CMSC/LING/STAT 208: Machine Learning

Abhishek Chakraborty [Much of the content in these slides have been adapted from *ISLR2* by James et al. and *HOMLR* by Boehmke & Greenwell]

Supervised Learning

More mathematically, the "true"/population model can be represented by

$$Y = f(\mathbf{X}) + \epsilon$$

where ϵ is a **random** error term (includes measurement error, other discrepancies) independent of $\mathbf{X}$ and has mean zero.

Supervised Learning: Why Estimate $f(\mathbf{X})$?

We wish to know about $f(\mathbf{X})$ for two reasons:

- Prediction at new unseen data points x_0

$$\hat{y}_0 = \hat{f}(x_0) \quad \text{or} \quad \hat{y}_0 = \hat{C}(x_0)$$

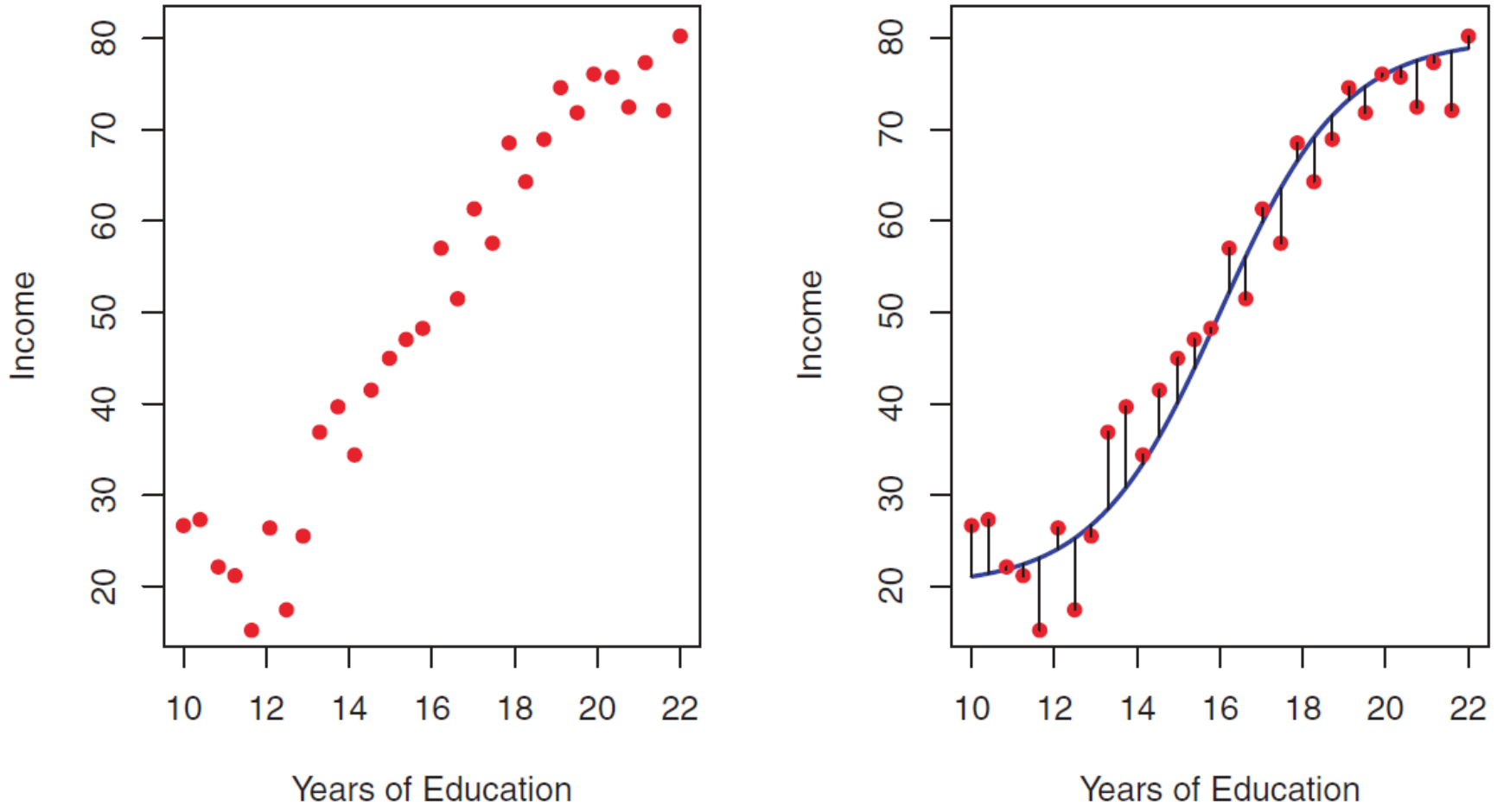
- Inference: Understand the relationship between $\mathbf{X}$ and Y.
- An ML algorithm that is developed mainly for predictive purposes is often termed as a **Black Box** algorithm.

The primary objective is to:

- **Regression** (response Y is quantitative): Build a model $\hat{Y} = \hat{f}(\mathbf{X})$
- **Classification** (response Y is qualitative): Build a classifier $\hat{Y} = \hat{C}(\mathbf{X})$

Supervised Learning: Prediction and Inference

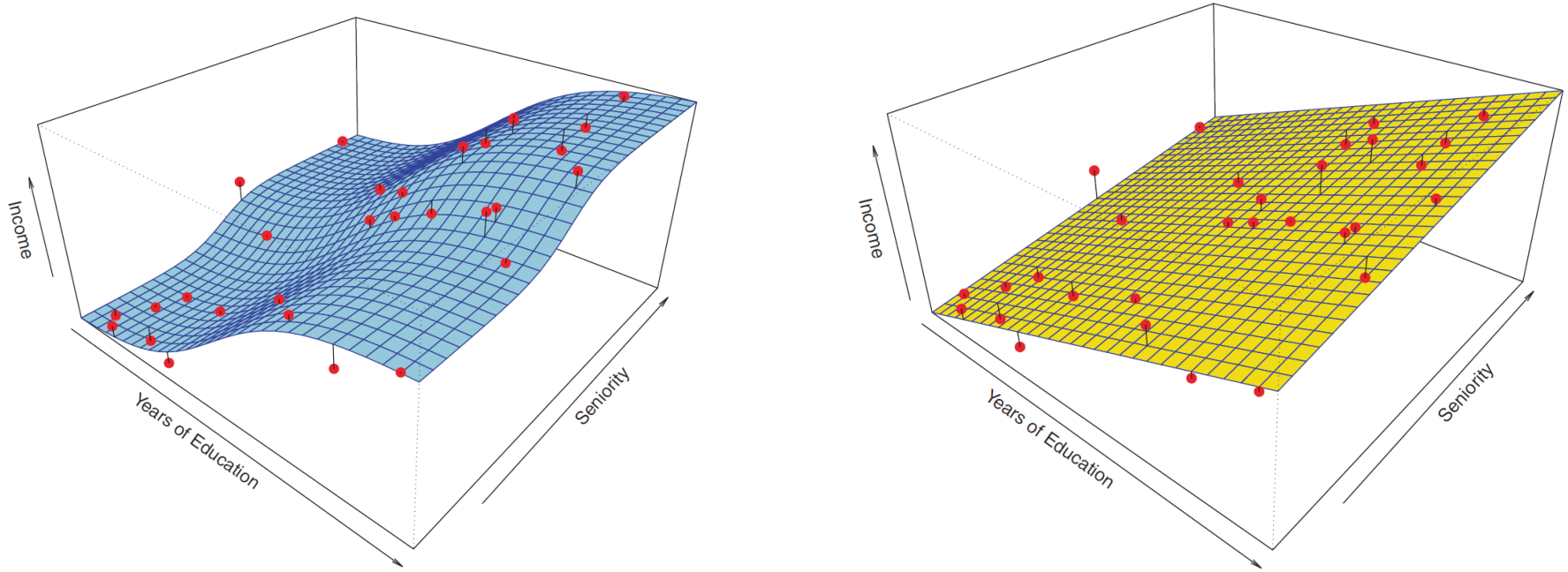
Income dataset



Why ML? (from ISLR2)

Supervised Learning: Prediction and Inference

Income dataset



Why ML? (from ISLR2)

Question!!!

Which of the following statements are correct?

1. As `Years of Education` increases, `Income` increases, keeping `Seniority` fixed.
2. As `Years of Education` increases, `Income` decreases, keeping `Seniority` fixed.
3. As `Years of Education` increases, `Income` increases.
4. As `Seniority` increases, `Income` increases, keeping `Years of Education` fixed.
5. As `Seniority` increases, `Income` decreases, keeping `Years of Education` fixed.
6. As `Seniority` increases, `Income` increases.

Question!!!

Which of the following statements are correct?

1. The increase in `Income` resulting from increase in `Years of Education` keeping other variables fixed is **more** than the increase in `Income` resulting from increase in `Seniority` keeping other variables fixed.

2. The increase in `Income` resulting from increase in `Years of Education` keeping other variables fixed is **less** than the increase in `Income` resulting from increase in `Seniority` keeping other variables fixed.

Supervised Learning: How Do We Estimate $f(\mathbf{X})$?

Broadly speaking, we have two approaches.

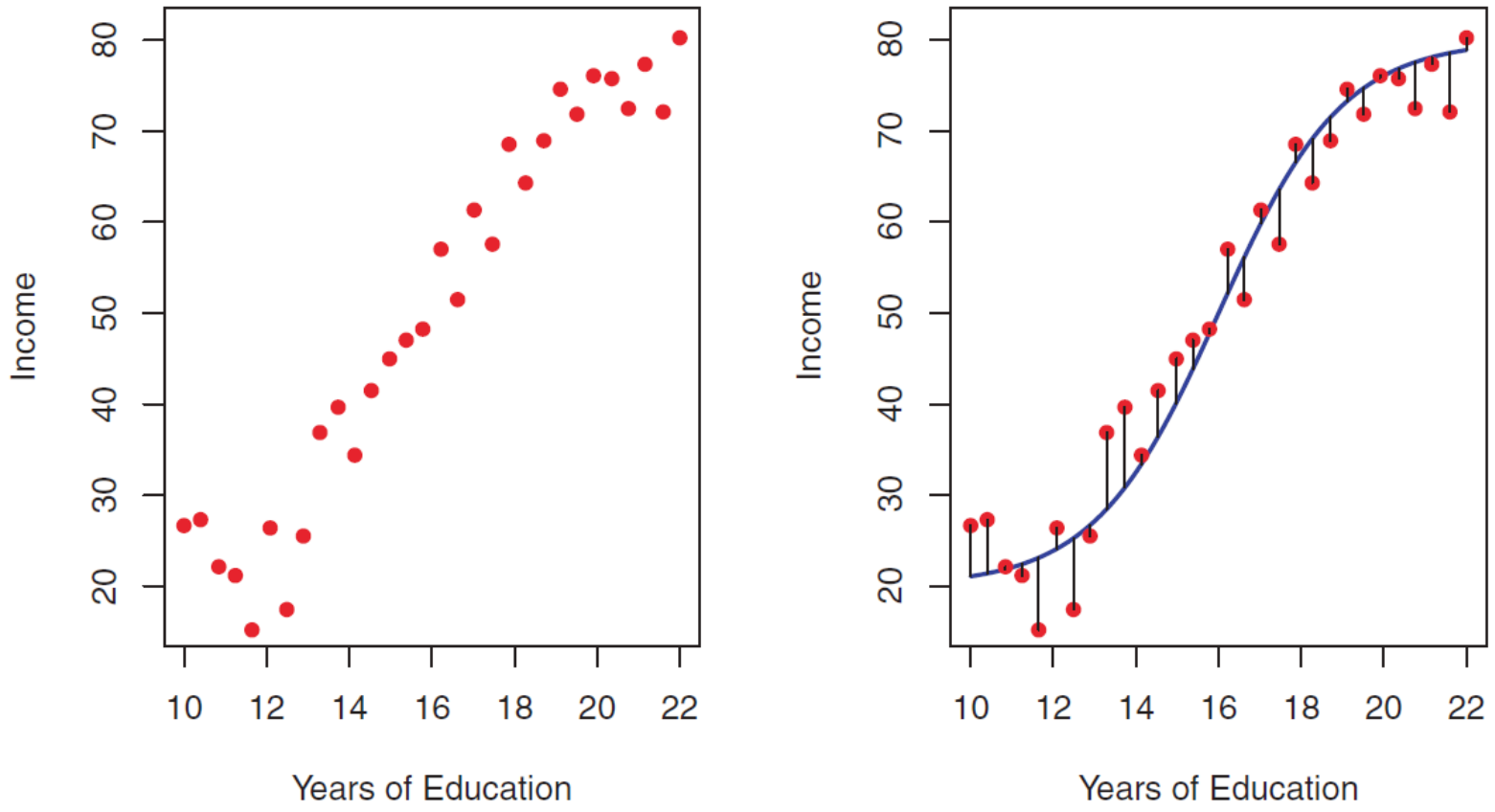
- Parametric and Structured Methods
  - A functional form of $f(\mathbf{X})$ is assumed, such as

$$f(\mathbf{X}) = \beta_0 + \beta_1 \mathbf{x}_1 + \beta_2 \mathbf{x}_2 + \ldots + \beta_p \mathbf{x}_p$$

  - We estimate the parameters $\beta_0, \beta_1, \ldots, \beta_p$ by fitting the model to labeled training data.

Supervised Learning: Parametric Methods

Income dataset

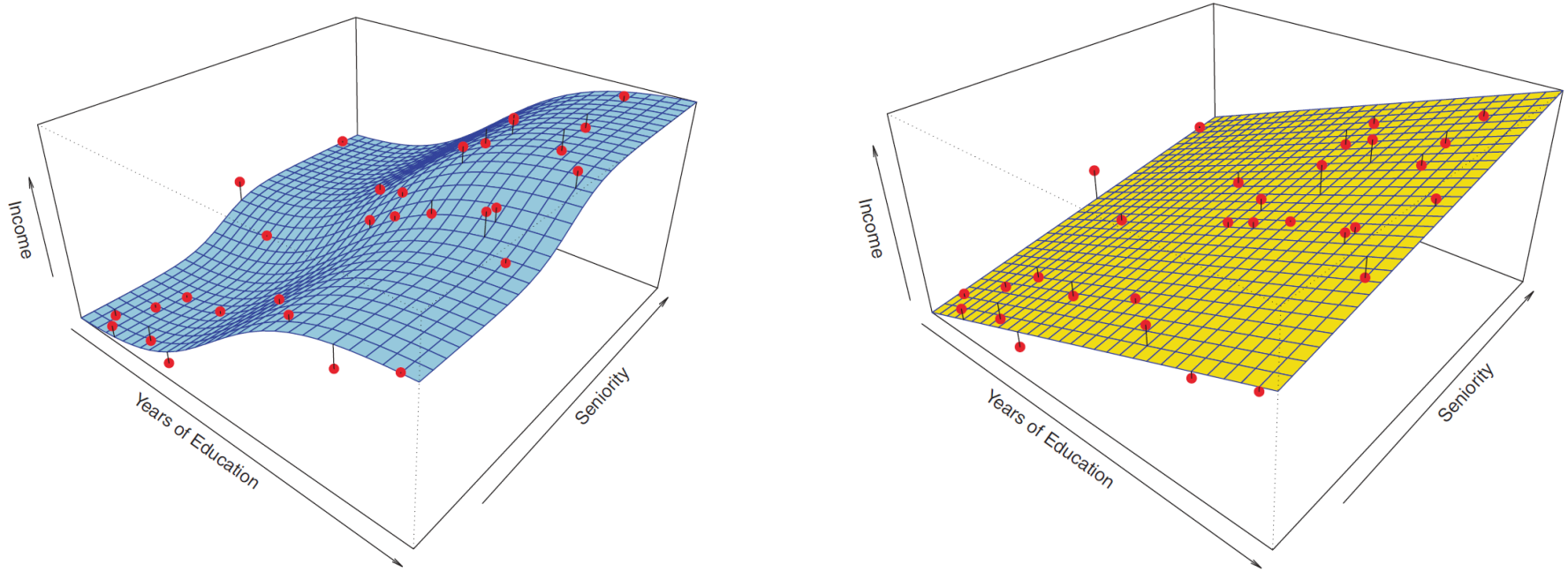


Parametric model fit (from ISLR2)

$$\text{Income} \approx \beta_0 + \beta_1 \times \text{Years of Education}$$

Supervised Learning: Parametric Methods

Income dataset



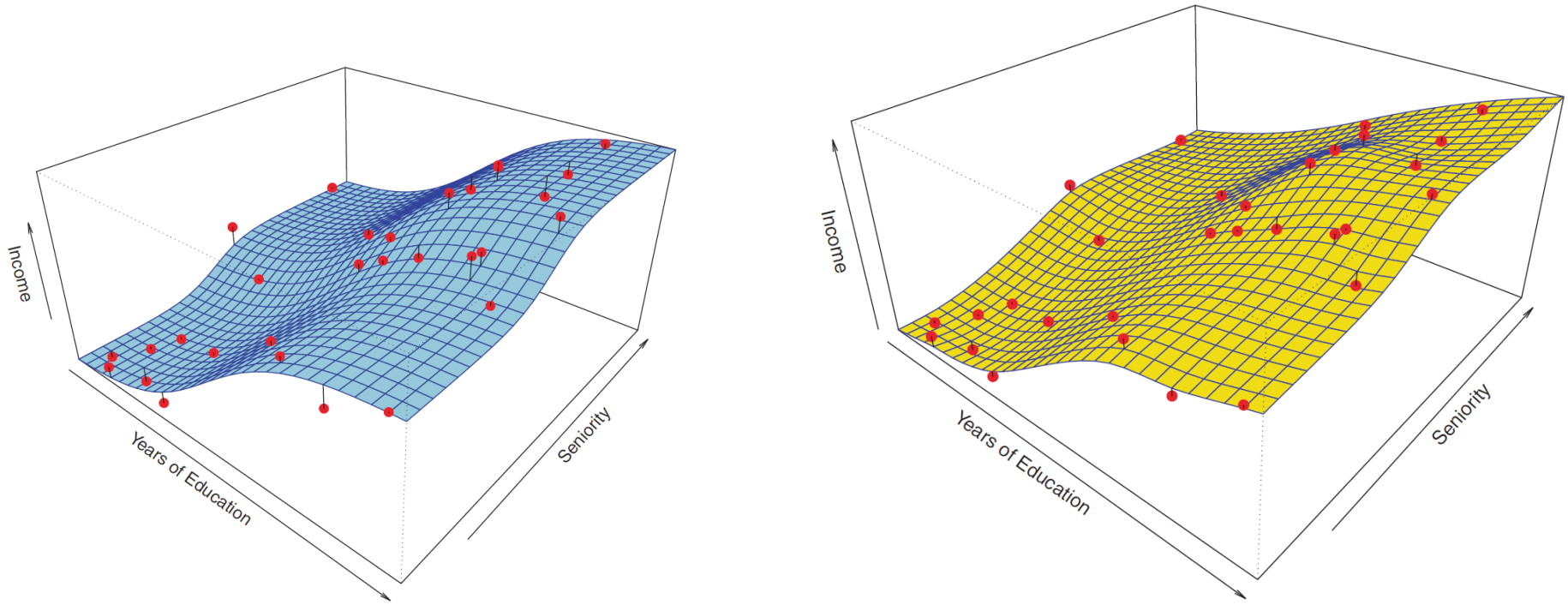
Parametric model fit (from ISLR2)

$$\text{Income} \approx \beta_0 + \beta_1 \times \text{Years of Education} + \beta_2 \times \text{Seniority}$$

Supervised Learning: Non-parametric Methods

Non-parametric approaches do not make any explicit assumptions about the functional form of $f(\mathbf{X})$. A very large number of observations (compared to a parametric approach) is required to fit a model using the non-parametric approach.

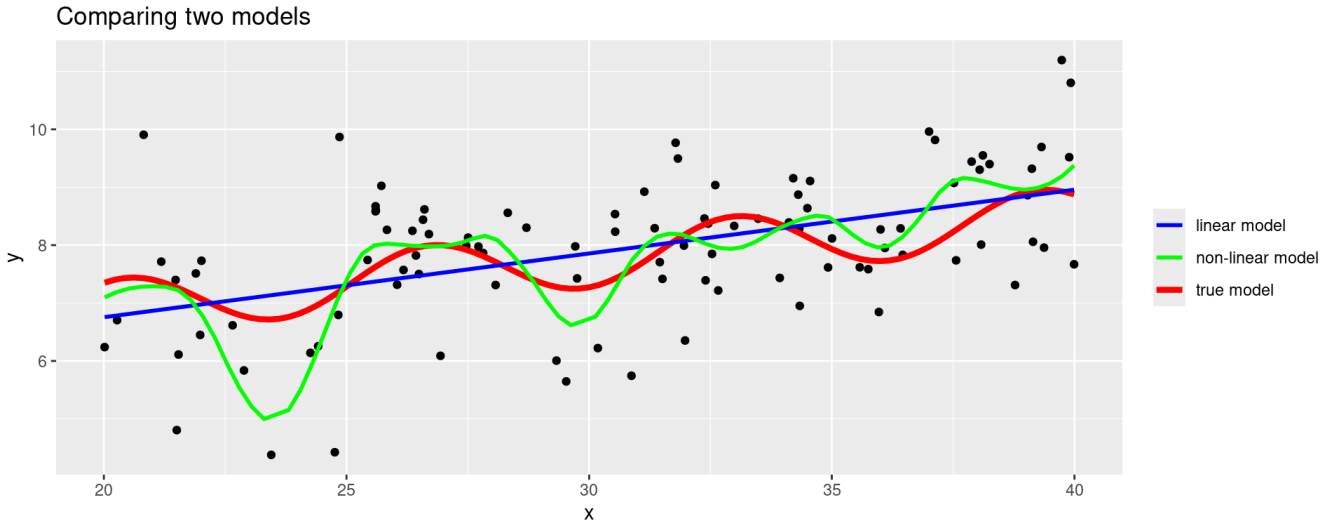
Income dataset



Non-parametric model fit (from ISLR2)

Supervised Learning: Flexibility of Models

Flexibility refers to the smoothness of functions. (More theoretically, flexibility depends on the number of parameters of the function).

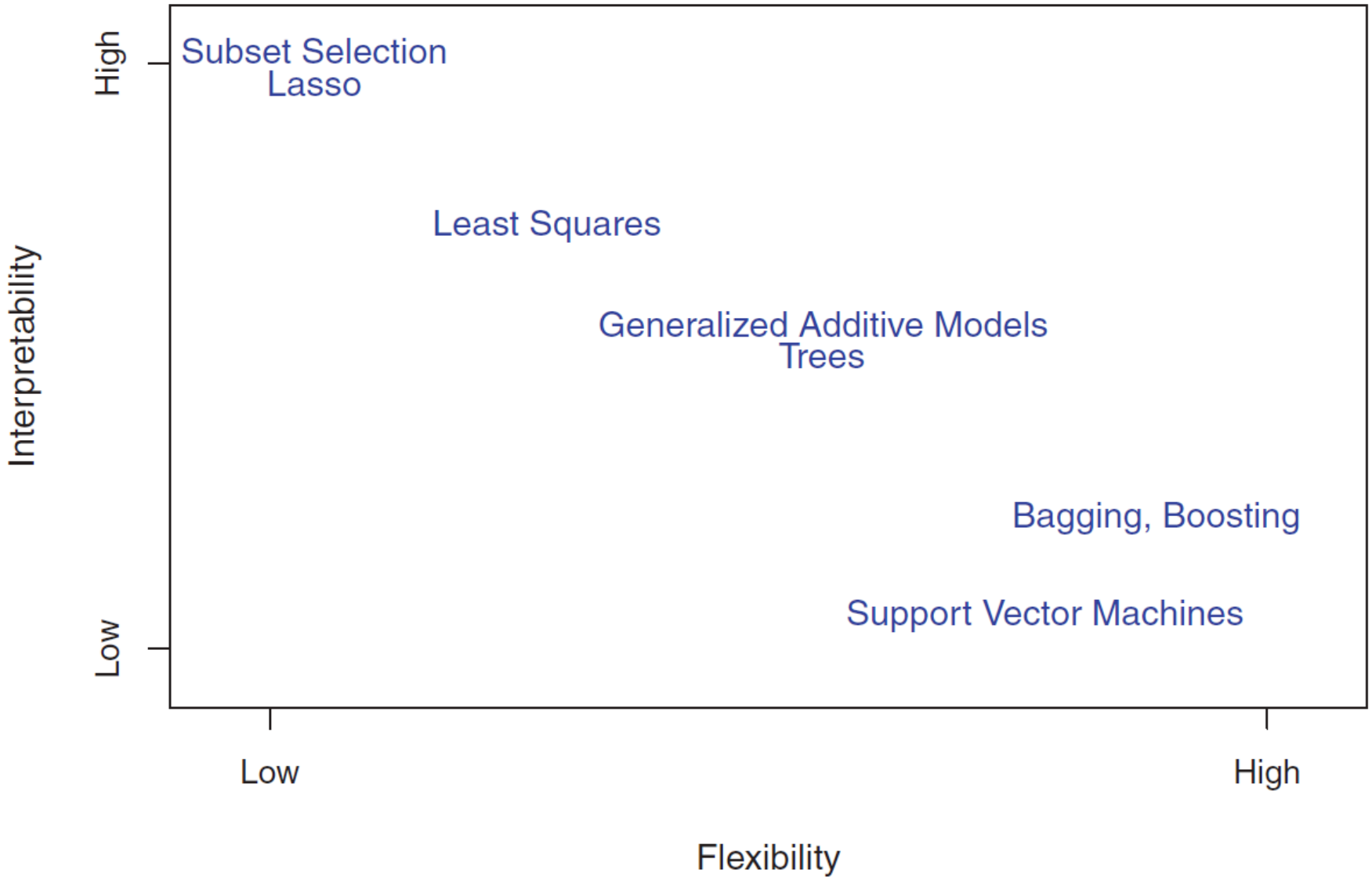


More flexible $\Longrightarrow$ More complex $\Longrightarrow$ Less Smooth $\Longrightarrow$ Less Restrictive $\Longrightarrow$ Less Interpretable

Supervised Learning: Some Trade-offs

- Prediction Accuracy versus Interpretability
- Good Fit versus Over-fit or Under-fit

Supervised Learning: Some Trade-offs



Trade-off between flexibility and interpretability (from ISLR2)

Supervised Learning: Assessing Model Accuracy

Why are we going to study so many different ML techniques?

There is no free lunch in statistics: No one method dominates all others over all possible datasets.

When we estimate $f(\mathbf{X})$ using $\hat{f}\left(\mathbf{X}\right)$, then,

$$E\left[Y-\hat{Y}\right]^2 = E\left[f(\mathbf{X})+\epsilon-\hat{f}\left(\mathbf{X}\right)\right]^2 = \underbrace{\left[f(\mathbf{X})-\hat{f}\left(\mathbf{X}\right)\right]^2}_{Reducible} + \underbrace{Var(\epsilon)}_{Irreducible}$$

$E\left[Y-\hat{Y}\right]^2$: Expected (average) squared difference between predicted and actual (observed) response.

We will focus on techniques for estimating $f(\mathbf{X})$ with the objective of minimizing the reducible error.

Supervised Learning: Assessing Model Accuracy

Suppose we have labeled training data $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$, i.e, n training data points/observations.

We fit/train a model $\hat{y} = \hat{f}(x)$ (or, a classifier $\hat{y} = \hat{C}(x)$) on the training data and obtain estimates $\hat{f}(x_1), \hat{f}(x_2), \ldots, \hat{f}(x_n)$ (or, $\hat{C}(x_1), \hat{C}(x_2), \ldots, \hat{C}(x_n)$).

We could then compute the

- **Regression**

$$\text{Training MSE} = \text{Average}_{Training}\left(y - \hat{f}(x)\right)^2 = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{f}(x_i)\right)^2$$

- **Classification**

$$\text{Training Error Rate} = \text{Average}_{Training}\left[I\left(y \neq \hat{C}(x)\right)\right] = \frac{1}{n}\sum_{i=1}^{n} I\left(y_i \neq \hat{C}(x_i)\right)$$

Supervised Learning: Assessing Model Accuracy

But in general, we are not interested in how the method works on the training data. We want to measure the accuracy of the method on previously unseen test data.

Suppose, if possible, we have fresh test data, $(x_1^{test}, y_1^{test}), (x_2^{test}, y_2^{test}), \ldots, (x_m^{test}, y_m^{test})$. Then we can compute,

- **Regression**

$$\text{Test MSE} = \text{Average}_{Test}\left(y - \hat{f}\left(x\right)\right)^2 = \frac{1}{m}\sum_{i=1}^{m}\left(y_i^{test} - \hat{f}\left(x_i^{test}\right)\right)^2$$

- **Classification**

$$\text{Test Error Rate} = \text{Average}_{Test}\left[I\left(y \neq \hat{C}(x)\right)\right] = \frac{1}{m}\sum_{i=1}^{m} I\left(y_i^{test} \neq \hat{C}(x_i^{test})\right)$$

Supervised Learning: Bias-Variance Trade-off

Suppose we have fit a model $\hat{f}(x)$ to some training data. Let the "true" model be $Y = f(x) + \epsilon$. Let (x_0, y_0) be a test observation.

We have,

$$\underbrace{\text{Average}_{Test}\left(y_0 - \hat{f}(x_0)\right)^2}_{total\ error} = \underbrace{Var\left(\hat{f}(x_0)\right)}_{source\ 3} + \underbrace{\left[Bias\left(\hat{f}(x_0)\right)\right]^2}_{source\ 2} + \underbrace{Var(\epsilon)}_{source\ 1}$$

where $Bias\left(\hat{f}(x_0)\right) = E\left(\hat{f}(x_0)\right) - f(x_0)$

- source 1: how y differs from "true" $f(x)$
- source 2: how $\hat{f}(x)$ (when fitted to the test data) differs from $f(x)$
- source 3: how $\hat{f}(x)$ varies among different randomly selected possible training data

Supervised Learning: Comparing Bias and Variance

Bias-Variance Trade-off: Example

In the following slides, we look at three different examples with simulated toy datasets. We work within the regression setting (but the ideas also extend to the classification setting) and three different $\hat{f}(.)$'s.

- Linear Regression (**orange**)
- Smoothing Spline 1 (**blue**)
- More flexible Smoothing Spline 2 (**green**)

The "true" function (simulated) is $f(.)$ (**black**).

Bias-Variance Trade-off: Example

Simulated toy dataset

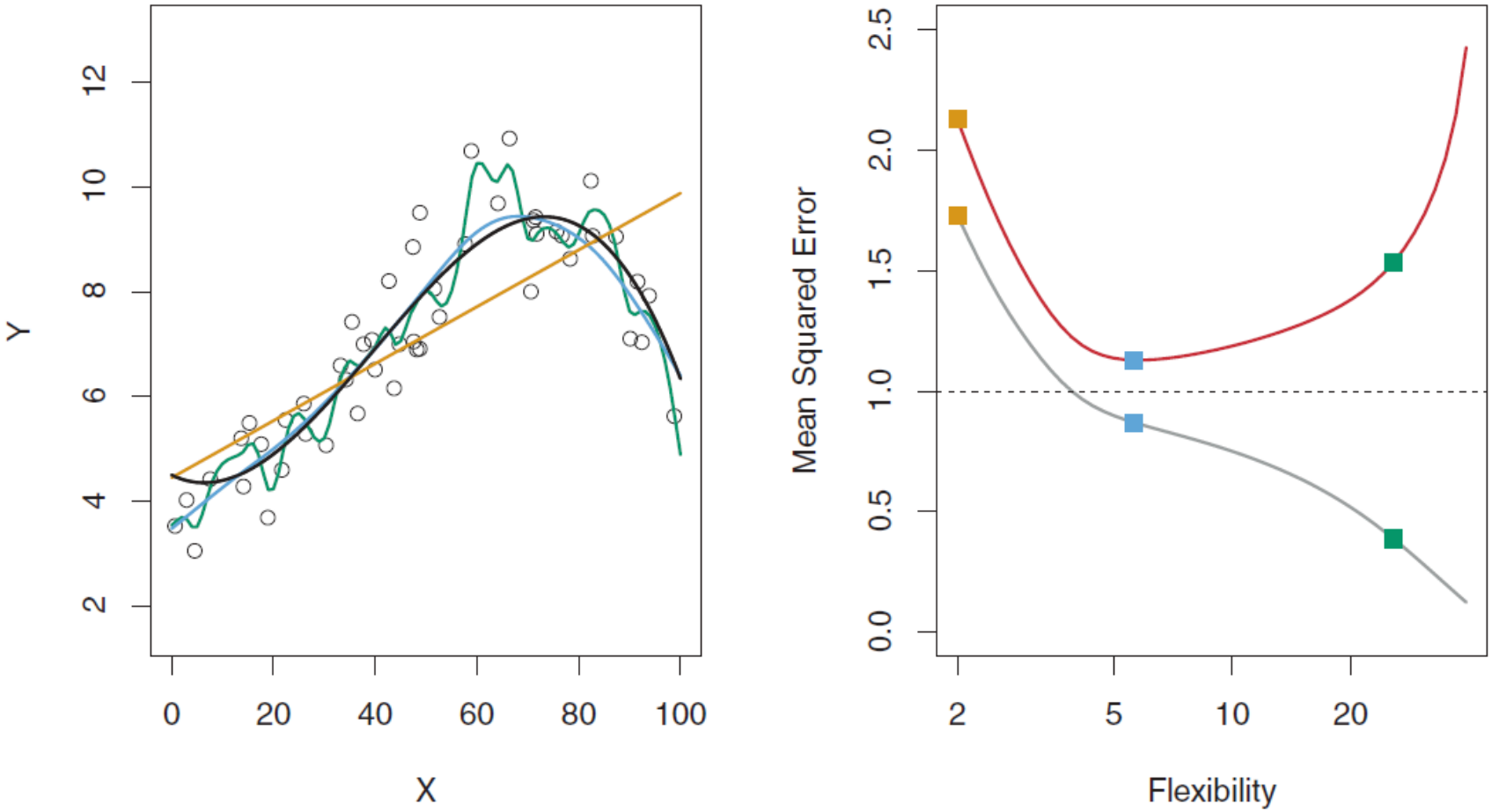


Figure from ISLR2

Bias-Variance Trade-off: Example

Simulated toy dataset

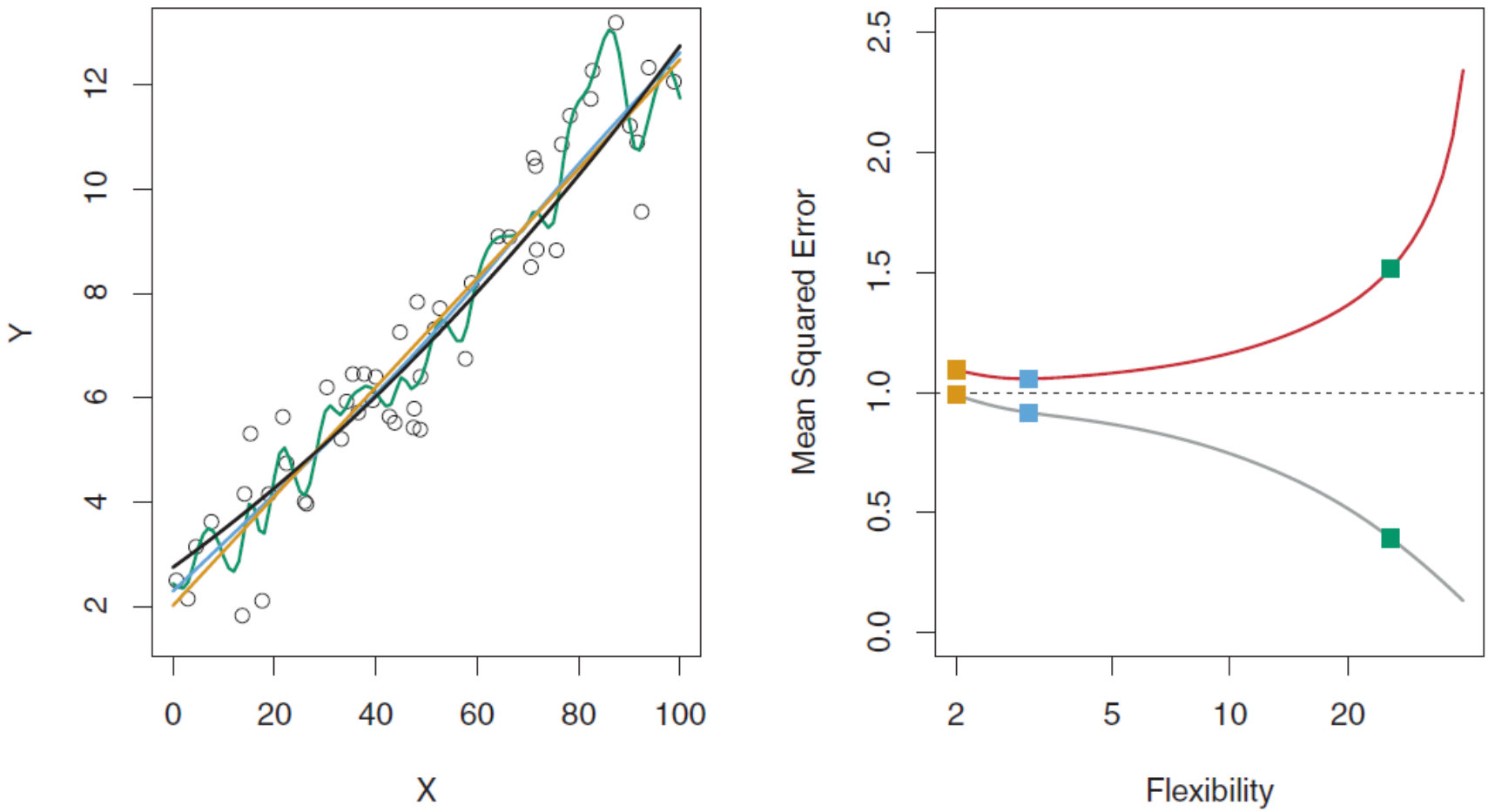


Figure from ISLR2

Bias-Variance Trade-off: Example

Simulated toy dataset

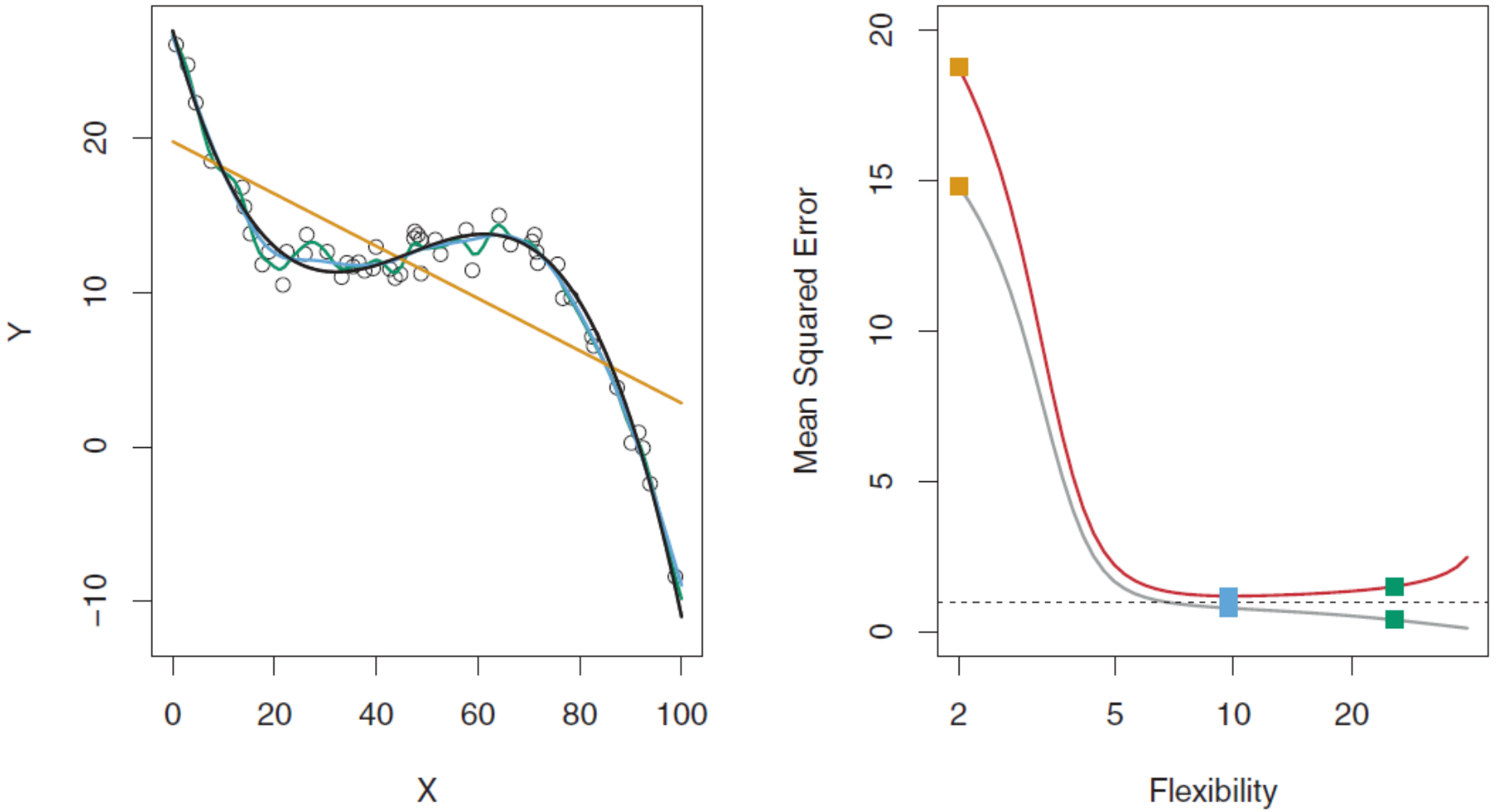


Figure from ISLR2

Bias-Variance Trade-off: Example

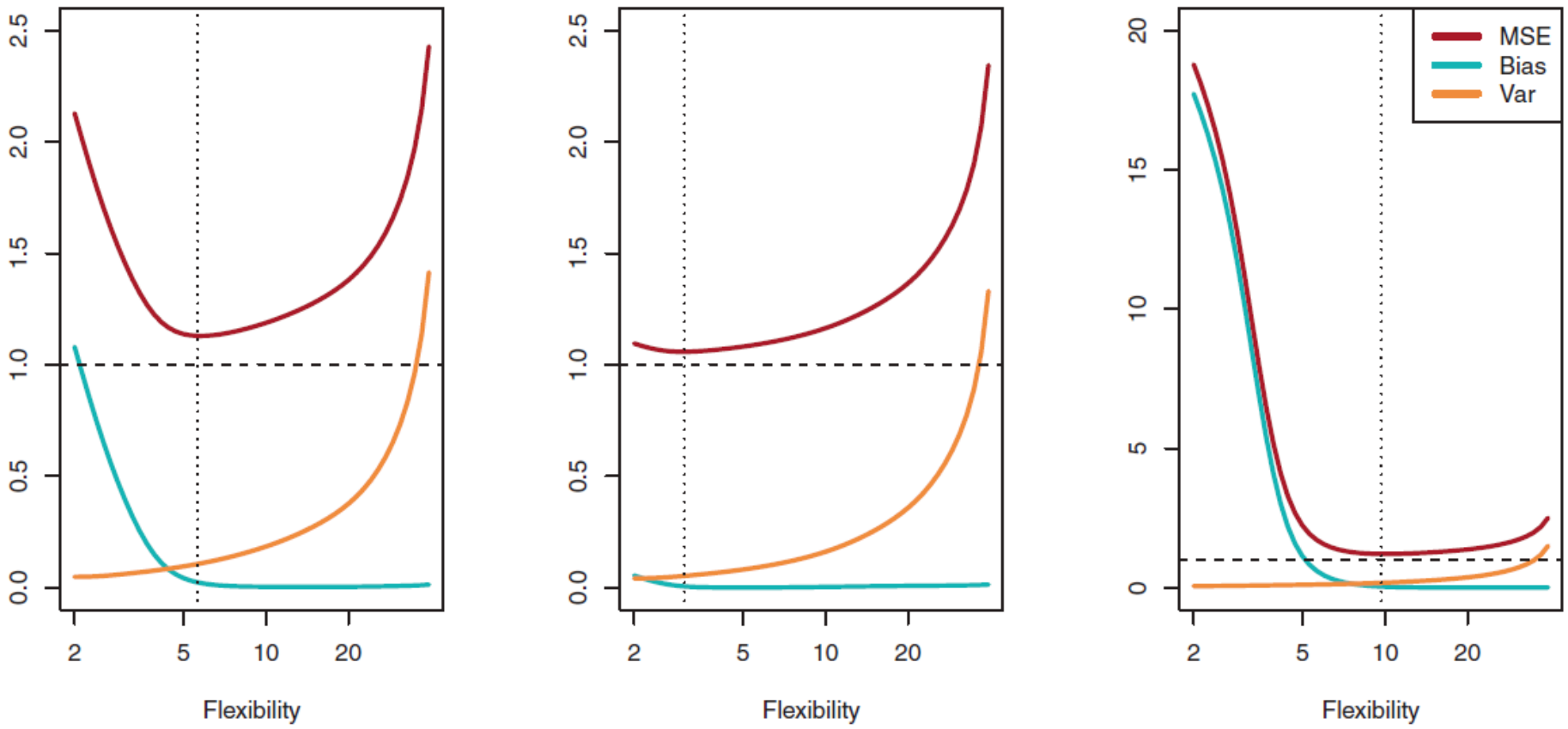


Figure from ISLR2

Question!!!

As the flexibility of a model $\hat{f}(\mathbf{X})$ increases,

1. its variance _____ (increases/decreases)
2. its bias _____ (increases/decreases)
3. its training MSE _____ (increases/decreases)
4. its test MSE _____ (increases/decreases/U-shaped)

Linear Regression

Mathematically, a supervised learning problem can be represented by

$$Y = f(\mathbf{X}) + \epsilon$$

where ϵ is a **random** error term (includes measurement error, other discrepancies) independent of $\mathbf{X}$ and has mean zero.

Objective: To approximate/estimate $f(\mathbf{X})$

For linear regression, we assume that $f(\mathbf{X})$ is a linear function of $\mathbf{X}$, that is, for $p = 1$

$$f(\mathbf{X}) = \beta_0 + \beta_1 X_1$$

Linear Regression

Suppose the CEO of a restaurant franchise is considering opening new outlets in different cities. They would like to expand their business to cities that give them higher profits with the assumption that highly populated cities will probably yield higher profits.

They have data on the population (in 100,000) and profit (in $1,000) at 97 cities where they currently have outlets.

```
outlets <- readRDS("outlets.rds")   # load dataset

head(outlets, 10)   # first ten observations of the dataset
```

```
##    population  profit
## 1      6.1101 17.5920
## 2      5.5277  9.1302
## 3      8.5186 13.6620
## 4      7.0032 11.8540
## 5      5.8598  6.8233
## 6      8.3829 11.8860
## 7      7.4764  4.3483
## 8      8.5781 12.0000
## 9      6.4862  6.5987
## 10     5.0546  3.8166
```

Linear Regression

Given this dataset, our **objective** is to estimate β_0 and β_1.

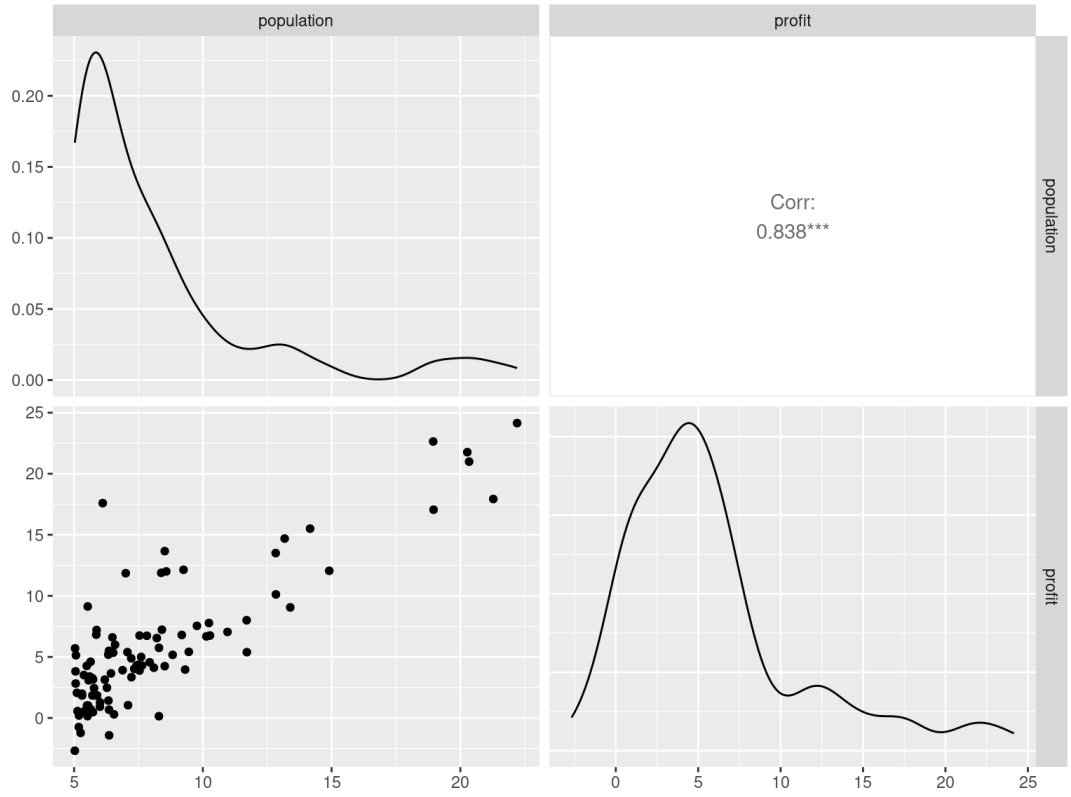
If we are able to estimate β_0 and β_1, we can

- predict the profit for a new city with a given population,
- understand the relationship between `population` and `profit` better.

Linear Regression

Some Exploratory Data Analysis (EDA)

```
ggpairs(data = outlets)
```



Linear Regression: Estimating Parameters

We use training data to find b_0 and b_1 such that

$$\hat{y} = b_0 + b_1 \, x$$

Observed response: y_i for $i = 1, \ldots, n$

Predicted response: $\hat{y}_i$ for $i = 1, \ldots, n$

Residual: $e_i = \hat{y}_i - y_i$ for $i = 1, \ldots, n$

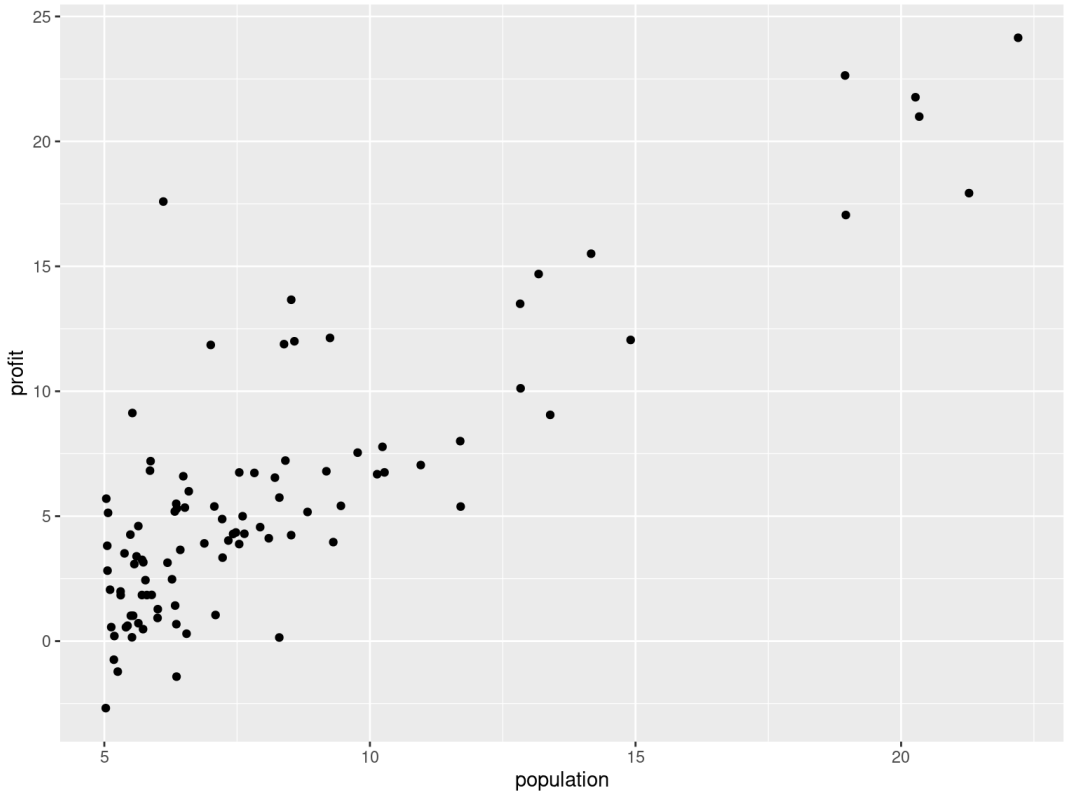
Mean Squared Error (MSE): $MSE = \dfrac{e_1^2 + e_2^2 + \ldots + e_n^2}{n}$ also known as the **loss/cost function**

Problem: Find b_0 and b_1 which minimizes MSE

Linear Regression: Estimating Parameters

A scatterplot of the dataset

```
ggplot(data = outlets) +
  geom_point(mapping = aes(x = population, y = profit))
```



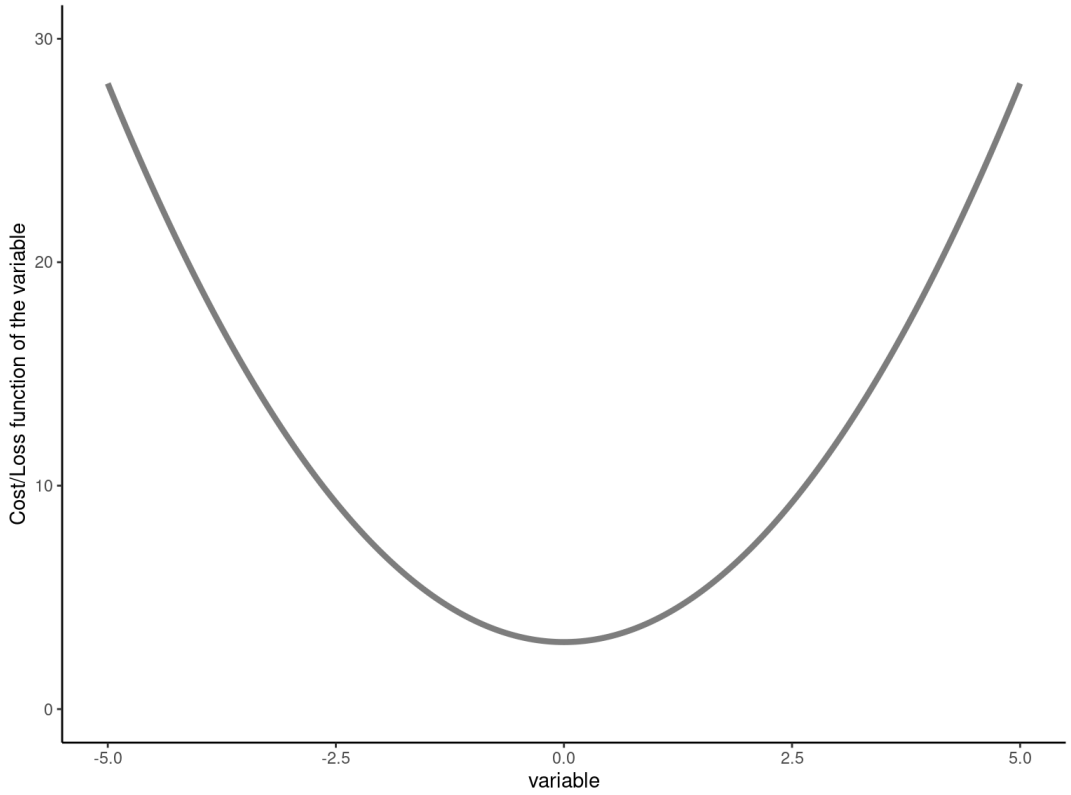
Gradient Descent Algorithm

How do we minimize the MSE?

One optimization technique we can use is called the **gradient descent**.

NOTE: Gradient Descent is not a machine learning technique. It is an optimization technique that helps to build machine learning models.

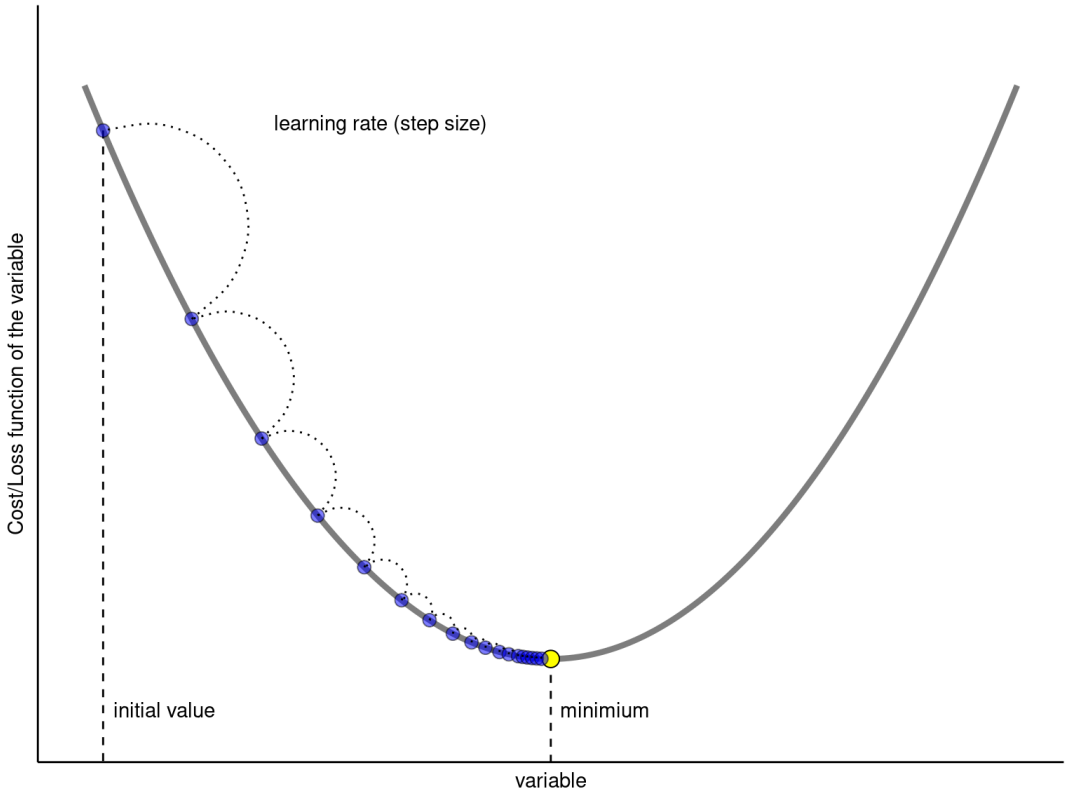
Gradient Descent Algorithm



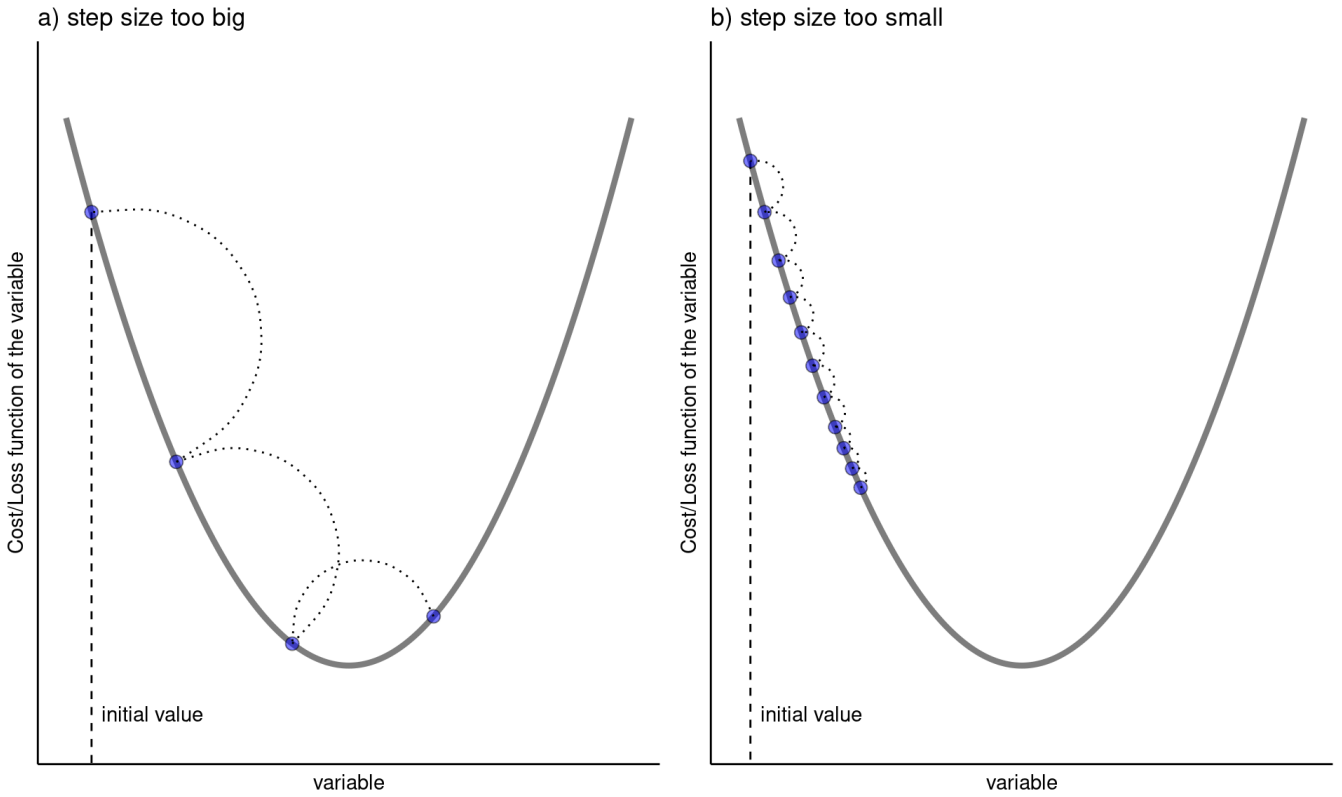
Updates to the variable:

$$\text{new value of variable} = \text{old value of variable} - \big(\text{step size} \times \text{gradient of function with respect to variable}\big)$$

Gradient Descent Algorithm



Gradient Descent Algorithm



Gradient Descent for Linear Regression

Objective: We want to find b_0 and b_1 which minimizes

$$MSE = \frac{e_1^2 + e_2^2 + \ldots + e_n^2}{n} = \frac{(\hat{y}_1 - y_1)^2 + (\hat{y}_2 - y_2)^2 + \ldots + (\hat{y}_n - y_n)^2}{n}$$

$$MSE = \frac{(b_0 + b_1\ x_1 - y_1)^2 + (b_0 + b_1\ x_2 - y_2)^2 + \ldots + (b_0 + b_1\ x_n - y_n)^2}{n}$$

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(b_0 + b_1\ x_i - y_i)^2 = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 = \frac{1}{n}\sum_{i=1}^{n}(e_i)^2$$

Gradient Descent for Linear Regression

To minimize MSE with respect to b_0 and b_1, we need the derivatives/gradients of MSE with respect to b_0 and b_1.

$$\text{gradient of MSE with respect to } b_0 = \frac{2}{n}\sum_{i=1}^{n}(b_0 + b_1\ x_i - y_i) = \frac{2}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)$$

$$\text{gradient of MSE with respect to } b_1 = \frac{2}{n}\sum_{i=1}^{n}x_i(b_0 + b_1\ x_i - y_i) = \frac{2}{n}\sum_{i=1}^{n}x_i(\hat{y}_i - y_i)$$

Gradient Descent for Linear Regression

The gradient descent updates will be as follows:

- For b_0

$$b_0 \text{ (new)} = b_0 \text{ (old)} - \left(\text{step size} \times \text{gradient with respect to } b_0 \right)$$

$$b_0 \text{ (new)} = b_0 \text{ (old)} - \left(\text{step size} \times \frac{2}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i) \right)$$

- For b_1

$$b_1 \text{ (new)} = b_1 \text{ (old)} - \left(\text{step size} \times \text{gradient with respect to } b_1 \right)$$

$$b_1 \text{ (new)} = b_1 \text{ (old)} - \left(\text{step size} \times \frac{2}{n} \sum_{i=1}^{n} x_i (\hat{y}_i - y_i) \right)$$

Let's now implement this algorithm in R.

Linear Regression in R

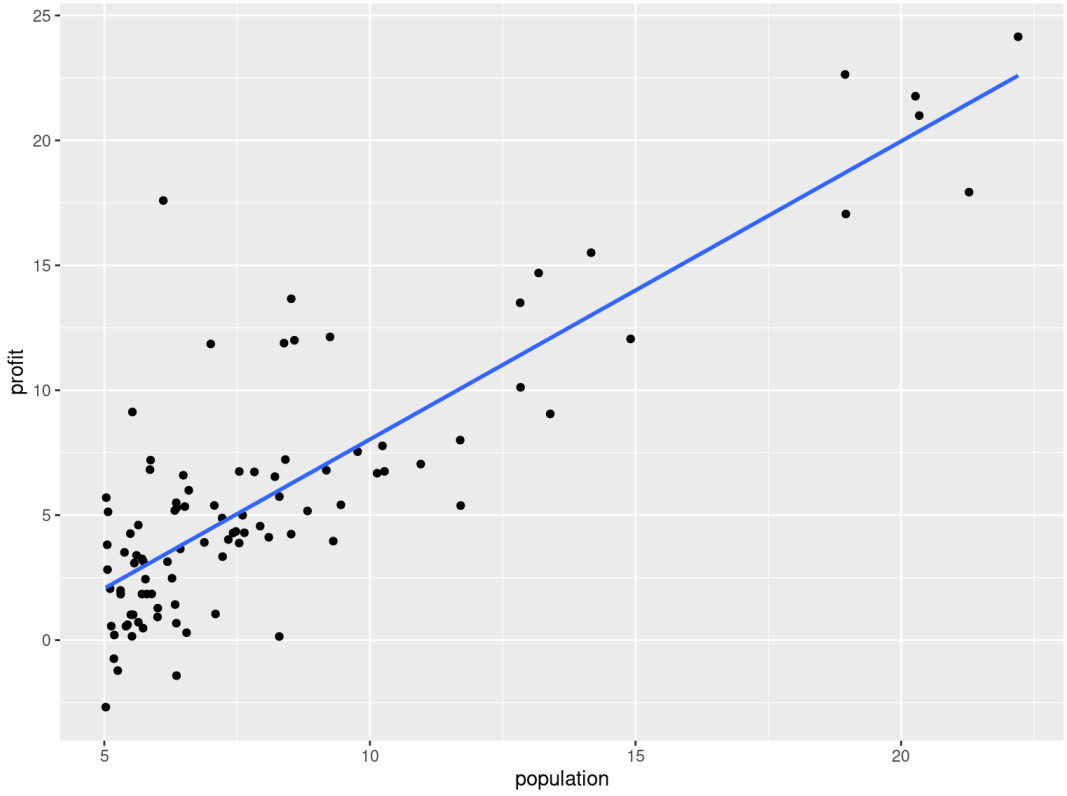
```
outlets_model <- lm(profit ~ population, data = outlets)

outlets_model
```

```
##
## Call:
## lm(formula = profit ~ population, data = outlets)
##
## Coefficients:
## (Intercept)   population
##      -3.896        1.193
```

Linear Regression in R

```
ggplot(data = outlets) +
  geom_point(mapping = aes(x = population, y = profit)) +    # create scatterplot
  geom_smooth(mapping = aes(x = population, y = profit), method = "lm", se = FALSE)   # add the regression line
```



Linear Regression in R: Prediction

```
predict(outlets_model, newdata = data.frame(population = 17))
```

```
##        1
## 16.38579
```