Learning Agents

Chapter 18

Supervised Learning and Decision Trees

Jim Rehg
Georgia Tech

Learning

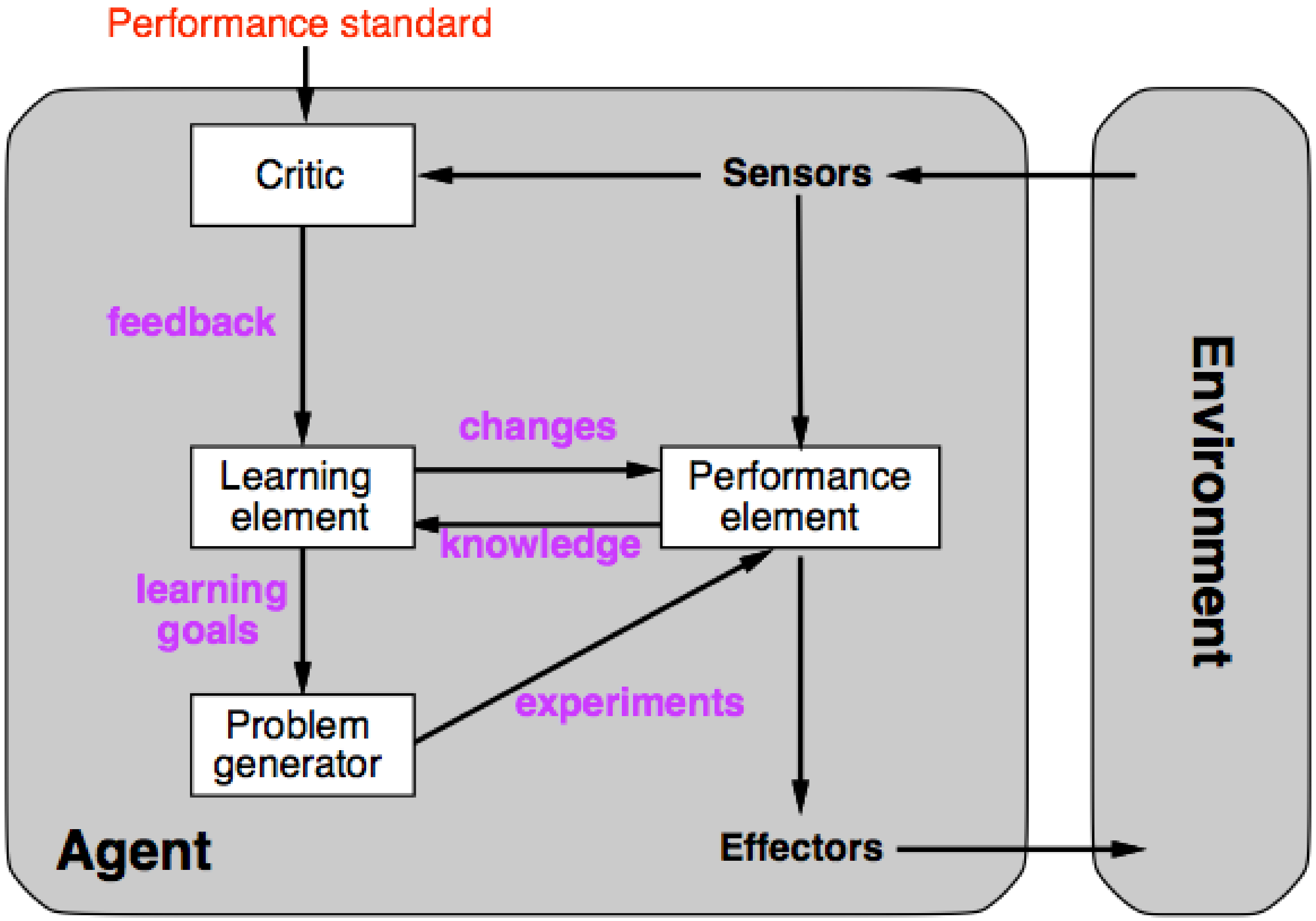
- Percepts can be used not only for making decisions and plans about acting, but to improve performance over time
- Needed for unknown environments, or when designer can't predict everything

Learning Overview

- CH 18: Learning from Examples
  - Decision Trees
  - Neural Nets, SVMs, KNN
- CH 20: Learning Probabilistic Models
  - Bayesian Learning

Project 4!

Remember from Chapter 2...



Learning Element

Performance element	Component	Representation	Feedback
Alpha–beta search	Eval. fn.	Weighted linear function	Win/loss
Logical agent	Transition model	Successor–state axioms	Outcome
Utility–based agent	Transition model	Dynamic Bayes net	Outcome
Simple reflex agent	Percept–action fn	Neural net	Correct action

Type of feedback is determining factor in class of learning algorithm used

Types of Learning Problems

- Supervised:
  - given examples of input-output pairs, learn a function mapping inputs to outputs
- Unsupervised:
  - given examples, find patterns
- Reinforcement Learning:
  - learn desired behavior from a reward signal

Supervised Learning

x – *Input vector*

(e.g. list of attributes, vector of reals, etc.)

y – *Output label*: $y \in \{-1,+1\}$ or $y \in \{0,1\}$

Binary classification (or labeling) problem

Multi-class classification when $y \in \{1,2,\ldots,K\}$

D – *Training dataset* containing N pairs:

$$D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$$

E.g. pairs:

O	O	X
	X	
X		

, $+1$

(T,F,F,T,Full,$$), 0

$\begin{bmatrix} -2 \\ 0.7 \\ 5.1 \end{bmatrix}, -1$

Supervised Learning

Target function f maps from input to output:

$y = f(x)$ (f is unknown)

Our goal is to use D to learn function h which approximates f $(h \approx f)$

h is hypothesis, H is hypothesis space, $h \in H$

Examples of H:

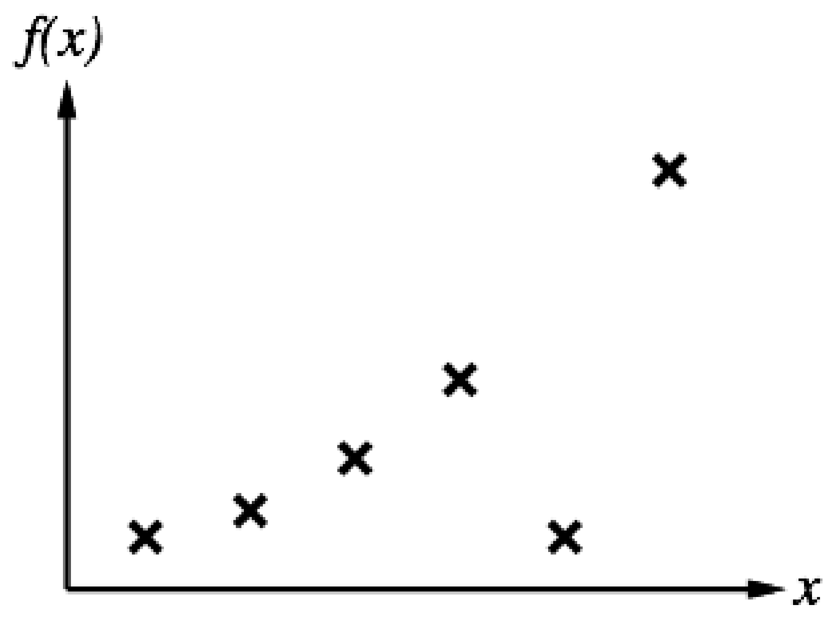
- Linear functions
- Quadratic decision boundaries
- Decision trees
- Neural networks

Simplified Model of Learning

- Ignores prior knowledge
  - Can be addressed using Bayesian learning
- Assumes fixed, observable f
  - Can be addressed with on-line stochastic learning
- Assumes labels are available for all examples
  - Can be addressed through active learning and weakly-supervised learning

Basic Example

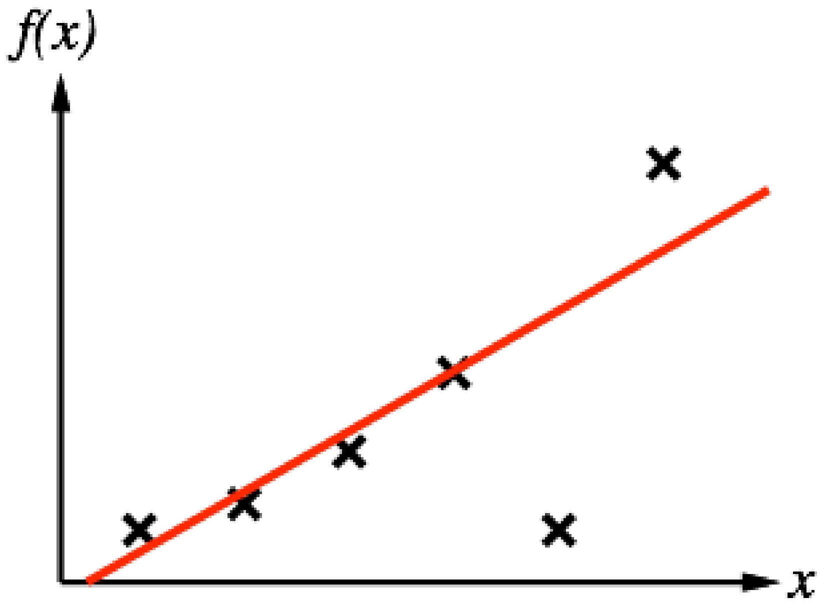
Curve fitting:



Dataset D consists of (x, $f(x)$) pairs

Basic Example

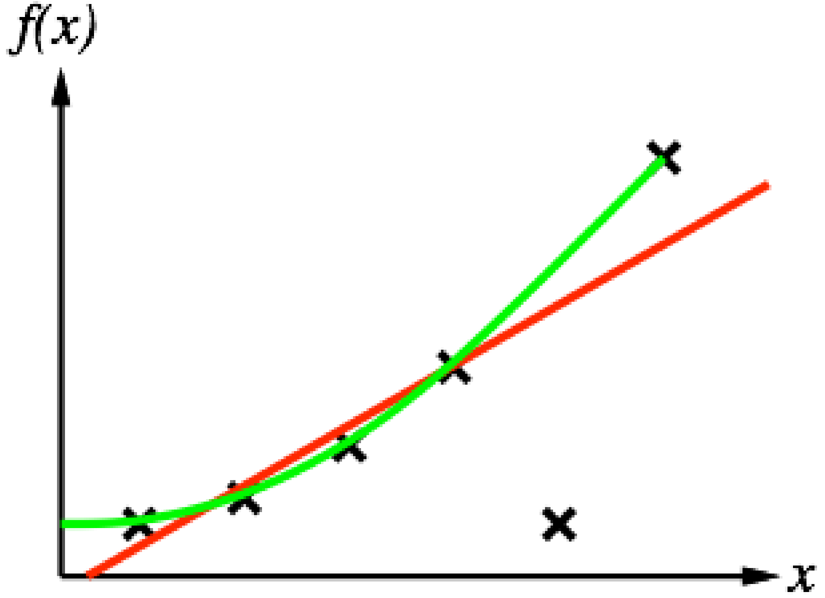
Curve fitting:



$$H = \{\text{ straight lines }\} = \{w_1 x + w_0 \mid w_1, w_0 \in \Re\}$$

Basic Example

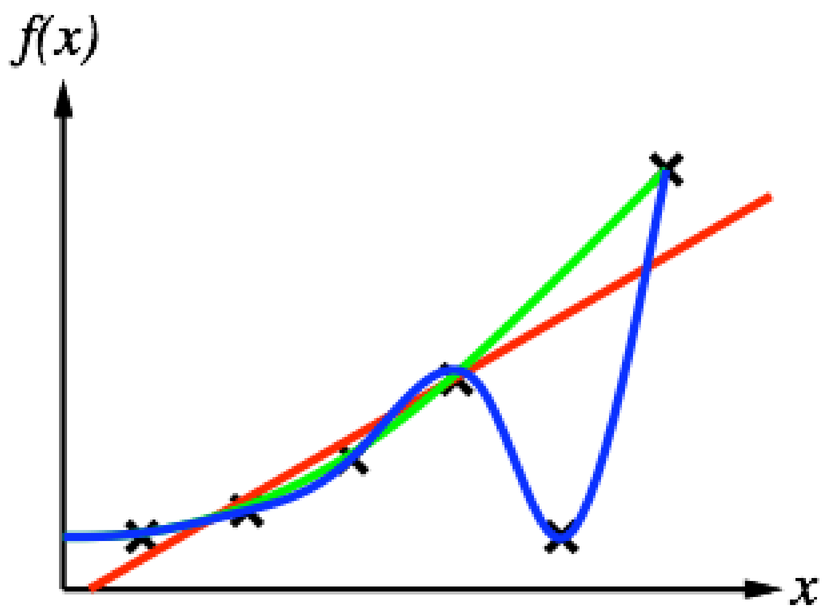
Curve fitting:



$$H = \{\, \text{quadratic curves} \,\}$$

Basic Example

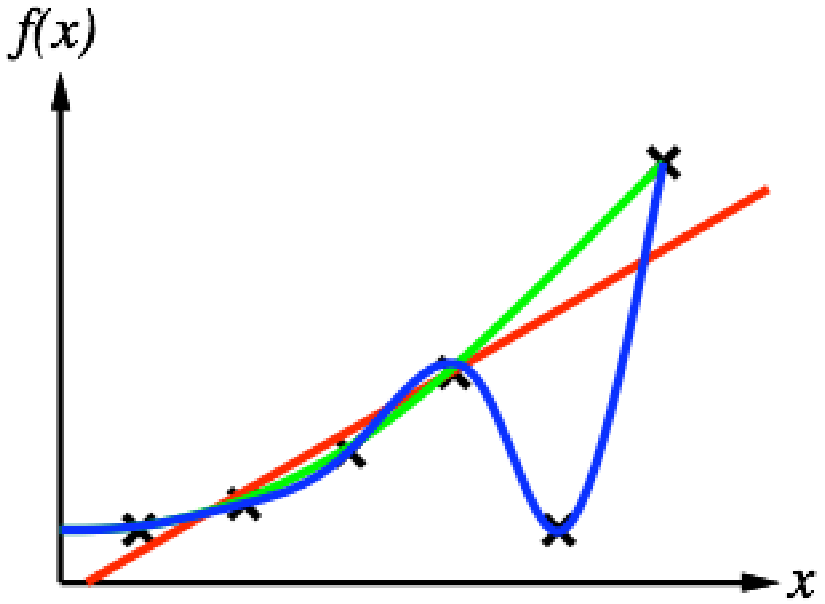
Curve fitting:



$$H = \{\text{ 5th order polynomials }\}$$

Basic Example

Which model is the best? (linear, quadratic, quintic)



Principle: *Occam's Razor*

Choose the *simplest* model which *fits the data* well

=> Quadratic fit is the best choice

Variance and Overfitting

Variance – Variation in model parameters as we fit the model to multiple random samples of data

Complex models tend to have high variance

Overfitting – Model parameters determined by noise in training data, don’t generalize well

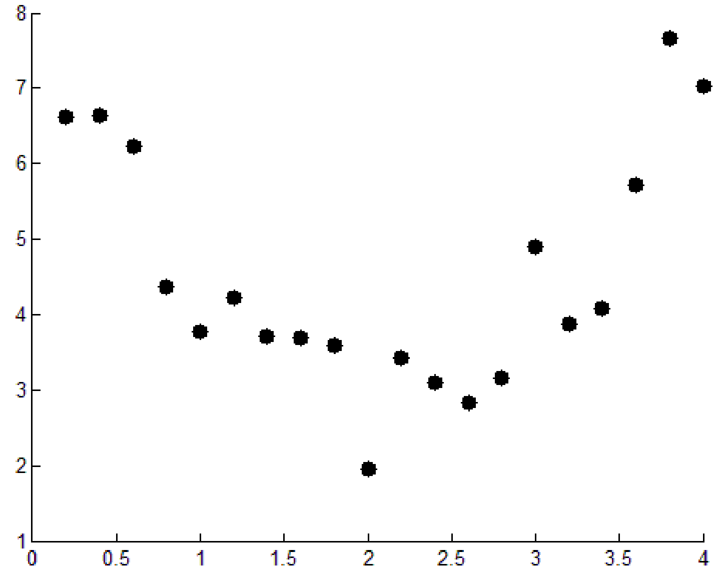
Complex models are prone to overfitting

Generalization – Performance of trained model on new datasets, e.g. during testing phase

Example of Variance

Data points from quadratic function with added noise

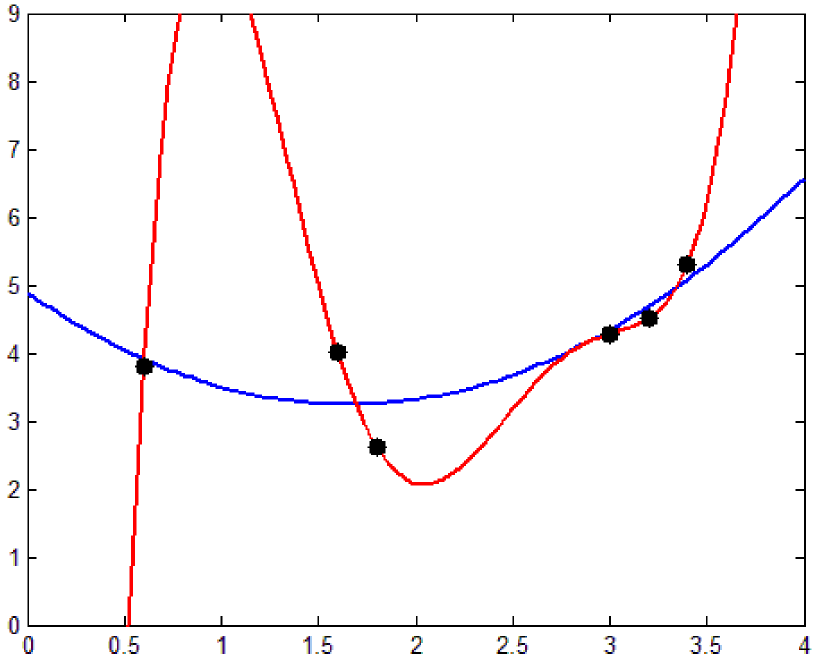
Three datasets (below) generated by sampling 6 points at random



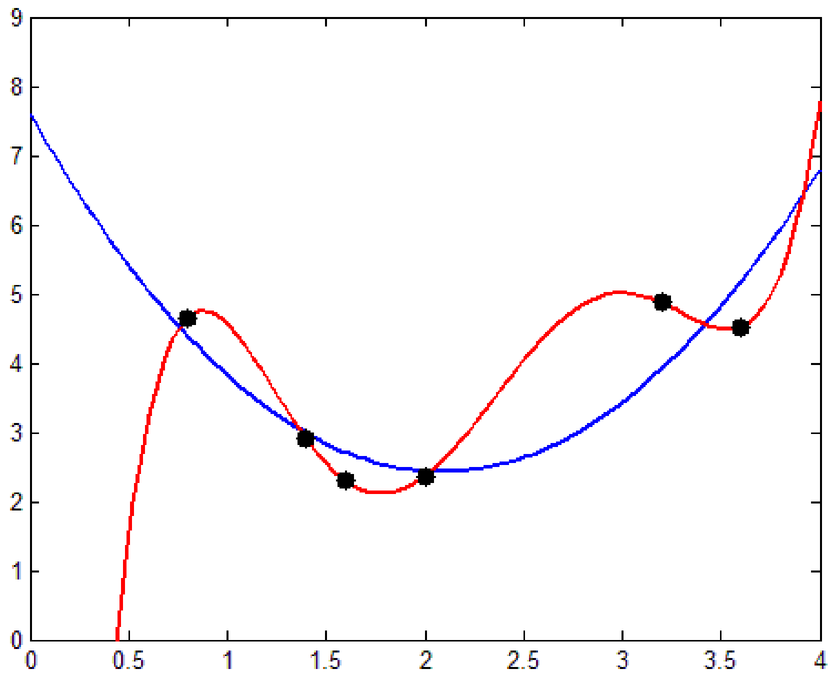
Blue curves are 2nd order (quadratic) fit to data

Red curves are 5th order (quintic) fit to data

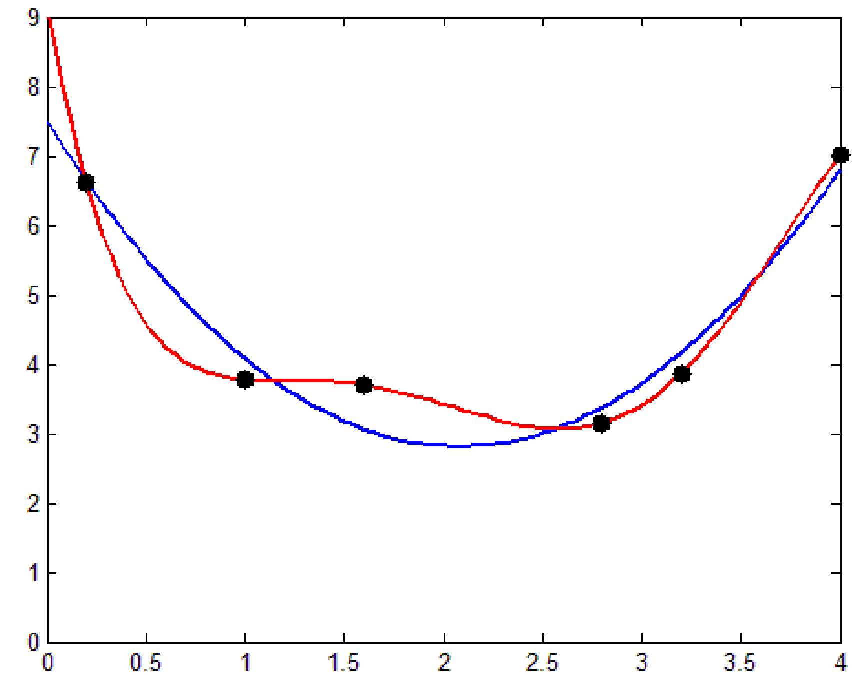
Dataset #1



Dataset #2



Dataset #3



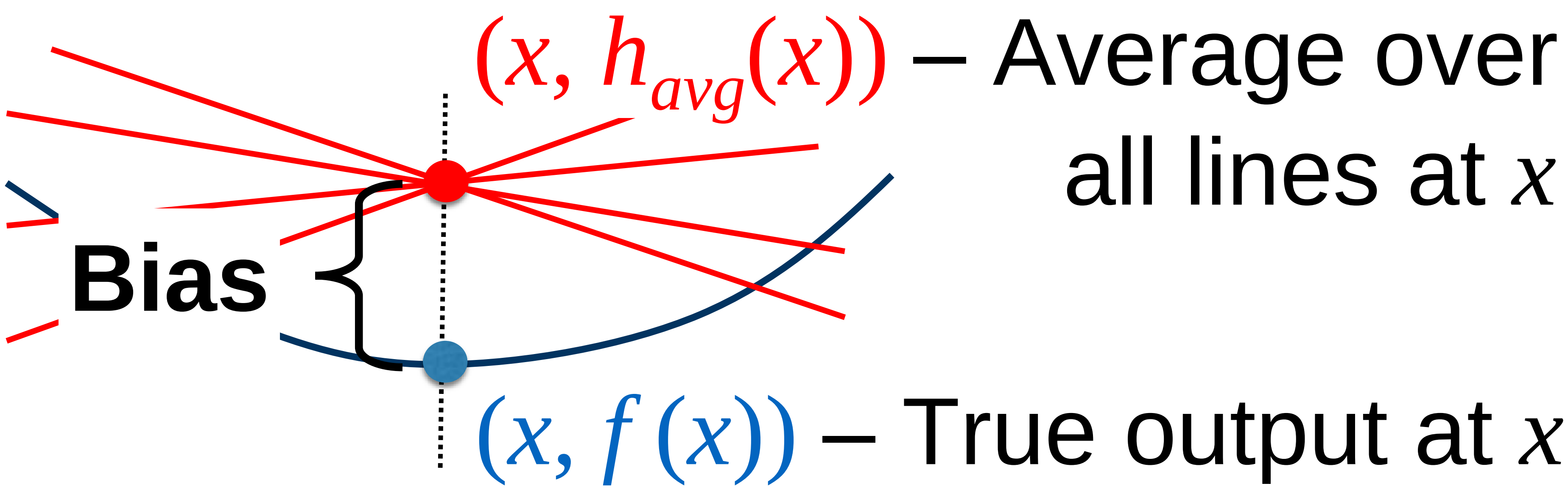
2nd order fit (blue) has much lower variance than 5th order fit (red)

Bias and Underfitting

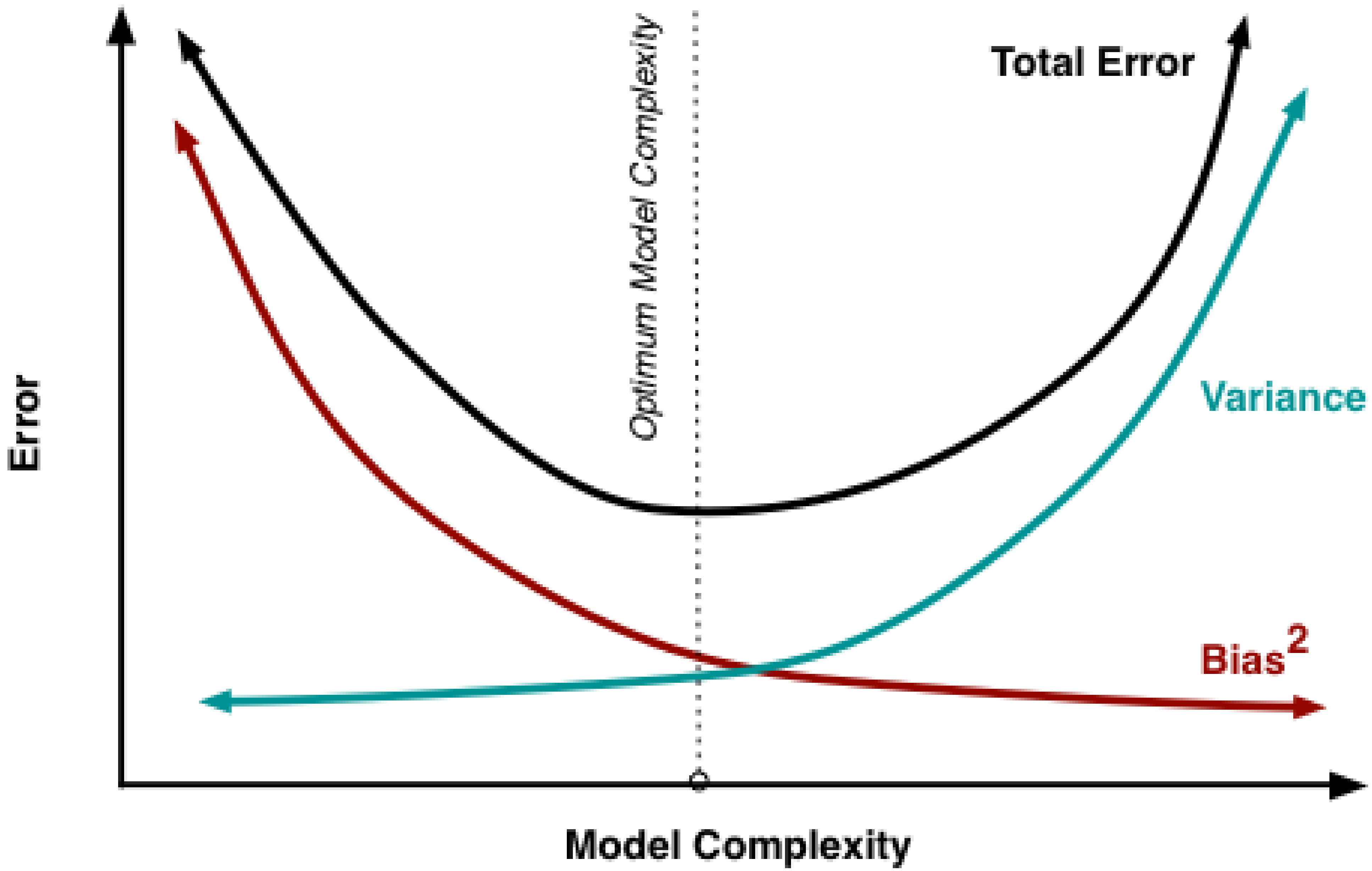
Related to the variance of a model is the *bias* of the model

The bias is the difference (error) between the target output and the best output of the model (averaged over all datasets of size N).

Large bias means insufficient model capacity to learn the target concept (even with large N).



Bias-Variance Tradeoff



Good models minimize bias and variance

Generalization and Overfitting

The goal of learning is *good generalization*, meaning that performance on unseen testing data (when the system is deployed in world) matches performance seen during training.
We need:

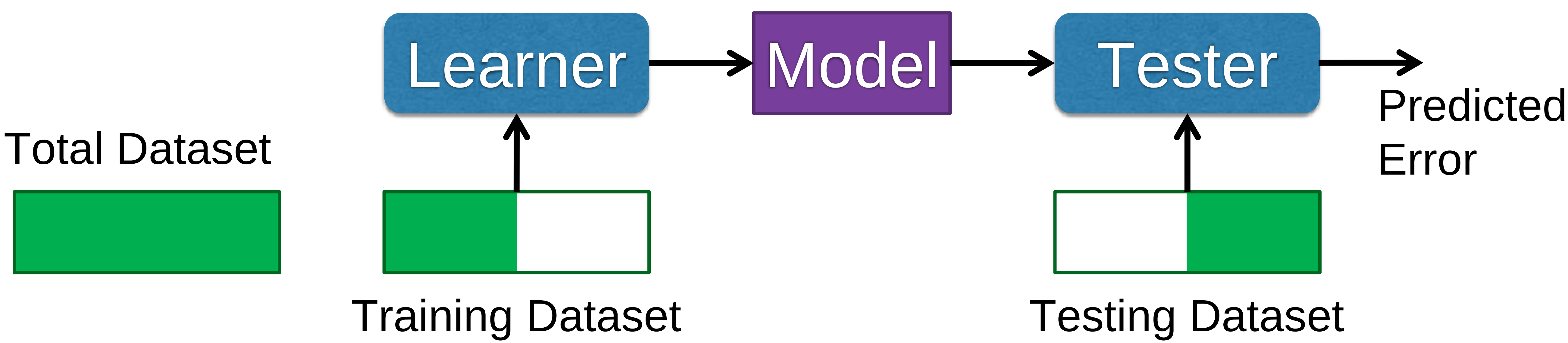
- Procedure for measuring the *generalization error* (expected error on unseen data).
- Procedure for automatically *selecting model capacity* (complexity) to ensure good generalization (avoid overfitting)

Training and Testing Split

Split points into separate training and testing datasets (e.g. 50-50 split)

Train on the *training dataset*, evaluate the resulting model on the *testing dataset*

The testing error gives a reliable measure of *generalization error*, as the testing data points were unseen during training

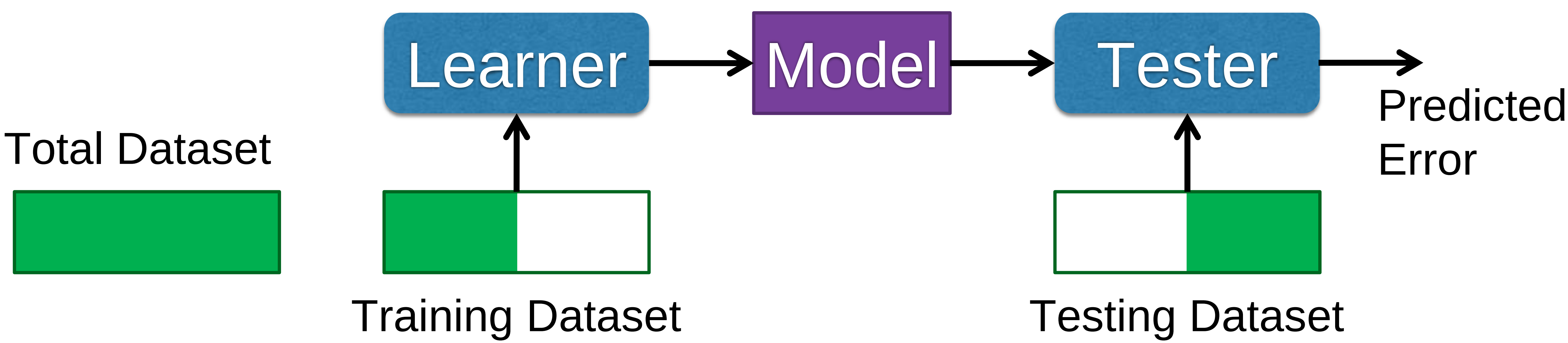


Training and Testing Split

Split points into separate training and testing datasets (e.g. 50-50 split)

Never Test on Your Training Data !!!!

The testing error gives a reliable measure of *generalization error*, as the testing data points were unseen during training



Training and Testing Split

While a basic 50-50 training-testing split is not a bad starting point for model development, it does not use your hard-won data effectively

Why should each data point be used only once, and in the randomly-assigned role of either training or testing?

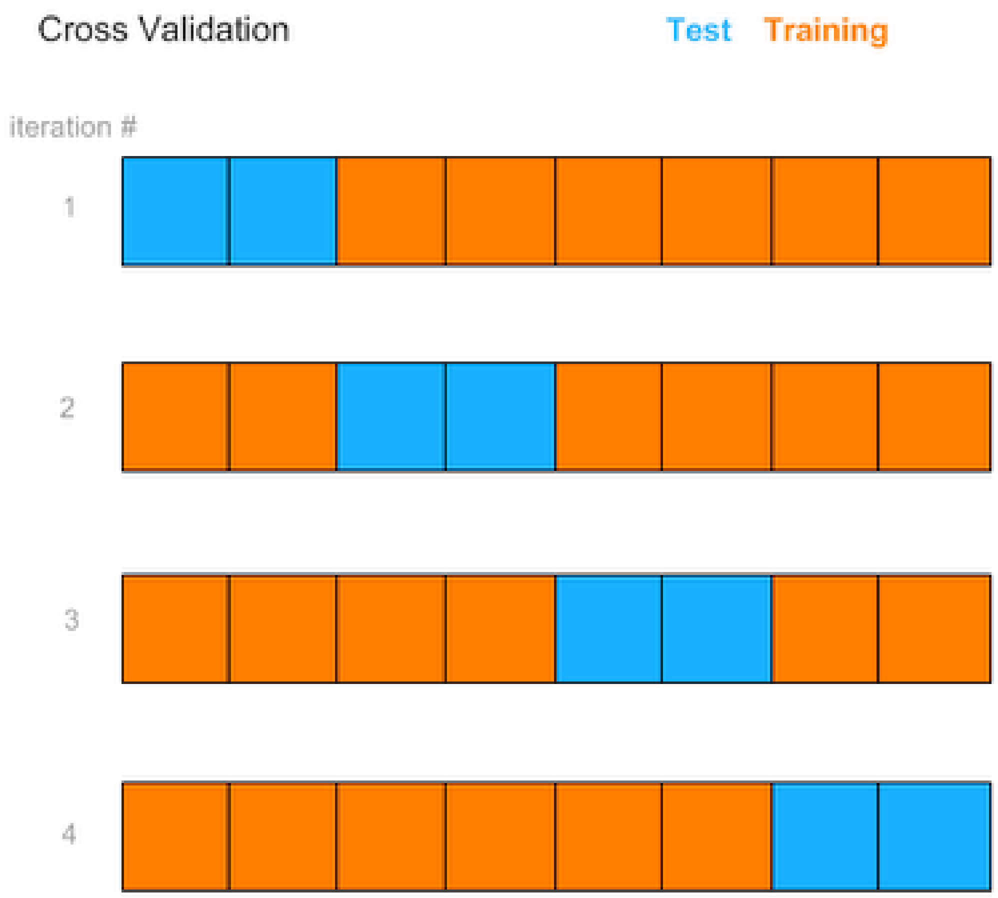
Why can’t each data point contribute to both, and why can’t we use averaging to improve the variance in our reported generalization error?

Cross Validation

Cross validation is a procedure for using your data effectively to measure generalization error

In 5-fold cross validation:

- Divide your dataset into 1/5 splits at random
- Each split is used once for testing and 4 times for training
- Average the errors over the five iterations



Model Selection

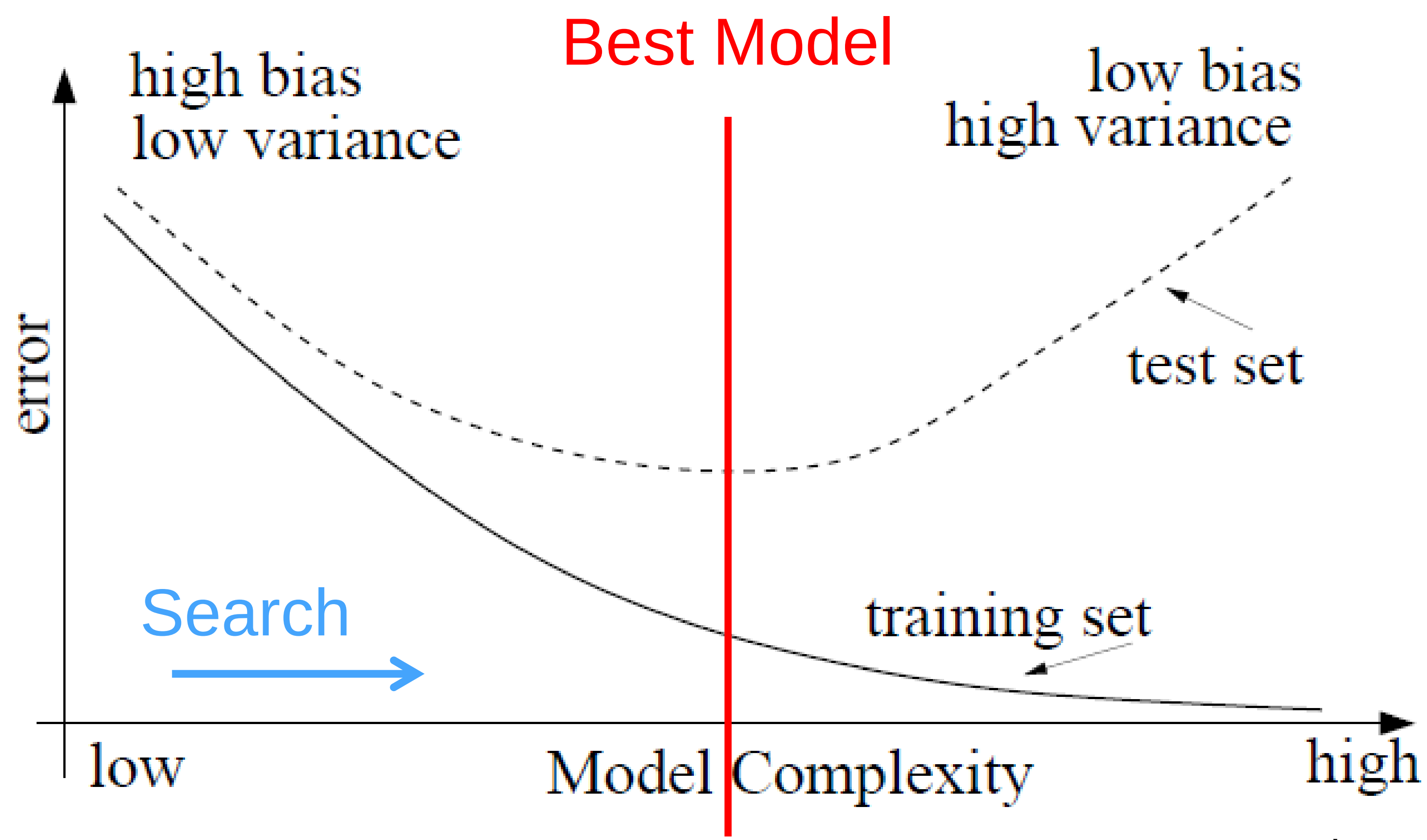


Figure credit: Sam Roweis

Estimate testing error using cross-validation

Search over increasing model complexity to find the best model (low variance, low bias)

Model Selection

Given a "control knob" which can increase the complexity of your model, search over increasing complexity to find "minimum"

See Figure 18.8 in your textbook for pseudocode

Different control knobs for different types of models

Examples:

- Regression: Polynomial order
- Decision trees: Amount of pruning