepsilon] = 0$ and $\text{Var}_\varepsilon[\varepsilon] = \sigma^2$

Bias-variance decomposition

Assume the following data generation model:

$$y = f(x) + \varepsilon$$

for

- some arbitrary and unknown function f
- additive iid noise ε with $\mathbb{E}_\varepsilon[\varepsilon] = 0$ and $\mathrm{Var}_\varepsilon[\varepsilon] = \sigma^2$

We further assume that each pair (x, y) is a sample of the distribution $\mathscr{D}$

Bias-variance decomposition

Assume the following data generation model:

$$y = f(x) + \varepsilon$$

for

- some arbitrary and unknown function f
- additive iid noise ε with $\mathbb{E}_\varepsilon[\varepsilon] = 0$ and $\text{Var}_\varepsilon[\varepsilon] = \sigma^2$

We further assume that each pair (x, y) is a sample of the distribution $\mathcal{D}$

Training data: $S_t := \left\{(x_i, y_i) \text{ iid } \sim \mathcal{D}\right\}_{i=1}^{s}$

Bias-variance decomposition

Assume the following data generation model:

$$y = f(x) + \varepsilon$$

for

- some arbitrary and unknown function f
- additive iid noise ε with $\mathbb{E}_{\varepsilon}[\varepsilon] = 0$ and $\text{Var}_{\varepsilon}[\varepsilon] = \sigma^2$

We further assume that each pair (x, y) is a sample of the distribution $\mathscr{D}$

Training data: $S_t := \left\{(x_i, y_i) \text{ iid } \sim \mathscr{D}\right\}_{i=1}^{s}$

Prediction function: f_t

Bias-variance decomposition

For fixed input $\tilde{x}$, look at error between model and prediction function:

$$\left(f(\tilde{x}) + \varepsilon - f_t(\tilde{x})\right)^2$$

Bias-variance decomposition

For fixed input $\tilde{x}$, look at error between model and prediction function:

$$\left(f(\tilde{x}) + \varepsilon - f_t(\tilde{x})\right)^2$$

Imagine we do this for many different instances of S_t and ε

then we can look at the expected value of the error:

$$\mathbb{E}_{t,\varepsilon}\left[\left(f(\tilde{x}) + \varepsilon - f_t(\tilde{x})\right)^2\right]$$

Bias-variance decomposition

$$\mathbb{E}_{t,\varepsilon}\left[\left(f(\tilde{x}) + \varepsilon - f_t(\tilde{x})\right)^2\right]$$

$$= \sigma^2 \quad + \quad \left(f(\tilde{x}) - \mathbb{E}_t[f_t(\tilde{x})]\right)^2 \quad + \quad \mathbb{E}_t\left[\left(\mathbb{E}_t[f_t(\tilde{x})] - f_t(\tilde{x})\right)^2\right]$$

Bias-variance decomposition

$$\mathbb{E}_{t,\varepsilon}\left[\left(f(\tilde{x}) + \varepsilon - f_t(\tilde{x})\right)^2\right]$$

$$= \underbrace{\sigma^2}_{\text{noise variance}} \quad + \quad \left(f(\tilde{x}) - \mathbb{E}_t[f_t(\tilde{x})]\right)^2 \quad + \quad \mathbb{E}_t\left[\left(\mathbb{E}_t[f_t(\tilde{x})] - f_t(\tilde{x})\right)^2\right]$$

Bias-variance decomposition

$$\mathbb{E}_{t,\varepsilon}\left[\left(f(\tilde{x}) + \varepsilon - f_t(\tilde{x})\right)^2\right]$$

$$= \underbrace{\sigma^2}_{\text{noise variance}} \quad + \quad \left(f(\tilde{x}) - \mathbb{E}_t[f_t(\tilde{x})]\right)^2 \quad + \quad \mathbb{E}_t\left[\left(\mathbb{E}_t[f_t(\tilde{x})] - f_t(\tilde{x})\right)^2\right]$$

Always there
in the data!

Bias-variance decomposition

$$\mathbb{E}_{t,\varepsilon}\left[\left(f(\tilde{x}) + \varepsilon - f_t(\tilde{x})\right)^2\right]$$

$$= \underbrace{\sigma^2}_{\text{noise variance}} + \underbrace{\left(f(\tilde{x}) - \mathbb{E}_t[f_t(\tilde{x})]\right)^2}_{\text{bias}} + \mathbb{E}_t\left[\left(\mathbb{E}_t[f_t(\tilde{x})] - f_t(\tilde{x})\right)^2\right]$$

Always there in the data!

Bias-variance decomposition

$$\mathbb{E}_{t,\varepsilon}\left[\left(f(\tilde{x}) + \varepsilon - f_t(\tilde{x})\right)^2\right]$$

$$= \underbrace{\sigma^2}_{\text{noise variance}} \quad + \quad \underbrace{\left(f(\tilde{x}) - \mathbb{E}_t[f_t(\tilde{x})]\right)^2}_{\text{bias}} \quad + \quad \mathbb{E}_t\left[\left(\mathbb{E}_t[f_t(\tilde{x})] - f_t(\tilde{x})\right)^2\right]$$

Always there in the data!

Systematic error/bias

Bias-variance decomposition

$$\mathbb{E}_{t,\varepsilon}\left[\left(f(\tilde{x}) + \varepsilon - f_t(\tilde{x})\right)^2\right]$$

$$= \underbrace{\sigma^2}_{\text{noise variance}} + \underbrace{\left(f(\tilde{x}) - \mathbb{E}_t[f_t(\tilde{x})]\right)^2}_{\text{bias}} + \underbrace{\mathbb{E}_t\left[\left(\mathbb{E}_t[f_t(\tilde{x})] - f_t(\tilde{x})\right)^2\right]}_{\text{variance}}$$

Always there in the data!

Systematic error/bias

Bias-variance decomposition

$$\mathbb{E}_{t,\varepsilon}\left[\left(f(\tilde{x}) + \varepsilon - f_t(\tilde{x})\right)^2\right]$$

$$= \underbrace{\sigma^2}_{\text{noise variance}} + \underbrace{\left(f(\tilde{x}) - \mathbb{E}_t[f_t(\tilde{x})]\right)^2}_{\text{bias}} + \underbrace{\mathbb{E}_t\left[\left(\mathbb{E}_t[f_t(\tilde{x})] - f_t(\tilde{x})\right)^2\right]}_{\text{variance}}$$

Always there in the data!

Systematic error/bias

Variance in the prediction function. How much one instance deviates from the average

Bias-variance decomposition

Example: polynomial regression

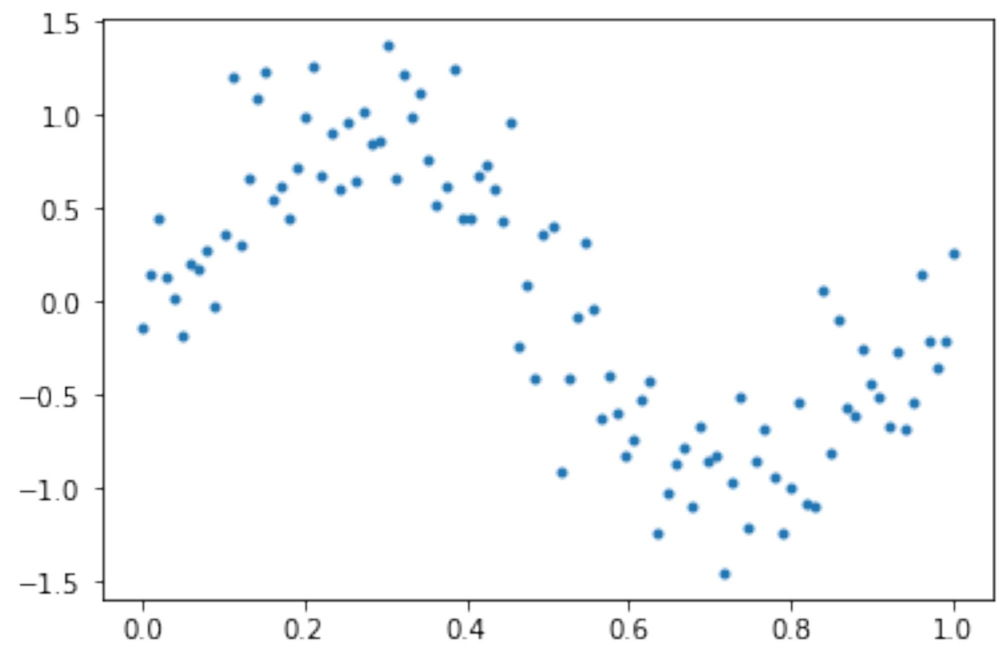
$$f(x) = \sin(2\pi x)\,, \quad x \in [0,1]$$

Bias-variance decomposition

Example: polynomial regression

$$f(x) = \sin(2\pi x)\,, \quad x \in [0,1]$$

Different noisy instances for training & testing

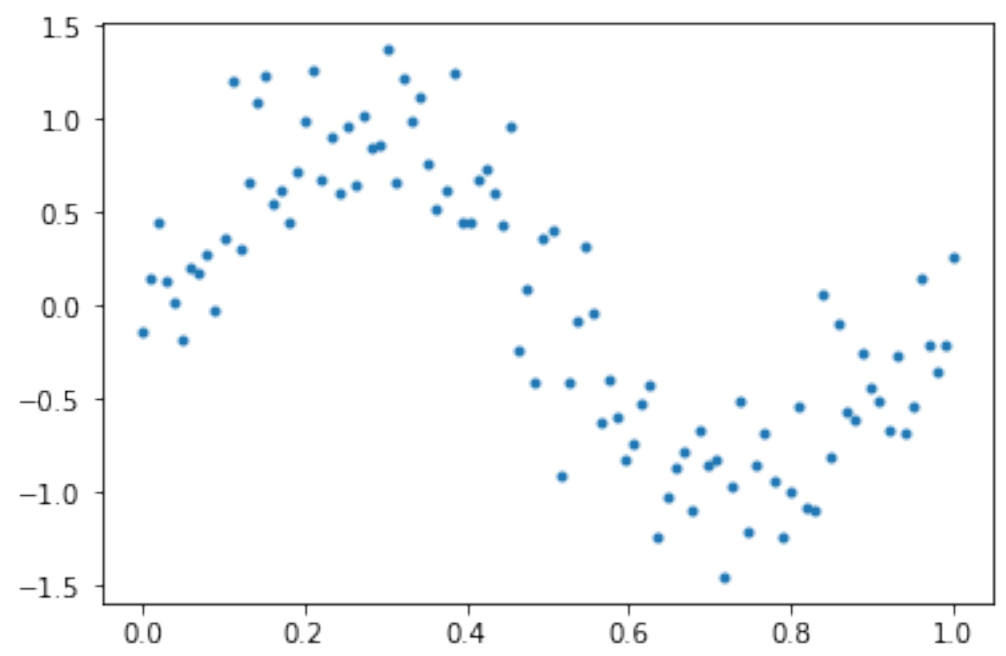


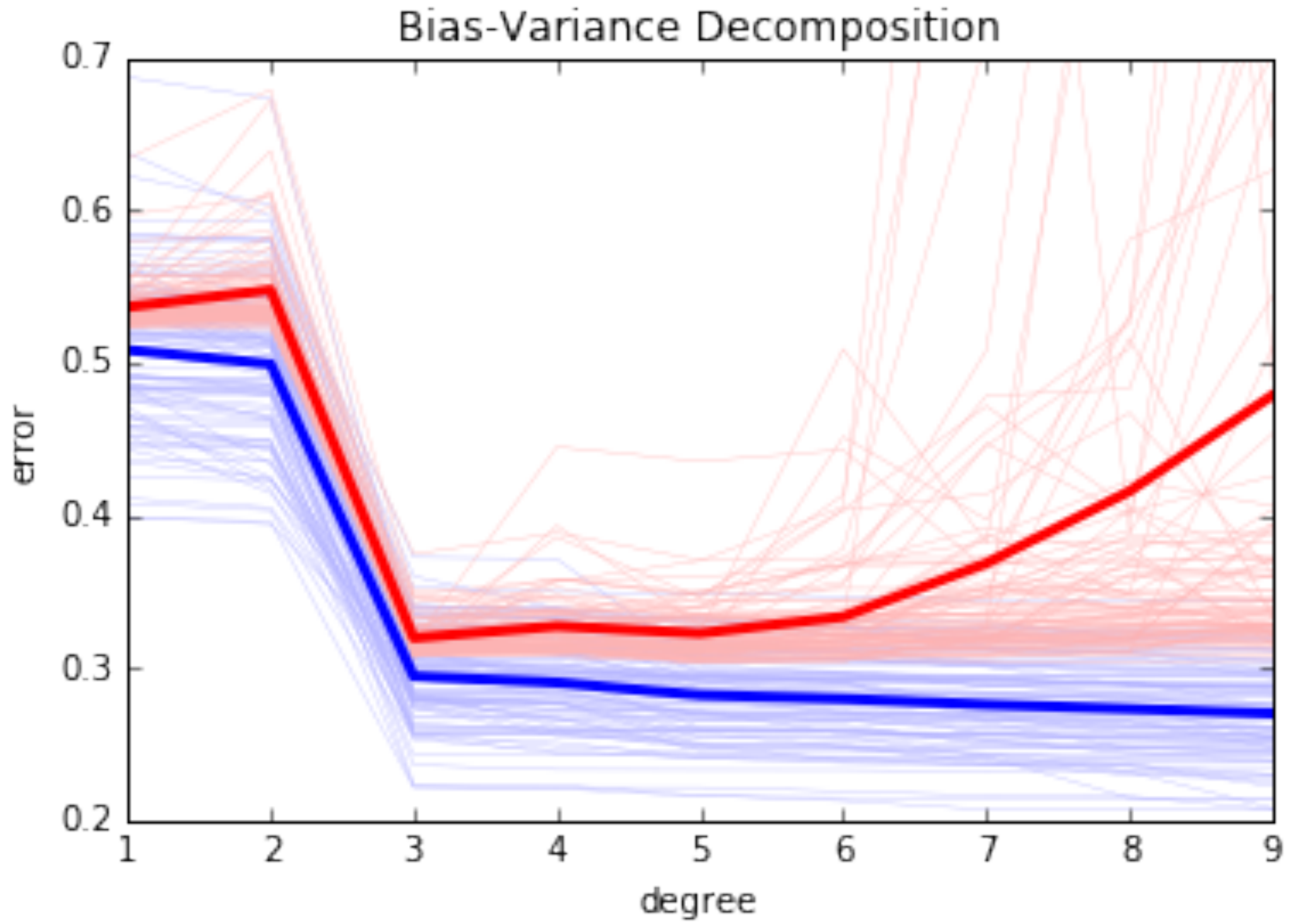
Bias-variance decomposition

Example: polynomial regression

$$f(x) = \sin(2\pi x)\,, \quad x \in [0,1]$$

Different noisy instances for training & testing





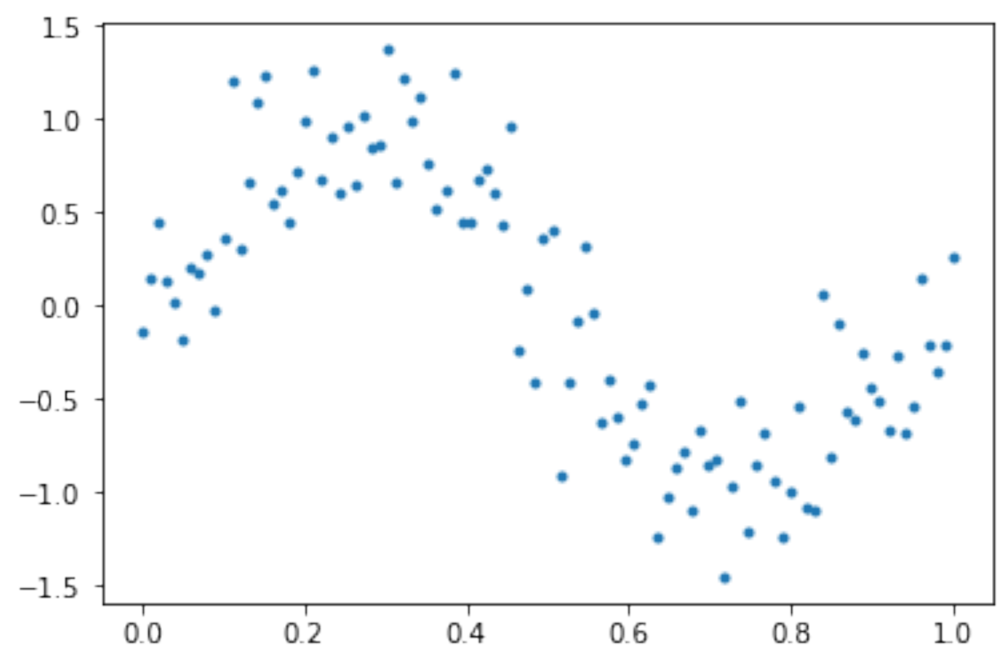
Model: $f_t(x) = \sum_{k=0}^{d} x^k w_k$ for $\hat{w} = \arg\min_w \left\{ \frac{1}{2} \sum_{i=1}^{s} \left| \sum_{k=0}^{d} x_{ij}^k w_k - y_i \right|^2 \right\}$ and $S_t := \{(x_i, y_i) \text{ iid } \sim \mathscr{D}\}_{i=1}^{s}$

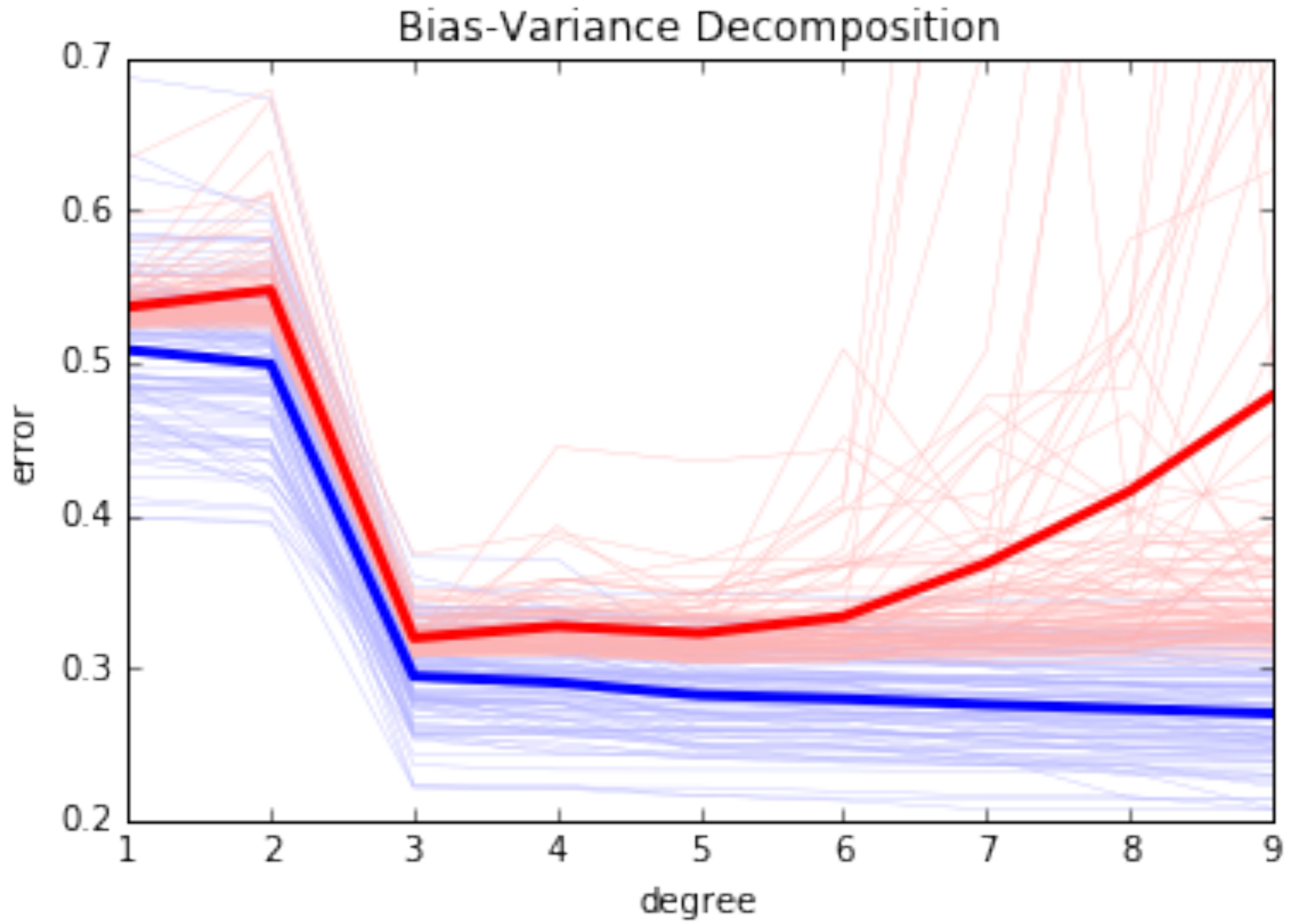
Bias-variance decomposition

Example: polynomial regression

$$f(x) = \sin(2\pi x)\,, \quad x \in [0,1]$$

Different noisy instances for training & testing





Training error

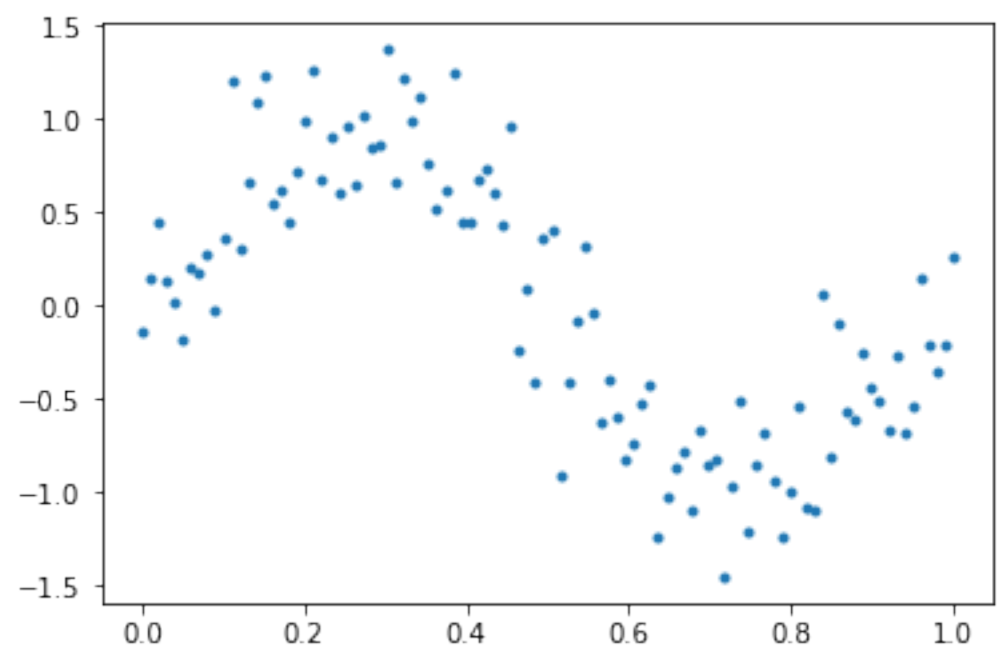
Model: $f_t(x) = \sum_{k=0}^{d} x^k w_k$ for $\hat{w} = \arg\min_w \left\{ \frac{1}{2} \sum_{i=1}^{s} \left| \sum_{k=0}^{d} x_{ij}^k w_k - y_i \right|^2 \right\}$ and $S_t := \{(x_i, y_i) \text{ iid } \sim \mathscr{D}\}_{i=1}^{s}$

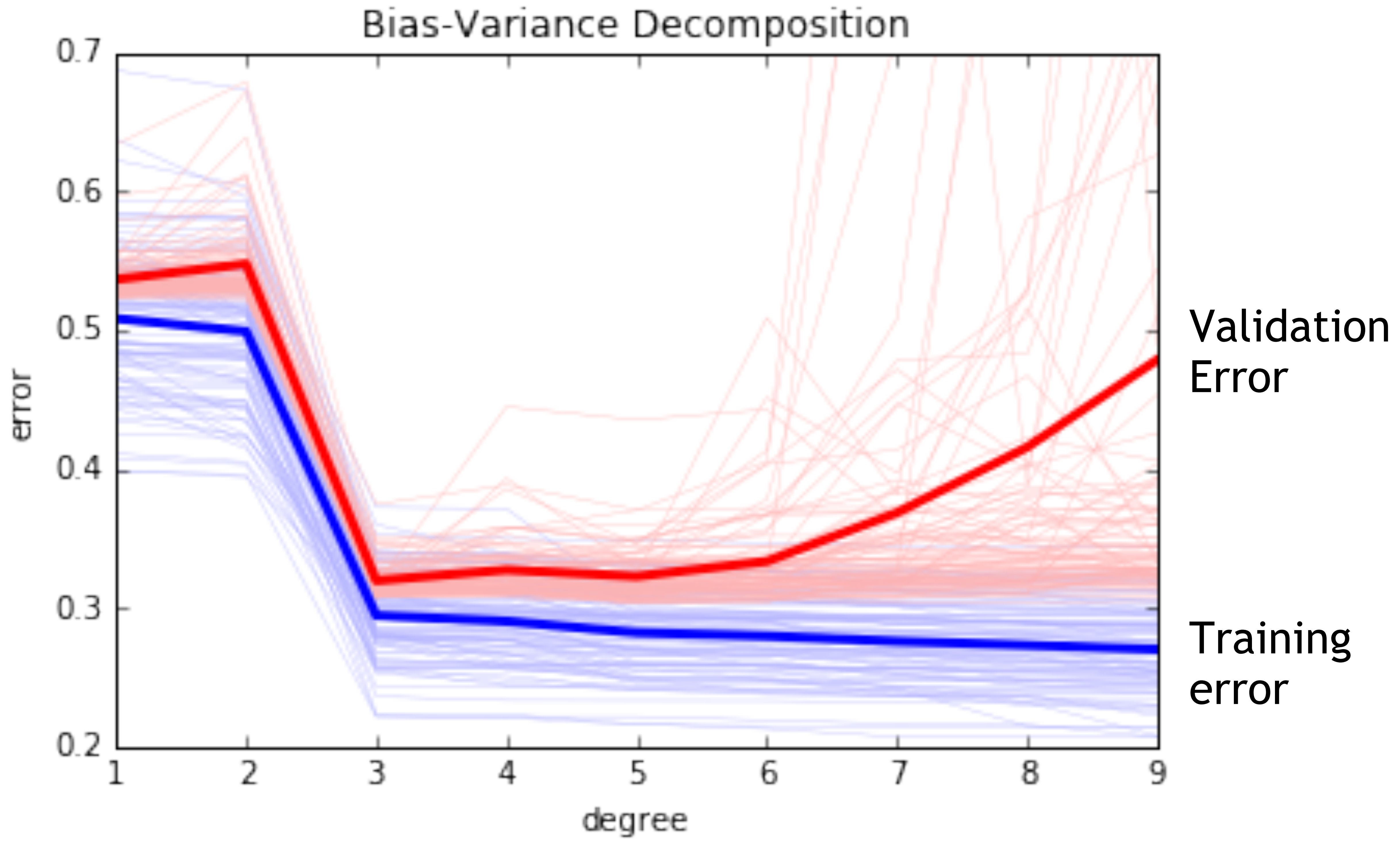
Bias-variance decomposition

Example: polynomial regression

$$f(x) = \sin(2\pi x)\,, \quad x \in [0,1]$$

Different noisy instances for training & testing

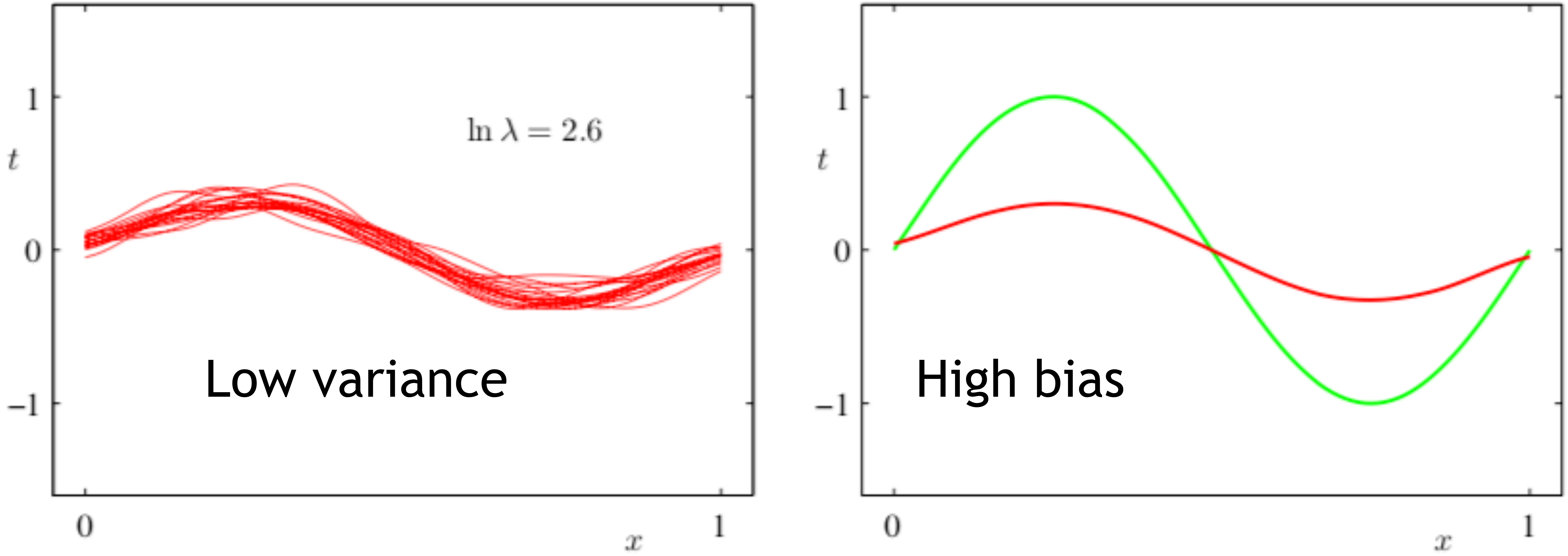




Model: $f_t(x) = \sum_{k=0}^{d} x^k w_k$ for $\hat{w} = \arg\min_w \left\{ \frac{1}{2} \sum_{i=1}^{s} \left| \sum_{k=0}^{d} x_{ij}^k w_k - y_i \right|^2 \right\}$ and $S_t := \{(x_i, y_i) \text{ iid } \sim \mathscr{D}\}_{i=1}^{s}$

Bias-variance decomposition

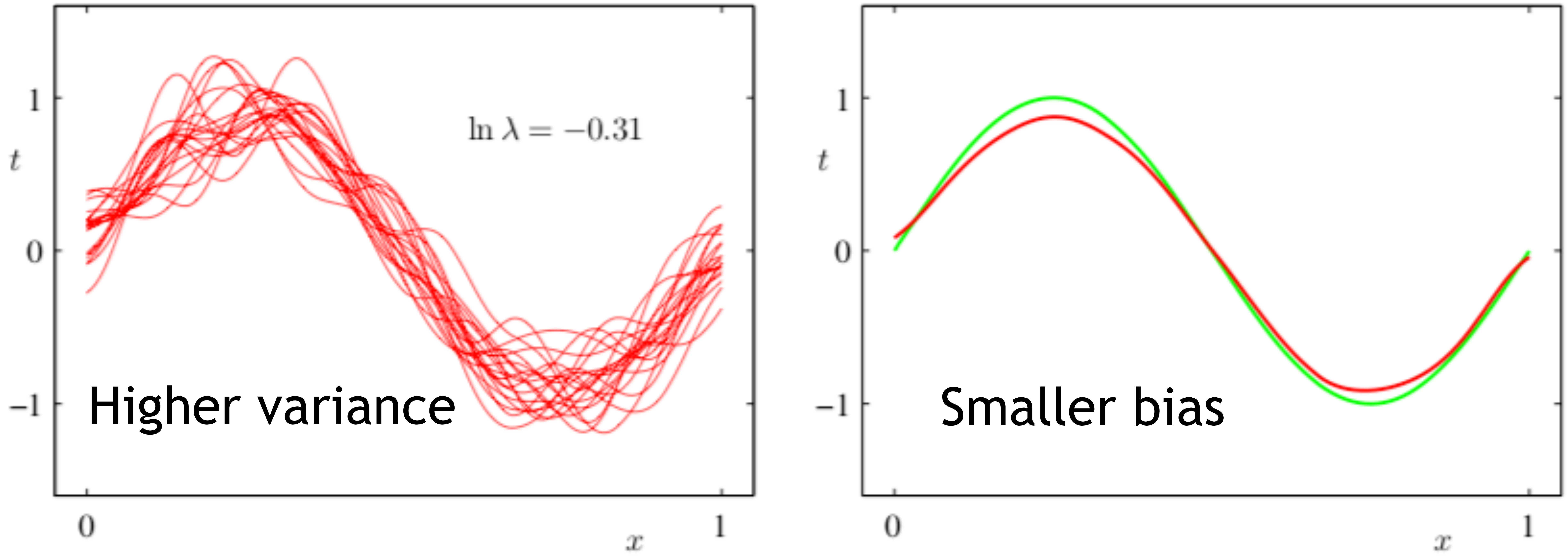
Example: ridge regression $$\hat{\mathbf{w}} = \arg\min_{w} \left\{ \frac{1}{2} \|\boldsymbol{\Phi}(\mathbf{X})\mathbf{w} - \mathbf{y}\|^2 + \frac{\lambda}{2} \|\mathbf{w}\|^2 \right\}$$



From Bishop. Pattern Recognition & Machine Learning

Bias-variance decomposition

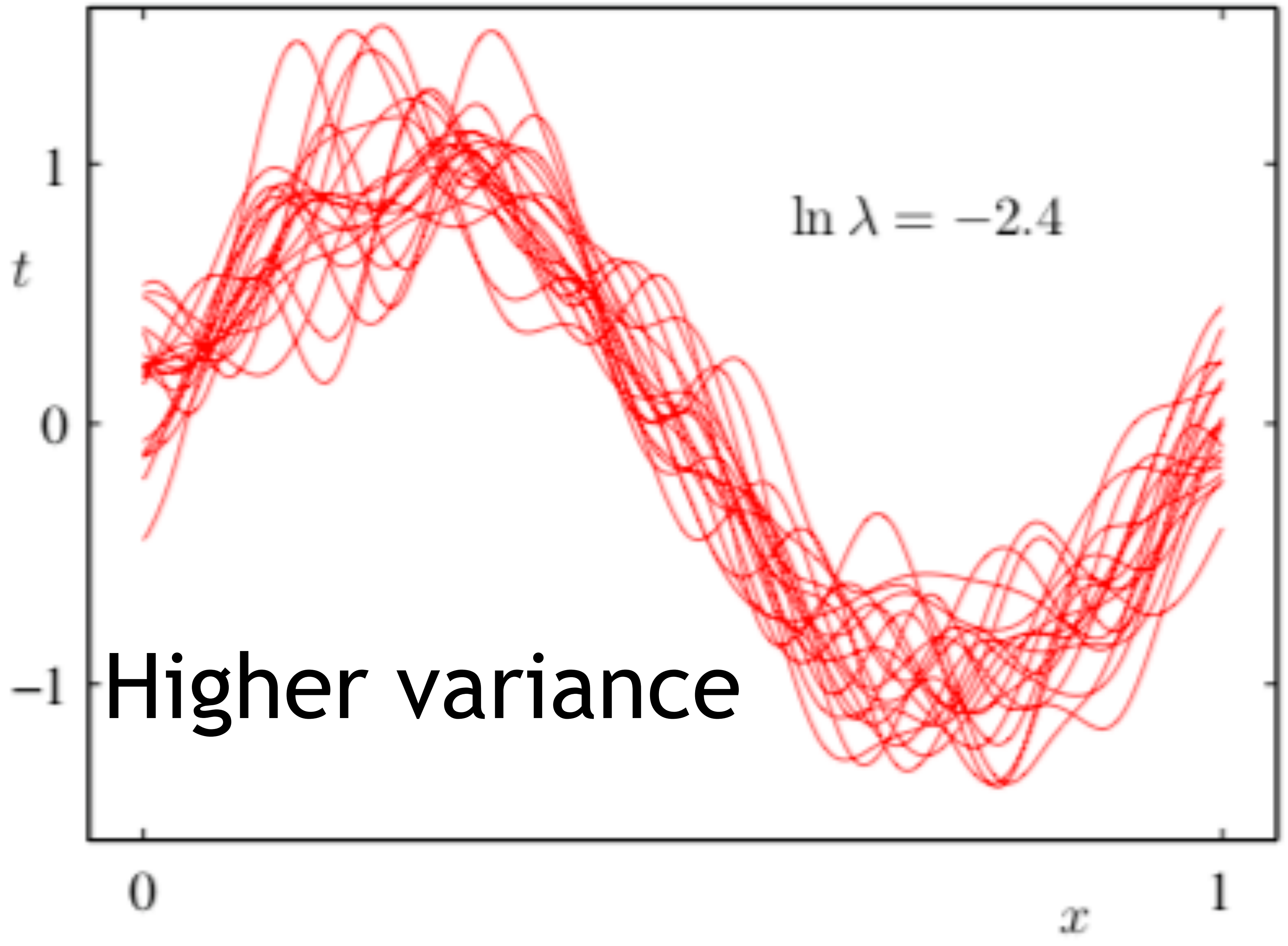
Example: ridge regression $$\mathbf{w} = \arg\min_{\mathbf{w}} \left\{ \frac{1}{2}\|\Phi(\mathbf{X})\mathbf{w} - \mathbf{y}\|^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2 \right\}$$

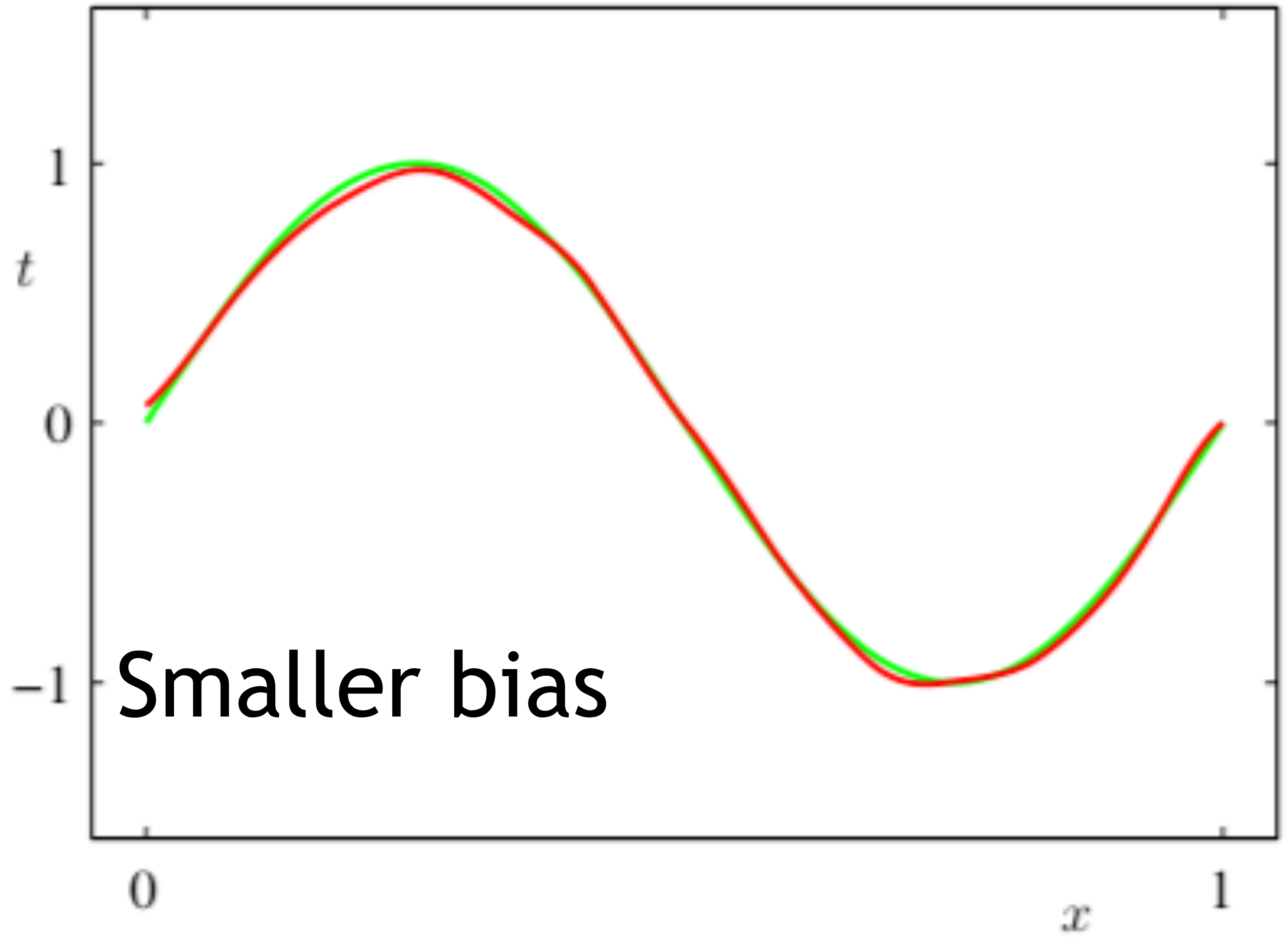


From Bishop. Pattern Recognition & Machine Learning

Bias-variance decomposition

Example: ridge regression $\hat{\mathbf{w}} = \arg\min_{w} \left\{ \frac{1}{2} \|\mathbf{\Phi}(\mathbf{X})\mathbf{w} - \mathbf{y}\|^2 + \frac{\lambda}{2} \|\mathbf{w}\|^2 \right\}$

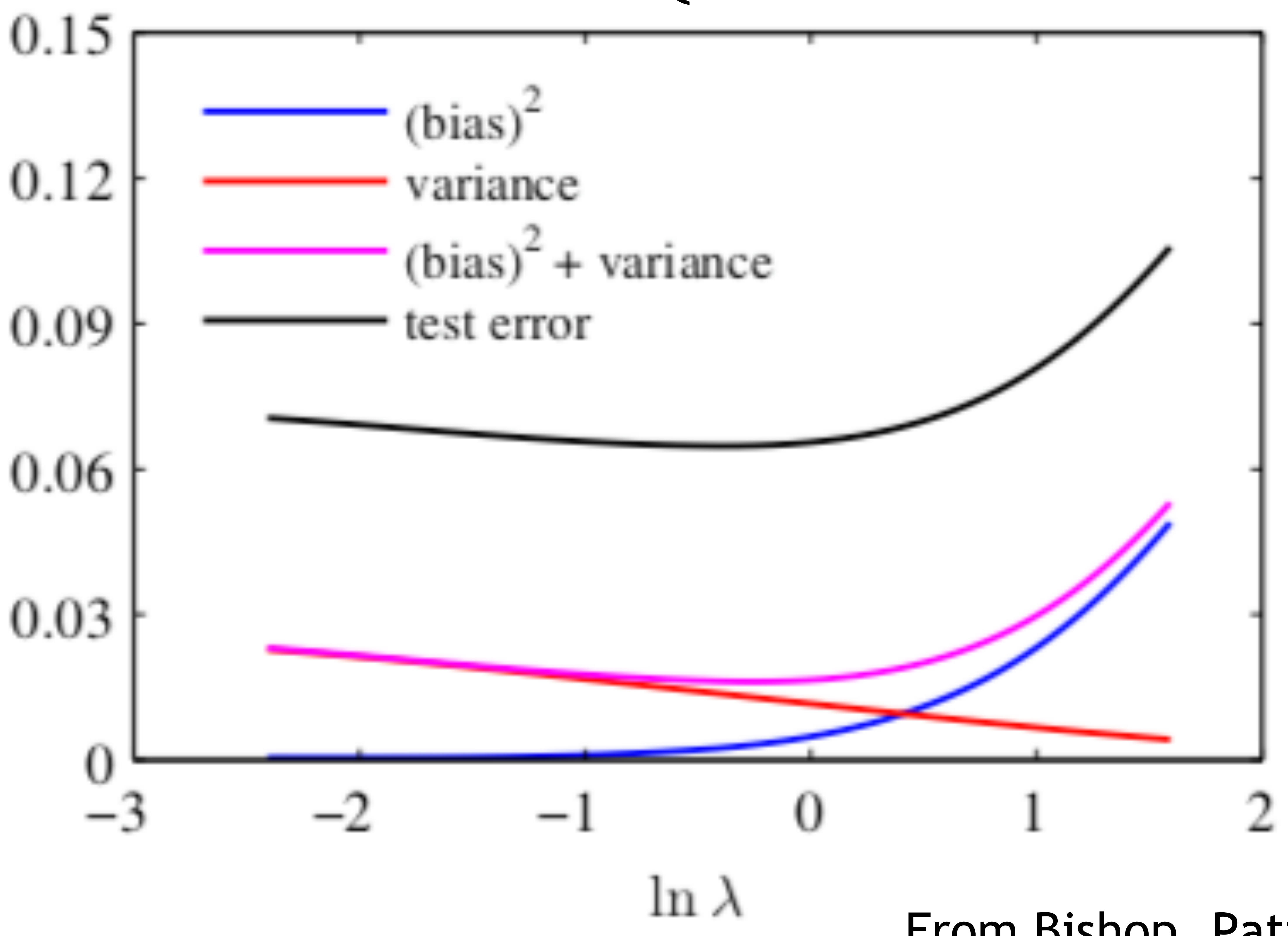




From Bishop. Pattern Recognition & Machine Learning

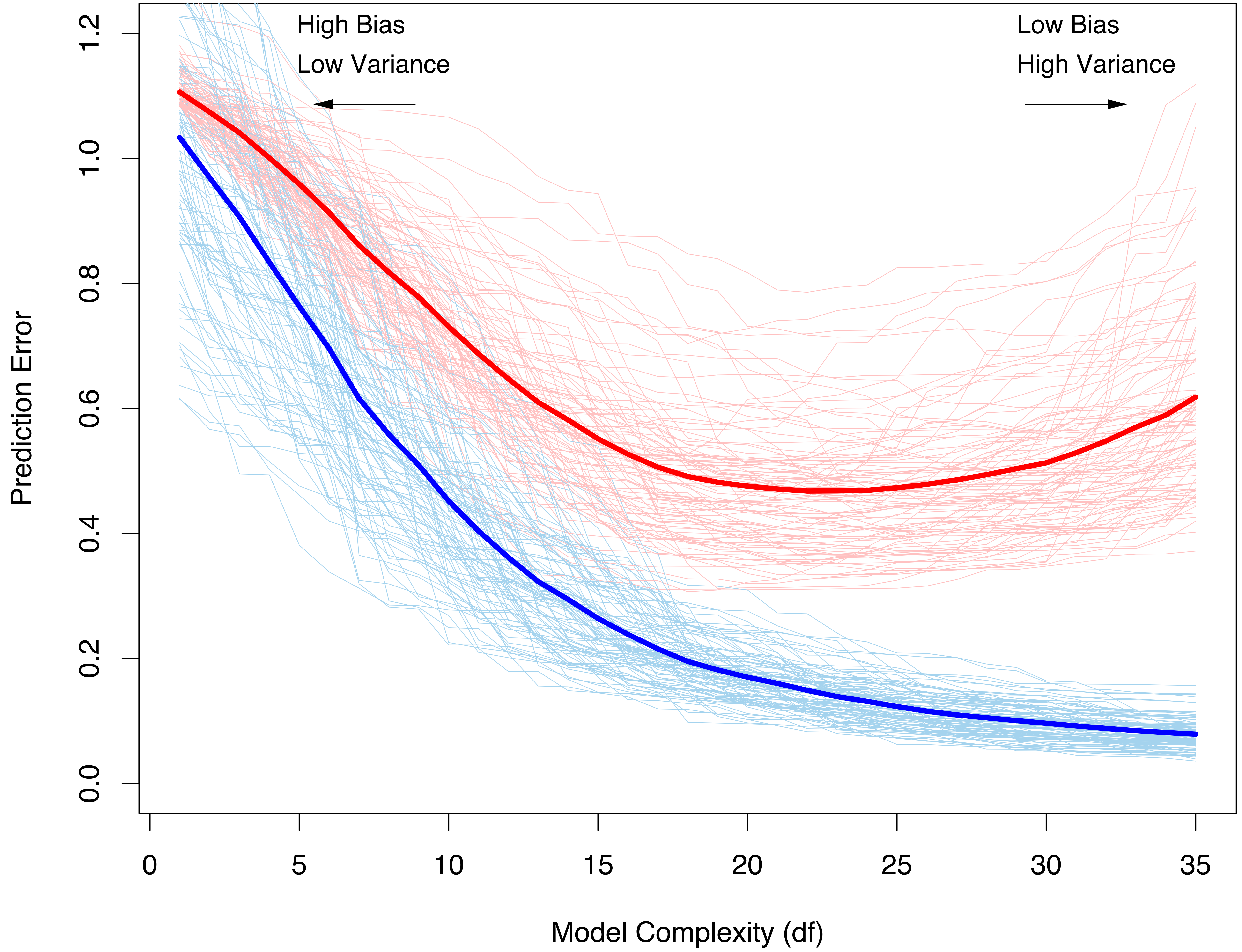
Bias-variance decomposition

Example: ridge regression $\hat{\mathbf{w}} = \arg\min_{w} \left\{ \frac{1}{2} \|\boldsymbol{\Phi}(\mathbf{X})\mathbf{w} - \mathbf{y}\|^2 + \frac{\lambda}{2} \|\mathbf{w}\|^2 \right\}$



From Bishop. Pattern Recognition & Machine Learning

Bias-variance decomposition



Bias-variance decomposition

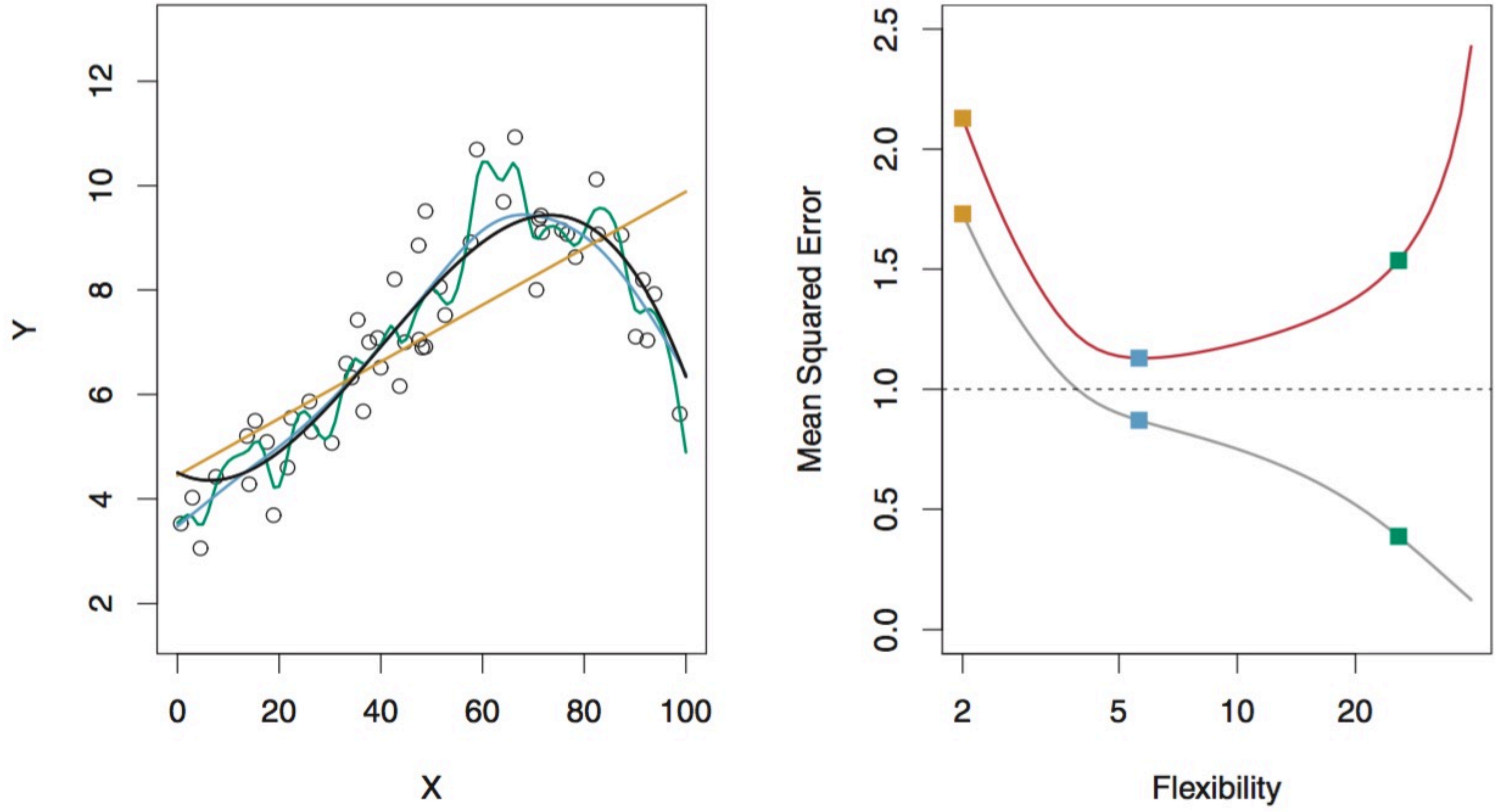
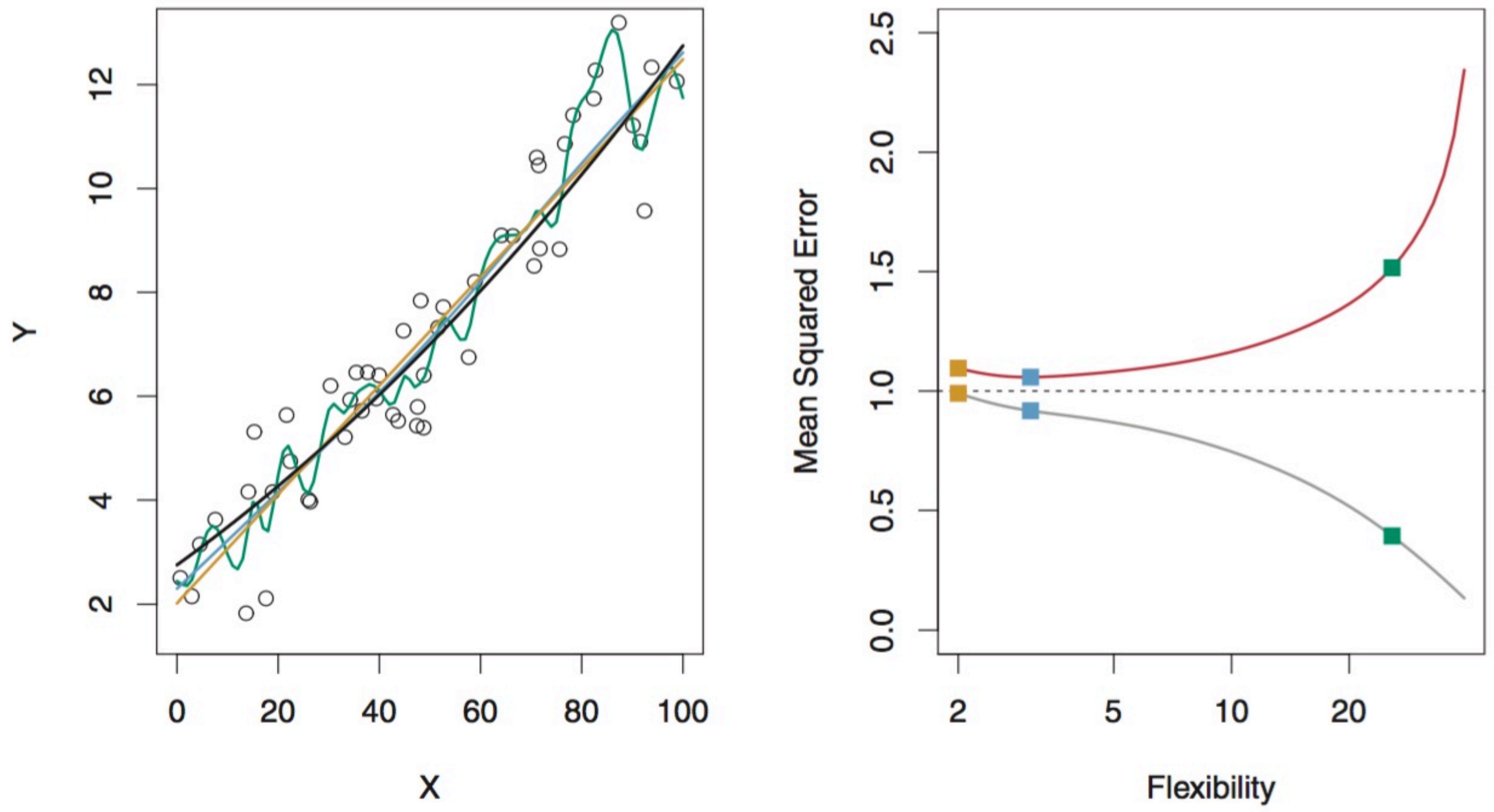


FIGURE 2.9. Left: *Data simulated from f, shown in black. Three estimates of f are shown: the linear regression line (orange curve), and two smoothing spline fits (blue and green curves).* Right: *Training MSE (grey curve), test MSE (red curve), and minimum possible test MSE over all methods (dashed line). Squares represent the training and test MSEs for the three fits shown in the left-hand panel.*

From Hastie, Tibshirani & Friedman, The Elements of Statistical Learning

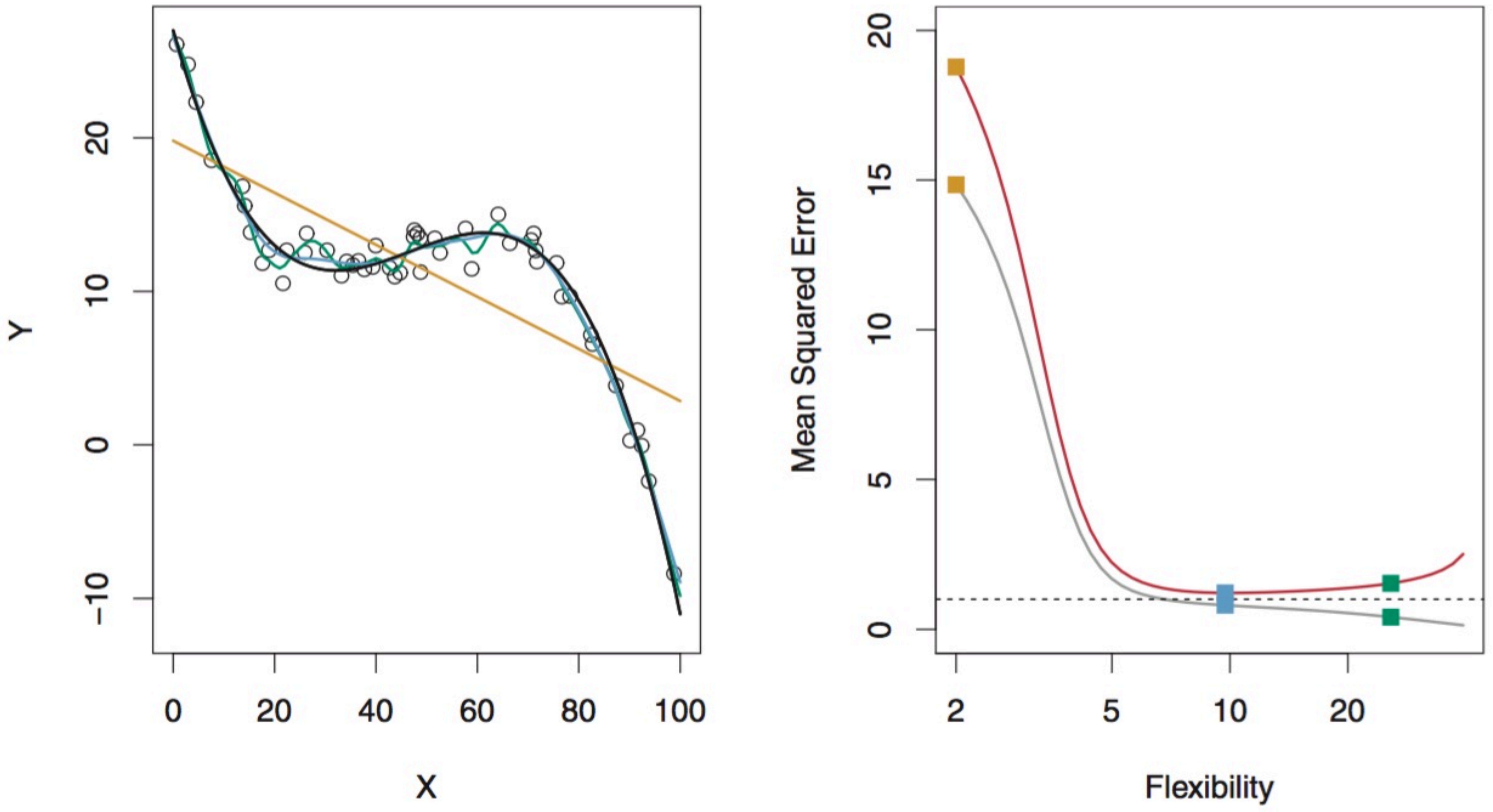
Bias-variance decomposition



From Hastie, Tibshirani & Friedman, The Elements of Statistical Learning

FIGURE 2.10. *Details are as in Figure 2.9, using a different true f that is much closer to linear. In this setting, linear regression provides a very good fit to the data.*

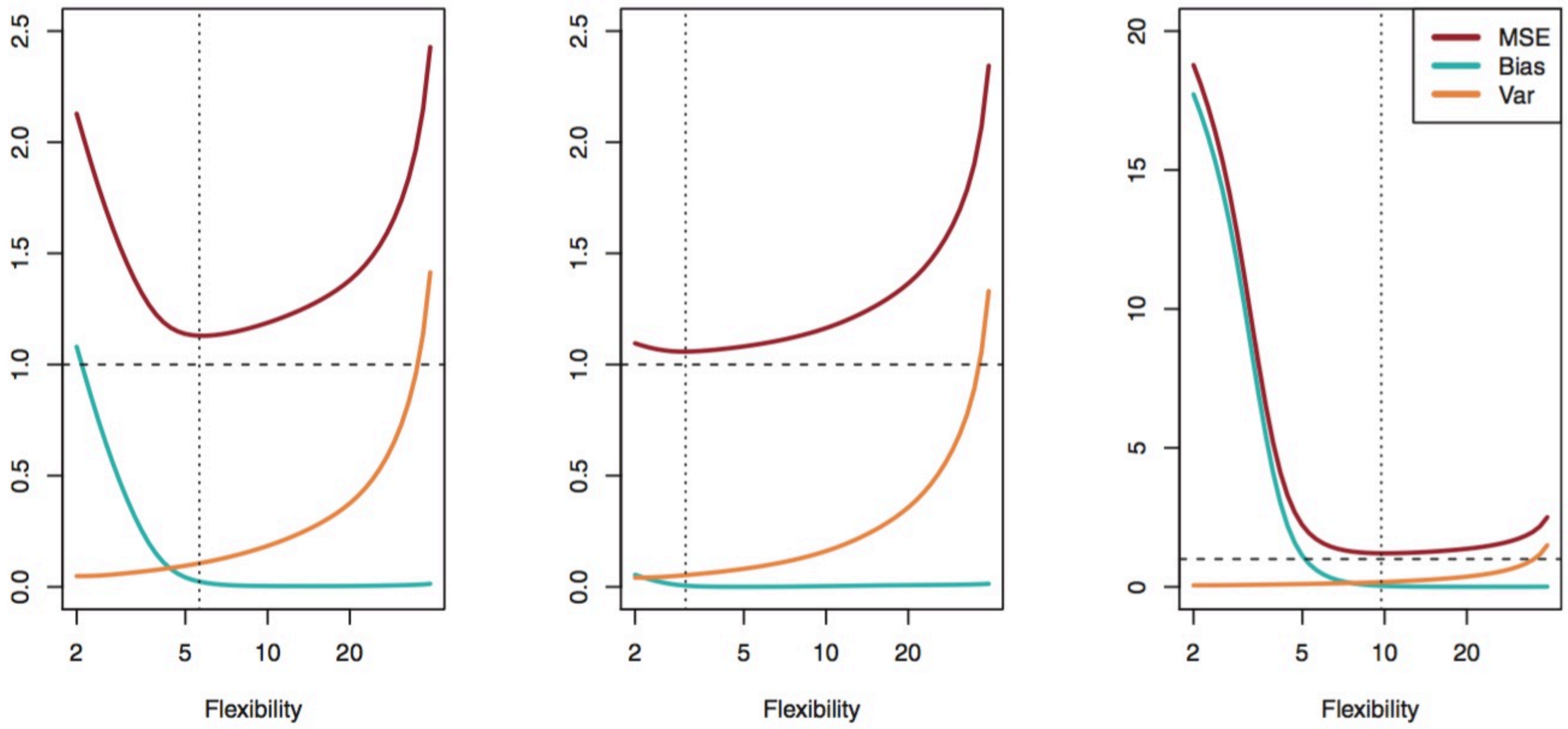
Bias-variance decomposition



From Hastie, Tibshirani & Friedman, The Elements of Statistical Learning

FIGURE 2.11. *Details are as in Figure 2.9, using a different f that is far from linear. In this setting, linear regression provides a very poor fit to the data.*

Bias-variance decomposition



From Hastie, Tibshirani & Friedman, The Elements of Statistical Learning

FIGURE 2.12. *Squared bias (blue curve), variance (orange curve), Var(ϵ) (dashed line), and test MSE (red curve) for the three data sets in Figures 2.9–2.11. The vertical dotted line indicates the flexibility level corresponding to the smallest test MSE.*