STA 314: Statistical Methods for Machine Learning I

Lecture 1 - Introduction to Statistical Learning and the Bias-Variance Tradeoff

Xin Bing

Department of Statistical Sciences
University of Toronto

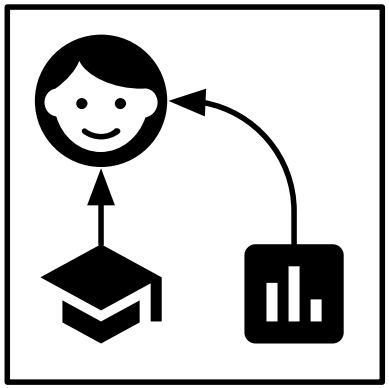
What is machine learning?

- Machine learning approach: program an algorithm to automatically learn from data, or from experience

- Why might you want to use a learning algorithm?
  - hard to code up a solution by hand (e.g. vision, speech)
  - system needs to adapt to a changing environment (e.g. spam detection)
  - want the system to perform *better* than the human programmers

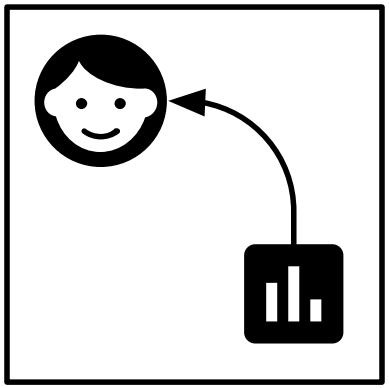
Types of machine learning problems

Supervised Learning



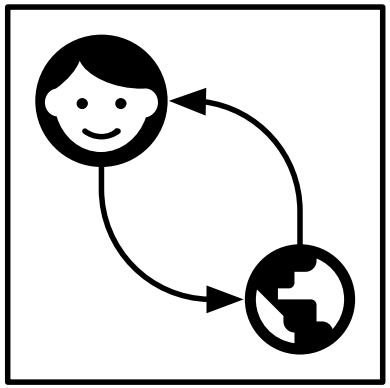
Machine is given data and examples of what to predict.

Unsupervised Learning



Machine is given data, but not what to predict.

Reinforcement Learning



Machine gets data by interacting with an environment and tries to minimize a cost.

Supervised Learning

- Outcome measurement Y (also called dependent variable, response, target).
- Vector of p predictor measurements X (also called inputs, regressors, covariates, features, independent variables).
- In **regression** problems, Y is quantitative (e.g price, blood pressure).
- In **classification** problems, Y takes values in a finite, unordered set (survived/died, digit 0-9, cancer class of tissue sample).
- We have training data $(x_1, y_1), ..., (x_n, y_n)$. These are observations (instances, realizations) of the measurement (X, Y).

Main tasks in Supervised Learning

On the basis of the training data we would like to:

- Prediction: accurately predict future outcome (Y).
- Estimation: understand how features (X) affect the outcome (Y).
- Model selection: find the best model for predicting the outcome (Y) or which features (X) affect the outcome (Y).
- Inference: assess the quality of our prediction, estimation and model selection.

Spam emails detection

- data from $n = 4601$ emails sent to an individual (named George, at HP labs). Each is labeled $Y \in \{\textbf{spam}, \textbf{email}\}$.
- goal: build a customized spam filter
- input features X: relative frequency of 57 words and punctuation marks in the email message

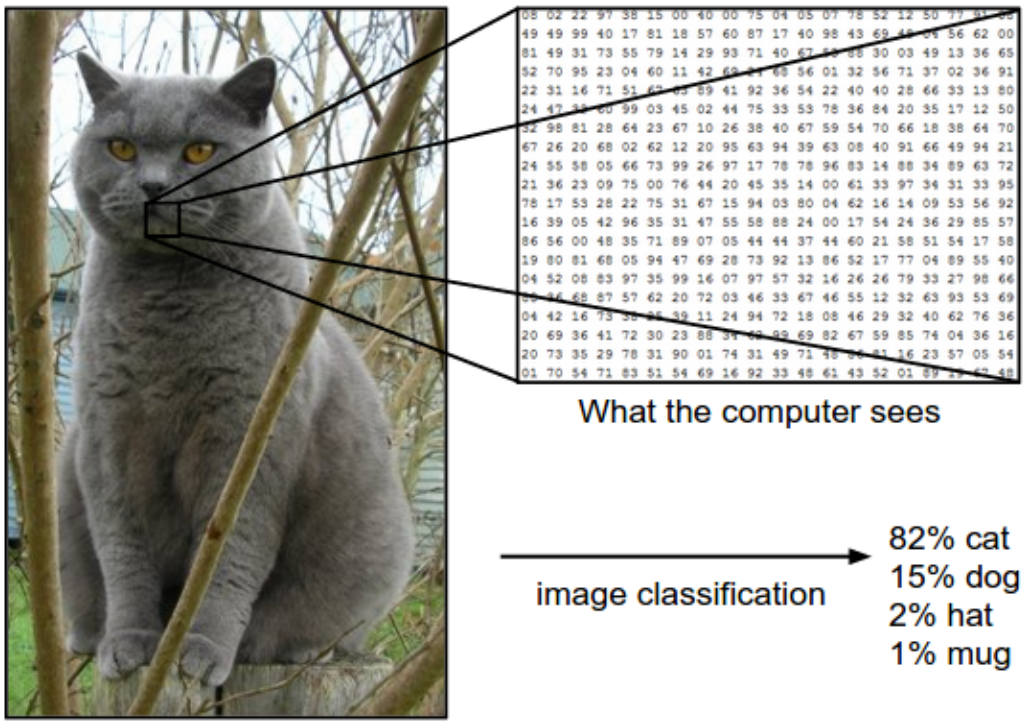
	george	you	hp	free	!	edu	remove
spam	0.00	2.26	0.02	0.54	0.51	0.01	0.28
email	1.27	1.27	0.90	0.07	0.11	0.29	0.01

Image recognition

For image data, we observe n images with annotated labels

$$y_i \in \{\text{cat, dog, hat, mug}\}, \qquad 1 \le i \le n.$$

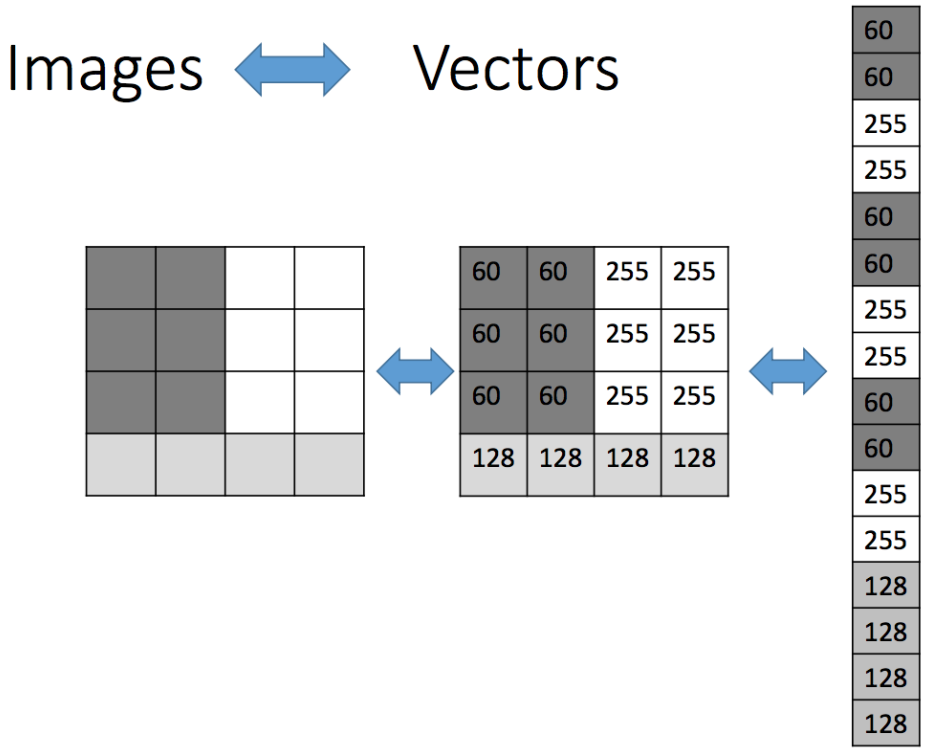
What an image looks like to the computer:



[Image credit: Andrej Karpathy]

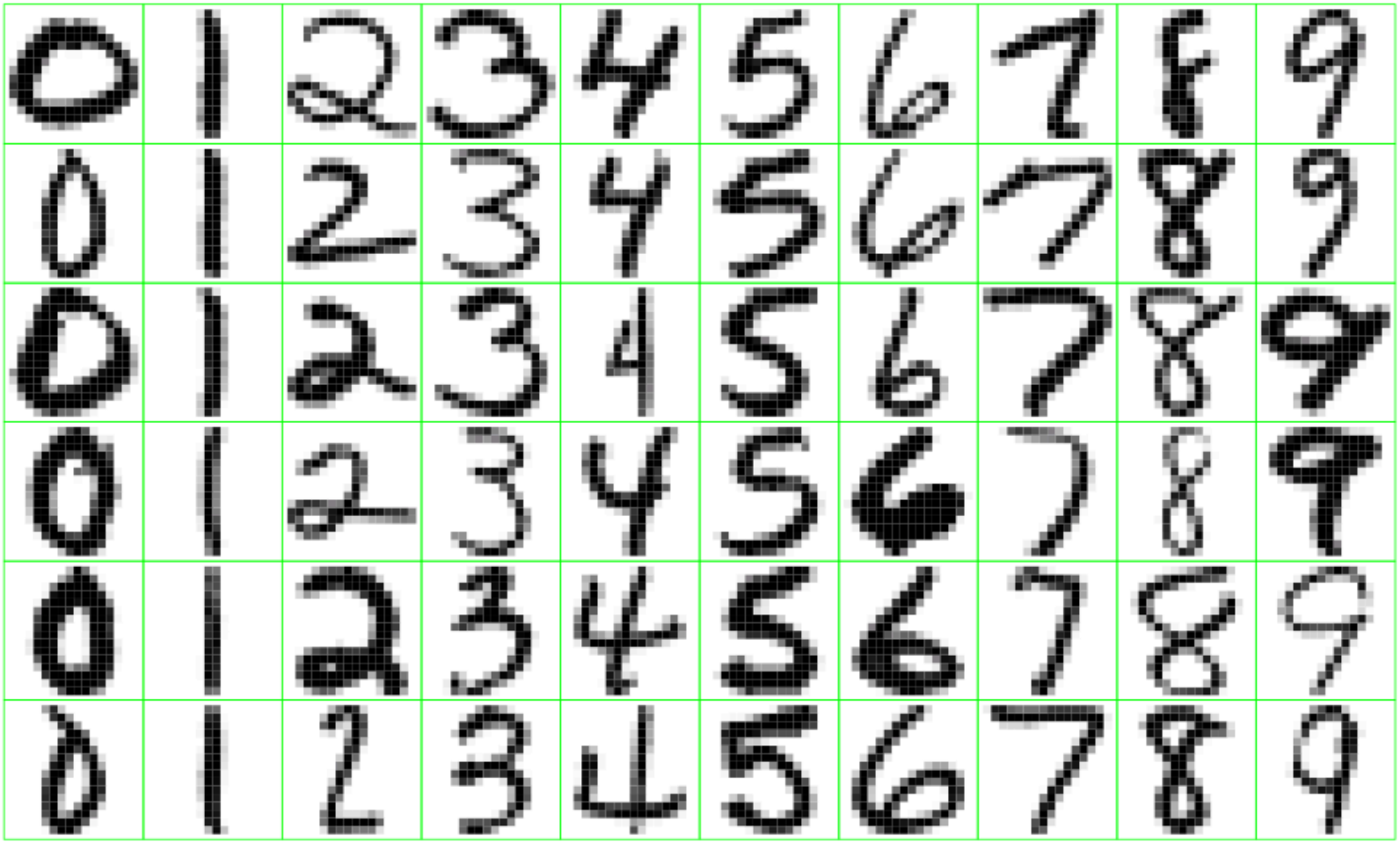
Image recognition

Features x_i are represented as a vector



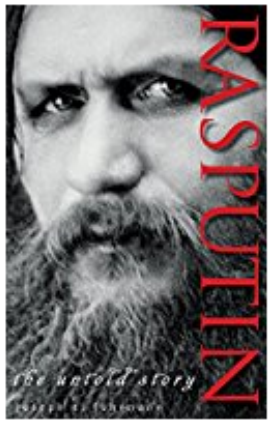
The goal is to let the machine learn the function $f : x_i \rightarrow y_i$ to predict the labels for unanotated images.

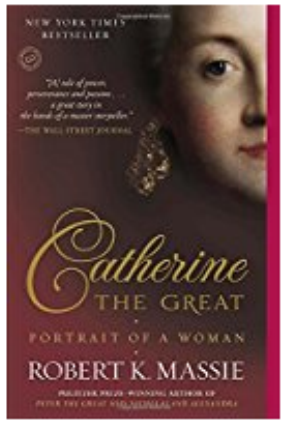
Detect numbers in a handwritten zip code

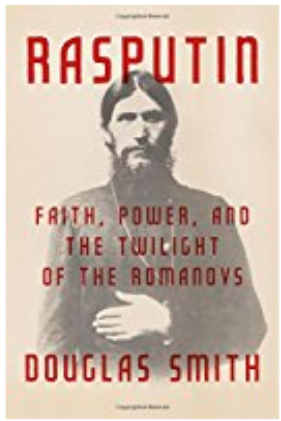


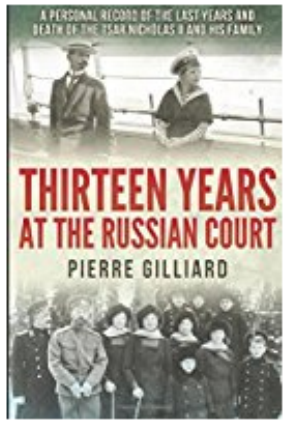
Recommender Systems : Amazon, Netflix, ...

Inspired by your shopping trends

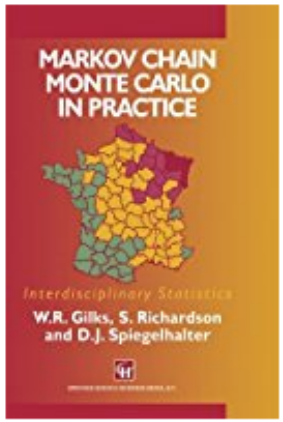


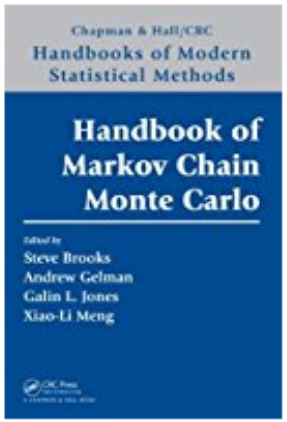


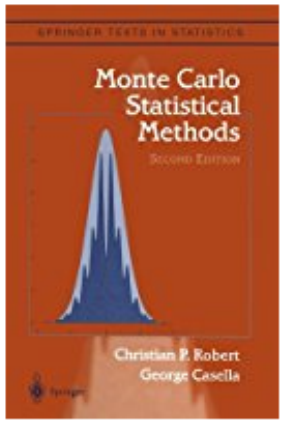


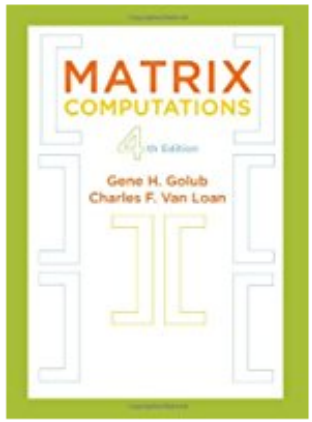


Related to items you've viewed See more









Computer vision: Object detection, semantic segmentation, pose estimation, and almost every other task is done with ML.

Object detection

DAQUAR 1553

What is there in front of the sofa?

Ground truth: table

IMG+BOW: table (0.74)

2-VIS+BLSTM: table (0.88)

LSTM: chair (0.47)

COCOQA 5078

How many leftover donuts is the red bicycle holding?

Ground truth: three

IMG+BOW: two (0.51)

2-VIS+BLSTM: three (0.27)

BOW: one (0.29)

Speech: speech to text, personal assistants, speaker identification...

Unsupervised Learning

No outcome variable Y, just a set of features X measured on a set of samples.

- objective is more fuzzy – find groups of samples that behave similarly, find features that behave similarly, find linear combinations of features with the most variation.
- difficulty in model assessment: difficult to quantify how well you are doing.
- different from supervised learning, but can be useful as a pre-processing step for supervised learning.

Image segmentation

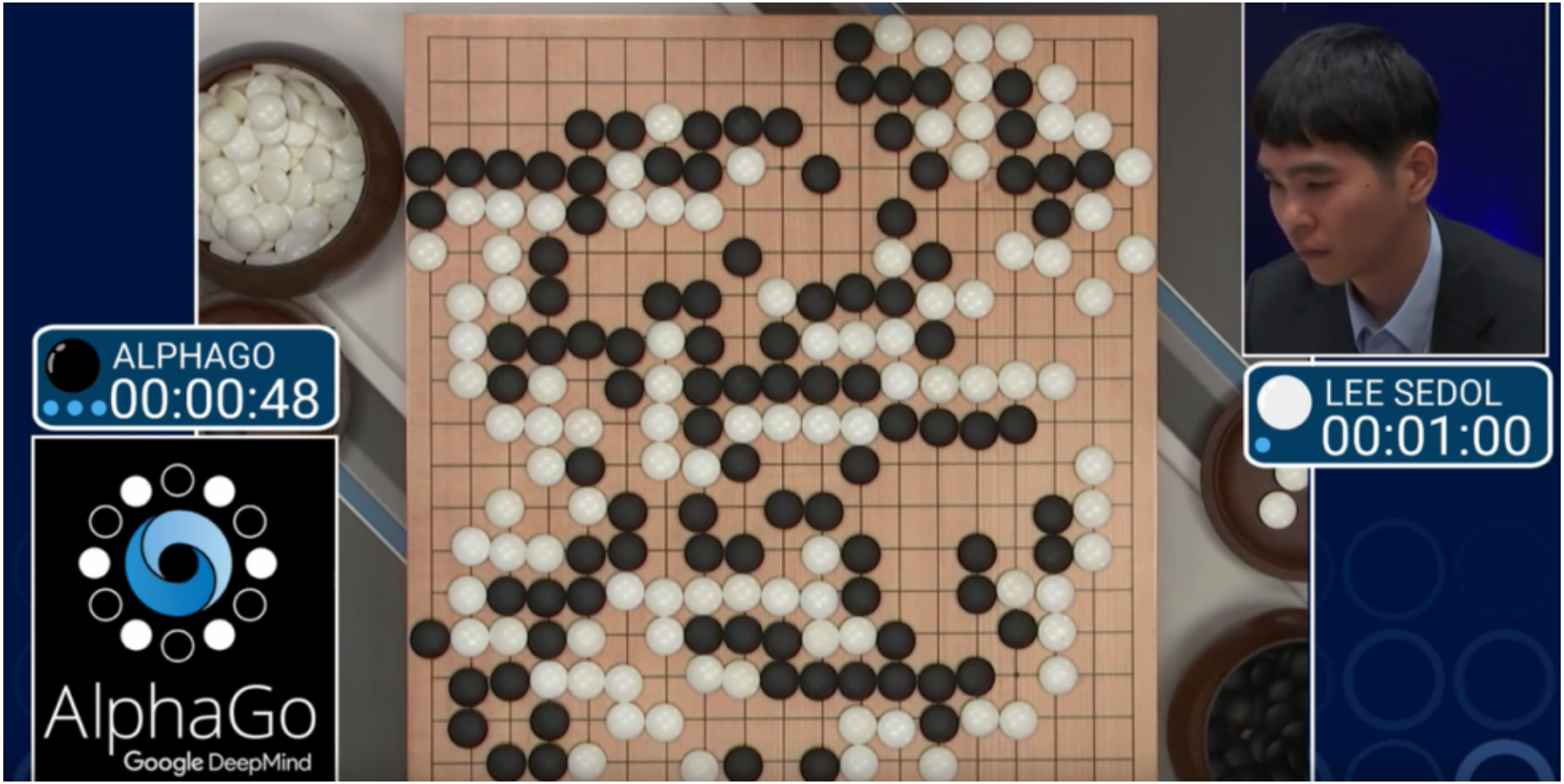
Natural language processing: machine translation, sentiment analysis, topic modelling.

Real world example: The New York Times

LDA analysis of 1.8M New York Times articles:

music band songs rock album jazz pop song singer night	book life novel story books man stories love children family	art museum show exhibition artist artists paintings painting century works	game knicks nets points team season play games night coach	show film television movie series says life man character know
theater play production show stage street broadway director musical directed	clinton bush campaign gore political republican dole presidential senator house	stock market percent fund investors funds companies stocks investment trading	restaurant sauce menu food dishes street dining dinner chicken served	budget tax governor county mayor billion taxes plan legislature fiscal

An example of Reinforcement Learning



DOTA2 - ▸ Link

Introduction to Supervised Learning

Today (and for much of this course) we focus on supervised learning: where we are given

- training set $\{(y_1, x_1), \ldots, (y_n, x_n)\}$ consisting of n samples of
  - features $x_1, \ldots, x_n \in \mathcal{X}$ and corresponding
  - outcome $y_1, \ldots, y_n \in \mathcal{Y}$.

Our goal is to learn a predictor (function / mapping) g: $\mathcal{X} \to \mathcal{Y}$ such that

- for a new test data $x_* \in \mathcal{X}$,

$$g(x_*) \approx y_*.$$

What is the intuition?

Introduction to Supervised Learning

Mathematically, write the underlying generating mechanism between Y and X as

$$Y = f(X) + \epsilon$$

where ϵ represents some measurement errors and other discrepancies.

We are given the training data $\mathcal{D}^{train}$ consisting of n i.i.d. samples of (X, Y) following the above model, that is, $(x_1, y_1), \ldots, (x_n, y_n)$.

Our goal is to estimate / learn the mapping / function f based on $\mathcal{D}^{train}$.

Why estimate f?

- Prediction: Given a new point $X = x$, $f(x)$ is typically a good prediction, and it is in fact the best prediction one can hope for with respect to certan criterion.

- Feature selection: Understand which components of $X = (X_1, X_2, ..., X_p)$ are important / irrelevant in explaining Y. E.g.

$$f(X_1, X_2, X_3) = 0.5 + 4X_1 + X_1^2 - 2X_2^3.$$

A Textbook Example: Sales prediction and inference

We are given information of n products: each of them consists

- y_i: **Sales** of product i in 200 different markets
- x_{i1}: **TV** budget of product i
- x_{i2}: **radio** budget of product i
- x_{i3}: **newspaper** budget of product i

Suppose the true generating model is

$$Y = f(X) + \epsilon = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \epsilon.$$

Knowing f helps to understand how **Sales** changes if one increases one unit of **TV** budget (X_1).

How to estimate f?

Notation:

- Let x_{ij} denote the value of the jth feature for observation i, where $i = 1, 2, ..., n$ and $j = 1, 2, ..., p$.
- Let y_i denote the response variable for the ith observation.
- Training data consist of $\mathcal{D}^{train} = \{(x_1, y_1), ..., (x_n, y_n)\}$, where $x_i = (x_{i1}, ..., x_{ip})^T$.

Two categories of approaches to estimate f based on $\mathcal{D}^{train}$:

- **Parametric method**
- **Non-parametric method**

Parametric approach

Assume parametric form of f, i.e. assuming f is a function of certain parameters.

Example (Linear model / predictor)

The linear model is an important example of a parametric model:

$$Y = f(X) + \epsilon, \quad \text{with} \quad f(X) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p.$$

Correspondingly, we would estimate f by

$$\hat{f}_{linear}(X) = \hat{\beta}_0 + \hat{\beta}_1 X_1 + \cdots + \hat{\beta}_p X_p.$$

Remark: constructing $\hat{f}_{linear}$ reduces the problem of estimating a function to that of estimating $(p+1)$ parameters $(\beta_0, \beta_1, \ldots, \beta_p)$.

More on linear model

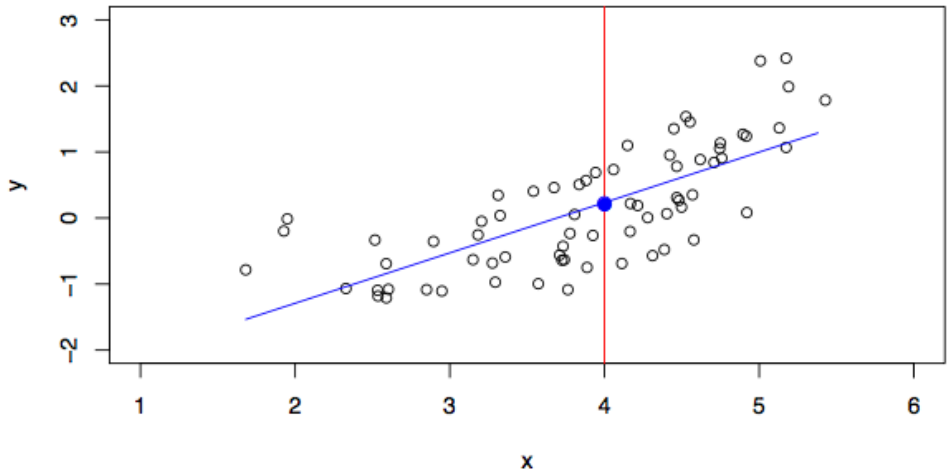
$$\hat{f}_{linear}(X) = \hat{\beta}_0 + \hat{\beta}_1 X_1 + \cdots + \hat{\beta}_p X_p.$$

- The procedure of obtaining $\hat{\beta}_0, \ldots, \hat{\beta}_p$ is called **fitting**, i.e. fitting the model to the training data.

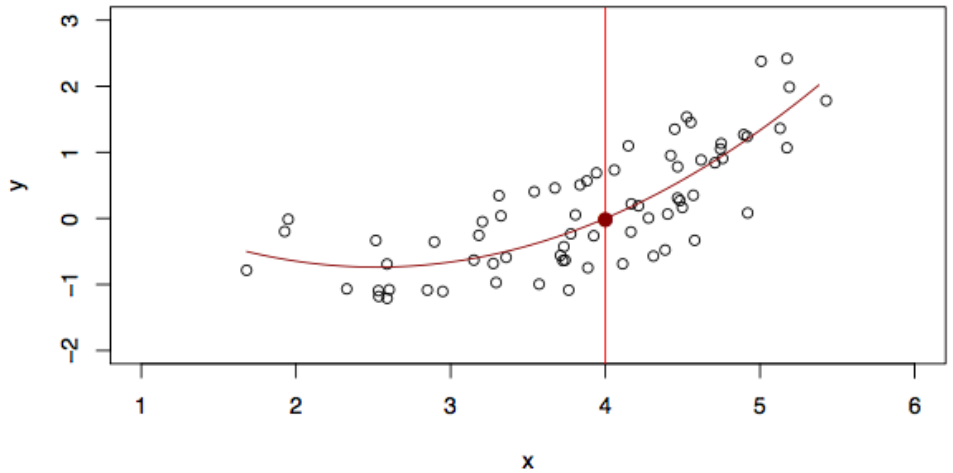
- Even for the same parametric model, there are different ways of fitting (algorithms)! We will get back to this later.

A toy example of linear predictor

A fitted linear model $\hat{f}_{linear}(X) = \hat{\beta}_0 + \hat{\beta}_1 X$ via the least squares approach.



A more flexible model $\hat{f}_{quad}(X) = \hat{\beta}_0 + \hat{\beta}_1 X + \hat{\beta}_2 X^2$ gives a slightly better fit



More (complex) parametric methods

As we have seen, if the linear predictor does a poor job for fitting the data, one can consider more complex forms of f, such as:

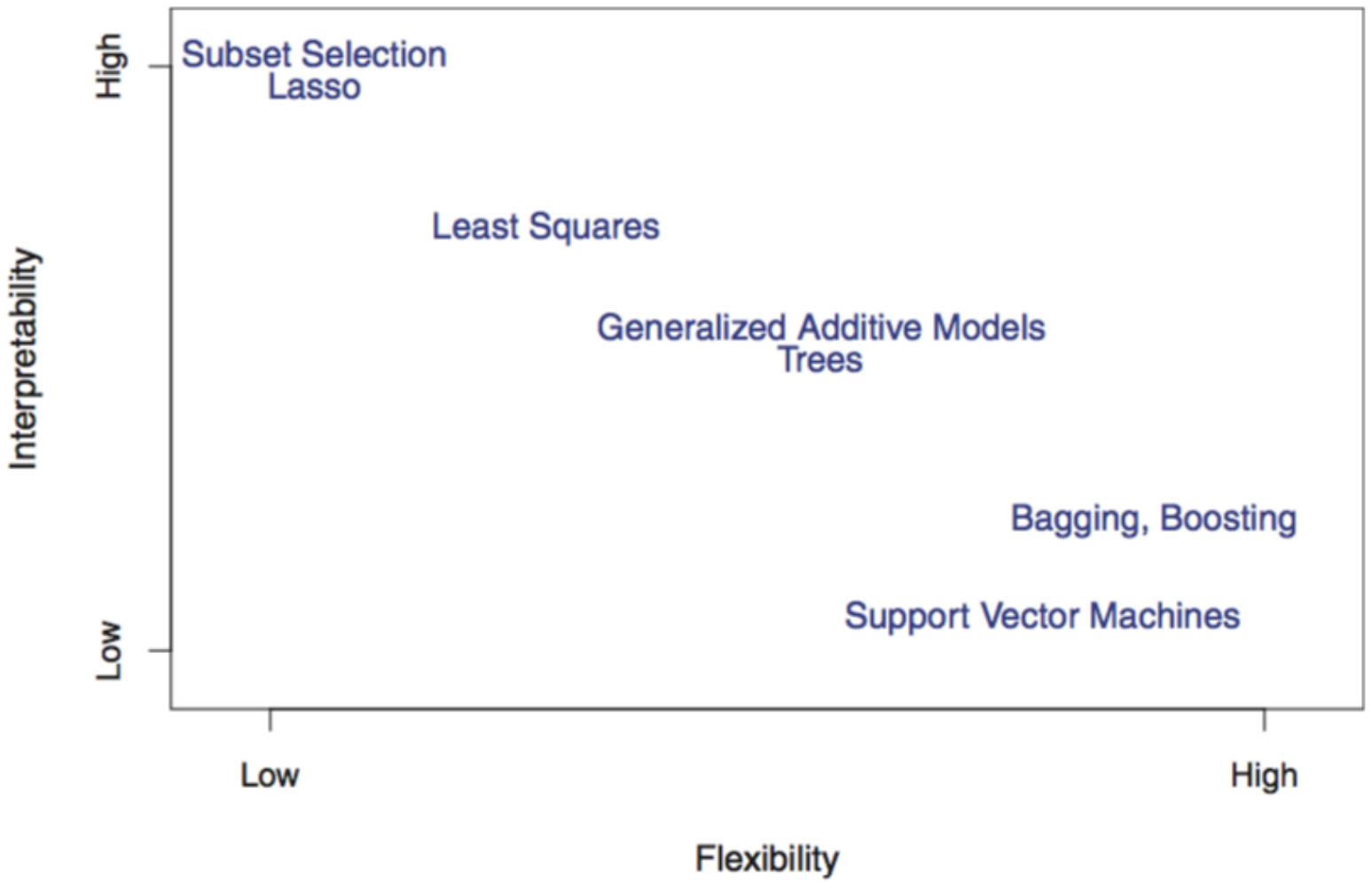
- Quadratic form: $f(X) = \beta_0 + \beta_1 X + \beta_2 X^2$
- Step-wise form: $f(X) = \beta_0 + \beta_1 1\{X <= 0.5\} + \beta_2 1\{X > 0.5\}$.
- Polynomial form: $f(X) = \beta_0 + \beta_1 X + \beta_2 X^2 + \cdots + \beta_d X^d$
- Two layer neural net: $f(X) = \sigma\left(W_1 \sigma(W_0 X + b_0) + b_1\right)$
- You can keep adding complexity by considering more complicated f

Remark: f gets less interpretable as its form gets more complicated!

Trade-off between complexity and interpretability

Model complexity (fitting flexibility) versus interpretability.

- The more complex, the more flexible to fit f, but less interpretable.
- Linear models are easy to interpret; neural nets are not.
- Interpretation means to understand how the predictors X contribute to predicting Y.


Subset Selection
Lasso
Least Squares
Generalized Additive Models
Trees
Bagging, Boosting
Support Vector Machines
High
Low
Interpretability
Low
High
Flexibility

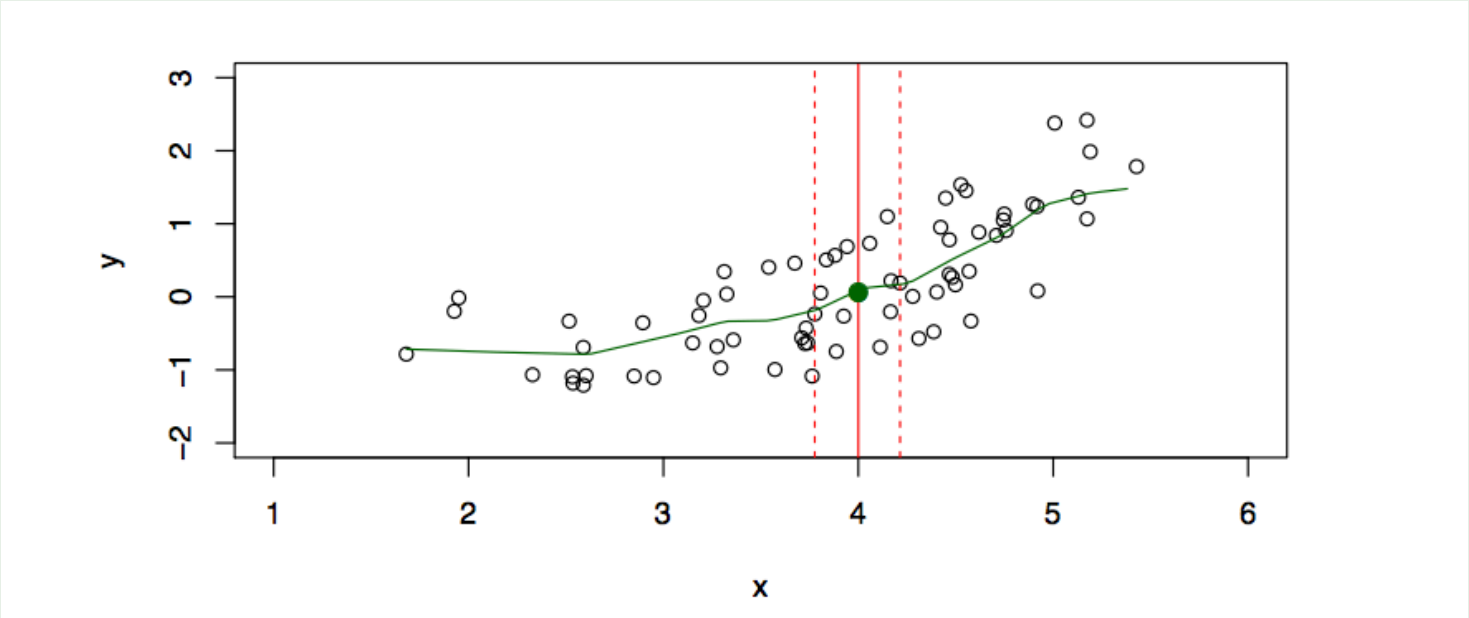
Non-parametric method

Make no / little assumption on f.

Example (Nearest neighbours)

To predict at $X = x$ with $\mathcal{N}(x)$ being some neighborhood of x,

$$\hat{f}(x) = \frac{1}{|\mathcal{N}(x)|} \sum_{i:x_i \in \mathcal{N}(x)} y_i.$$



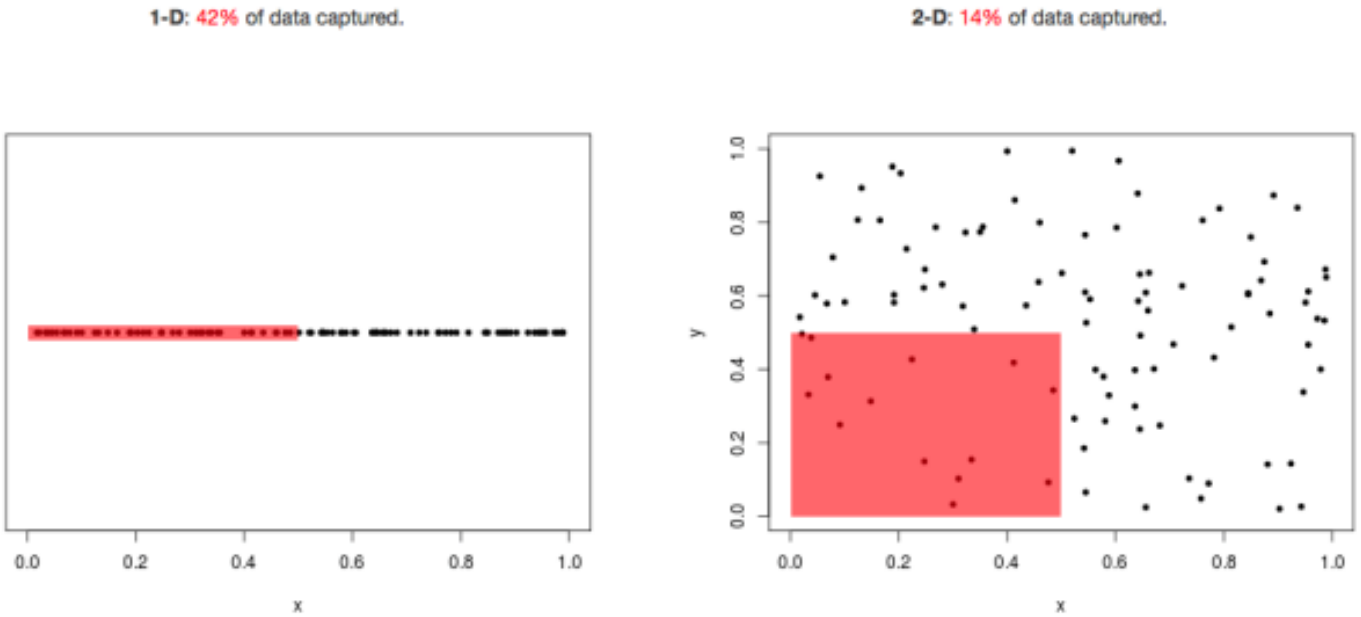
General comments on non-parametric methods

- More sophisticated versions, e.g. kernel estimator, spline estimator.

- **Pros**:
  - little assumption on f
  - good prediction for large n and small p, e.g. $p \leq 4$.

- **Cons**:
  - Poor performance when p is large.
  - **Curse of dimensionality**: There are very few data points in the nearby neighbors when p is large.

Curse of Dimensionality



Model selection

How do we choose the best model?

- parametric vs parametric
- parametric vs non-parametric
- non-parametric vs non-parametric

We need a systematic way of choosing the best $\hat{f}$ among a set of $\hat{f}$'s.

- What is a good metric for evaluating any given $\hat{f}$?

We start with the regression problems where Y is quantatitive.

Metric of $\hat{f}$ for regression problems

Recall the setup:

$$Y = f(X) + \epsilon$$

and $\mathcal{D}^{train}$ contains n i.i.d. samples of (X, Y).

Given any $\hat{f}$, ideally, we want to evaluate $\hat{f}$ by the expected **mean squared error** (MSE)

$$\mathbb{E}\left[\left(Y_* - \hat{f}(X_*)\right)^2\right]$$

where

- (X_*, Y_*) is a new random pair that is independent of $\mathcal{D}^{train}$.
- the expectation is taken w.r.t. the random pair (X_*, Y_*) as well as the randomness in $\hat{f}$.

Metric of $\hat{f}$

We cannot compute the expected MSE as we do not know either the distribution of (X_*, Y_*) or that of $\mathcal{D}^{train}$.

One natural option is to use $\mathcal{D}^{train}$ to approximate the expectation by

$$MSE(\hat{f}) := \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{f}(x_i)\right)^2.$$

This is called the **training MSE** as it uses $\mathcal{D}^{train}$.

- However, it is **NOT** a valid metric of the fit for $\hat{f}$.
- It always favors more complex $\hat{f}$'s.

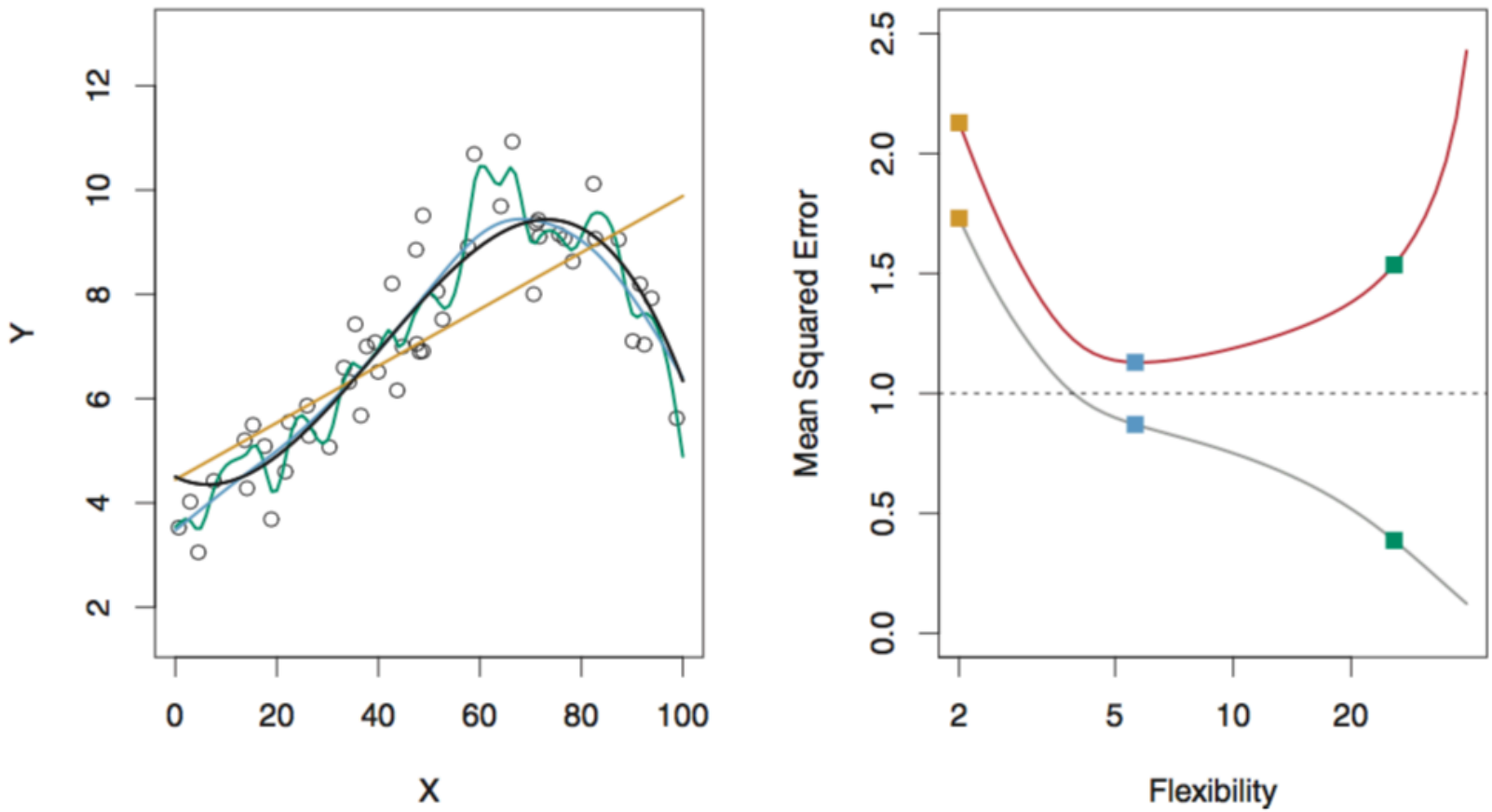
Metric of $\hat{f}$

- **Test data** refers to the data which is not used to train the statistical model (i.e., not used to compute $\hat{f}$).
- **Test MSE**. Suppose we have the test data $\mathcal{D}_{test}$ containing $\{(x_{T1}, y_{T1}), ..., (x_{Tm}, y_{Tm})\}$

$$MSE_T(\hat{f}) = \frac{1}{m}\sum_{i=1}^{m}(y_{Ti} - \hat{f}(x_{Ti}))^2.$$

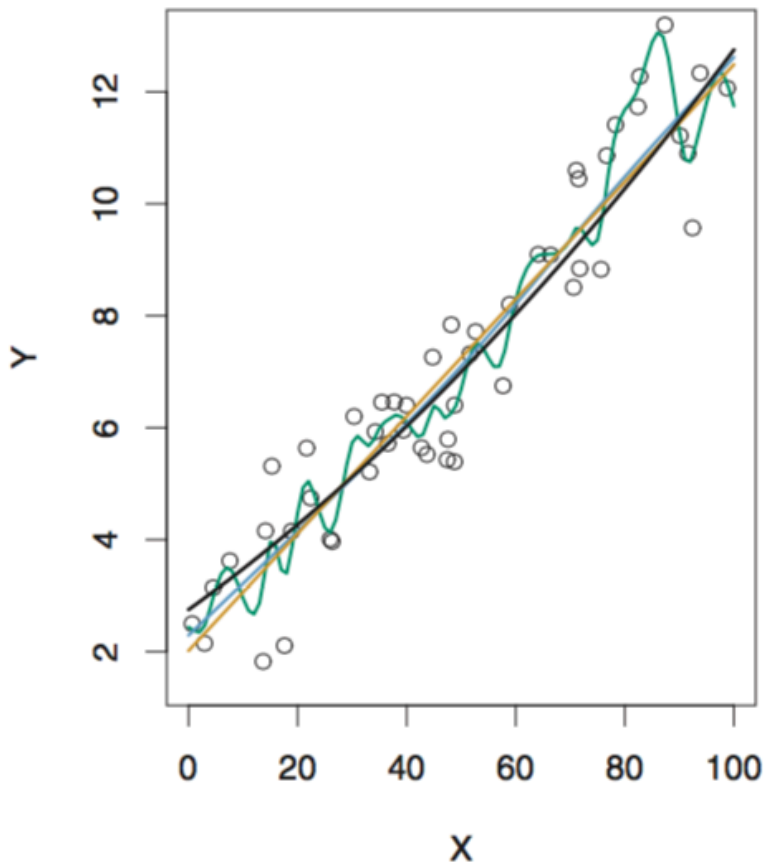
- Instead of using the training MSE, we should look at the test MSE. We'd like to select the model which yields the smallest test MSE.
- How to calculate $MSE_T(\hat{f})$?
  - If test data is available, we can directly compute $MSE_T(\hat{f})$.
  - Otherwise, we use a resampling technique called *cross-validation* (later in Lecture 3).

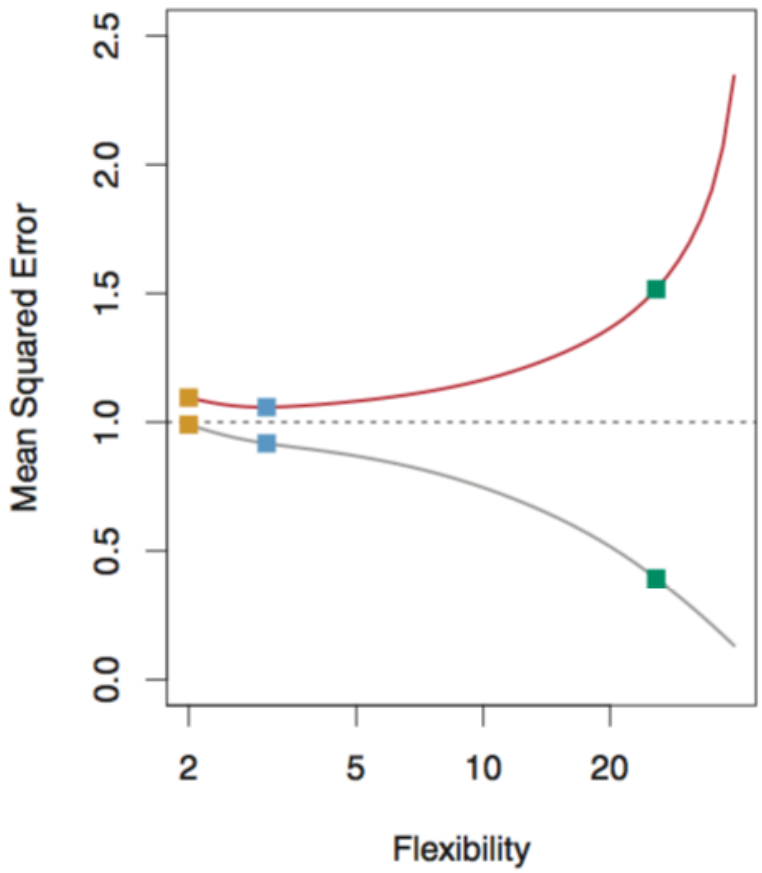
Training MSE vs Test MSE



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

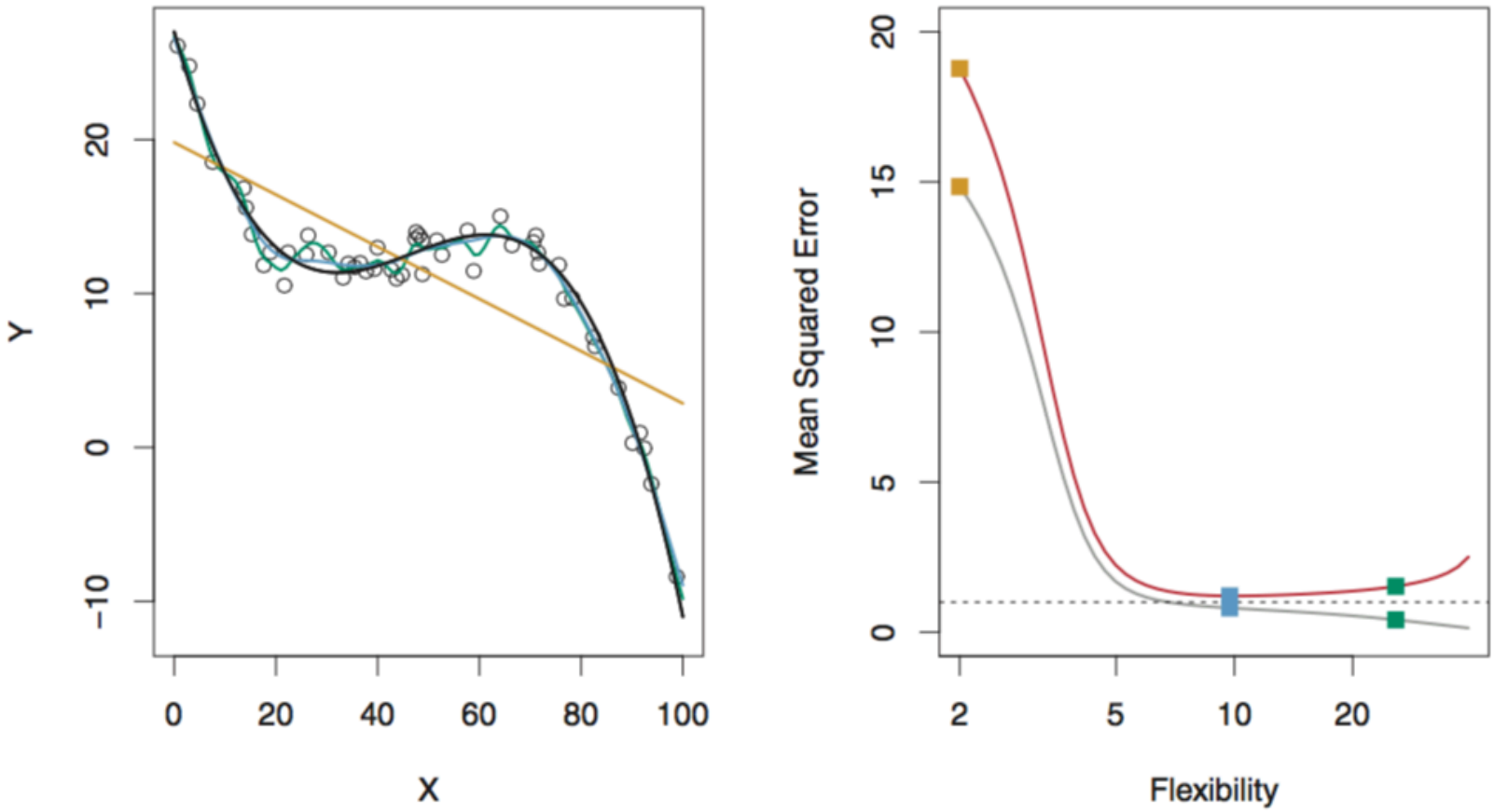
Training MSE vs Test MSE: a linear f





When f is close to linear, the linear predictor provides a very good fit to the data.

Training MSE vs Test MSE: a highly non-linear f



When f is highly non-linear, the linear predictor provides a very poor fit to the data.

Overfitting

Training MSE

$$MSE(\hat{f}) := \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{f}(x_i)\right)^2.$$

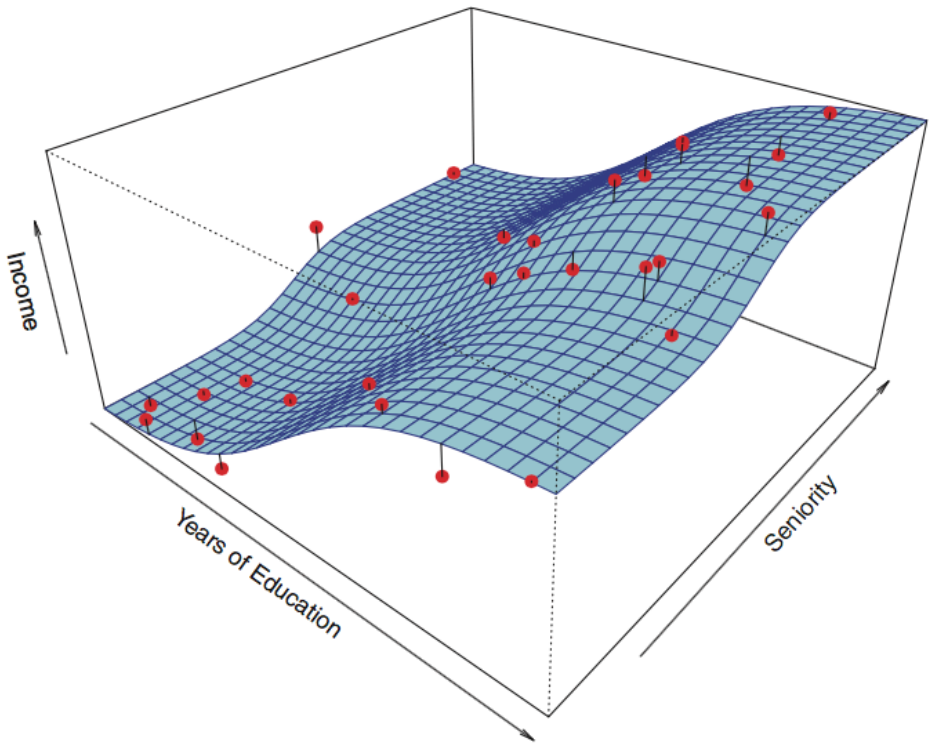
decreases as $\hat{f}$ gets more complex.

- $\hat{f}$ is typically obtained by minimizing $MSE(g)$ over all possible g in the specified model class.

A highly complex $\hat{f}$ can lead to a phenomenon known as **overfitting** the data, which essentially means it follows the noise ϵ too closely.

- A simple example of overfitting: $\hat{f}(x_i) = y_i$ for all $1 \leq i \leq n$.

Simulated example

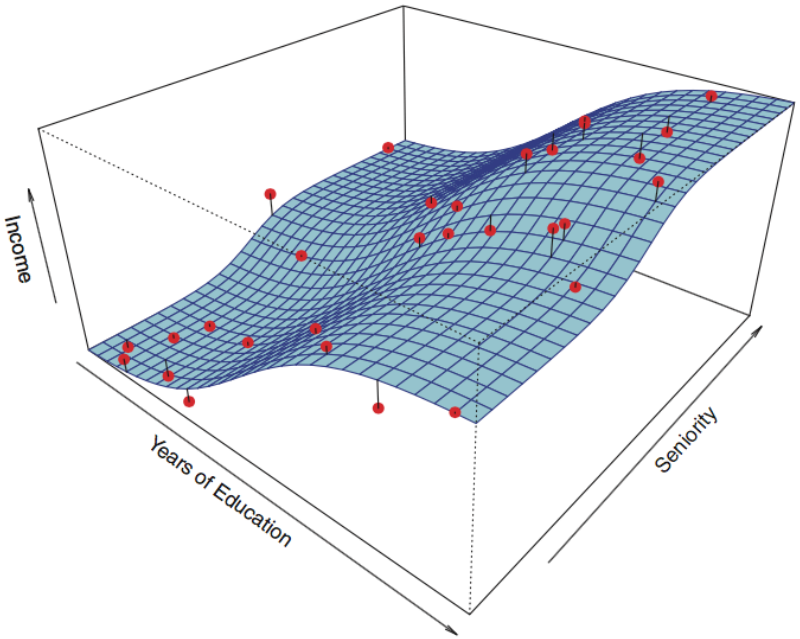


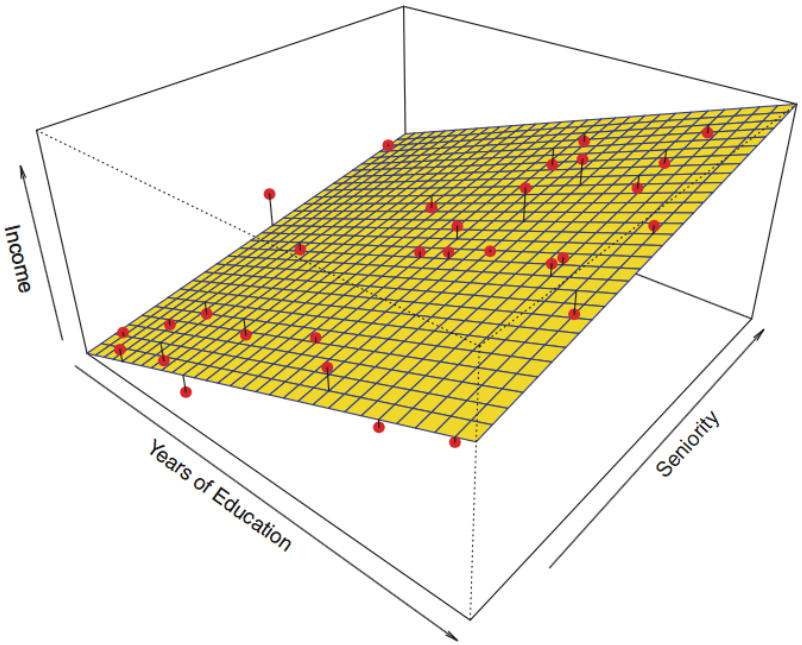
Red points are simulated values for income from the model

$$income = f(education, seniority) + \epsilon$$

where f is the blue surface.

Example



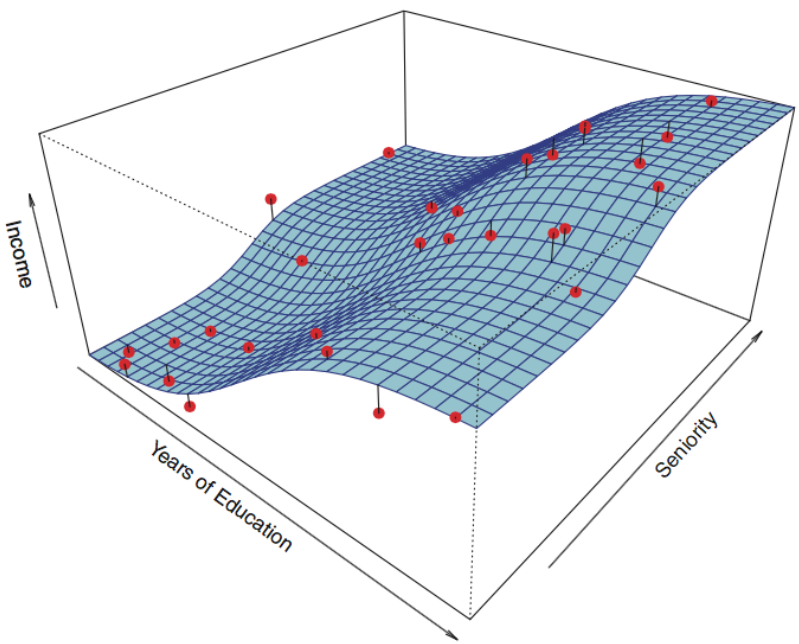


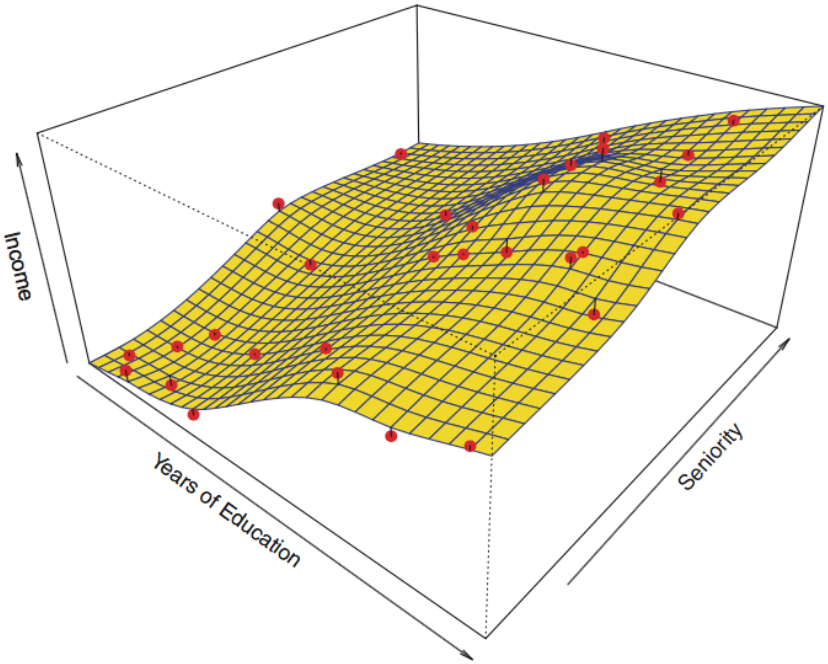
Linear regression model

$$\hat{f}_{linear}(education, seniority) = \hat{\beta}_0 + \hat{\beta}_1 \times education + \hat{\beta}_2 \times seniority$$

Underfit the training, poor prediction.

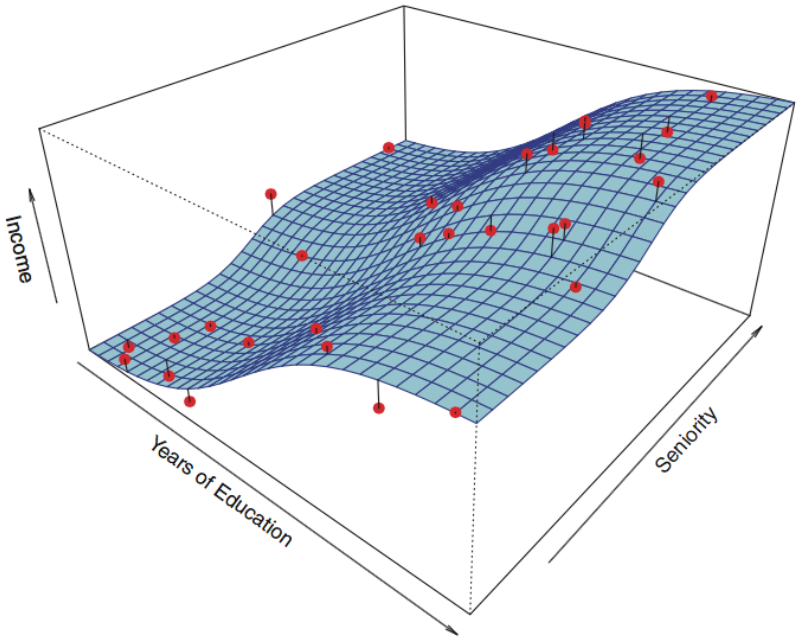
Example

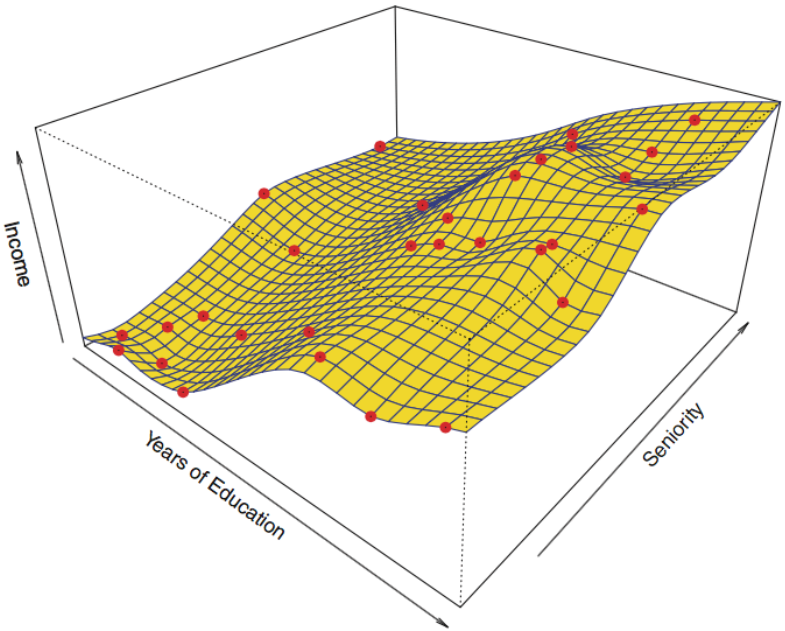




A more flexible model (good nonparametric model).

Example





A even more flexible nonparametric model.

Zero error on the training data! An indicator for **overfitting**.

Test MSE

- Tradeoff between the test MSE and the flexibility (complexity) of the fitted model $\hat{f}$.
- Question: Is there a universal rule or explanation about this?

Bias-Variance decomposition

Let (X_*, Y_*) be a new random pair (independent from $\mathcal{D}^{train}$) following $Y_* = f(X_*) + \epsilon_*$ with $\mathbb{E}[\epsilon_*] = 0$.

For any estimator $\hat{f}$ (obtained from $\mathcal{D}^{train}$), its conditional **expected MSE** at any $X_* = x_*$ is

$$\begin{aligned}
&\mathbb{E}\left[\left(Y_* - \hat{f}(X_*)\right)^2 | X_* = x_*\right] \\
&= \underbrace{\mathrm{Var}(\hat{f}(x_*))}_{Variance} + \left(\underbrace{\mathbb{E}[\hat{f}(x_*)] - f(x_*)}_{Bias}\right)^2 + \underbrace{\mathrm{Var}(\epsilon_*)}_{Irreducible\ \ error} \\
&\geq \mathrm{Var}(\epsilon_*)
\end{aligned}$$

- The first expectation is over ϵ_* as well as $\mathcal{D}^{train}$.
- The expected MSE $\geq$ the Irreducible error.
- An ideal $\hat{f}$ should minimize the expected MSE.

What is the Bias-Variance Trade-off?

Variance: how much $\hat{f}$ would change if we estimated it using a different training data set.

Bias: refers to the error that is introduced by parametrizing f.

E.g., the real relationship between response and predictors is nonlinear

$$f(X) = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3,$$

but we fit a linear model

$$\hat{f}(X) = \hat{\beta}_0 + \hat{\beta}_1 X.$$

This causes a bias in $\mathbb{E}[\hat{f}(x_*)] - f(x_*)$ at $X_* = x_*$.

What is the Bias-Variance Trade-off?

- As the complexity (a.k.a. flexibility) of $\hat{f}$ increases (e.g., linear method $\rightarrow$ non-parametric methods), the variance of $\hat{f}$ typically increases whereas its bias decreases.
  - The variance of any fitted model $\hat{f}$ also depends on the sample size (n), and is roughly proportional to

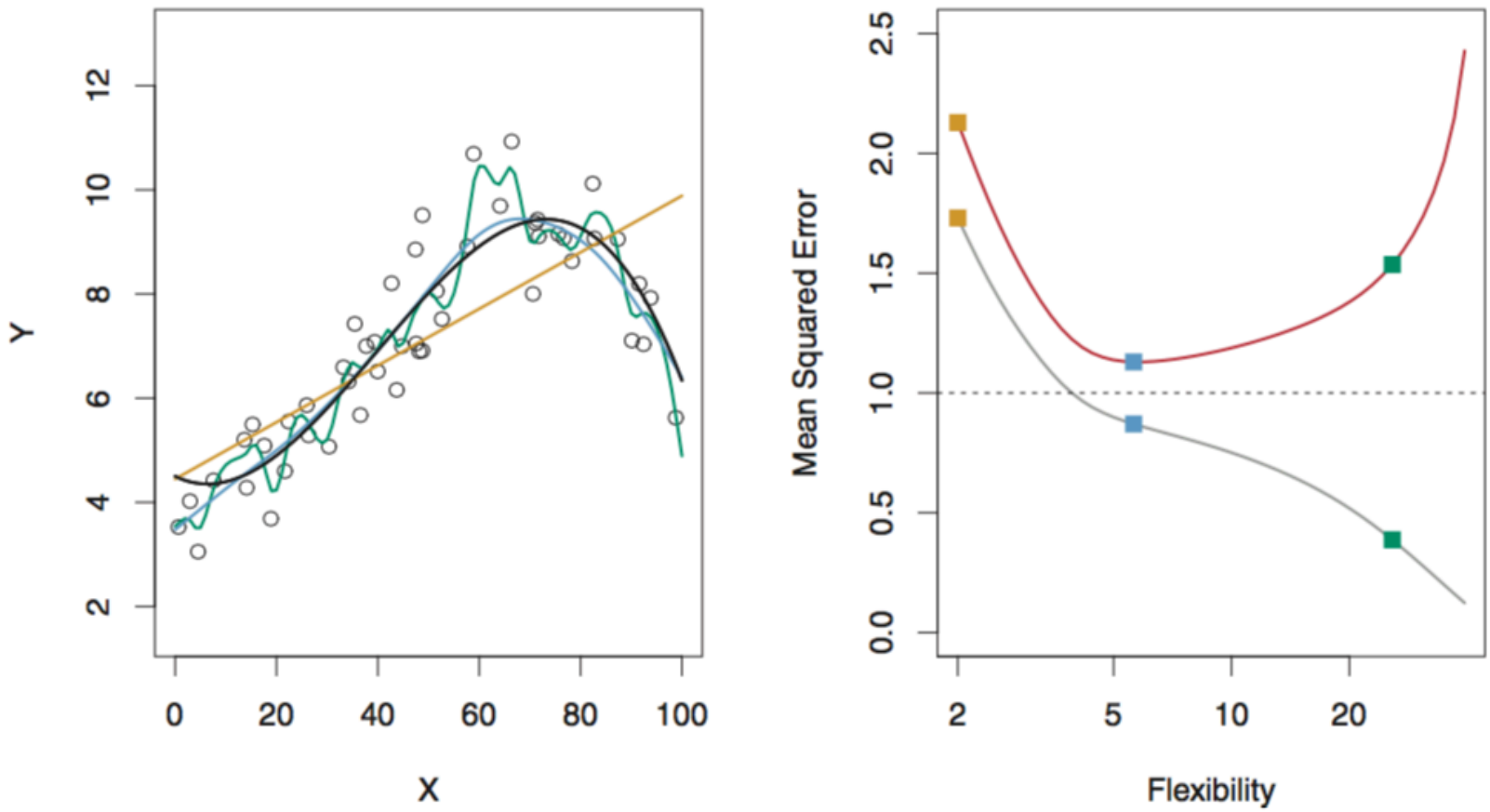
$$\frac{\text{complexity of } \hat{f}}{n}.$$

  - When n is small, a fitted model with high complexity performs poorly due to large variance.
  - When n is large enough, a more complex fitted model tends to peform better as they have smaller bias than simpler models.

More on the tradeoff

- So choosing the complexity of $\hat{f}$ based on the expected MSE has a bias-variance trade-off.
- When two $\hat{f}_1$ and $\hat{f}_2$ have similar expected MSEs, we usually prefer the more parsimonious (less complex) one.

Example

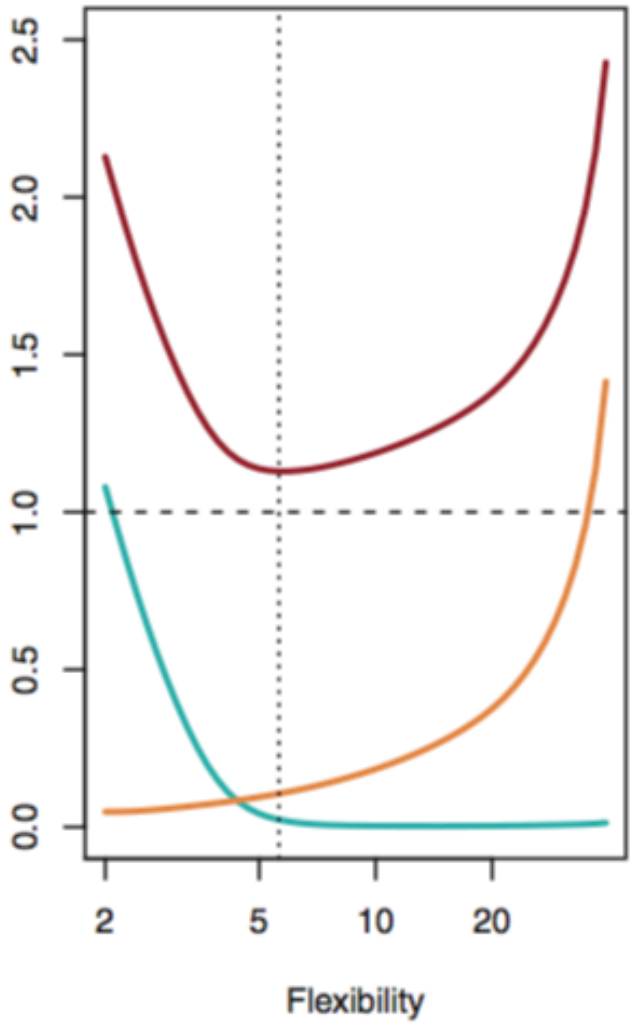


- Left: Data simulated from f, shown in black. Three estimates of f are shown: the linear regression line (orange curve), and two nonparametric fits (blue and green curves).
- Right: Training MSE (grey curve), test MSE (red curve), and minimum test MSE over all possible methods (dashed line).
  Squares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

Example

- red curve: the test MSE.
- blue curve: $(\mathbb{E}[\hat{f}(x)] - f(x))^2$
- orange curve: $\mathrm{Var}(\hat{f}(x))$

- dashed horizontal line: $\mathrm{Var}(\epsilon)$
- dotted vertical line: the best flexibility corresponding to the smallest test MSE.



- There are alternative metrics for measuring $\hat{f}$, such as the Sum of Absolute Difference (SAM):

$$\mathbb{E}[|Y - \hat{f}(X)|], \qquad \frac{1}{n}\sum_{i=1}^{n}\left|y_i - \hat{f}(x_i)\right|.$$

- Both MSE and SAM are only appropriate for quantatitive Y!

- What about categorical or ordinal Y?
  - Spam email detection: $Y = 0$ for non-spam, $Y = 1$ for spam
  - Hand-written digit recoginition: $Y \in \{0, 1, ..., 9\}$

Metric of $\hat{f}$ for classification

When Y is categorical or ordinal, the **expected error rate** is defined as

$$\mathbb{E}\Big[1\{Y \neq \hat{f}(X)\}\Big].^{1}$$

Analogously, the **training error rate** is

$$\frac{1}{n}\sum_{i=1}^{n} 1\left\{y_i \neq \hat{f}(x_i)\right\}$$

and the **test error rate** is

$$\frac{1}{m}\sum_{i=1}^{m} 1\left\{y_{Ti} \neq \hat{f}(x_{Ti})\right\}.$$

Of course, there also exists other metrics that can be used when Y is categorical or ordinal.

[1] $1\{\}$ is the indicator function. $1\{\text{A}\} = 1$ if A is true and $1\{\text{A}\} = 0$ otherwise.

Summary on the metrics of the fit

- Metrics:
  - In regression problems, we have the expected MSE, the training MSE and the test MSE.
  - In classification problems, we have the expected error rate, the training error rate and the test error rate.
- Model selection:
  - The best model yields the smallest expected (test) MSE (error rate).
  - Among models that have similar expected MSE (error rate), we always prefer the more parsimonious one.
- Bias and variance trade-off:
  - A more complex / flexible $\hat{f}$ has smaller bias but larger variance

- In practice, how should we compute the expected MSE to select the best $\hat{f}$ when we do not have test data?
  (We will come back to this later in Lecture 3).

- What's next?
  - Different algorithms of computing $\hat{f}$.....

Questions?