Bias-variance Tradeoff

Taehoon Ko (taehoonko@snu.ac.kr)

목표

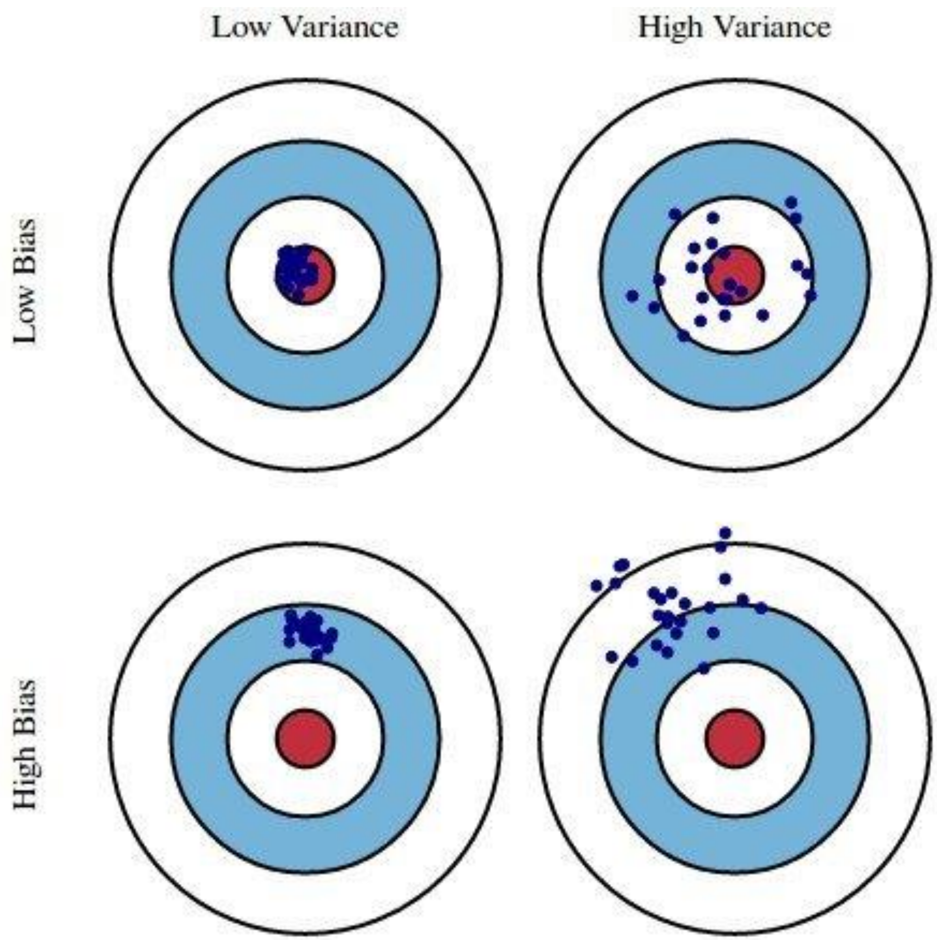
- **다음을 이해한다.**
  - 모델의 편향과 분산
  - 편향과 분산의 트레이드오프 관계
  - 모델의 복잡도, underfitting, overfitting 관계

Bias-variance tradeoff

- Bias (편향성)
  - 지도학습 알고리즘이 학습데이터(training set) 내 입력변수들과 출력변수의 관계를 잘 fitting하지 못해 발생하는 오차
- Variance (변동성)
  - 학습데이터에 내재되어 있는 변동(fluctuation)에 의해 발생하는 오차
  - 학습데이터가 모집단을 완벽하게 대표할 수 없기 때문에 발생

Bias-variance tradeoff

- Graphical illustration of bias and variance



출처: http://scott.fortmann-roe.com/docs/BiasVariance.html

Bias-variance tradeoff

- Mathematical definition
  - Assume that there is a relationship such as $Y = f(X) + \epsilon$
    - Y: output variable we are trying to predict
    - X: input variables related to Y
    - ϵ: error term normally distributed with a mean of zero ($\epsilon \sim N(0, \sigma_\epsilon^2)$)
  - We estimate a model $\hat{f}(X)$ of $f(X)$ using regression algorithms.
  - Expected squared prediction error at a point x is

$$\mathrm{Err}(x) = \mathrm{E}\left[(y - \hat{f}(x))^2 \right]$$

Bias-variance tradeoff

$$\mathrm{Err}(x) = \mathrm{E}\left[(y - \hat{f})^2\right]$$

$$= \mathrm{E}\left[y^2 + \hat{f}^2 - 2y\hat{f}\right]$$

$$= \mathrm{E}[y^2] + \mathrm{E}[\hat{f}^2] - 2\mathrm{E}[y\hat{f}]$$

$$= \mathrm{Var}[y] + \mathrm{E}[y]^2 + \mathrm{Var}[\hat{f}]$$

$$+ \mathrm{E}[\hat{f}]^2 - 2f\mathrm{E}[\hat{f}]$$

$$= \mathrm{Var}[y] + \mathrm{Var}[\hat{f}] + (f - \mathrm{E}[\hat{f}])^2$$

$$= \sigma_\varepsilon{}^2 + \mathrm{Var}[\hat{f}] + \mathrm{Bias}[\hat{f}]^2$$

For an arbitrary random variable X,

$$\mathrm{E}[X^2] = \mathrm{Var}[X] + \mathrm{E}[X]^2$$

$$\mathrm{E}[f] = f \quad (\because f \text{ is deterministic})$$

$$\mathrm{E}[y] = \mathrm{E}[f + \varepsilon] = \mathrm{E}[f] + \mathrm{E}[\varepsilon] = \mathrm{E}[f] = f$$

$$\mathrm{Var}[y] = \mathrm{E}[y^2] - \mathrm{E}[y]^2$$

$$= \mathrm{E}[f^2 + \varepsilon^2 + 2f\varepsilon] - \mathrm{E}[f]^2$$

$$= \mathrm{E}[f]^2 + \mathrm{E}[\varepsilon^2] + 2f\mathrm{E}[\varepsilon] - \mathrm{E}[f]^2$$

$$= \mathrm{E}[\varepsilon^2] - E[\varepsilon]^2 \quad (\because E[\varepsilon] = 0)$$

$$= \mathrm{Var}[\varepsilon]$$

Bias-variance tradeoff

- Expected squared prediction error at a point x is decomposed into bias, variance and irreducible error

$$\mathrm{Err}(x) = \mathrm{Bias}[\hat{f}(x)]^2 + \mathrm{Var}[\hat{f}(x)] + \sigma_\varepsilon^{\;2}$$

$$\text{Expected prediction error} = \text{Bias}^2 + \text{Variance} + \text{Irreducible Error}$$

  - Irreducible error: derived from the noise on the true relationship

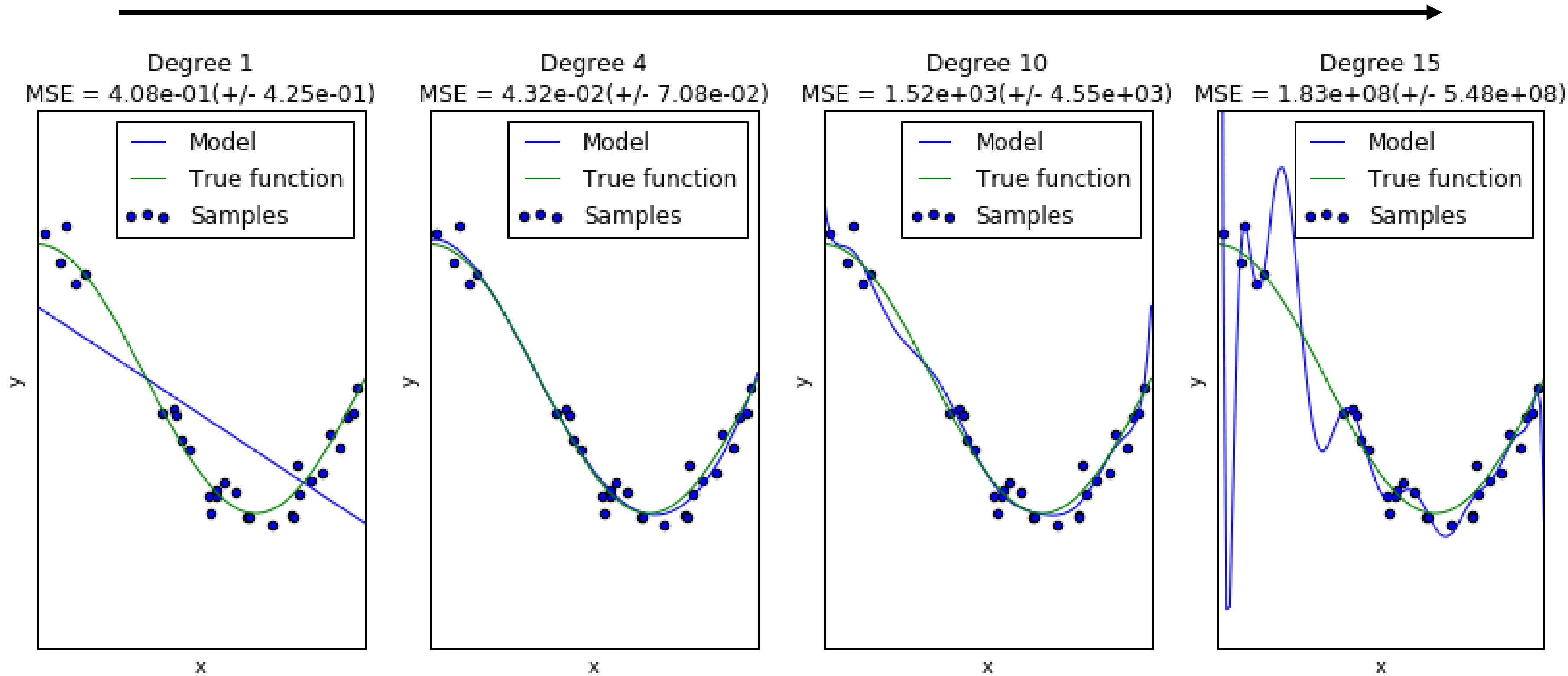
- If we can train perfect model and have infinite data, we should able to reduce both the bias and variance terms to 0.

- It is almost impossible to satisfy above proposition.
  ➔ There is a tradeoff between minimizing the bias and minimizing the variance!

Underfitting vs. Overfitting

- Example: polynomial fitting

Increasing model complexity



Underfitting

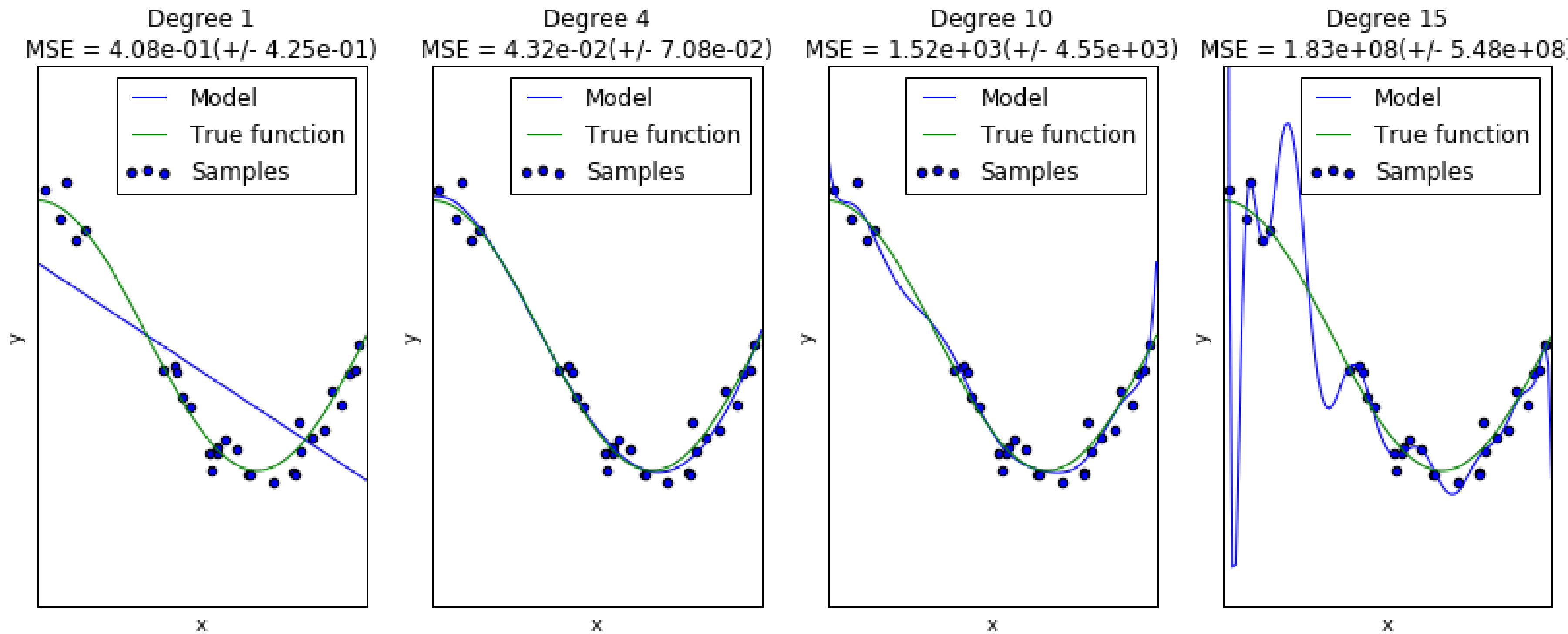
Overfitting

출처: http://scikit-learn.org/stable/auto_examples/model_selection/plot_underfitting_overfitting.html

Underfitting vs. Overfitting

- Example: polynomial fitting

Increasing model complexity



High bias
Low variance

Low bias
High variance

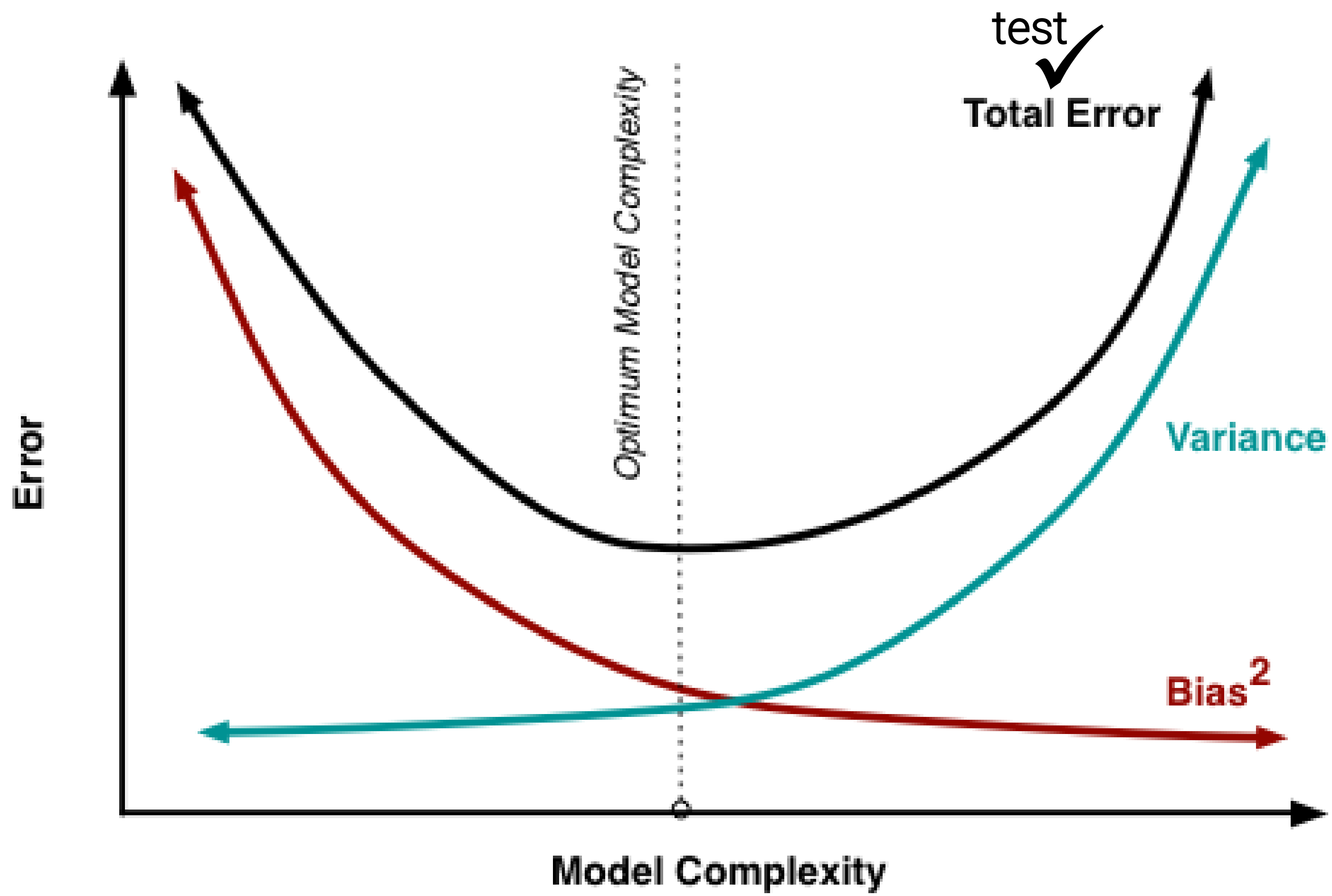
출처: http://scikit-learn.org/stable/auto_examples/model_selection/plot_underfitting_overfitting.html

Underfitting vs. Overfitting

```
print(__doc__)

import numpy as np
import matplotlib.pyplot as plt
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn import cross_validation

np.random.seed(0)

n_samples = 30
degrees = [1, 4, 10, 15]

true_fun = lambda X: np.cos(1.5 * np.pi * X)
X = np.sort(np.random.rand(n_samples))
y = true_fun(X) + np.random.randn(n_samples) * 0.1

plt.figure(figsize=(14, 5))
for i in range(len(degrees)):
    ax = plt.subplot(1, len(degrees), i + 1)
    plt.setp(ax, xticks=(), yticks=())

    polynomial_features = PolynomialFeatures(degree=degrees[i],
                                             include_bias=False)
    linear_regression = LinearRegression()
    pipeline = Pipeline([("polynomial_features", polynomial_features),
                         ("linear_regression", linear_regression)])
    pipeline.fit(X[:, np.newaxis], y)

    # Evaluate the models using crossvalidation
    scores = cross_validation.cross_val_score(pipeline,
        X[:, np.newaxis], y, scoring="mean_squared_error", cv=10)

    X_test = np.linspace(0, 1, 100)
    plt.plot(X_test, pipeline.predict(X_test[:, np.newaxis]), label="Model")
    plt.plot(X_test, true_fun(X_test), label="True function")
    plt.scatter(X, y, label="Samples")
    plt.xlabel("x")
    plt.ylabel("y")
    plt.xlim((0, 1))
    plt.ylim((-2, 2))
    plt.legend(loc="best")
    plt.title("Degree {}₩nMSE = {:.2e}(+/- {:.2e})".format(
        degrees[i], -scores.mean(), scores.std()))
plt.show()
```

Underfitting vs. Overfitting

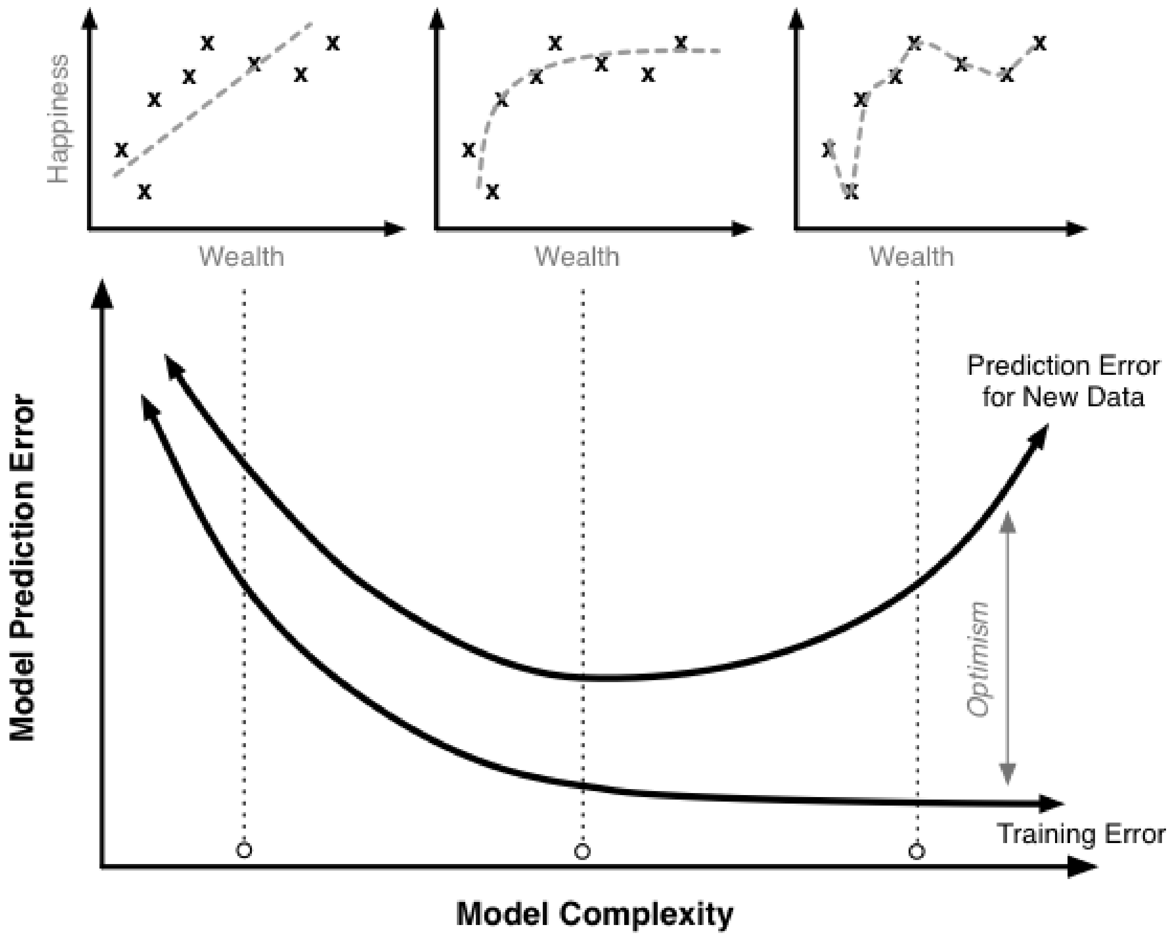
- Bias and variance contributing to total test error



출처: http://scott.fortmann-roe.com/docs/BiasVariance.html

Underfitting vs. Overfitting

- Training optimism and true prediction error (or test error)



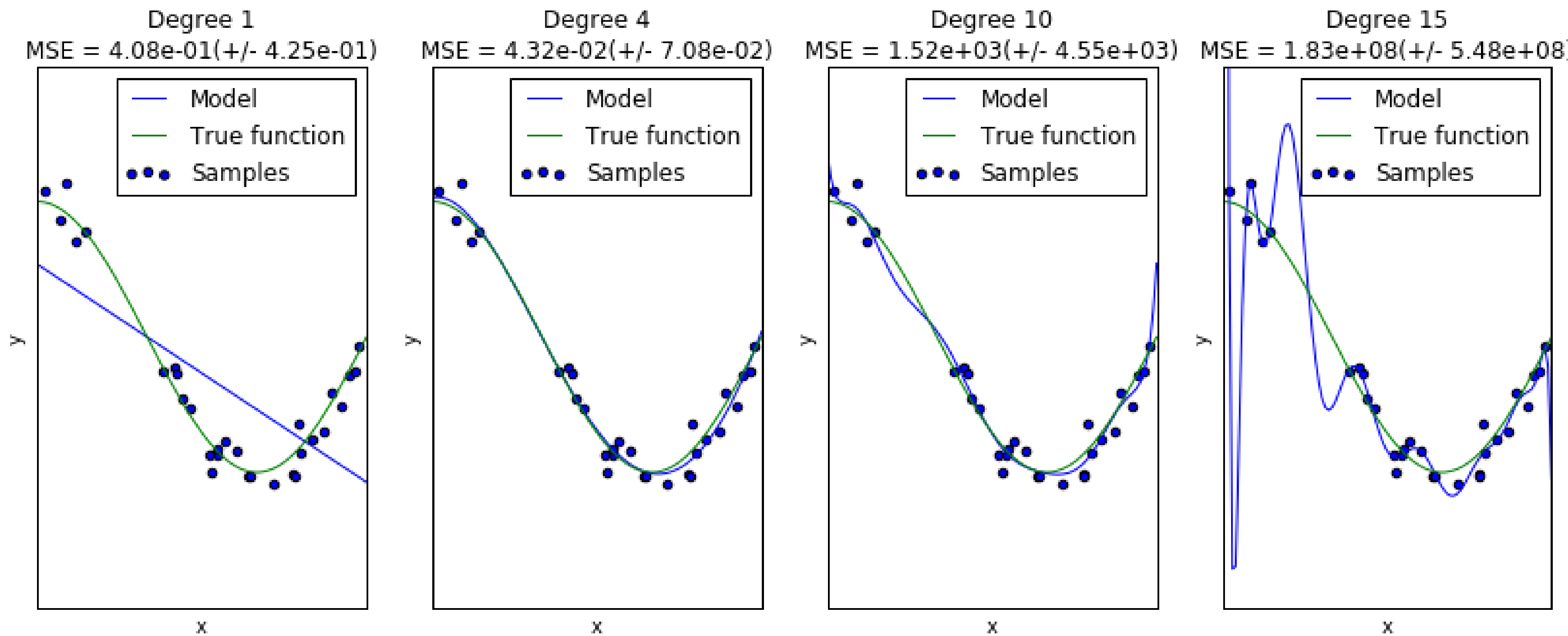
출처: http://scott.fortmann-roe.com/docs/MeasuringError.html

How to solve underfitting and overfitting problem

- Actions to reduce bias (for underfitted models)
  - Add more features.
    - Derived variables, data integration (or mash-up), etc.
  - Use more sophisticated algorithms.
    - Adding complexity to the model
- Actions to reduce variance (for overfitted models)
  - Use fewer features.
    - Using feature selection and extraction methods
  - Use more data points.
  - Use regularization terms.
    - Adding the penalty term for model complexity
  - Average models

By increasing training points,

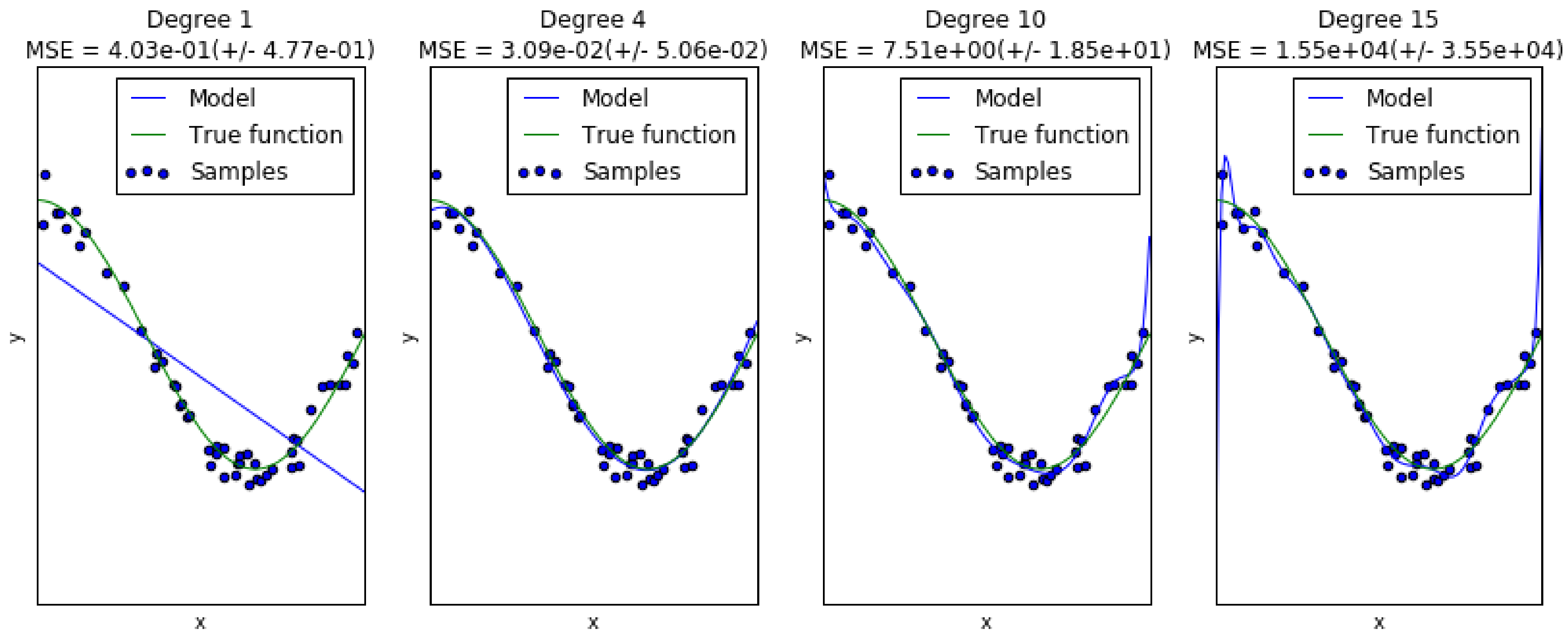
- n = 30



overfitting

By increasing training points,

- n = 50



not overfitting

다음 중 모델 제약에 사용해야 할 데이터셋은?

- Training set
  - 모델 생성, 학습에 이용
- Validation set (Development set)
  - 모델의 오버피팅 방지
  - 모델의 복잡도 축소
  - 모델의 파라미터 탐색
- Test set
  - 모델의 예측 성능 (predictive performance) 평가

Next: Cross-validation (교차검증)

- Cross-validation은 학습 데이터와 검증 데이터를 변화시키면서 모델을 학습하는 방법
- 모델의 'variance'를 측정하기에 효과적