Polynomial Regression

Shusen Wang

Warm-up: Linear Regression

Linear Regression (Task)

Input: vectors $\mathbf{x}_1, \cdots, \mathbf{x}_n \in \mathbb{R}^d$ and labels $y_1, \cdots, y_n \in \mathbb{R}$

Output: a vector $\mathbf{w} \in \mathbb{R}^d$ and scalar $b \in \mathbb{R}$ such that $\mathbf{x}_i^T \mathbf{w} + b \approx y_i$.

assume y_i is a linear function of $\mathbf{x}_i$.

Tasks

Linear Regression

Least Squares Regression (Method)

Input: vectors $\mathbf{x}_1, \cdots, \mathbf{x}_n \in \mathbb{R}^d$ and labels $y_1, \cdots, y_n \in \mathbb{R}$

1. Add one dimension to $\mathbf{x} \in \mathbb{R}^d$: $\bar{\mathbf{x}}_j = [\mathbf{x}_j; 1] \in \mathbb{R}^{d+1}$.
2. Solve least squares regression: $\min_{\mathbf{w} \in \mathbb{R}^{d+1}} \left\| \bar{\mathbf{X}}\, \mathbf{w} - \mathbf{y} \right\|_2^2$.

Tasks

Methods

Linear Regression

Least Squares Regression

Least Squares Regression (Method)

Input: vectors $\mathbf{x}_1, \cdots, \mathbf{x}_n \in \mathbb{R}^d$ and labels $y_1, \cdots, y_n \in \mathbb{R}$

1. Add one dimension to $\mathbf{x} \in \mathbb{R}^d$: $\bar{\mathbf{x}}_j = [\mathbf{x}_j; 1] \in \mathbb{R}^{d+1}$.
2. Solve least squares regression: $\min_{\mathbf{w} \in \mathbb{R}^{d+1}} ||\bar{\mathbf{X}}\,\mathbf{w} - \mathbf{y}||_2^2$.

Tasks	Methods	Algorithms
Linear Regression	Least Squares Regression	Analytical Solution
		Gradient Descent
		Conjugate Gradient

Polynomial Regression

The Regression Task

Input: vectors $\mathbf{x}_1, \cdots, \mathbf{x}_n \in \mathbb{R}^d$ and labels $y_1, \cdots, y_n \in \mathbb{R}$.

Output: a function $f: \mathbb{R}^d \mapsto \mathbb{R}$ such that $f(\mathbf{x}) \approx y$.

Question: f is unknown! So how to learn f?

The Regression Task

Input: vectors $\mathbf{x}_1, \cdots, \mathbf{x}_n \in \mathbb{R}^d$ and labels $y_1, \cdots, y_n \in \mathbb{R}$.

Output: a function f: $\mathbb{R}^d \mapsto \mathbb{R}$ such that $f(\mathbf{x}) \approx y$.

Question: f is unknown! So how to learn f?

Answer: polynomial approximation; f is a polynomial function.

Taylor expansion: $f(x) = f(a) + f'(a)(a - x) + \frac{f''(a)}{2!}(a - x)^2 + \cdots$

Polynomial Regression: 1D Example

Input: scalars $x_1, \cdots, x_n \in \mathbb{R}$ and labels $y_1, \cdots, y_n \in \mathbb{R}$.

Output: a function $f : \mathbb{R} \mapsto \mathbb{R}$ such that $f(x) \approx y$.

One-dimensional example: $f(x) = w_0 + w_1 x + w_2 x^2 + \cdots + w_p x^p$.

Polynomial regression:

1. Define a feature map $\boldsymbol{\phi}(x) = [1,\ x,\ x^2,\ x^3, \cdots, x^p]$.
2. For $j = 1$ to n, do the mapping $x_j \mapsto \boldsymbol{\phi}(x_j)$.
   - Let $\boldsymbol{\Phi} = [\boldsymbol{\phi}(x_1); \cdots, \boldsymbol{\phi}(x_n)]^T \in \mathbb{R}^{n \times (p+1)}$
3. Solve the least squares regression $\min_{\mathbf{w} \in \mathbb{R}^{p+1}} \ ||\boldsymbol{\Phi}\, \mathbf{w} - \mathbf{y}||_2^2$.

Polynomial Regression: 2D Example

Input: vectors $\mathbf{x}_1, \cdots, \mathbf{x}_n \in \mathbb{R}^2$ and labels $y_1, \cdots, y_n \in \mathbb{R}$.

Output: a function $f: \mathbb{R}^2 \mapsto \mathbb{R}$ such that $f(\mathbf{x}_i) \approx y_i$.

Two-dimensional example: how to do feature mapping?

Polynomial features:

$$\boldsymbol{\phi}(\mathbf{x}) = [\underbrace{1}_{\text{degree-0}}, \ \underbrace{x_1, \ x_2}_{\text{degree-1}}, \ \underbrace{x_1^2, \ x_2^2, \ x_1 x_2}_{\text{degree-2}}, \ \underbrace{x_1^3, \ x_2^3, \ x_1 x_2^2, \ x_1^2 x_2}_{\text{degree-3}}].$$

Polynomial Regression

```python
import numpy
X = numpy.arange(6).reshape(3, 2)
print('X = ')
print(X)
```

```
X =
[[0 1]
 [2 3]
 [4 5]]
```

```python
from sklearn.preprocessing import PolynomialFeatures
poly = PolynomialFeatures(degree=3)
Phi = poly.fit_transform(X)
print('Phi = ')
print(Phi)
```

```
Phi =
[[   1.    0.    1.    0.    0.    1.    0.    0.    0.    1.]
 [   1.    2.    3.    4.    6.    9.    8.   12.   18.   27.]
 [   1.    4.    5.   16.   20.   25.   64.   80.  100.  125.]]
```

degree-0   degree-1   degree-2   degree-3

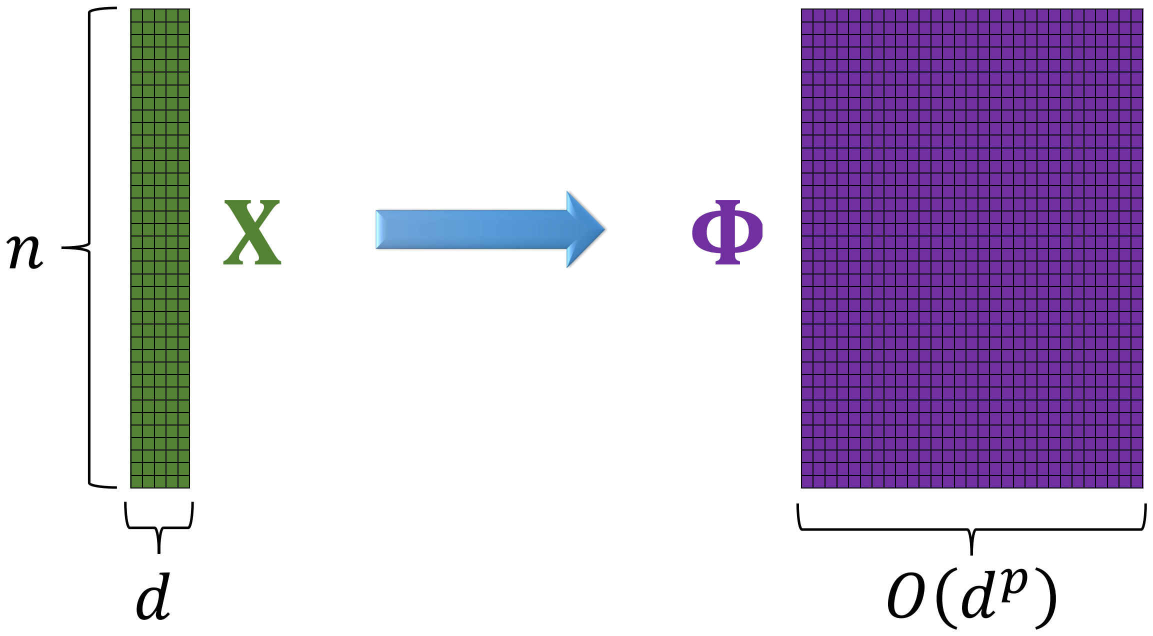
Polynomial Regression

- $\mathbf{x}$: d-dimensional
- $\boldsymbol{\phi}(\mathbf{x})$: degree-$p$ polynomial
- The dimension of $\boldsymbol{\phi}(\mathbf{x})$ is $O(d^p)$

Polynomial Regression

Input: vectors $\mathbf{x}_1, \cdots, \mathbf{x}_n \in \mathbb{R}^d$ and labels $y_1, \cdots, y_n \in \mathbb{R}$.

Output: a function $f: \mathbb{R}^d \mapsto \mathbb{R}$ such that $f(\mathbf{x}_i) \approx y_i$.



Training, Test, and Overfitting

Polynomial Regression: Training

Input: vectors $\mathbf{x}_1, \cdots, \mathbf{x}_n \in \mathbb{R}^d$ and labels $y_1, \cdots, y_n \in \mathbb{R}$.

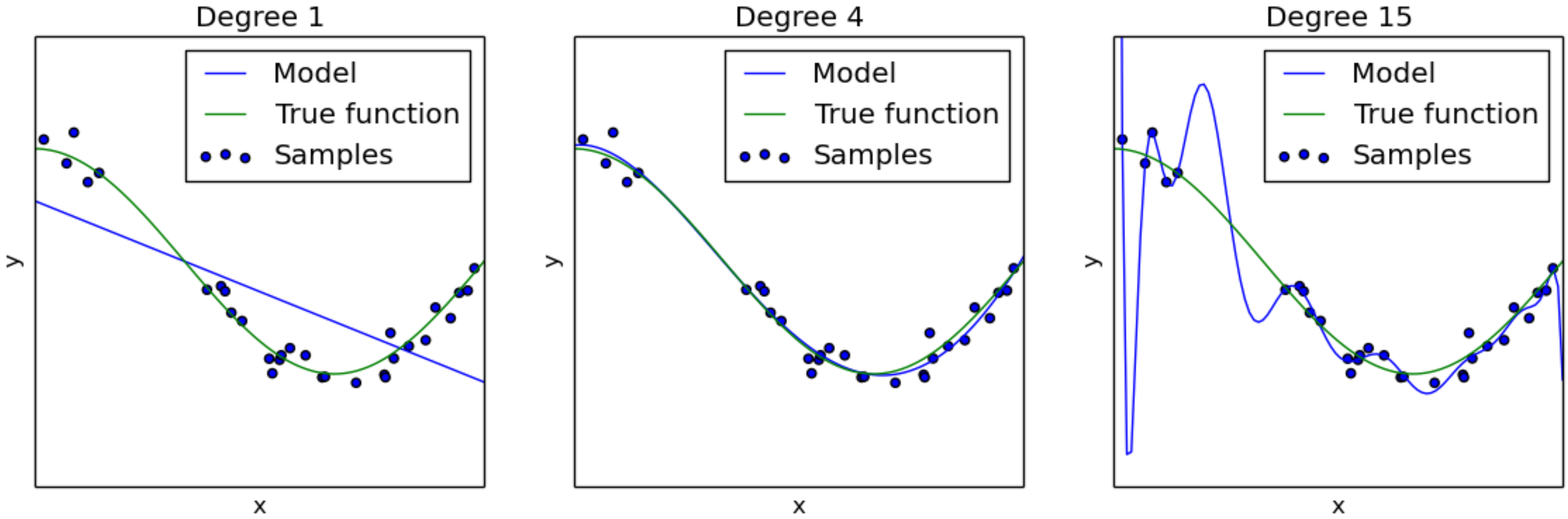
Feature map: $\boldsymbol{\phi}(\mathbf{x}) = \otimes^p \bar{\mathbf{x}}$. Its dimension is $O(d^p)$.

Least squares: $\min_{\mathbf{w}} \ ||\boldsymbol{\Phi}\, \mathbf{w} - \mathbf{y}||_2^2$.

Question: what will happen as p grows?

1. For sufficiently large p, the dimension of the feature $\boldsymbol{\phi}(\mathbf{x})$ exceeds n.
2. Then you can find $\mathbf{w}$ such that $\boldsymbol{\Phi}\, \mathbf{w} = \mathbf{y}$. (Zero training error!)

Training and Testing



Underfitting

Overfitting

Training and Testing

Train:

Input: vectors $\mathbf{x}_1, \cdots, \mathbf{x}_n \in \mathbb{R}^d$ and labels $y_1, \cdots, y_n \in \mathbb{R}$.

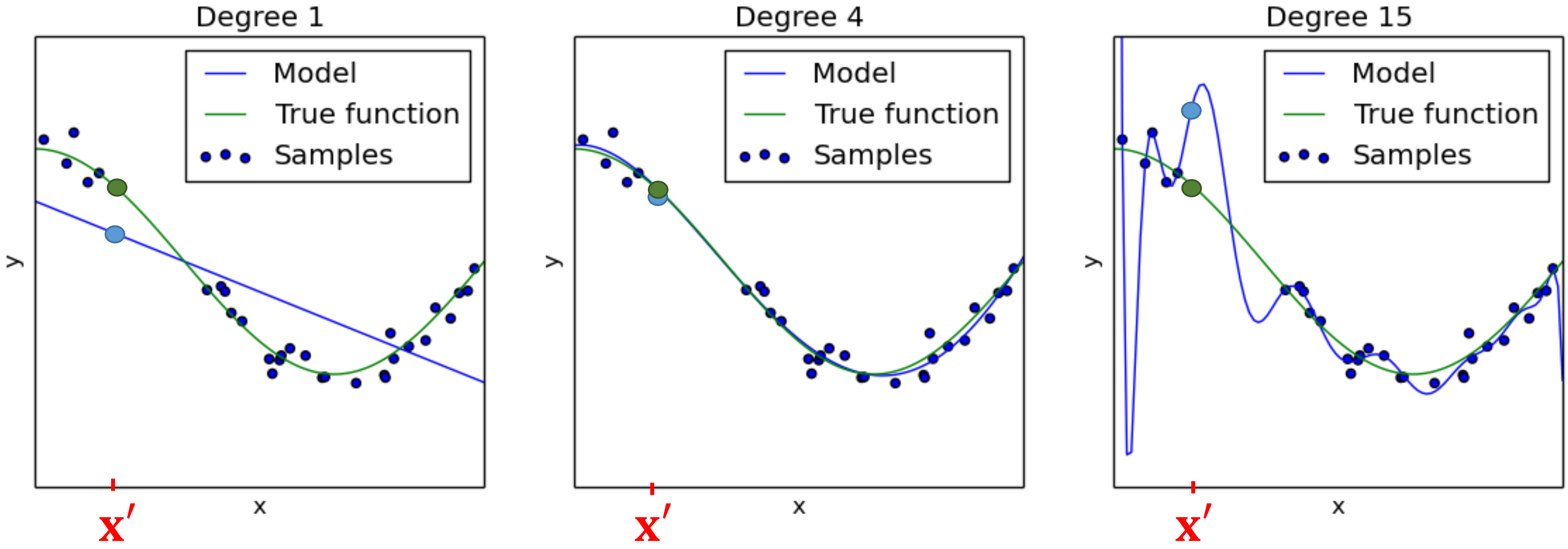
Output: a function $f: \mathbb{R}^d \mapsto \mathbb{R}$ such that $f(\mathbf{x}_i) \approx y_i$.

Test:

Input: a *never-seen-before* feature vectors $\mathbf{x}' \in \mathbb{R}^d$.

Input: predict its label by $f(\mathbf{x}')$.

Training and Testing

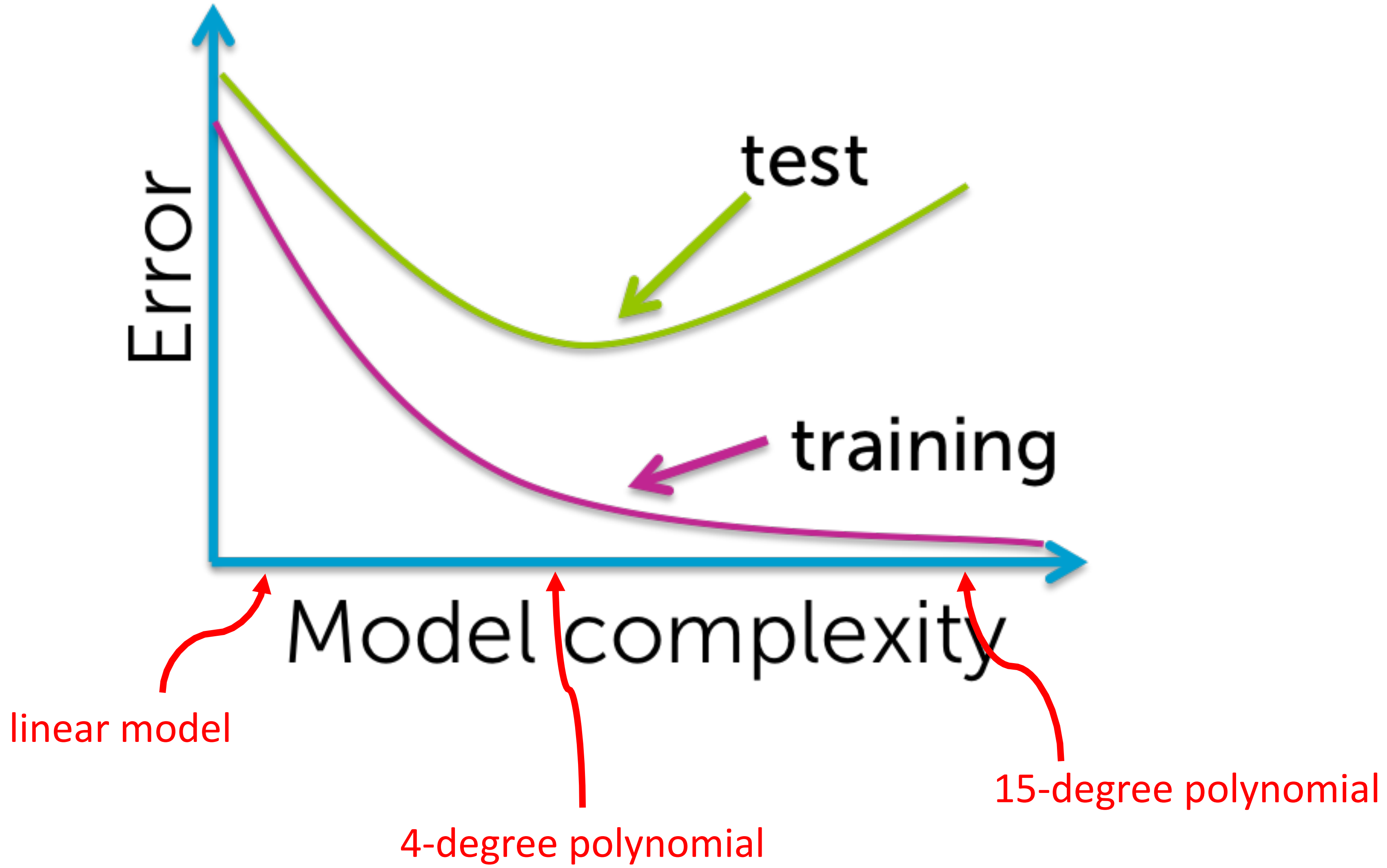


BAD

GOOD

BAD

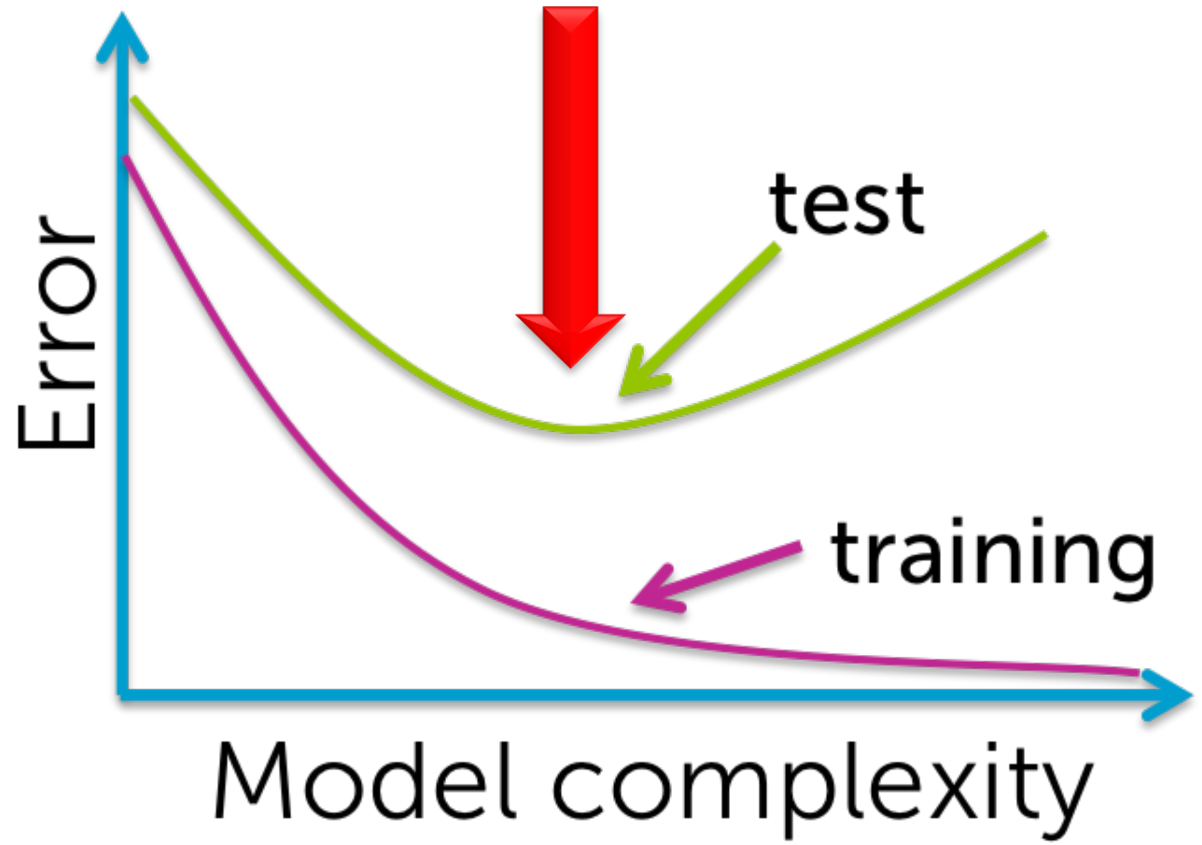
Training and Testing



Hyper-Parameter Tuning

Question: for the polynomial regression model, how to determine the degree p?

Answer: the degree p leads to the smallest test error.



Hyper-Parameter Tuning

Training Set		Test Set
Train a degree-1 polynomial regression	→	Test MSE = 23.2
Train a degree-2 polynomial regression	→	Test MSE = 19.0
Train a degree-3 polynomial regression	→	Test MSE = 16.7
Train a degree-4 polynomial regression	→	Test MSE = 12.2
Train a degree-5 polynomial regression	→	Test MSE = 14.8
Train a degree-6 polynomial regression	→	Test MSE = 25.1
Train a degree-7 polynomial regression	→	Test MSE = 39.4
Train a degree-8 polynomial regression	→	Test MSE = 53.0

Hyper-Parameter Tuning

Training Set		Test Set
Train a degree-1 polynomial regression	→	Test MSE = 23.2
Train a degree-2 polynomial regression	→	Test MSE = 19.0
Train a degree-3 polynomial regression	→	Test MSE = 16.7
Train a $\boxed{\text{degree-4 polynomial}}$ regression	→	$\boxed{\text{Test MSE = 12.2}}$
Train a degree-5 polynomial regression	→	Test MSE = 14.8
Train a degree-6 polynomial regression	→	Test MSE = 25.1
Train a degree-7 polynomial regression	→	Test MSE = 39.4
Train a degree-8 polynomial regression	→	Test MSE = 53.0

Hyper-Parameter Tuning

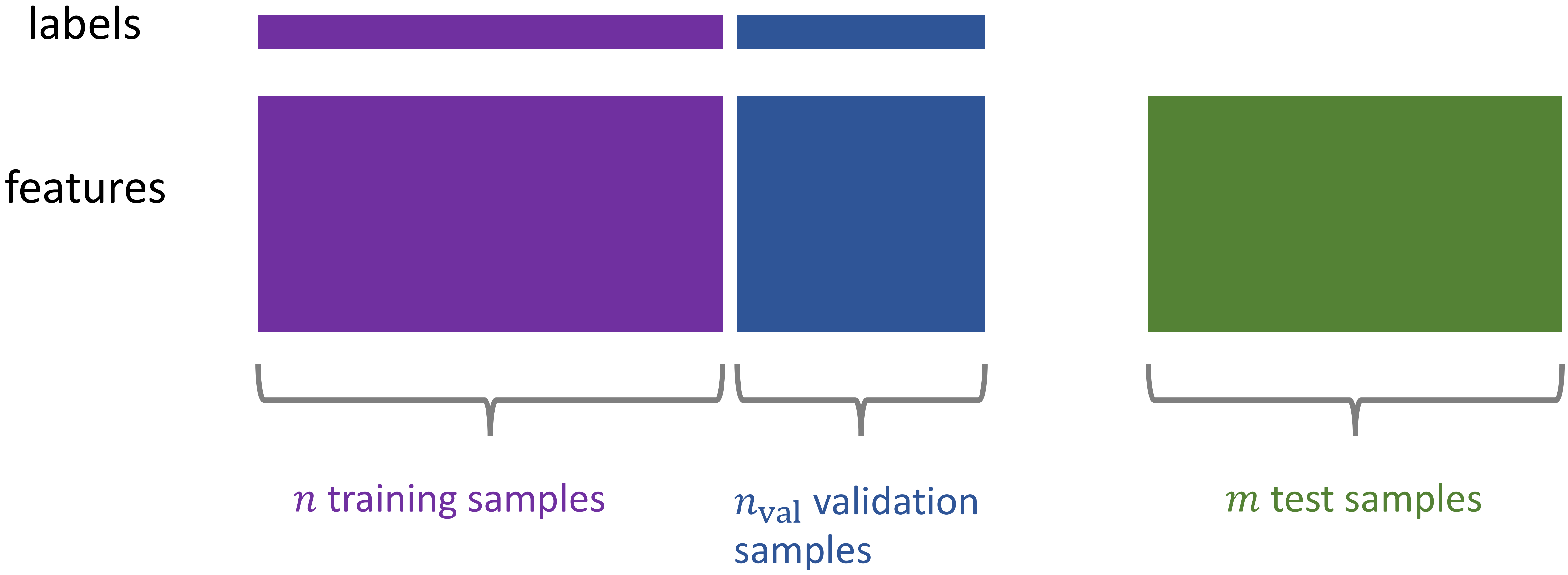
Training Set		Test Set
Train a degree-1 polynomial regression	⟶	Test MSE = 23.2
Train a degree-2 polynomial regression	⟶	Test MSE = 19.0
Train a degree-3 polynomial regression	⟶	Test MSE = 16.7
Train a degree-4 polynomial regression	⟶	Test MSE = 12.2
Train a degree-5 polynomial regression	⟶	Test MSE = 14.8
Train a degree-6 polynomial regression	⟶	Test MSE = 25.1
Train a degree-7 polynomial regression	⟶	Test MSE = 39.4
Train a degree-8 polynomial regression	⟶	Test MSE = 53.0

- **Wrong! The test labels are unavailable!**
- **Even if you have the test labels, never do this!**

Cross-Validation (Naïve Approach) for Hyper-Parameter Tuning

Cross-Validation (Naïve Approach)

Cross-Validation (Naïve Approach)



Cross-Validation (Naïve Approach)

Training Set	Test Set
Train a degree-1 polynomial regression	Test MSE = 23.2
Train a degree-2 polynomial regression	Test MSE = 19.0
Train a degree-3 polynomial regression	Test MSE = 16.7
Train a degree-4 polynomial regression	Test MSE = 12.2
Train a degree-5 polynomial regression	Test MSE = 14.8
Train a degree-6 polynomial regression	Test MSE = 25.1
Train a degree-7 polynomial regression	Test MSE = 39.4
Train a degree-8 polynomial regression	Test MSE = 53.0

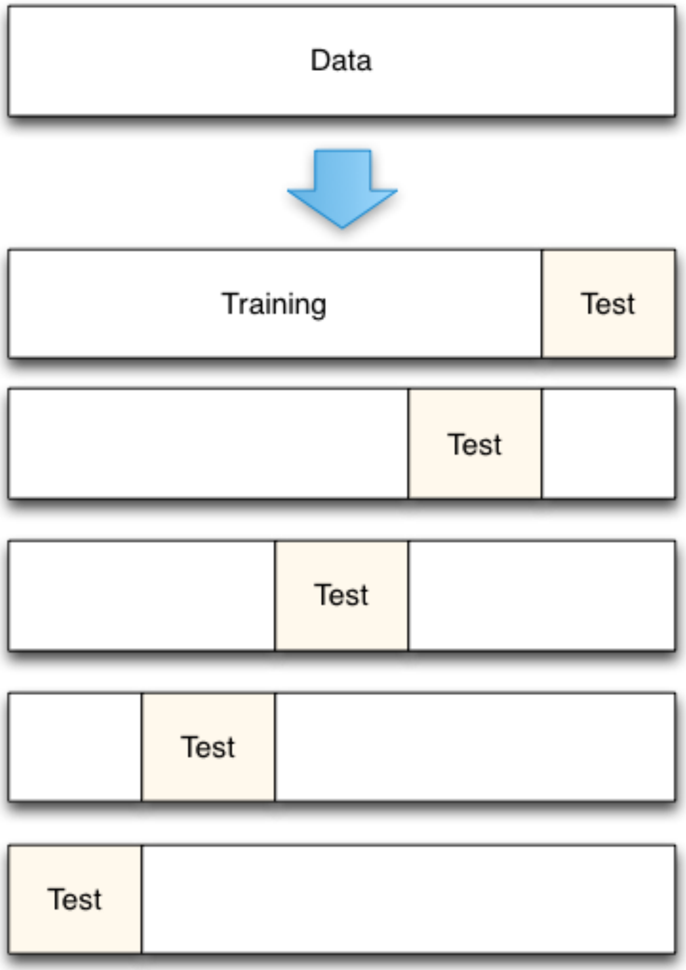
Cross-Validation (Naïve Approach)

Training Set	Validation Set ~~Test Set~~
Train a degree-1 polynomial regression	Valid. MSE = 23.1
Train a degree-2 polynomial regression	Valid. MSE = 19.2
Train a degree-3 polynomial regression	Valid. MSE = 16.3
Train a degree-4 polynomial regression	$\boxed{\text{Valid. MSE = 12.5}}$
Train a degree-5 polynomial regression	Valid. MSE = 14.4
Train a degree-6 polynomial regression	Valid. MSE = 25.0
Train a degree-7 polynomial regression	Valid. MSE = 39.1
Train a degree-8 polynomial regression	Valid. MSE = 53.5

k-Fold Cross-Validation

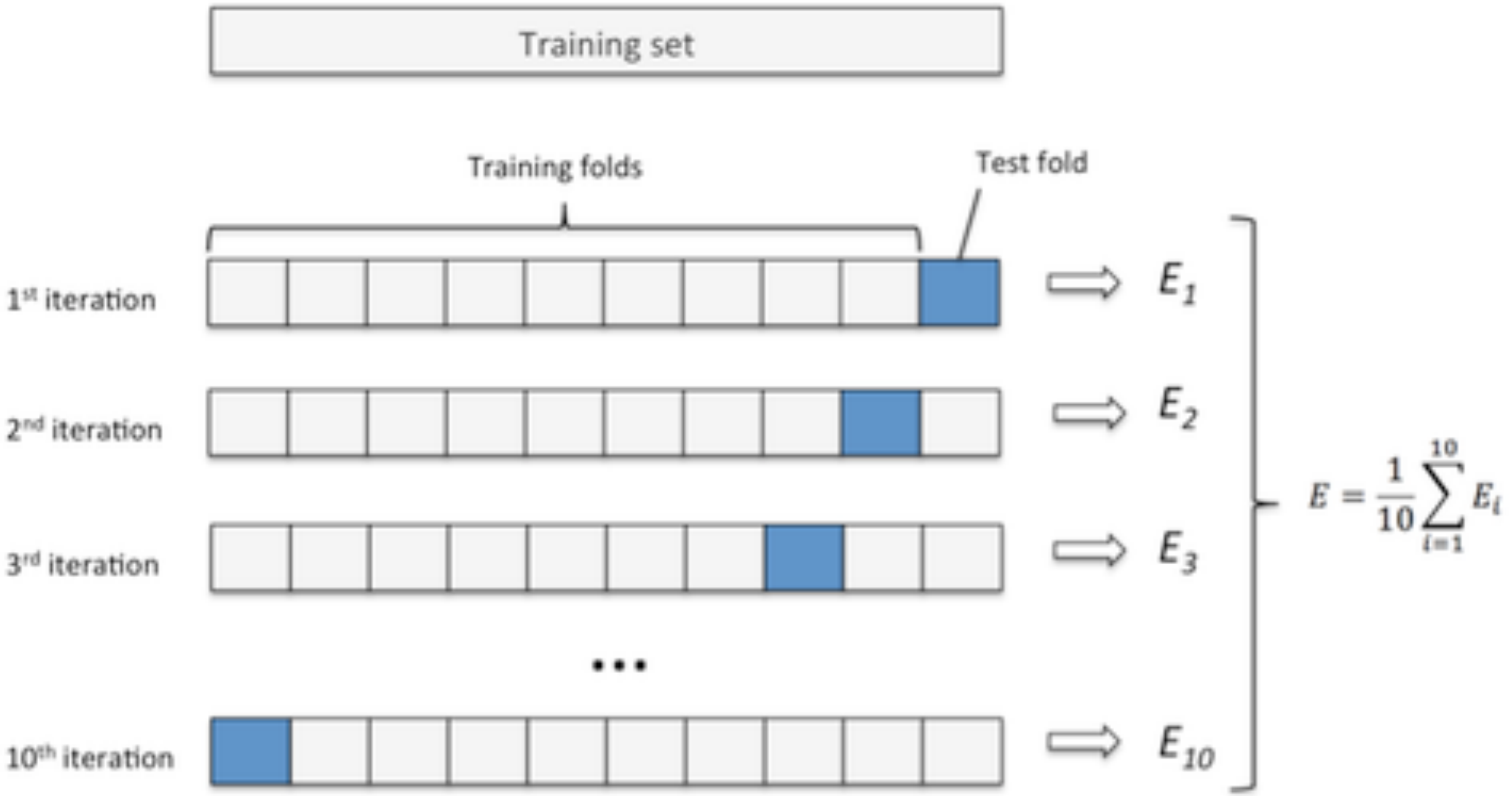
k-Fold Cross-Validation

1. Propose a grid of hyper-parameters.
   - E.g. $p \in \{1, 2, 3, 4, 5\}$.
2. Randomly partition the training samples to k parts.
   - $k - 1$ parts for training.
   - One part for test.
3. Compute the averaged test errors of the k repeats.
   - The average is called the validation error.
4. Choose the hyper-parameter p that leads to the smallest validation error.



Example: 5-fold cross-validation

Example: 10-Fold Cross-Validation



validation error

Example: 10-Fold Cross-Validation

hyper-parameter	validation error
p=1	23.19
p=2	21.00
p=3	18.54
p=4	24.36
p=5	27.96

Real-World Machine Learning Competition

The Available Data

	Training	Public	Private
Labels:	y	unknown	unknown
Features:	X	X_{public}	$X_{private}$

$$\underbrace{X_{public} \quad X_{private}}_{\textbf{Test Data}}$$

Test Data

The public and private are mixed;
Participants cannot distinguish them.

Train A Model

	Training	Public	Private
Labels:	$\mathbf{y}$	unknown	unknown
Features:	$\mathbf{X}$	$\mathbf{X}_{\text{public}}$	$\mathbf{X}_{\text{private}}$

↓

Model

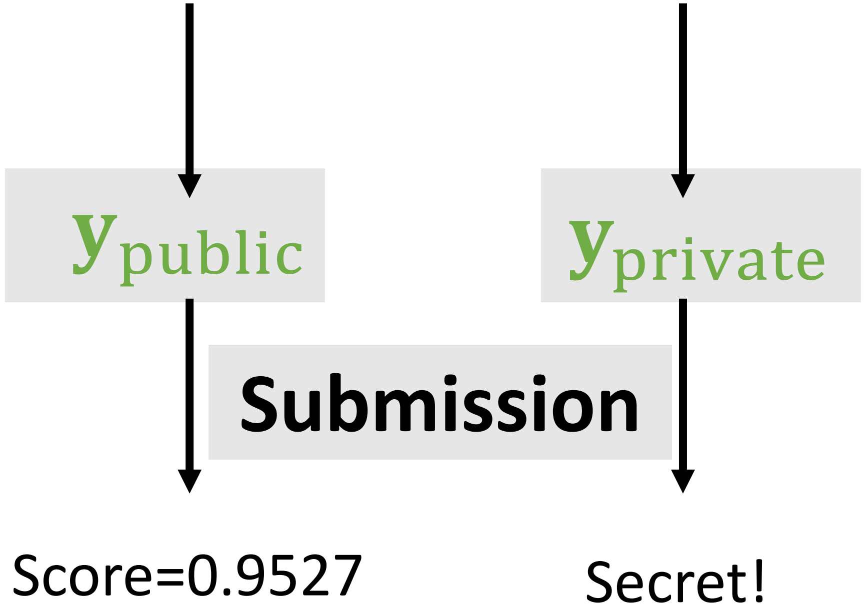
Prediction

	Training	Public	Private
Labels:	y	unknown	unknown
Features:	X	X_{public}	$X_{private}$

$$X_{public} \xrightarrow{\text{Model}} y_{public} \qquad X_{private} \xrightarrow{\text{Model}} y_{private}$$

Submission to Leaderboard

	Training	Public	Private
Labels:	y	unknown	unknown
Features:	X	X_{public}	$X_{private}$



Submission to Leaderboard

	Training	Public	Private
Labels:	$\mathbf{y}$	unknown	unknown
Features:	$\mathbf{X}$	$\mathbf{X}_{\text{public}}$	$\mathbf{X}_{\text{private}}$

$\mathbf{y}_{\text{public}}$ $\mathbf{y}_{\text{private}}$

Submission

Score=0.9527

Secret!

Question: Why two leaderboards?

Answer: The score can be evilly used for hyper-parameter tuning (cheating).

Summary

- Polynomial regression for non-linear problems.
- Polynomial regression has a hyper-parameter p.
- Underfitting (very small p) and overfitting (very big p) .
- Tune the hyper-parameters using cross-validation.
- Make your model parameters and hyper-parameters independent of the test set!!!