YaleNUSCollege

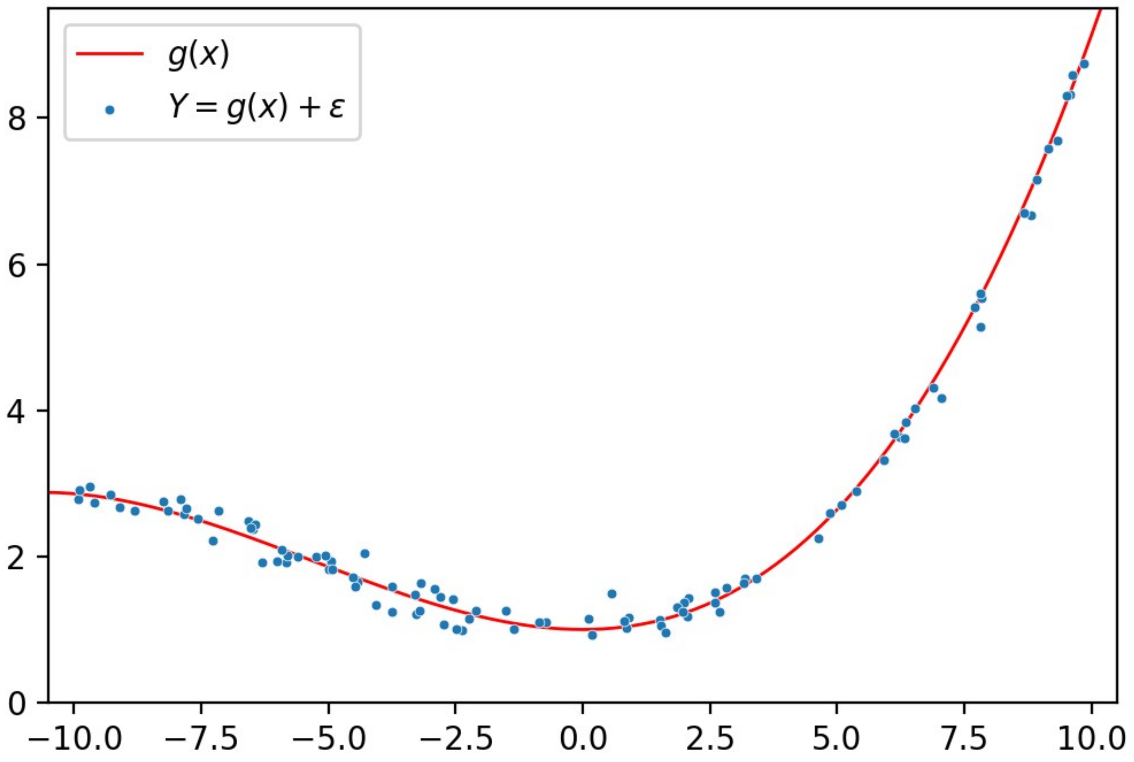
YSC2239 Lecture 17

Today's class

- Variance-bias tradeoff
- Overfitting
- Cross-validation

Data Generation Process

Data Generation Process



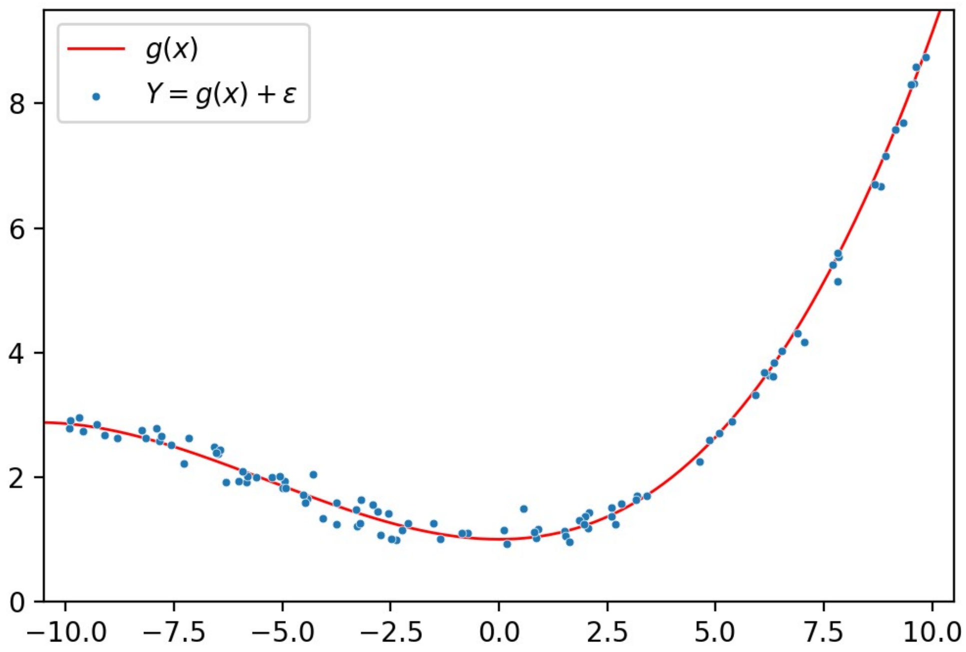
- **Assume** true relation *g*

- For example: $g(x) = \theta_0 + \theta_1 x$
- For each individual:
  - fixed value of *x*, so also *g(x)*
  - random error ε
  - Observation is: $$Y = g(x) + \epsilon$$

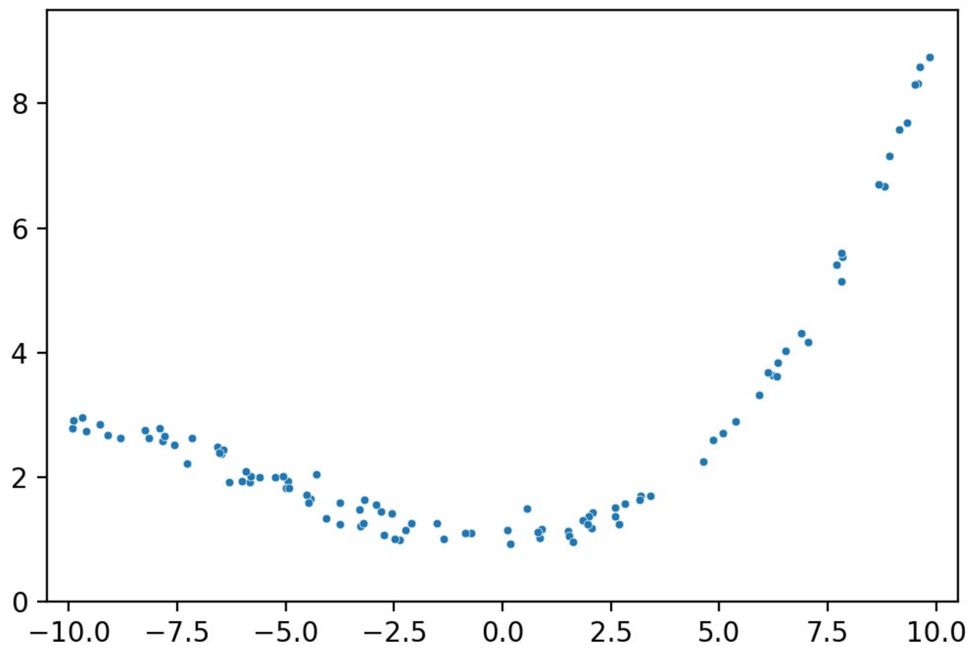
Errors ε have expectation 0, and are “independent and identically distributed” across individuals

The Data

- At each *x*, truth is *g(x)*
- noise is ε
- Observation is $Y = g(x) + \epsilon$

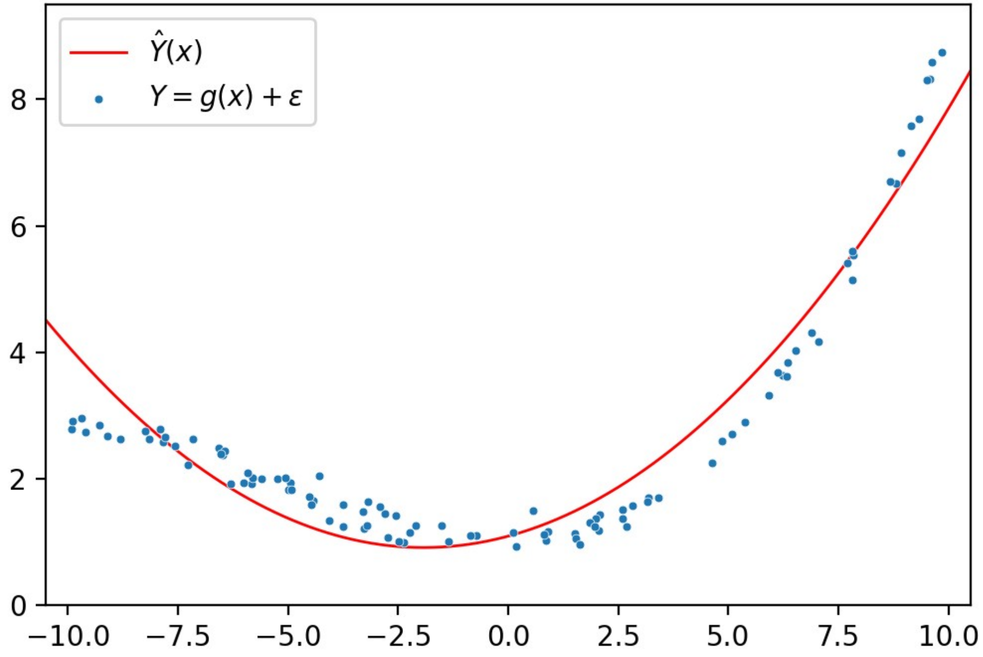


We only see Y



Our Predictions

- We **choose** a model and fit it to our data
  - Choosing a model is codifying our assumption of the form of *g(x)*
- The red line is our fitted function—the best possible function given *g(x)*



At every *x*, our prediction for *Y* is
- the height of the red line at *x*
- Denote this $\hat{Y}(x)$

Bias and Variance in Modeling

A Constant Model

Let's say you want to estimate how often a coin lands on heads when flipped.

- The result of a coin flip follows a Bernoulli(p) distribution, and you want to estimate p.
- You do not collect any data, but instead you are given a choice between two models.
- Suppose you are also told that $p = .5$.

Which of the following is the better model?

Model A: Select a random number between 0 and 1. This is your estimate of p. This is equivalent to running `np.random.random()` in Python.

Model B: Select .75 as your estimate of p.

A Constant Model

How do we define “better”?

We can calculate the expected MSE of each model. This is called the “model risk,” a term which we will formalize later on. The lower the risk, the better.

Model A: *On average*, we will select .5 as our estimate, so we *expect* 0 error. But, as we are only selecting one number, there is a chance we select a number really far away from .5.

Model B: With this model, we will never be exactly correct. But, we know there is zero chance of a really terrible prediction.

The Bias-Variance Tradeoff

When building models, we generally face a tradeoff between **bias** and **variance**.

- Lower bias means that our model will predict closer to the truth, *on average*.
- Lower variance means that our model will not change too much given the sample.

We want low bias *and* low variance, but oftentimes, when one decreases, the other increases.

Model A has zero bias, but lots of variance. **Model B** has zero variance, but lots of bias.

So which is better? The answer will be revealed later in the lecture.

Three Sources of Error in Our Predictions

Irreducible error: Recall the data generating process: $Y = g(x) + \epsilon$

$$\mathbb{V}ar(\epsilon) = \sigma^2 \qquad \mathbb{V}ar(Y) = \mathbb{V}ar(g(x) + \epsilon) = \mathbb{V}ar(\epsilon) = \sigma^2$$

There will be chance error in our predictions due to the natural randomness of the world.

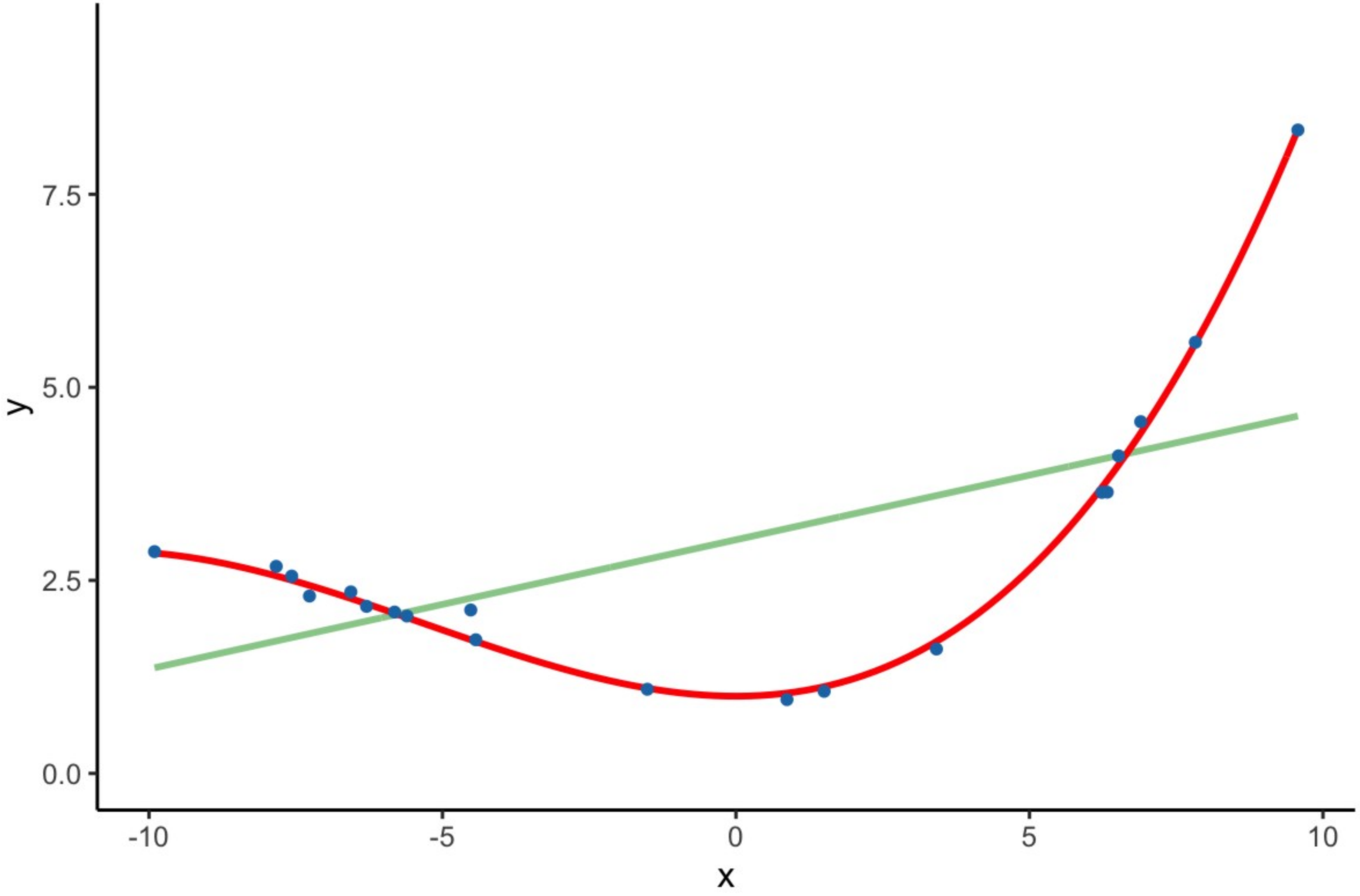
Model variance: Our fitted model is based on a random sample.

The sample could have come out differently, then the fitted model would have been different.

Model bias: This is the difference between the expected predictions, and the true *g(x)*.

Our model may be too limited to find the correct *g(x)*, for example if we pick a quadratic model to fit to cubic data.

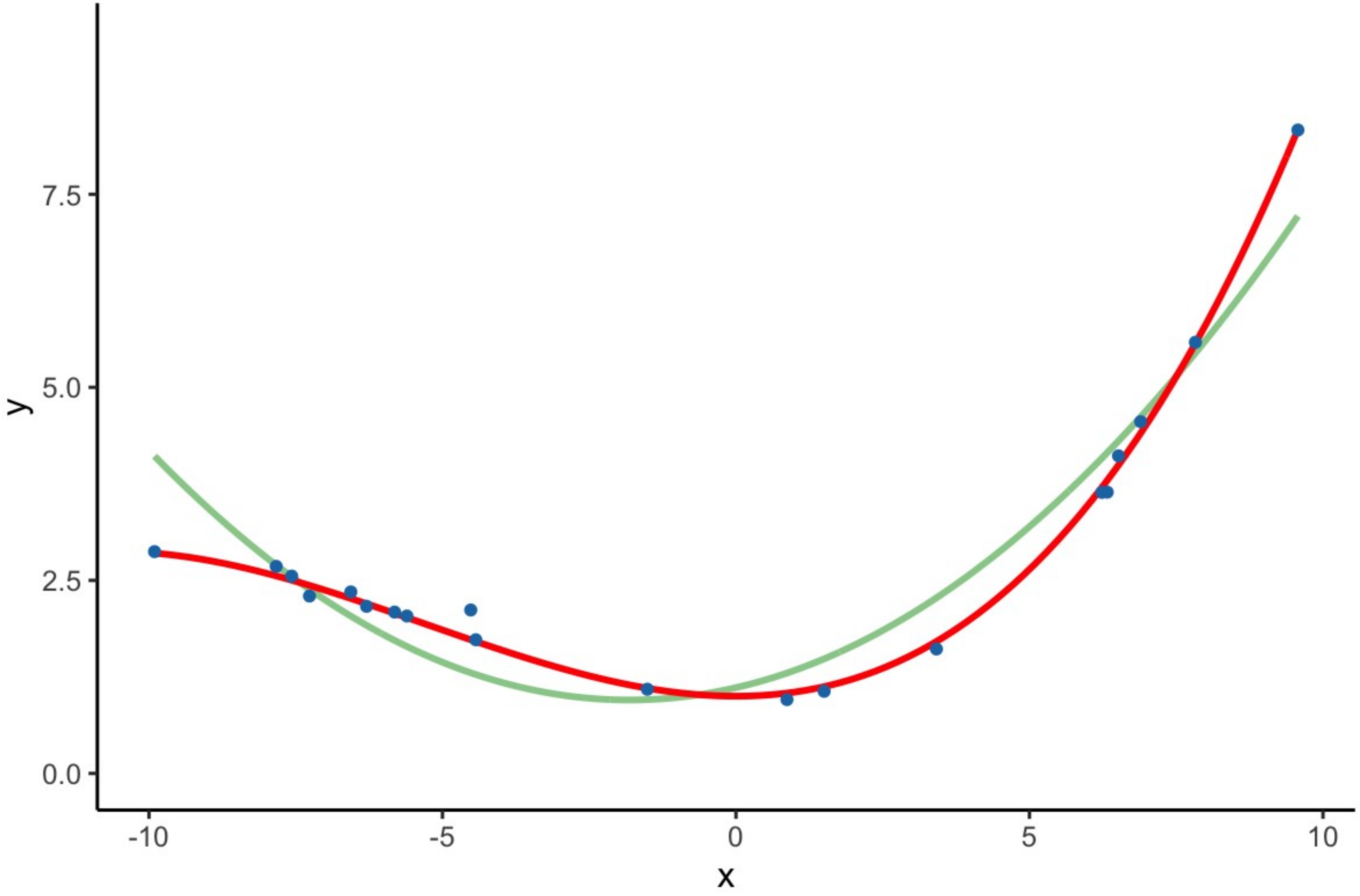
Simulation



Let's simulate the sampling and modeling process for a strictly linear model

$$g(x) = \theta_0 + \theta_1 x$$

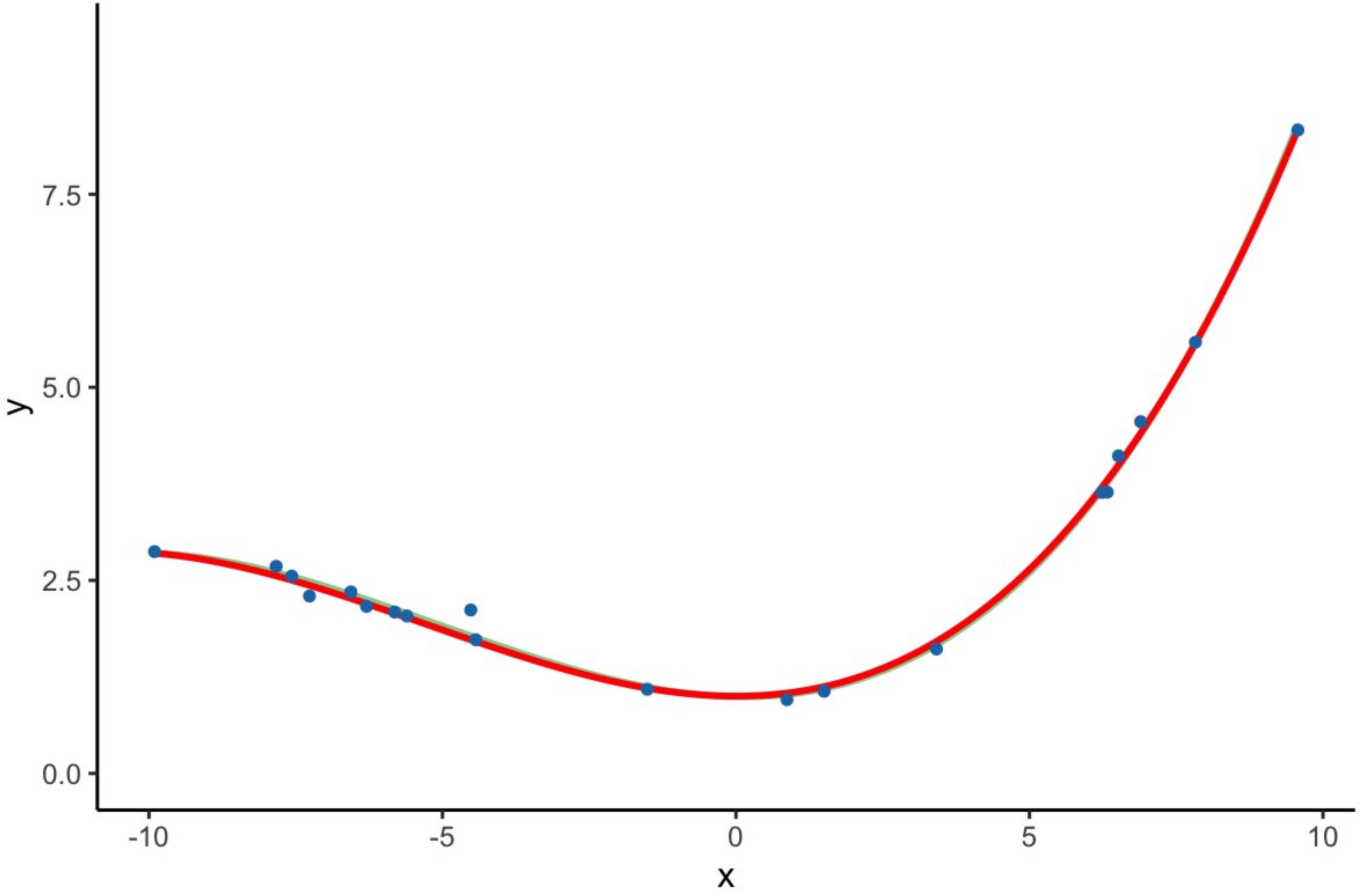
Simulation



Let's simulate the sampling and modeling process for a quadratic model.

$$g(x) = \theta_0 + \theta_1 x + \theta_2 x^2$$

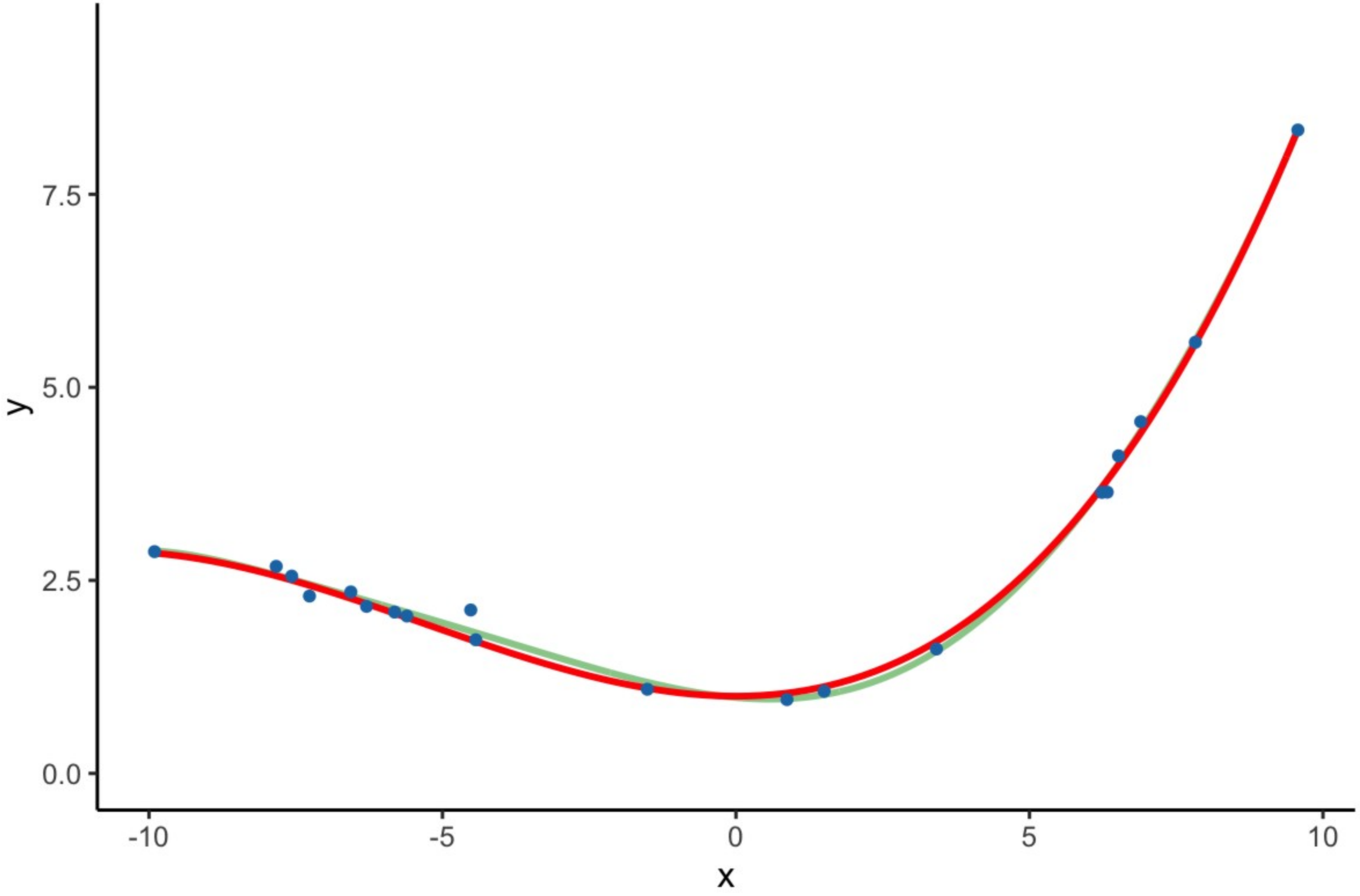
Simulation



Let's simulate the sampling and modeling process for a cubic model.

$$g(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3$$

Simulation



Let's simulate the sampling and modeling process for a septic model.

$$g(x) = \theta_0 + \sum_{i=1}^{7} \theta_i x^i$$

Decomposition of Risk

Model Risk

For a new individual at (x, Y):

- Expected mean squared error of prediction:

$$\text{model risk} = \mathbb{E}((Y - \hat{Y}(x))^2)$$

The expectation is taken over all possible samples that we could have collected.

- Remember, each new sample would generate a different $\hat{Y}(x)$
- Also, for some fixed x, Y can be different due to the random error ε

Decomposition of Error and Risk

The model risk can be decomposed into three pieces:

$$
\begin{aligned}
\mathbb{E}\big((Y - \hat{Y}(x))^2\big) &= \mathbb{E}(\epsilon^2) \\
&\quad + (g(x) - \mathbb{E}(\hat{Y}(x)))^2 \\
&\quad + \mathbb{E}\big((\mathbb{E}(\hat{Y}(x)) - \hat{Y}(x))^2\big)
\end{aligned}
$$

$$
\text{model risk} = \sigma^2 + (\text{model bias})^2 + \text{model variance}
$$

Bias-Variance Decomposition

$$\text{model risk} = \text{observation variance} + (\text{model bias})^2 + \text{model variance}$$

$$\mathbb{E}\big((Y - \hat{Y}(x))^2\big) \;=\; \sigma^2 \;+\; (\mathbb{E}(\hat{Y}(x)) - g(x))^2 \;+\; \mathbb{E}\big((\hat{Y}(x) - \mathbb{E}(\hat{Y}(x)))^2\big)$$

Remember our assumption about the true relationship *g(x)*. When we fit our model, we find some function $f_{\hat{\theta}}$ that estimates *g(x)*. $f_{\hat{\theta}}$ is random and is just another name for $\hat{Y}$

Observation Variance

$$\mathbb{V}ar(Y) = \mathbb{V}ar(g(x) + \epsilon) = \mathbb{V}ar(\epsilon) = \sigma^2$$

Some reasons:

- Measurement error
- Missing information acting like noise

Some remedies:

- Could try to get more precise measurements.
- Often this is beyond the control of the data scientist.

Model Variance

$$\text{model variance} = \mathbb{V}ar(\hat{Y}(x)) = \mathbb{E}\big((\hat{Y}(x) - \mathbb{E}(\hat{Y}(x)))^2\big)$$

Main reason:

- Overfitting: small differences in random samples lead to large differences in the fitted model

Some remedies:

- Reduce model complexity
- Don’t fit the noise

Model Bias

$$\text{model bias} = \mathbb{E}(\hat{Y}(x)) - g(x)$$

Some reasons:

- Underfitting
- Lack of domain knowledge

Remedies:

- Increase model complexity (but don't overfit)
- Consult domain experts to see which models make sense

A Constant Model

So which model is better? Model A or Model B?

Model A: Select a random number between 0 and 1. This is your estimate of *p*.
This is equivalent to running `np.random.random()` in Python.
Model B: Select .75 as your estimate of *p*.

We can calculate the model risks directly. Note that the observation variance is 0.

Model A:

Model Bias = .5 - .5 = 0
Model Variance = $(1 - 0)^2$ / 12 = 1/12

Risk = 0^2 + 1/12 = **1/12**

Model B:

Model Bias = .75 - .5 = .25
Model Variance = 0

Risk = $.25^2$ + 0 = **1/16**

Overfitting

Introduction to Overfitting

In the previous lectures, our goal has been to **minimize** a loss function (MSE)

- We do this by collecting more features, or through **feature engineering**

However, we only ever evaluated our model on the **data on which it was trained**

- The whole point in building a model is to *learn something about the world*
- Why do we care about finding a $\hat{Y}$ if we already know Y?

We care about how well our model performs on **new** data, for which we want to **predict** Y

Complexity

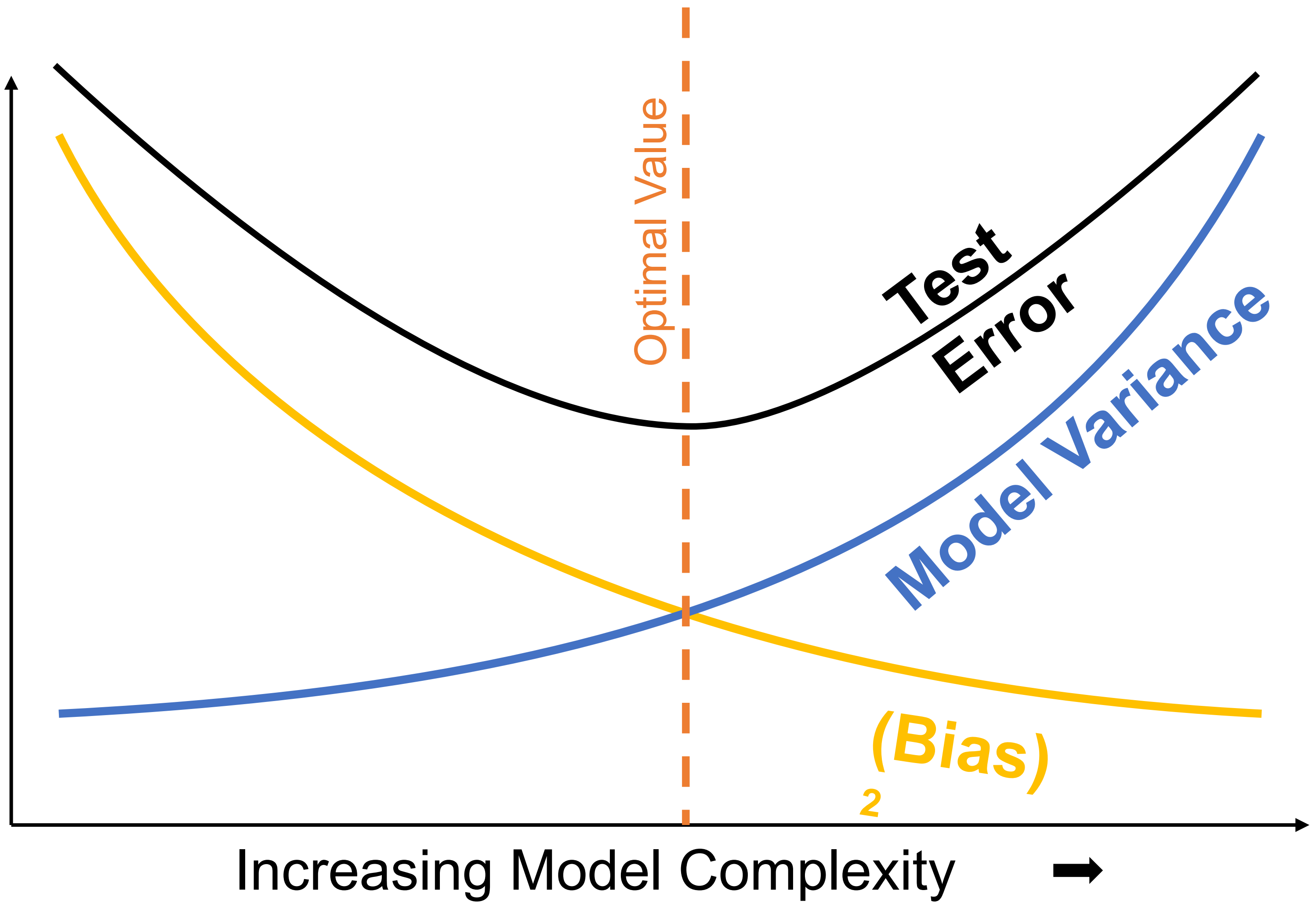
Modeling Goals

- Try to minimize all three of observation variance, model bias, and model variance.

But

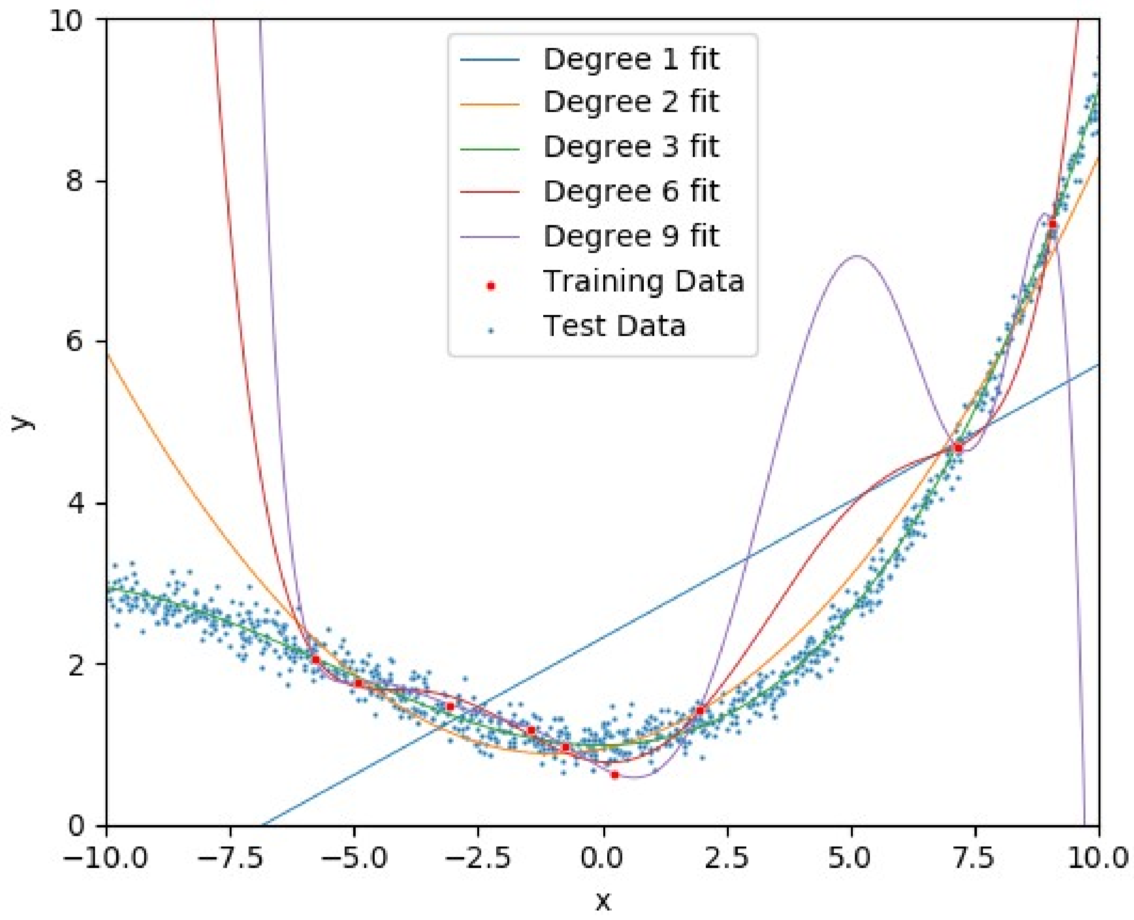
- Observation variance is often out of our control
- Reducing complexity to reduce model variance can increase bias
- Increasing model complexity to reduce bias can increase model variance
- Domain knowledge matters: the right model structure!

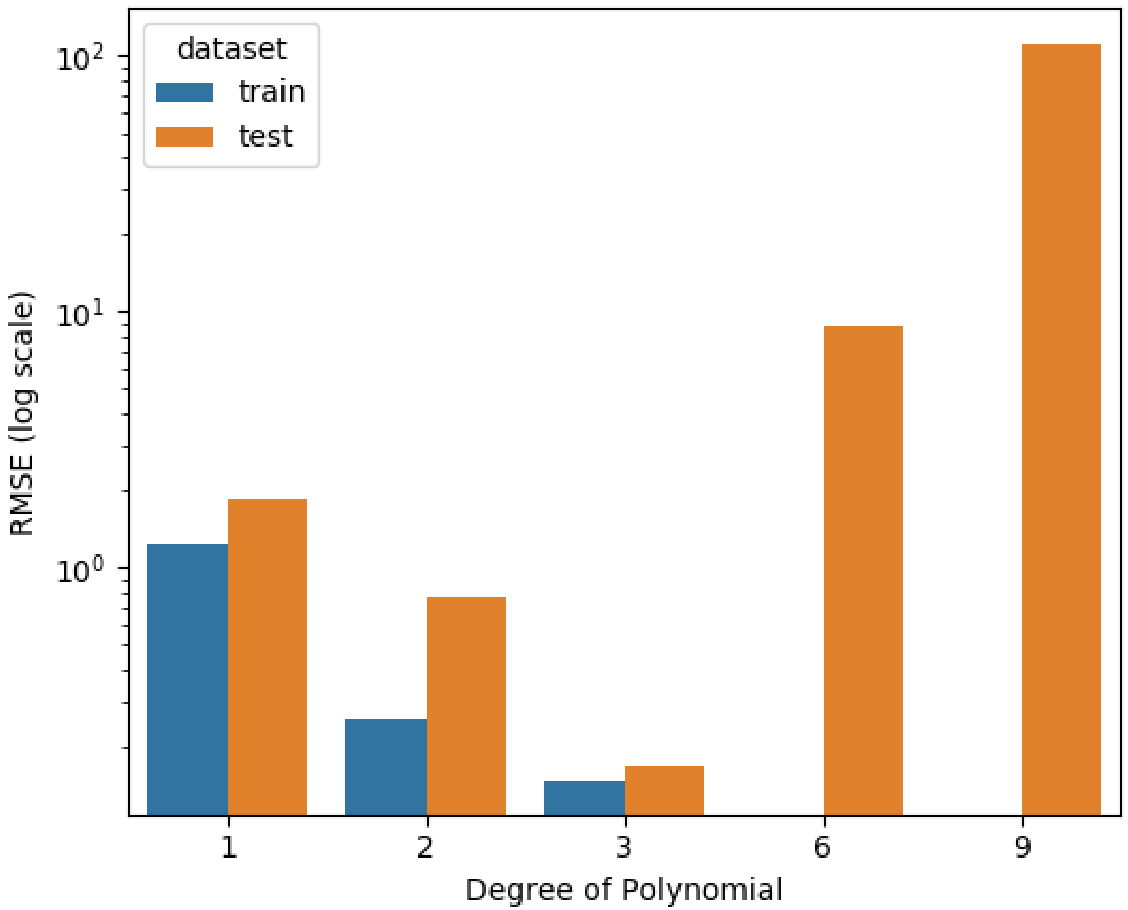
Bias Variance Plot



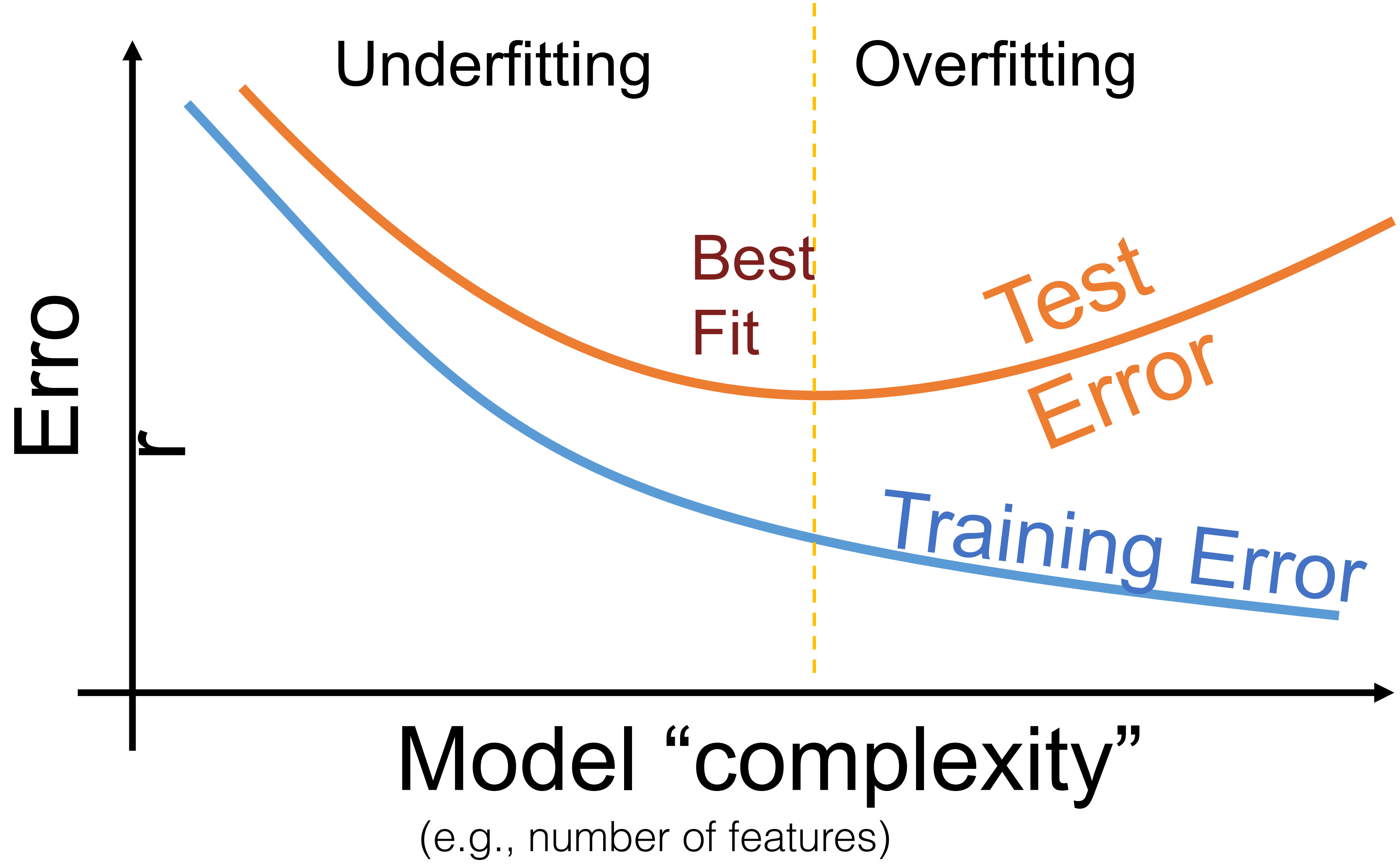
Cross-Validation

Training Error vs Test Error





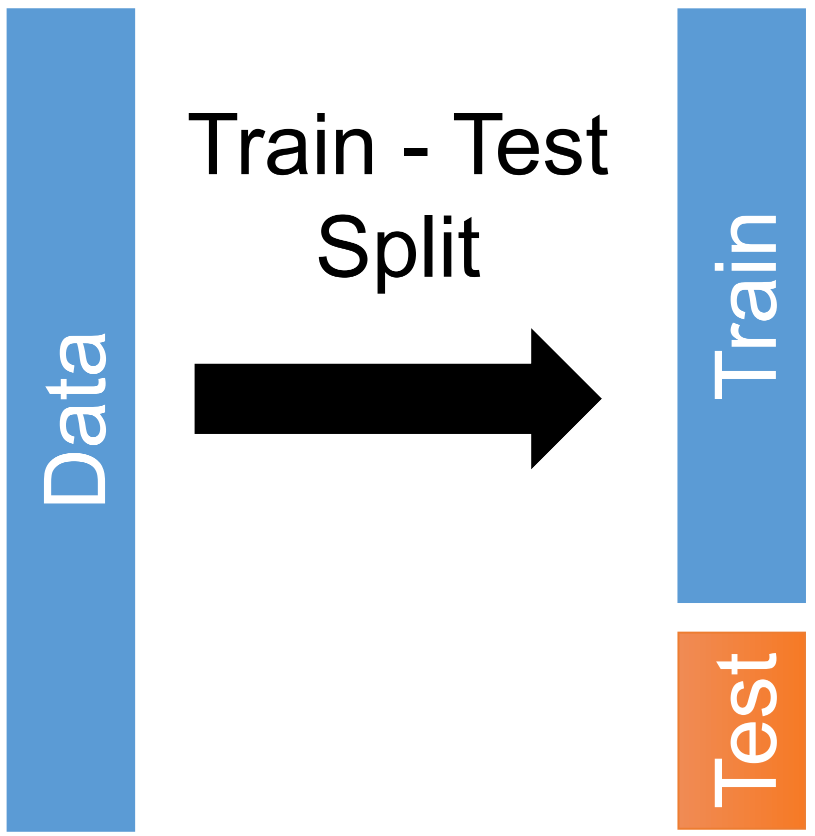
Training Error vs Test Error



Training error typically *underestimates* **test error**.

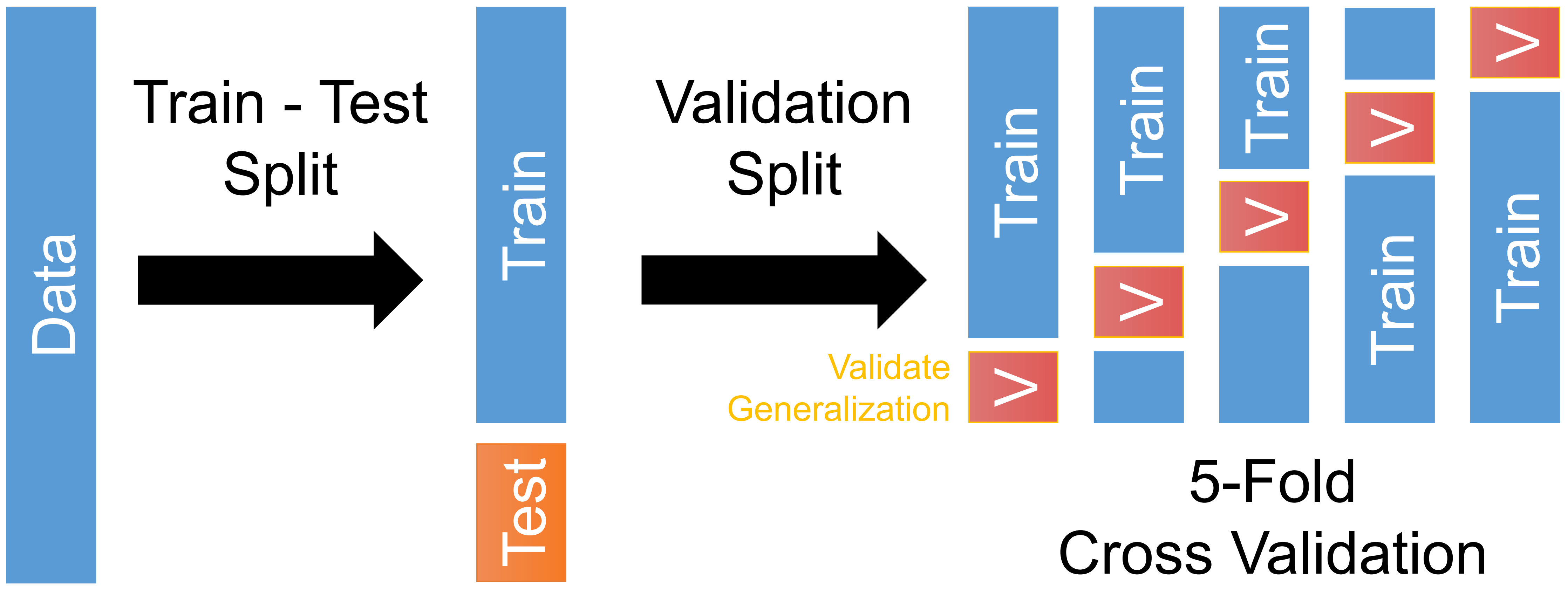
Generalization: *The Train-Test Split*

- **Training Data:** used to fit model
- **Test Data:** check generalization error
- How to split?
  - Depends on application (usually randomly)
- What size? (90%-10%)
  - Larger training set – more complex models
  - Larger test set – better estimate of generalization error
  - Typically between 70%-30% and 90%-10%



You can only use the test dataset once after deciding on the model.

Generalization: *Validation Split*



Cross validation ***simulates multiple train test-splits*** *within the training data.*

Recipe for Successful Generalization

1. Split your data into **training** and **test** sets (90%, 10%)
2. Use **only the training data** when designing, training, and tuning the model
   - Use **cross validation** to test *generalization* during this phase
   - **Do not look at the test data!**
1. Commit to your final model and train once more using **only the training data.**
2. Test the final model using the **test data**.
3. Train on **all available data** and ship it!

Demo

Next week

- Another strategy to possibly overcome overfitting:
  - Regularization