Introduction

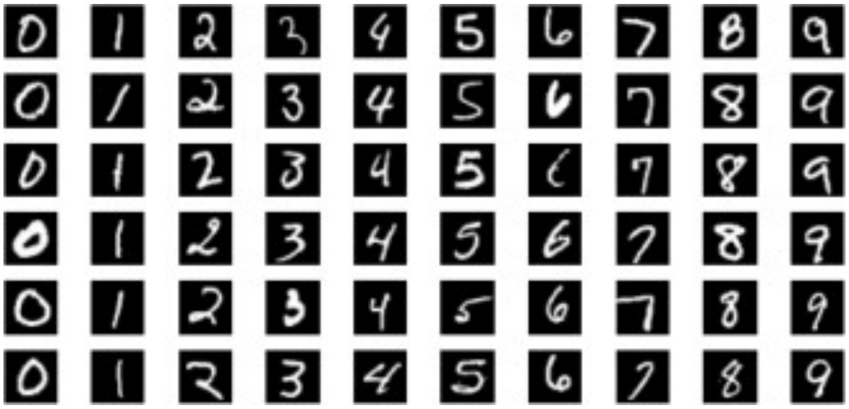
web.stanford.edu/class/stats202

Sergio Bacallado, Jonathan Taylor

Autumn 2022

Prediction challenges

The MNIST dataset is a library of handwritten digits

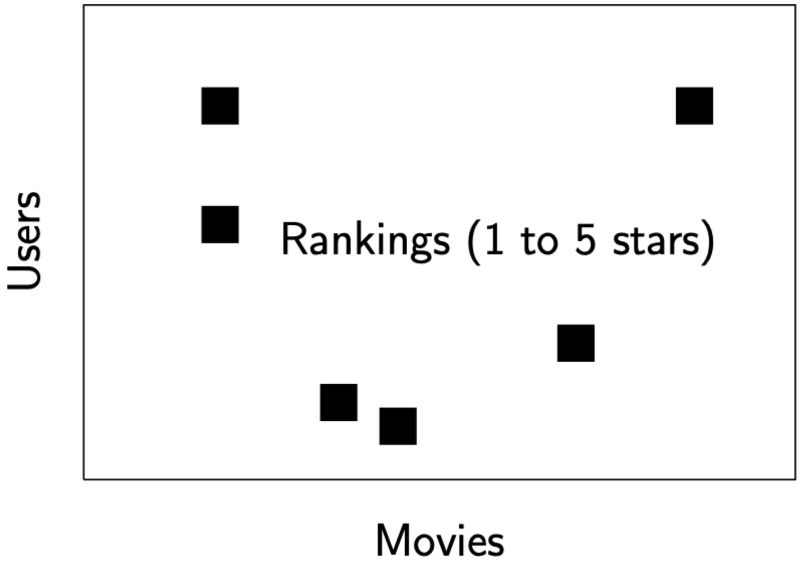


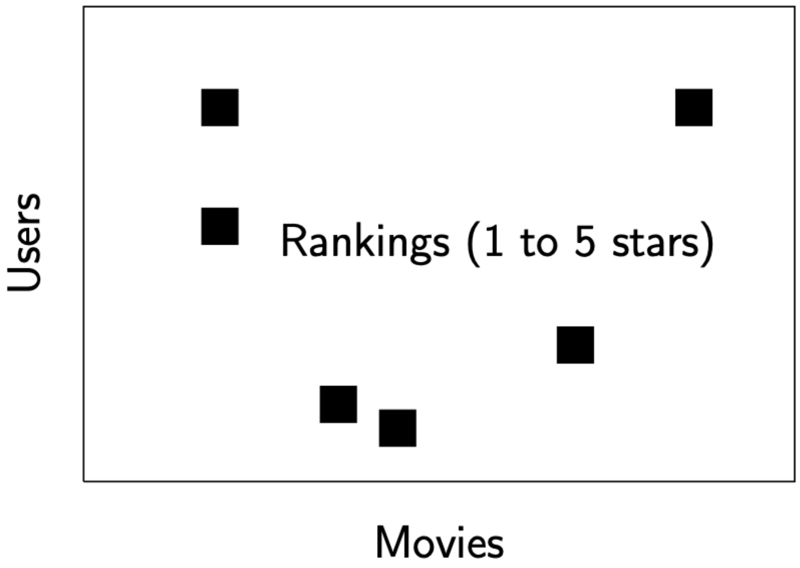
MNIST dataset

- A lot of the *coolest* applications of statistical and machine learning are prediction challenges.
- In a prediction challenge, you are given a training set of images of handwritten digits, which are labeled from 0 to 9.
- You are also given a test set of handwritten digits, which are not identified.
- Your job is to assign a digit to each image in the test set.

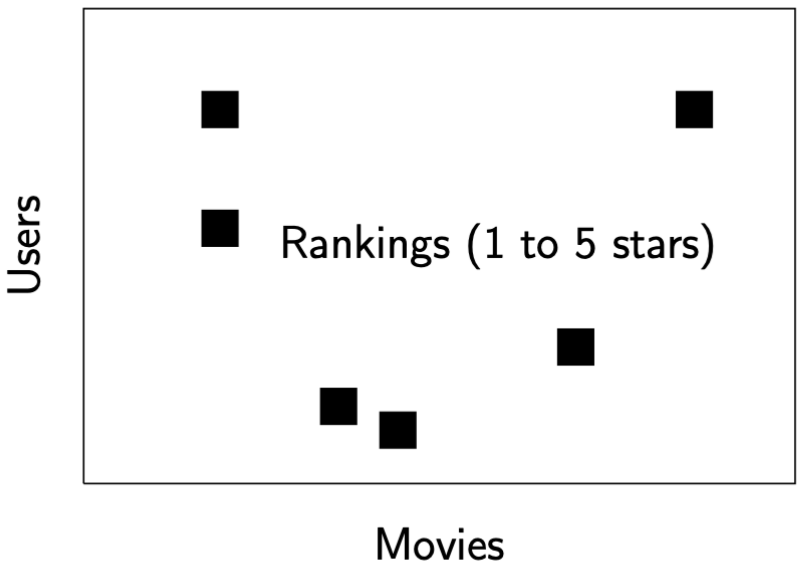
The Netflix prize

Netflix popularized prediction challenges by organizing an open, blind contest to improve its recommendation system.





Some rankings were hidden in the training data



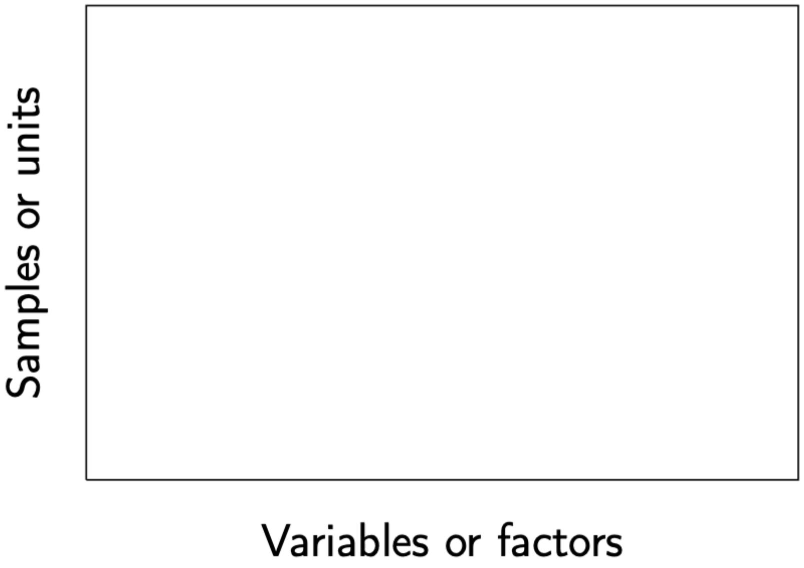
The challenge was to predict those rankings

The prize was $1 million.

(Cue Dr. Evil jokes if anyone knows Austin Powers movies…)

Unsupervised learning

In **unsupervised learning** we start with a data matrix:



Samples or units

Variables or factors

Unsupervised learning setup

- Quantitative, eg. weight, height, number of children, …;
- Qualitative, eg. college major, profession, gender, …;

Goals of unsupervised learning

In **unsupervised learning** we start with a data matrix:

Our goal is to:

1. Find meaningful relationships between the variables or units: Correlation analysis.

2. Find interpretable low-dimensional representations of the data which make it easy to visualize the variables and units. PCA, ICA, isomap, locally linear embeddings, etc.

3. Find meaningful groupings of the data. Clustering.

Unsupervised learning is sometimes referred to in Statistics as **exploratory data analysis**.

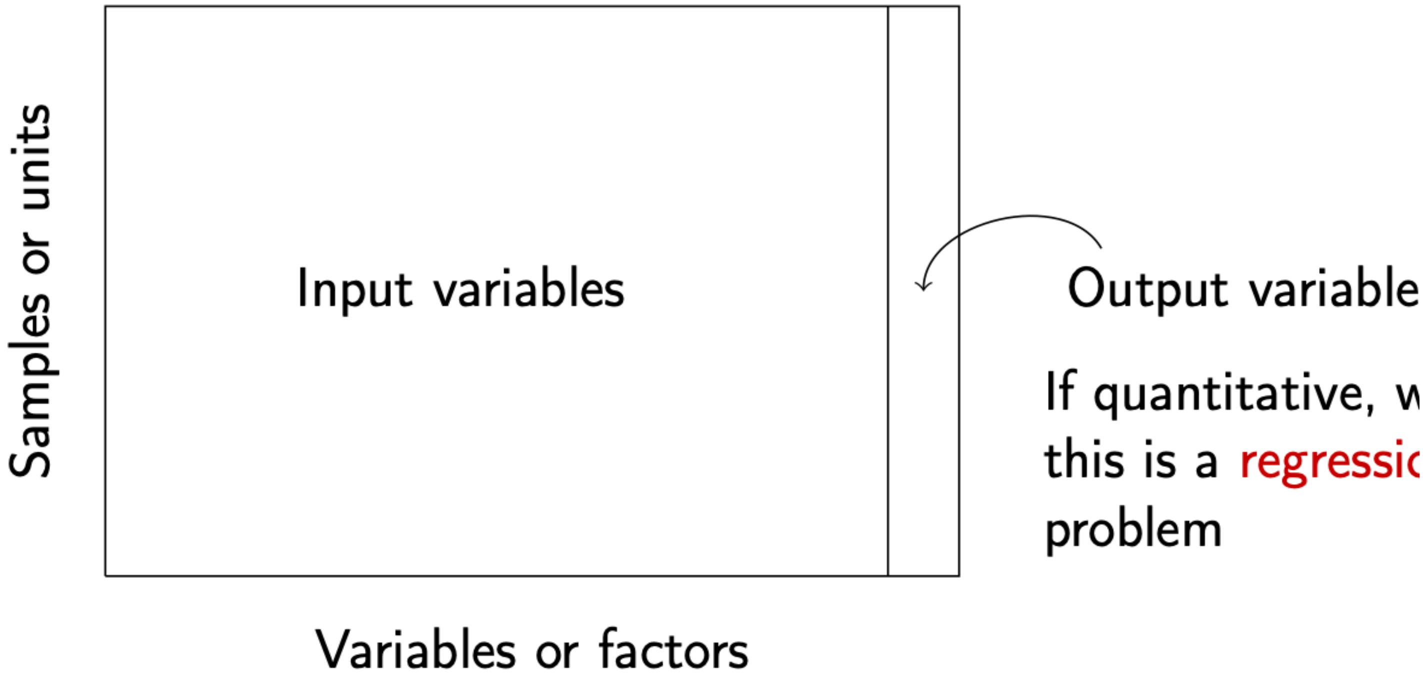
Striking example

- 1387 European subjects were genotyped and differences (SNPs) are measured
- Can be used to form a distance between subjects.
- This distance looks surprisingly close to a map of Europe

Supervised learning

Setup

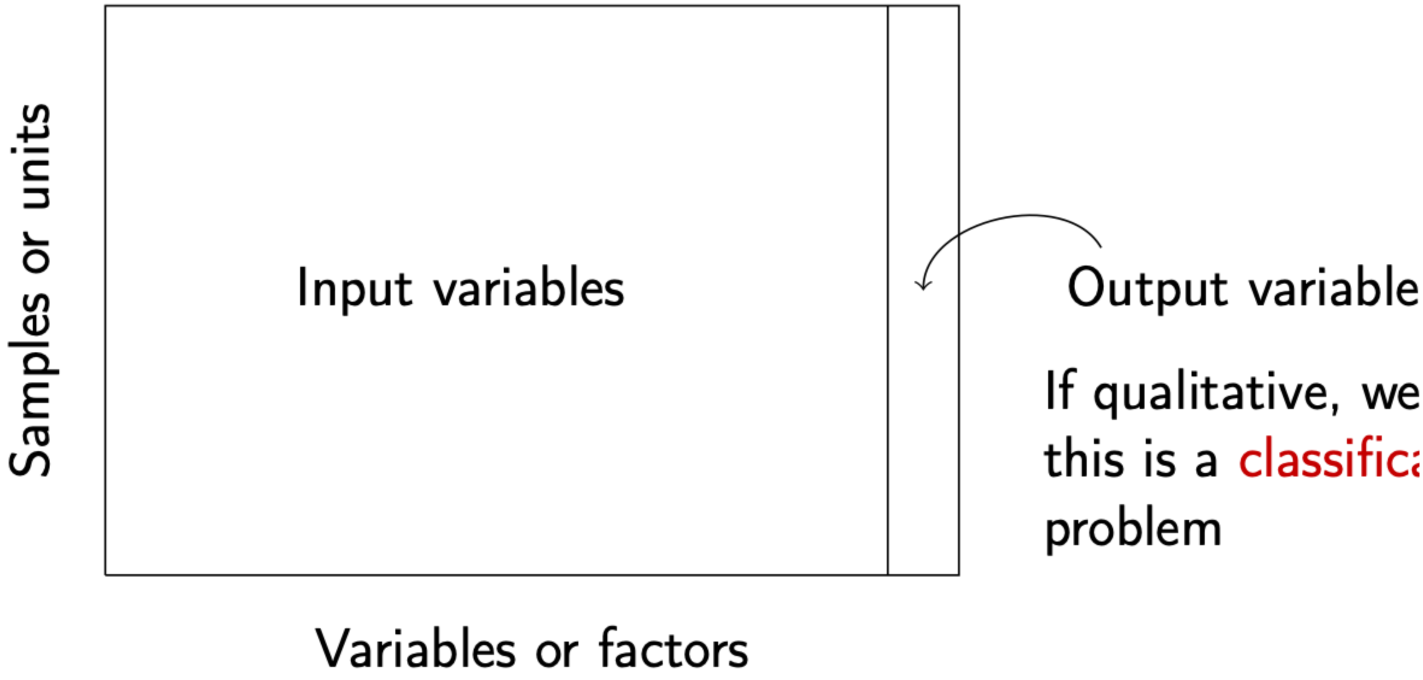
In **supervised learning**, there are *input* variables, and *output* variables:



Typical regression problem

Classification problems

In **supervised learning**, there are *input* variables, and *output* variables:



If qualitative, we this is a classifica problem

Typical classification problem

Goals of supervised learning

In **supervised learning**, there are *input* variables, and *output* variables:

- If X is the vector of inputs for a particular sample. The output variable for *regression* is modeled by:

$$Y = f(X) + \underbrace{\varepsilon}_{\text{Random error}}$$

- Our goal is to learn the function f, using a set of training samples $(x_1, y_1) \ldots (x_n, y_n)$

Regression model

$$\overset{\text{outcome}}{Y} = f(\overset{\text{features}}{X}) + \underset{\text{error}}{\varepsilon}$$

Task	**Example**
Prediction ML	- Useful when the input variable is readily available, but the output variable is not. - Predict $\underbrace{\text{stock prices}}_{Y}$ next month using $\underbrace{\text{data from last}}_{X}$ year.
Inference Statistics	- A model for f can help us understand the structure of the data: which variables influence the output, and which don't? - What is the relationship between each variable and the output, e.g. linear, non-linear? - What is the influence of genetic variations on the incidence of heart disease?

complex

Parametric and nonparametric methods

simple

There are (broadly) two kinds of supervised learning method:

- **Parametric methods:** We assume that f takes a specific form. For example, a linear form:

$M = \{ f(x_1, \dots, x_p) : f(x) = \beta_1 x_1 + \dots \beta_p x_p \}$

Parameters

$$f(X) = X_1\beta_1 + \cdots + X_p\beta_p$$

$(\beta_1, \dots, \beta_p)$

with parameters $\beta_1, \ldots, \beta_p$. Using the training data, we try to *fit* the parameters.

- **Non-parametric methods:** We don't make any assumptions on the form of f, but we restrict how *wiggly* or *rough* the function can be.

$f \approx$ complexity

semiparametric

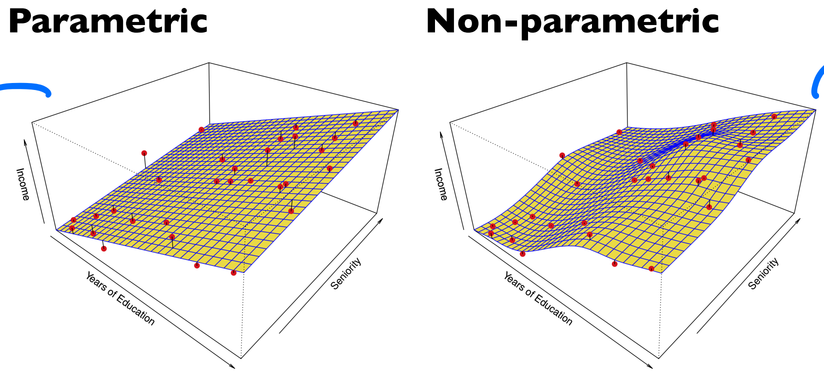
$f(x) = \beta_1 X_1 + w(X_2, \dots, X_p)$

"nonparametric"

H_0: Q: $\beta_1 = 0$?

Parametric vs. nonparametric prediction

$I = \beta_1 \cdot E + \beta_2 \cdot S + \varepsilon$



Parametric

Non-parametric

wigglier

1. Parametric methods have a limit of fit quality. Non-parametric methods can keep improving as we add more data to fit.
2. Parametric methods are often simpler to interpret.

Prediction error

- Our goal in supervised learning is to minimize the prediction error
- For regression models, this is typically the **Mean Squared Error (MSE)**. Let (x_0, y_0) denote a new sample from the population:

$$MSE(\hat{f}) = E\Big[(y_0 - \hat{f}(x_0))^2\Big]$$

- Unfortunately, this quantity cannot be computed, because we don't know the joint distribution of (x_0, y_0).
- We can compute a sample average using the training data; this is known as the training MSE:

$$MSE_{\text{training}}(\hat{f}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2.$$

- MSE_{training} can be used to learn $\hat{f}$

$$\hat{f} = \text{argmin}_{f \in \mathcal{M}} MSE_{\text{training}}(f)$$

Parametric

$$f(x) \approx f_\beta(x)$$

$$\hat{f} \approx \text{argmin}_\beta \, \underset{\nearrow\ \text{training}}{MSE}(f_\beta)$$

Key quantities

Term	Quantity
Training data	$(x_1, y_1), (x_2, y_2) \ldots (x_n, y_n)$ ← dataset
Predicted function	$\hat{f}$, a function of (i.e. learned on) training data
Future data	(x_0, y_0) having some distribution (usually related to distribution of training data)
Prediction error (MSE)	$MSE(\hat{f}) = E(y_0 - \hat{f}(x_0))^2.$

$$MSE_{train}(f) = G\left(f, \{(x_1,y_1), (x_2,y_2), \ldots, (\ )\}\right)$$

$$MSE(f) = \tilde{G}\left(f, \text{dbn of } (x_0, y_0)\right)$$

$$\hat{f} = \bar{G}\left(\{(x_1,y_1), \ldots (x_n,y_n)\}\right)$$

Test vs. training error

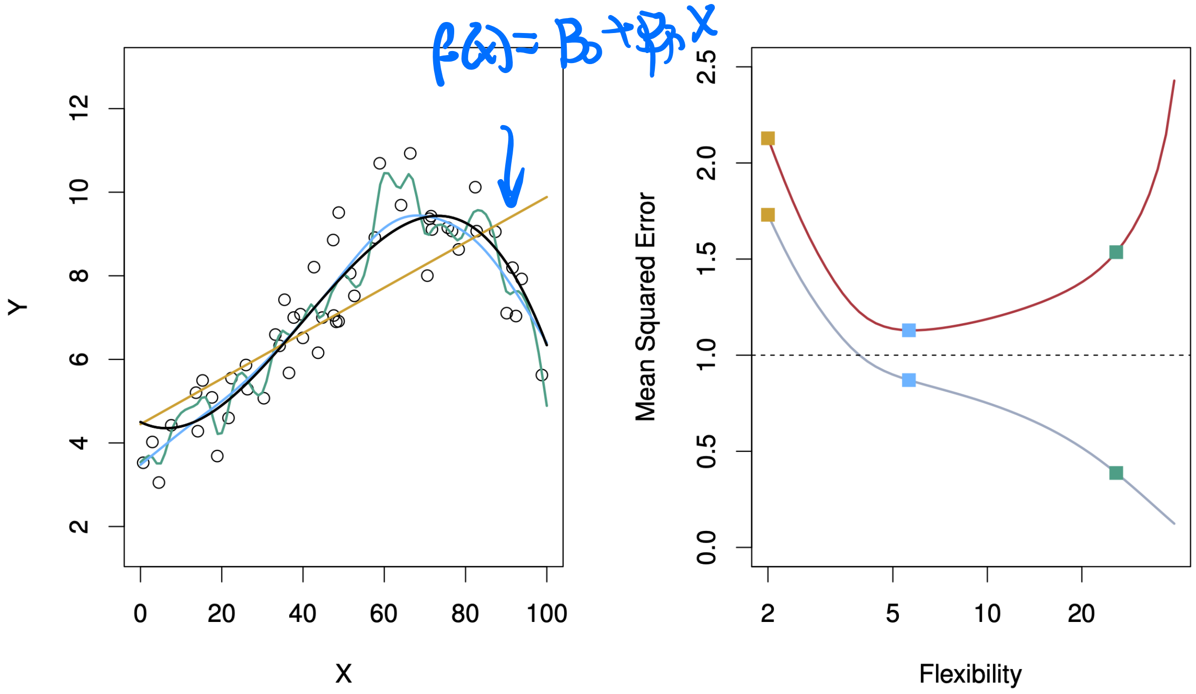
1. The main challenge of statistical learning is that **a low training MSE does not imply a low MSE.**

2. If we have test data $\{(x_i', y_i'); i = 1, \dots, m\}$ which were not used to fit the model, a better measure of quality for $\hat{f}$ is the test MSE:

$$MSE_{\text{test}}(\hat{f}) = \frac{1}{m} \sum_{i=1}^{m} (y_i' - \hat{f}(x_i'))^2.$$

$$\approx MSE\left(\hat{f}, \text{"empirical dbn of test data"}\right)$$

surrogate for $MSE(\hat{f})$...



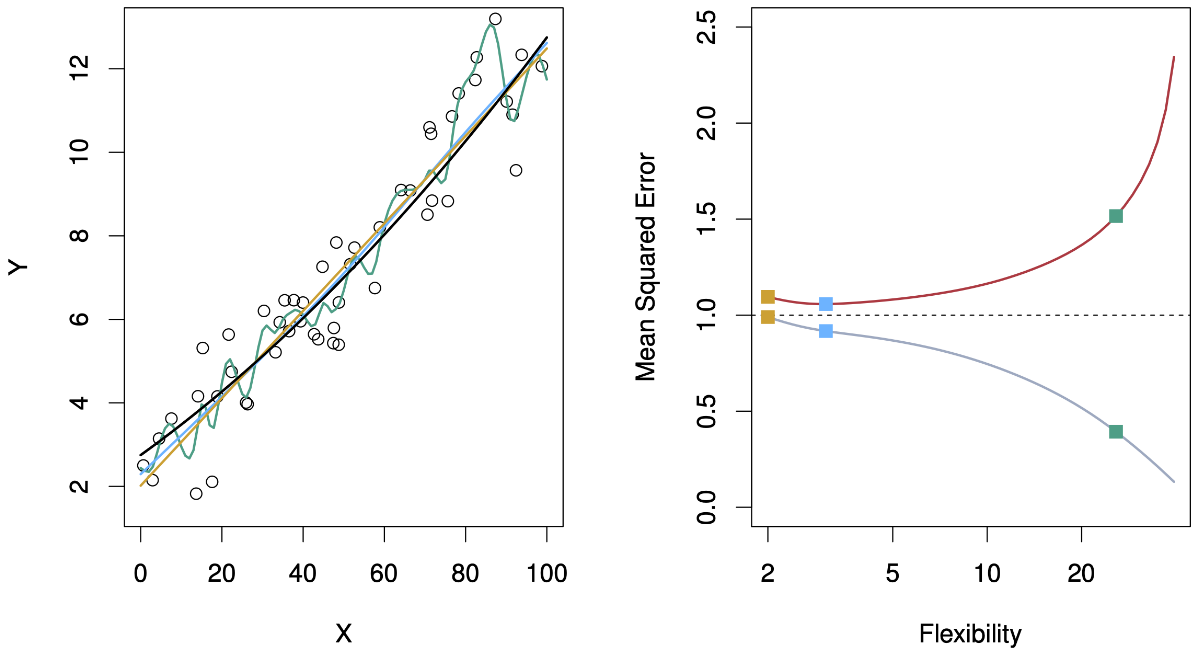
$f(x) = \beta_0 + \beta_1 X$

Prediction error vs. flexibility

The circles are simulated data from the black curve f by adding Gaussian noise. In this artificial example, we *know* what f is.

Three estimates $\hat{f}$ are shown: Linear regression, Splines (very smooth), Splines (quite rough).

Red line: Test MSE, Gray line: Training MSE.

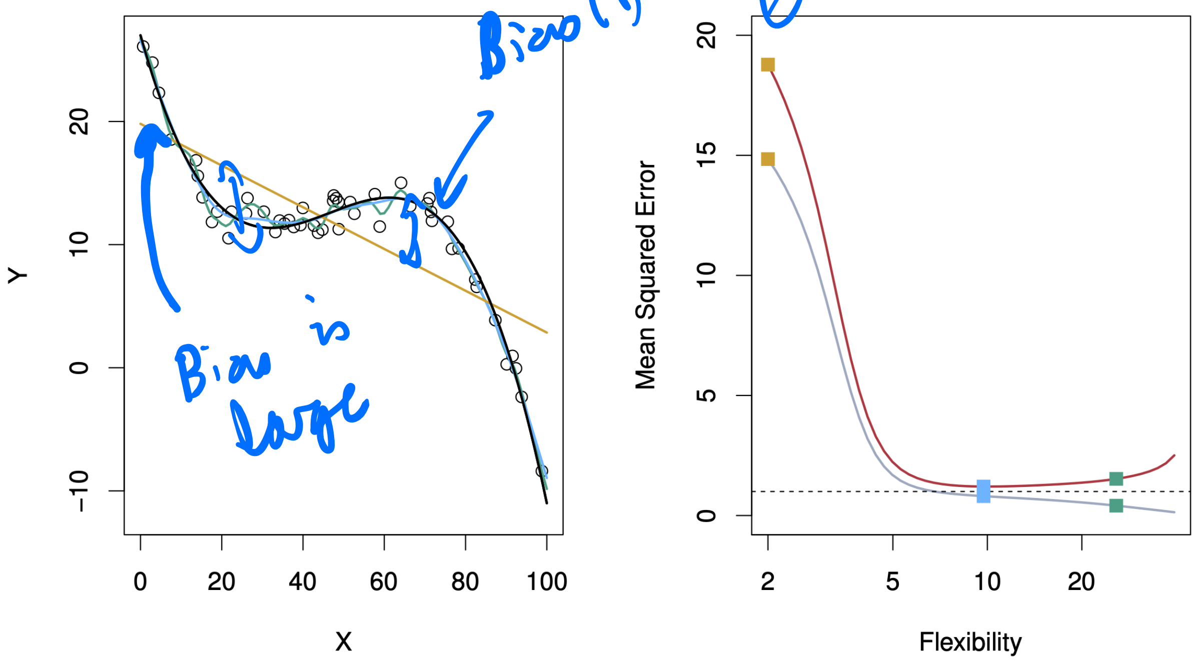


Prediction error vs. flexibility

The function f is now almost linear.

μ is too small

Bias(f, 60)



Bias is large

Prediction error vs. flexibility

When the noise ε has small variance relative to f, the third method does well.

Bias variance decomposition

- Let x_0 be a fixed test point, $y_0 = f(x_0) + \varepsilon_0$, and $\hat{f}$ be estimated from n training samples $(x_1, y_1) \ldots (x_n, y_n)$.

- Let E denote the expectation over y_0 and the training outputs $(y_1, \ldots, y_n)$. Then, the Mean Squared Error at x_0 can be decomposed:

$$\Big(\; MSE(\hat{f}, x_0) = E(y_0 - \hat{f}(x_0))^2 = \mathrm{Var}(\hat{f}(x_0)) + [\mathrm{Bias}(\hat{f}(x_0))]^2 + \mathrm{Var}(\varepsilon_0)$$

Term	**Quantity**
Variance	▪ $\mathrm{Var}(\hat{f}(x_0))$ ▪ The variance of the estimate of Y: $E[\hat{f}(x_0) - E(\hat{f}(x_0))]^2$
Bias (squared)	▪ $[\mathrm{Bias}(\hat{f}(x_0))]^2$ ▪ This measures how much the estimate of $\hat{f}$ at x_0 changes when we sample and average over new training data.
Irreducible error	▪ $\mathrm{Var}(\varepsilon_0)$ ▪ This is how much of the response is not predictable by any method.

- Each fitting method has its own variance and bias. No method can improve on irreducible error.

- A good method has small **variance + squared bias**.

$$MSE(f, x_0) = E\left[(y_0 - f(x_0))^2 \,\middle|\, x_0 = x_0\right]$$

$$\mathrm{Bias}(\hat{f}, x_0) = f_{true}(x_0) - E\left[\hat{f}(x_0)\right]$$

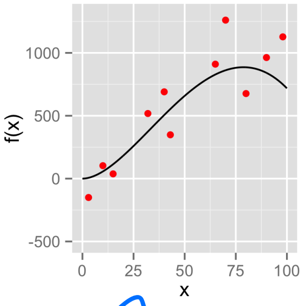
$$\overset{\displaystyle f_{true}}{\parallel}$$

$$E[Y \mid X = x_0]$$

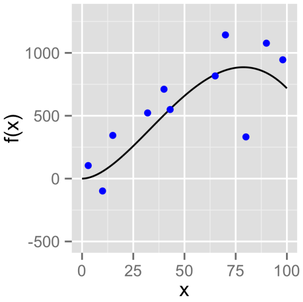
Simulation example

$\hat{f}_2$

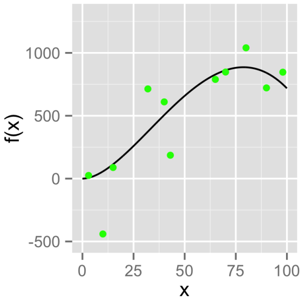
Sample #1



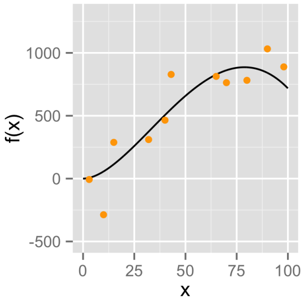
Sample #2



Sample #3

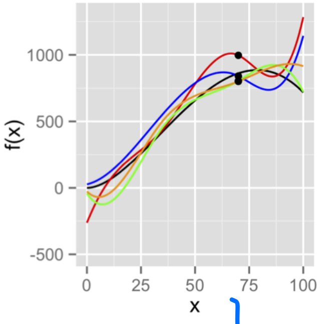


Sample #4

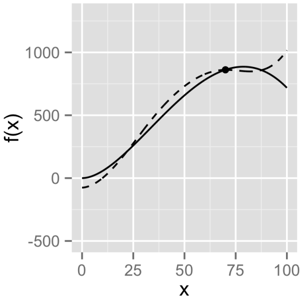


$\hat{f}_1$

All fits



Bias



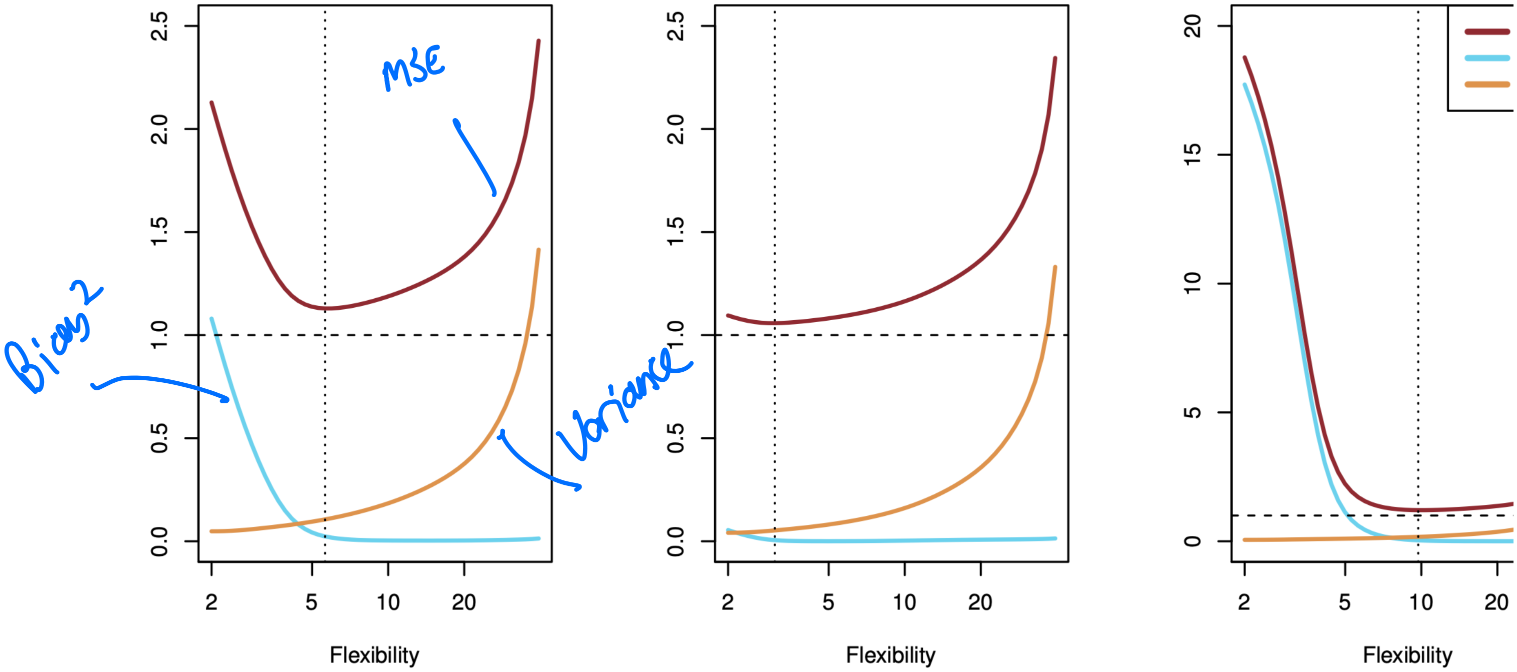
$x=70$

Implications of bias variance decomposition

$$MSE(x_0) = E(y_0 - \hat{f}(x_0))^2 = \text{Var}(\hat{f}(x_0)) + [\text{Bias}(\hat{f}(x_0))]^2 + \text{Var}(\varepsilon).$$

1. The MSE is always positive.
2. Each element on the right hand side is always positive.
3. Therefore, typically when we decrease the bias beyond some point, we increase the variance, and vice-versa.

More flexibility $\iff$ Higher variance $\iff$ Lower bias.



- Squiggly f, high noise
- Linear f, high noise
- Squiggly f, low noise

Classification problems

- In a classification setting, the output takes values in a discrete set.
- For example, if we are predicting the brand of a car based on a number of variables, the function f takes values in the set $\{\text{Ford, Toyota, Mercedes-Benz}, \ldots\}$.
- Model is no longer

$$\cancel{Y = f(X) + \epsilon} \qquad \cancel{E\left[(Y - f(X))^2\right]}$$

- We use slightly different notation:

$p(x|Y)$

$$\begin{aligned} P(X, Y) &: \text{joint distribution of } (X, Y), \\ P(Y \mid X) &: \text{conditional distribution of } Y \text{ given } X, \\ \hat{y}_i &: \text{prediction for } x_i. \end{aligned}$$

- For classification we are interested in learning $\boxed{P(Y \mid X).}$
- Connection between the classification and regression: both try to learn $\boxed{E(Y \mid X)}$ but the type of Y is different.

Loss function for classification

- There are many ways to measure the error of a classification prediction. One of the most common is the 0-1 loss:

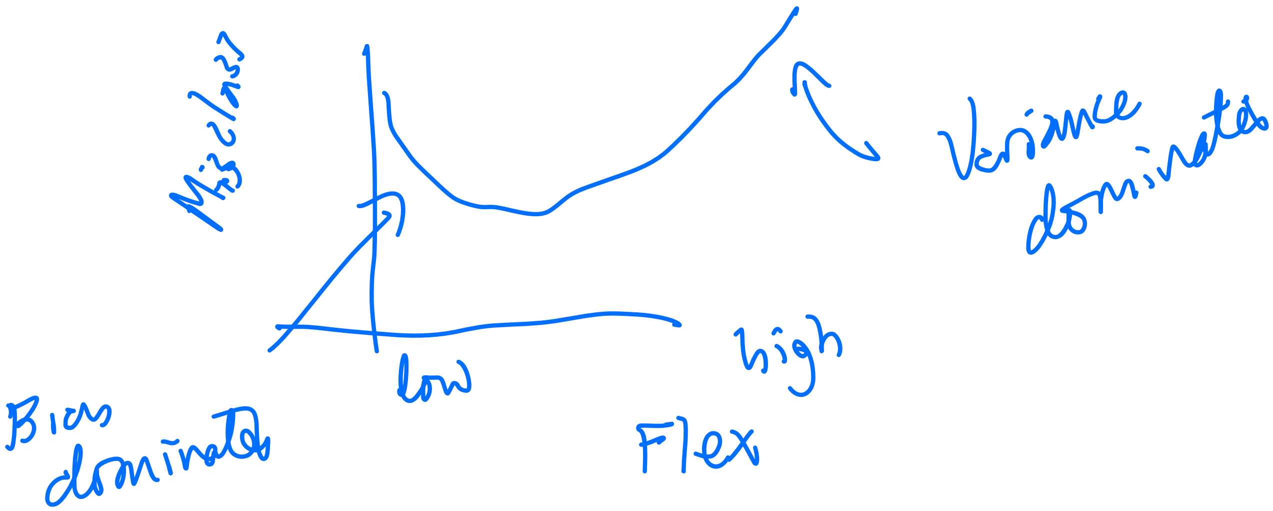
$$\text{Misclass} \approx E(\mathbf{1}(y_0 \neq \hat{y}_0)) \approx P(Y_0 \neq \hat{Y}_0)$$

- Like the MSE, this quantity can be estimated from training or test data by taking a sample average:

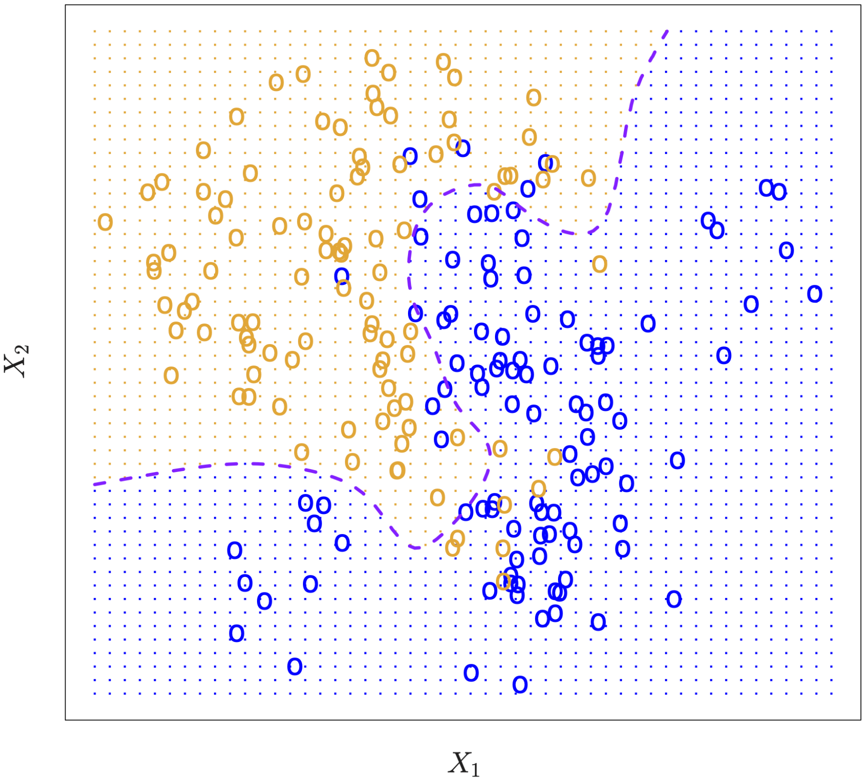
$$\text{Misclass}_{\text{test}} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}(y_i \neq \hat{y}_i)$$

- For training, we usually use smoother proxies of 0-1 loss – easier to minimize.

- While no bias-variance decomposition, test error curves look similar to regression.



Bayes classifier



Bayes decision boundary

- The Bayes classifier assigns:

$$\hat{y}_i = \text{argmax}_j \;\; P(Y = j \mid X = x_i)$$

- It can be shown that this is the best classifier under the 0-1 loss.

- In practice, we never know the joint probability P. However, we may assume that it exists.

- Many classification methods approximate $P(Y = j \mid X = x_i)$ with $\hat{P}(Y = j \mid X = x_i)$ and predict using this approximate Bayes classifier.