

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

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 $(x_1, y_1), \dots, (x_n, y_n)$. 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.

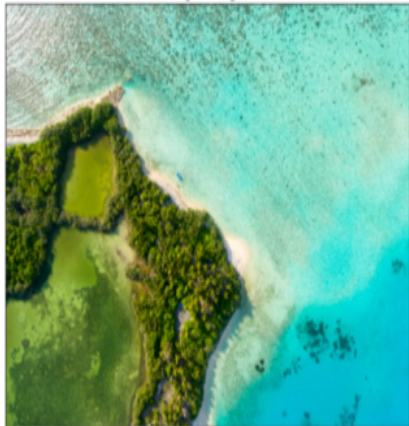
Unsupervised Learning

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.

Unsupervised learning example

Original Image



Segmented Image when $K = 3$



Introduction to Supervised Learning

- Today (and for much of this course) we focus on **supervised learning**.
- For many tasks of interest we are given some data consisting of features X and outcome Y , and we are interested in predicting some unseen Y by using its corresponding X .

Task	Input data (X, Y)	Y to predict from a new X
object recognition	image + object	object category
image captioning	image + caption	caption
document classification	text	document category
speech-to-text	audio waveform	text
⋮	⋮	⋮

- Supervised learning is applicable when we have outcome instances, i.e., we can supervise the learner by telling it exactly what to predict.

Introduction to Supervised Learning

More precisely, in supervised learning, we are given

- **training set** consisting of n samples of
 - ▶ **features** X and corresponding
 - ▶ **outcome** Y .

Our goal is to **learn the underlying generating mechanism from $X \rightarrow Y$** .

Introduction to Supervised Learning

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

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

where ϵ captures 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, $(x_1, y_1), \dots, (x_n, y_n)$.

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

Why estimate f ?

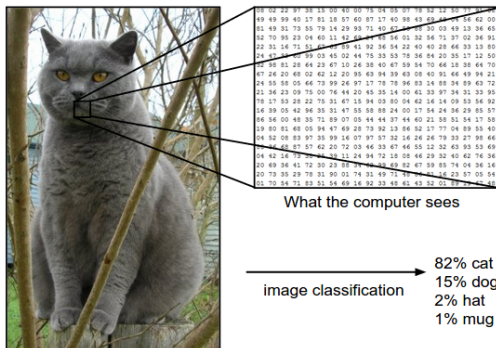
- **Prediction:** Given a new point $X = x$, $f(x)$ is typically a good prediction, and it is in fact the best prediction one can hope for with respect to certain criterion.
- **Inference:**
 - ▶ We can understand which components of $X = (X_1, X_2, \dots, X_p)$ are important in explaining Y , and which are irrelevant.
 - ▶ Depending on the complexity of f , we may be able to understand how each component X_j of X affects Y .

Example 1: image recognition

For image data, we observe n images with annotated labels

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

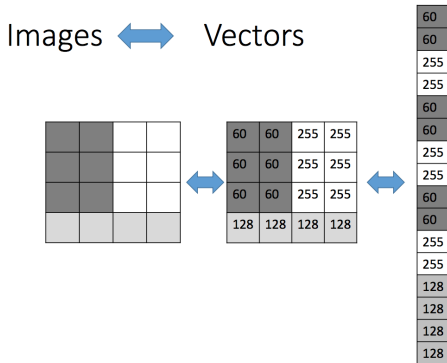
What an image looks like to the computer:



[Image credit: Andrej Karpathy]

Example 1: image recognition

Predictors 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 unannotated images.

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

We would like to estimate f based on \mathcal{D}^{train} .

Two different approaches:

- **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

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

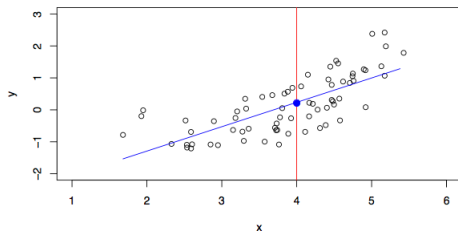
Remark: constructing \hat{f}_{linear} reduces the problem of estimating a function to that of estimating $(p + 1)$ parameters $(\beta_0, \beta_1, \dots, \beta_p)$.

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

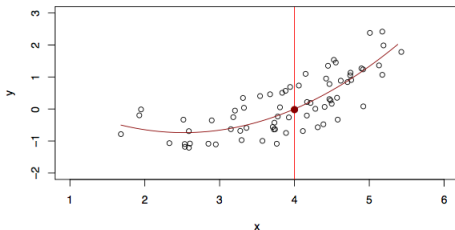
- The procedure of obtaining $\hat{\beta}_0, \dots, \hat{\beta}_p$ is called **fitting**, i.e. fitting the model to the training data.
- There are different ways of fitting (algorithms)! We will get back to this later.
- Although a linear model is almost never correct, it often serves as a good and interpretable approximation to the unknown true function $f(X)$.

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

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

Non-parametric method

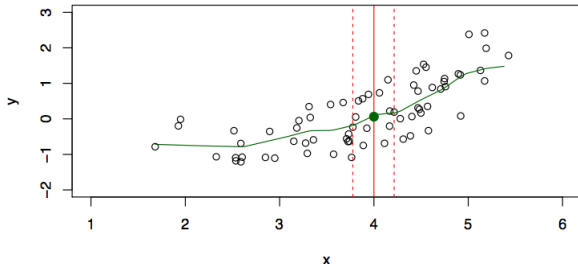
Make no / little assumption on f .

Example (Nearest neighbours)

To predict at $X = x$, consider the local average,

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

where $\mathcal{N}(x)$ is some neighborhood of x .

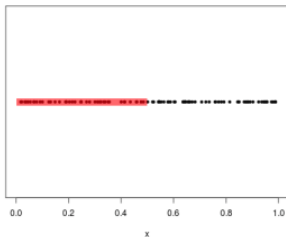


General comments on non-parametric methods

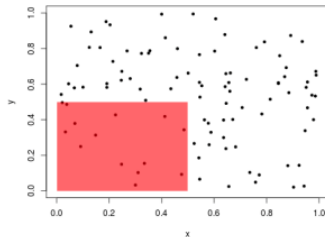
- Nearest neighbor averaging can yield pretty good prediction for small p , e.g. $p \leq 4$ and large n .
- More sophisticated versions, e.g. kernel estimator, spline estimator.
- **Curse of dimensionality:** There are very few data points in the nearby neighbors when the feature dimension, p , is large.

Curse of Dimensionality

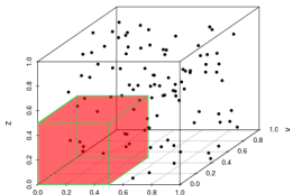
1-D: 42% of data captured.



2-D: 14% of data captured.

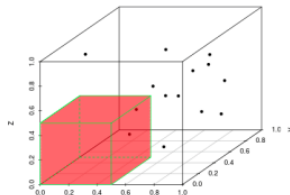


3-D: 7% of data captured.



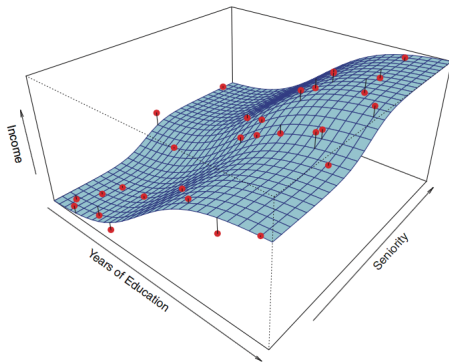
4-D: 3% of data captured.

$t = 0$



- In parametric/nonparametric approach, fitting a more complex model requires estimating more parameters, hence more training data points.
- Those highly complex models can lead to a phenomenon known as **overfitting** the data, which essentially means they follow the errors, or noise, too closely.
- A simple example of overfitting: $\hat{f}(x_i) = y_i$ for all $1 \leq i \leq n$.

Simulated example

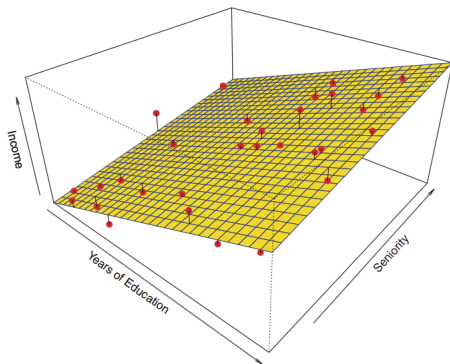


Red points are simulated values for income from the model

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

where f is the blue surface.

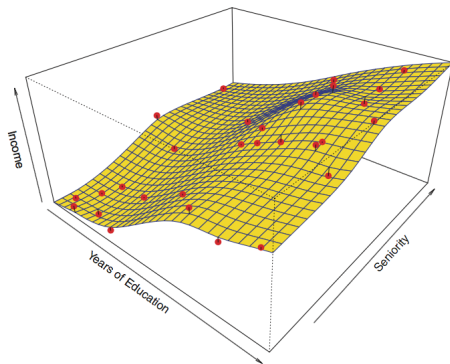
Example



Linear regression model

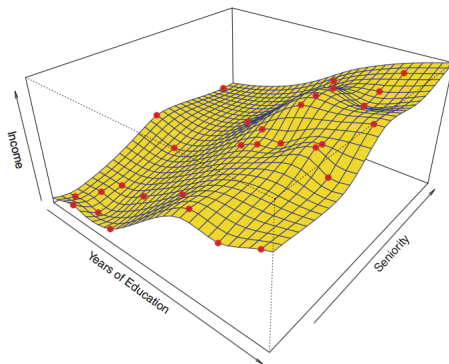
$$\hat{f}_L(\text{education}, \text{seniority}) = \hat{\beta}_0 + \hat{\beta}_1 \times \text{education} + \hat{\beta}_2 \times \text{seniority}$$

Example



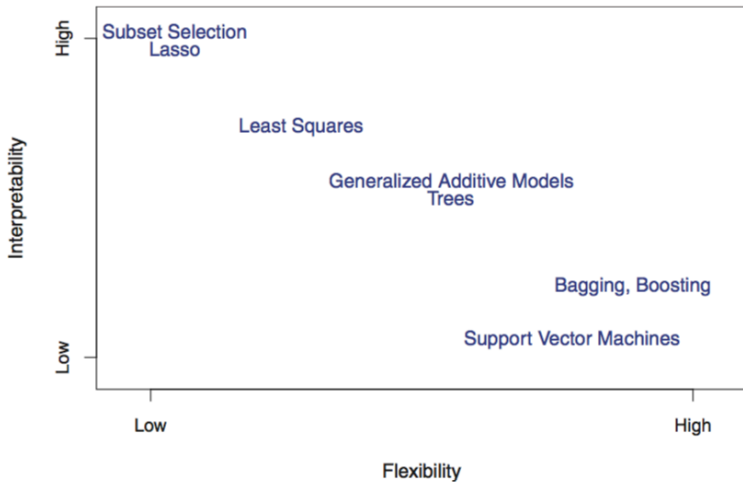
A more flexible model (good nonparametric model).

Example



A even more flexible model (nonparametric model with a bad choice of tuning parameters). This fit makes zero errors on the training data! Also known as **overfitting**. Overfit models tend to have larger variability!

- 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 .



- Parsimony versus black-box
 - ▶ Given the same fit, we often prefer a simpler model involving fewer variables over a black-box predictor involving them all.
- Good fit versus over-fit or under-fit.
 - ▶ How do we know when the fit is just right?
 - ▶ Is there a more 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} ?

Metric of \hat{f} for regression problems

We start with the regression problems where Y is quantitative.

Recall the setup:

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

and we are given \mathcal{D}^{train} consisting of 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 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

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

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} because \hat{f} is typically obtained by using \mathcal{D}^{train} as well.
- In fact, a ubiquitous way of constructing \hat{f} is to minimize $MSE(g)$ over all possible g in certain class.
- Overfitted \hat{f} usually have smaller (or even zero) $MSE(\hat{f})$.

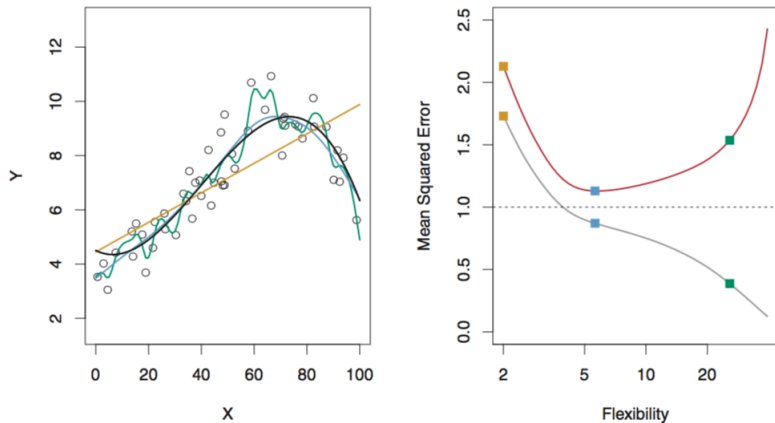
Metric of \hat{f}

- **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 $\{(x_{T1}, y_{T1}), \dots, (x_{Tm}, y_{Tm})\}$

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

- 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).

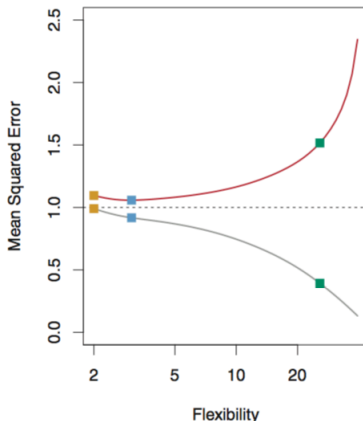
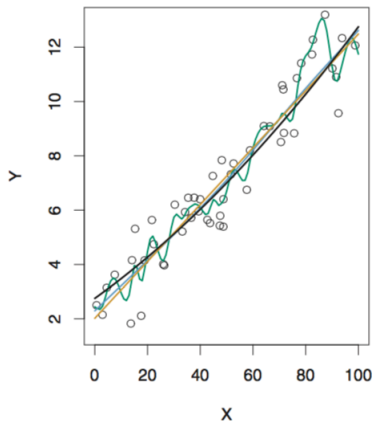
Training MSE vs Test MSE



- 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).

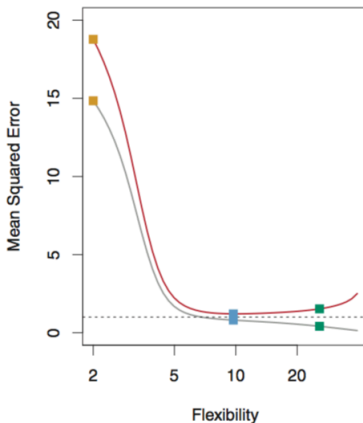
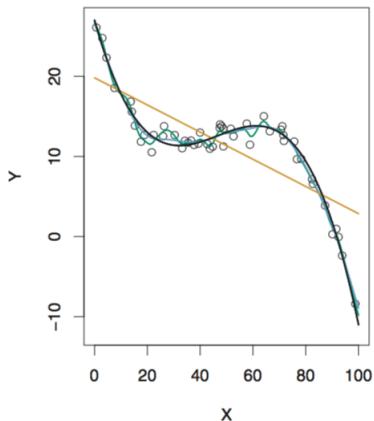
Squares represent the training and test MSEs for the three fits shown in the LHS panel.

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.

- You might have noticed the tradeoff between the test MSE and the flexibility (complexity) of the fitted model \hat{f} .
- Question: Is there a universal rule about this trade-off?

Bias-Variance decomposition

Suppose we have an estimate \hat{f} from \mathcal{D}^{train} . Let (X, Y) be a new random pair (independent from \mathcal{D}^{train}).

Recall that $Y = f(X) + \epsilon$, with $\mathbb{E}[\epsilon|X] = 0$. Then the conditional **expected MSE** at any $X = x$ is

$$\begin{aligned} & \mathbb{E} \left[\left(Y - \hat{f}(X) \right)^2 \mid X = x \right] \\ &= \underbrace{\text{Var}(\hat{f}(x))}_{\text{Variance}} + \underbrace{\left(\mathbb{E}[\hat{f}(x)] - f(x) \right)^2}_{\text{Bias}} + \underbrace{\text{Var}(\epsilon)}_{\text{Irreducible error}} \\ &\geq \text{Var}(\epsilon) \end{aligned}$$

- The expectation is over the variability of $Y|X = x$ 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

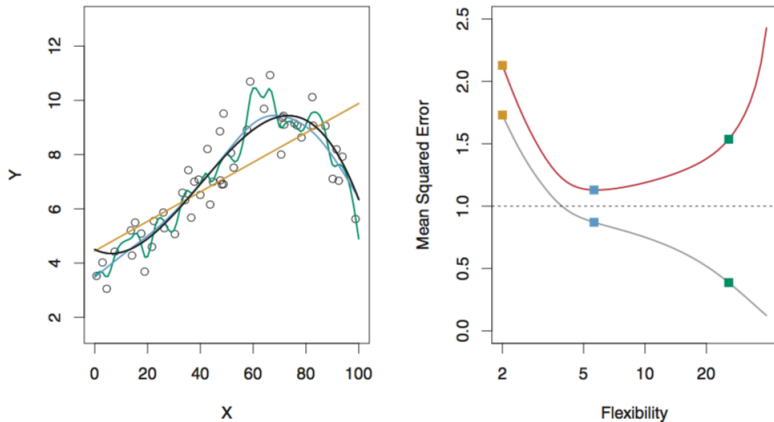
$$\hat{f}(X) = \hat{\beta}_0 + \hat{\beta}_1 X.$$

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.
- On the other hand, if \hat{f} is less complex (e.g., linear model), the variance of \hat{f} is usually small, and its bias is large.
- 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

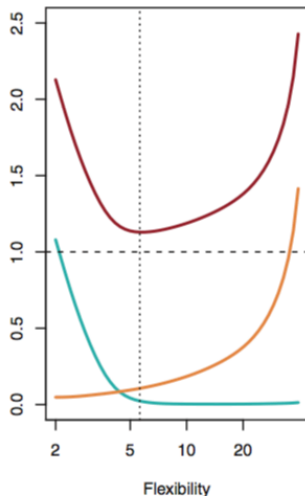


- Left: Data simulated from f , shown in black. Three estimates of f are shown: the linear regression line (orange curve), and two nonparametric fits (blue and green curves).
- Right: Training MSE (grey curve), test MSE (red curve), and minimum test MSE over all possible methods (dashed line).

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**: $\text{Var}(\hat{f}(x))$
- dashed horizontal line: $\text{Var}(\epsilon)$
- 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):

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

- Both MSE and SAM are only appropriate for quantitative Y !
- What about categorical or ordinal Y ?
 - ▶ Spam email detection: $Y = 0$ for non-spam, $Y = 1$ for spam
 - ▶ Hand-written digit recognition: $Y \in \{0, 1, \dots, 9\}$

Metric of \hat{f} for classification

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

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

Analogously, the **training error rate** is

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

and the **test error rate** is

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

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

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

Summary on the metrics of the fit

- 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.
- The best model yields the smallest expected MSE (error rate).
- Bias and variance trade-off exists in both scenarios.
 - ▶ A more complex / flexible \hat{f} has smaller bias but larger variance
- Among models that have similar expected MSE (error rate), we always prefer the more parsimonious one.

- We have learned the bias-variance-tradeoff phenomenon for different \hat{f} 's.
- In practice, how should we compute the expected MSE to select the best \hat{f} ?
(We will come back to this later in Lecture 3).
- What's next?
 - ▶ Different algorithms of computing \hat{f}

Questions?