SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

10.020 Data Driven World

Multiple Linear Regression

Peng Song, ISTD

Week 9, Lesson 3, 2021

Linear Regression: Single Feature

- Single Feature
  - x: house size
- Target
  - y: house price
- Hypothesis
  - $y = \beta_0 + \beta_1 x$

Linear Regression: Single Feature

- Training data

Size in feet^2 (x)	Price ($) in 1000's (y)
2104	460
1416	232
1534	315
852	178
…	…

- Learned model

$- \ y = \hat{\beta}_0 + \hat{\beta}_1 x$

Linear Regression: Multiple Features

- Multiple features
  - x_1: house size
  - x_2: number of bedrooms
  - x_3: number of floors
  - x_4: age of home
- Target
  - y: house price
- Hypothesis
  - $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4$

Linear Regression: Multiple Features

Size in feet^2 (x_1)	Number of bedrooms (x_2)	Number of floors (x_3)	Age of home (years) (x_4)	Price ($) in 1000's (y)
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
…	…	…	…	…

Hypothesis: $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4$

Linear Regression: Multiple Features

TV x_1	radio x_2	newspaper x_3	sales y
230.1	37.8	69.2	22.1
44.5	39.3	45.1	10.4
17.2	45.9	69.3	9.3
151.5	41.3	58.5	18.5
180.8	10.8	58.4	12.9

Hypothesis: $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3$

Simple Linear Regression

Simple linear regression models the relationship between a **single** input variable and an output variable with a linear equation.

Input variable (1 feature)		Output variable (target)
x	$\longrightarrow$	y

Hypothesis: $y = \beta_0 + \beta_1 x$

Multiple Linear Regression

Multiple linear regression models the relationship between **multiple** input variables and an output variable with a linear equation.

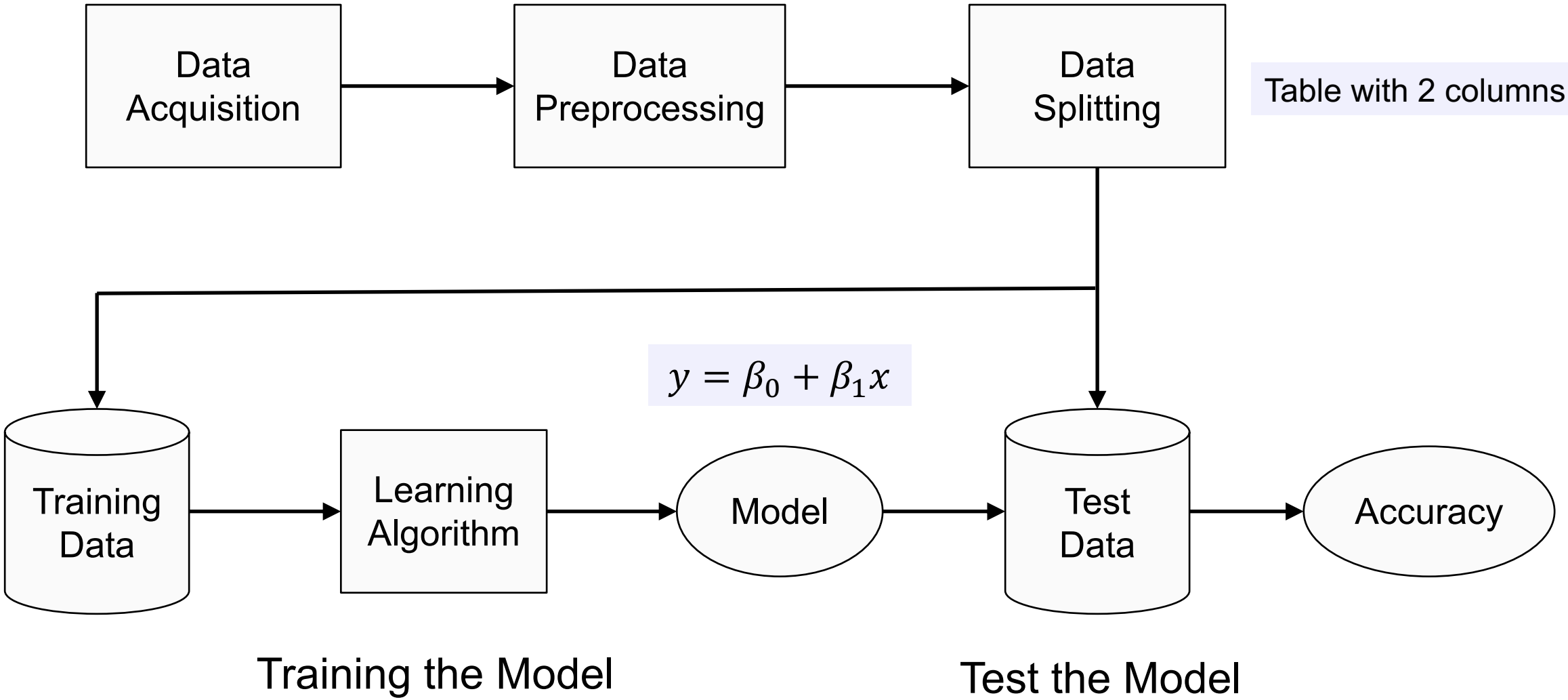
Input variables (n features) → Output variable (target)

$$x_1, x_2, \ldots, x_n \longrightarrow y$$

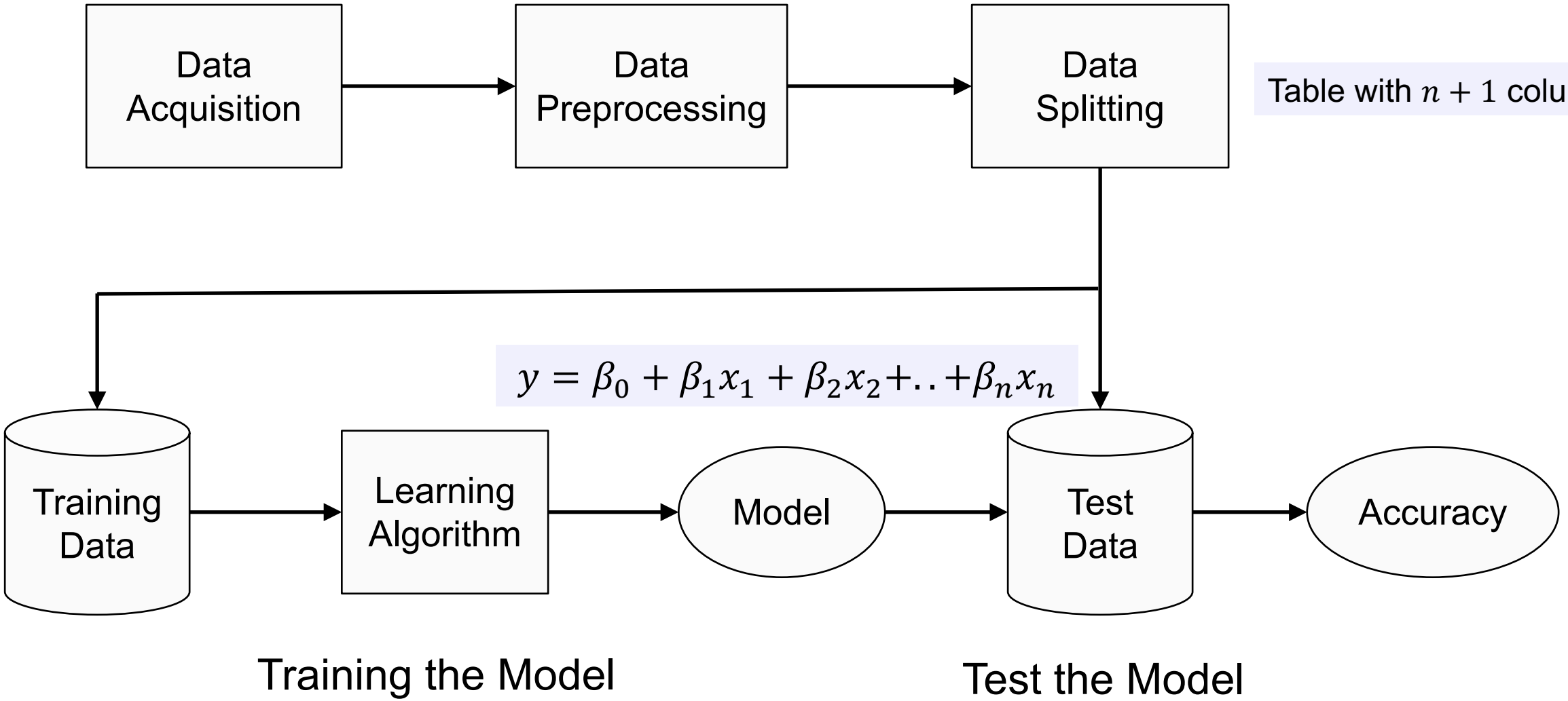
Hypothesis: $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + .. + \beta_n x_n$

How to determine the values of $\beta_0,\ \beta_1, \beta_2, .., \beta_n$?

Simple Linear Regression



Multiple Linear Regression



Simple Linear Regression

How about equations for multiple linear regression?

- Training the model
  - Hypothesis: $\hat{\boldsymbol{y}} = \boldsymbol{X} \times \boldsymbol{b}$
  - Cost Function: $J(\beta_0, \beta_1) = \frac{1}{2m} (\hat{\boldsymbol{y}} - \boldsymbol{y})^{\mathrm{T}} \times (\hat{\boldsymbol{y}} - \boldsymbol{y})$
  - Gradient Descent: $\boldsymbol{b} = \boldsymbol{b} - \alpha \frac{1}{m} \boldsymbol{X}^{\mathrm{T}} \times (\boldsymbol{X} \times \boldsymbol{b} - \boldsymbol{y})$

- Testing the model
  - Mean Squared Error: $MSE = \frac{1}{n} \sum_{i=1}^{n} \left(y^i - \hat{y}^i \right)^2$
  - R^2 Coefficient of Determination: $R^2 = 1 - \frac{\sum_{i=1}^{n} \left(y^i - \hat{y}^i \right)^2}{\sum_{i=1}^{n} \left(y^i - \bar{y} \right)^2}$

Hypothesis

Previously ($\mathbf{1}$ feature):

$$y = \beta_0 + \beta_1 x$$

Now ($\boldsymbol{n}$ features):

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + .. + \beta_n x_n$$

Hypothesis

Previously ($\mathbf{1}$ feature):

$$y = [1 \quad x] \times \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix}$$

$$\boldsymbol{x} = [1 \quad x] \qquad \boldsymbol{b} = \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix}$$

Now ($\boldsymbol{n}$ features):

$$y = [1 \quad x_1 \quad .. \quad x_n] \times \begin{bmatrix} \beta_0 \\ \beta_1 \\ .. \\ \beta_n \end{bmatrix}$$

$$\boldsymbol{x} = [1 \quad x_1 \quad .. \quad x_n] \qquad \boldsymbol{b} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ .. \\ \beta_n \end{bmatrix}$$

Hypothesis

Previously ($\mathbf{1}$ feature):

$$y = \boldsymbol{x} \times \boldsymbol{b}$$

where

$$\boldsymbol{x} = [1 \quad x] \qquad \boldsymbol{b} = \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix}$$

Now ($\boldsymbol{n}$ features):

$$y = \boldsymbol{x} \times \boldsymbol{b}$$

where

$$\boldsymbol{x} = [1 \quad x_1 \quad \cdots \quad x_n] \qquad \boldsymbol{b} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_n \end{bmatrix}$$

Assume $x_0 = 1$ with β_0 as its coefficient

Hypothesis

Previously ($\mathbf{1}$ feature):

$$y = \boldsymbol{x} \times \boldsymbol{b}$$

where

$$\boldsymbol{x} = [1 \quad x] \qquad \boldsymbol{b} = \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix}$$

Now ($\boldsymbol{n}$ features):

$$y = \boldsymbol{x} \times \boldsymbol{b}$$

where

$$\boldsymbol{x} = [x_0 \quad x_1 \quad .. \quad x_n] \qquad \boldsymbol{b} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ .. \\ \beta_n \end{bmatrix}$$

Assume $x_0 = 1$ with β_0 as its coefficient

Prediction: 1 Data Point

Previously ($\mathbf{1}$ feature):

$$\hat{y}^{(i)} = \boldsymbol{x}^{(i)} \times \boldsymbol{b}$$

where

$$\boldsymbol{x}^{(i)} = [1 \quad x^{(i)}] \qquad \boldsymbol{b} = \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix}$$

Now ($\boldsymbol{n}$ features):

ith data point

$$\hat{y}^{(i)} = \boldsymbol{x}^{(i)} \times \boldsymbol{b}$$

where

$$\boldsymbol{x}^{(i)} = \begin{bmatrix} x_0^{(i)} & x_1^{(i)} & .. & x_n^{(i)} \end{bmatrix} \quad \boldsymbol{b} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ .. \\ \beta_n \end{bmatrix}$$

Prediction: m Data Points

Previously ($\mathbf{1}$ feature):

$$\hat{\boldsymbol{y}} = \boldsymbol{X} \times \boldsymbol{b}$$

where

$$\hat{\boldsymbol{y}} = \begin{bmatrix} \hat{y}^{(1)} \\ \hat{y}^{(2)} \\ .. \\ \hat{y}^{(m)} \end{bmatrix} \quad \boldsymbol{X} = \begin{bmatrix} 1 & x^{(1)} \\ 1 & x^{(2)} \\ .. & .. \\ 1 & x^{(m)} \end{bmatrix} \quad \boldsymbol{b} = \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix}$$

Now ($\boldsymbol{n}$ features):

$$\hat{\boldsymbol{y}} = \boldsymbol{X} \times \boldsymbol{b}$$

where

$$\boldsymbol{X} = \begin{bmatrix} x_0^{(1)} & x_1^{(1)} & \vdots & x_n^{(1)} \\ x_0^{(2)} & x_1^{(2)} & \vdots & x_n^{(2)} \\ .. & .. & .. & .. \\ x_0^{(m)} & x_1^{(m)} & \vdots & x_n^{(m)} \end{bmatrix} \quad \boldsymbol{b} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ .. \\ \beta_n \end{bmatrix}$$

Cost Function

Previously ($\mathbf{1}$ feature):

$$J(\beta_0, \beta_1) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\beta\left(x^{(i)}\right) - y^{(i)}\right)^2$$

Now ($\boldsymbol{n}$ features):

$$J(\beta_0, \beta_1, .., \beta_n) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\beta\left(\boldsymbol{x}^{(i)}\right) - y^{(i)}\right)^2$$

Cost Function

Previously ($\mathbf{1}$ feature):

$$J(\beta_0, \beta_1) = \frac{1}{2m}\sum_{i=1}^{m}\left(\hat{y}^{(i)} - y^{(i)}\right)^2$$

where

$$\hat{y}^{(i)} = \boldsymbol{x}^{(i)} \times \boldsymbol{b}$$

Now ($\boldsymbol{n}$ features):

$$J(\beta_0, \beta_1, .., \beta_n) = \frac{1}{2m}\sum_{i=1}^{m}\left(\hat{y}^{(i)} - y^{(i)}\right)^2$$

where

$$\hat{y}^{(i)} = \boldsymbol{x}^{(i)} \times \boldsymbol{b}$$

Cost Function

Previously ($\mathbf{1}$ feature):

$$J(\boldsymbol{b}) = \frac{1}{2m}(\hat{\boldsymbol{y}} - \boldsymbol{y})^{\mathrm{T}} \times (\hat{\boldsymbol{y}} - \boldsymbol{y})$$

where

$$\hat{\boldsymbol{y}} = \boldsymbol{X} \times \boldsymbol{b}$$

Now ($\boldsymbol{n}$ features):

$$J(\boldsymbol{b}) = \frac{1}{2m}(\hat{\boldsymbol{y}} - \boldsymbol{y})^{\mathrm{T}} \times (\hat{\boldsymbol{y}} - \boldsymbol{y})$$

where

$$\hat{\boldsymbol{y}} = \boldsymbol{X} \times \boldsymbol{b}$$

What are the differences between these two equations?

Cost Function

Previously ($\mathbf{1}$ feature):

$$J(\boldsymbol{b}) = \frac{1}{2m}(\hat{\boldsymbol{y}} - \boldsymbol{y})^{\mathrm{T}} \times (\hat{\boldsymbol{y}} - \boldsymbol{y})$$

where

$$\hat{\boldsymbol{y}} = \boldsymbol{X} \times \boldsymbol{b}$$

$\boldsymbol{X}$ is a $m \times 2$ matrix
$\boldsymbol{b}$ is a 2×1 matrix

Now ($\boldsymbol{n}$ features):

$$J(\boldsymbol{b}) = \frac{1}{2m}(\hat{\boldsymbol{y}} - \boldsymbol{y})^{\mathrm{T}} \times (\hat{\boldsymbol{y}} - \boldsymbol{y})$$

where

$$\hat{\boldsymbol{y}} = \boldsymbol{X} \times \boldsymbol{b}$$

$\boldsymbol{X}$ is a $m \times (n+1)$ matrix
$\boldsymbol{b}$ is a $(n+1) \times 1$ matrix

Gradient Descent

Previously ($\mathbf{1}$ feature):

Start with some $\beta_0,\ \beta_1$

Repeat until convergence

{

$$\beta_0 := \beta_0 - \alpha \frac{\partial}{\partial \beta_0} J(\beta_0, \beta_1)$$

$$\beta_1 := \beta_1 - \alpha \frac{\partial}{\partial \beta_1} J(\beta_0, \beta_1)$$

}

Now ($\boldsymbol{n}$ features):

Start with some $\beta_0,\ \beta_1, .., \beta_n$

Repeat until convergence

{

$$\beta_0 := \beta_0 - \alpha \frac{\partial}{\partial \beta_0} J(\beta_0, \beta_1, .., \beta_n)$$

$$\beta_1 := \beta_1 - \alpha \frac{\partial}{\partial \beta_1} J(\beta_0, \beta_1, .., \beta_n)$$

…

$$\beta_n := \beta_n - \alpha \frac{\partial}{\partial \beta_n} J(\beta_0, \beta_1, .., \beta_n)$$

}

Gradient Descent

Previously ($\mathbf{1}$ feature):

Start with some $\beta_0,\ \beta_1$

Repeat until convergence
{

$$\beta_j := \beta_j - \alpha \frac{\partial}{\partial \beta_j} J(\beta_0, \beta_1)$$

for any $0 \leq j \leq 1$
}

Now ($\boldsymbol{n}$ features):

Start with some $\beta_0,\ \beta_1, .., \beta_n$

Repeat until convergence
{

$$\beta_j := \beta_j - \alpha \frac{\partial}{\partial \beta_j} J(\beta_0, \beta_1, .., \beta_n)$$

for any $0 \leq j \leq n$
}

Gradient Descent

Previously ($\mathbf{1}$ feature):

$$\frac{\partial}{\partial \beta_0} J(\beta_0, \beta_1) = \frac{1}{m} \sum_{i=1}^{m} \left(h_\beta\left(x^{(i)}\right) - y^{(i)} \right)$$

$$\frac{\partial}{\partial \beta_1} J(\beta_0, \beta_1) = \frac{1}{m} \sum_{i=1}^{m} \left(h_\beta\left(x^{(i)}\right) - y^{(i)} \right) x^{(i)}$$

Now ($\boldsymbol{n}$ features):

Recall that $x_0 = 1$

$$\frac{\partial}{\partial \beta_0} J(\beta_0, \beta_1, .., \beta_n) = \frac{1}{m} \sum_{i=1}^{m} \left(h_\beta\left(\boldsymbol{x}^{(i)}\right) - y^{(i)} \right)$$

$$\frac{\partial}{\partial \beta_1} J(\beta_0, \beta_1, .., \beta_n) = \frac{1}{m} \sum_{i=1}^{m} \left(h_\beta\left(\boldsymbol{x}^{(i)}\right) - y^{(i)} \right) x_1^{(i)}$$

…

$$\frac{\partial}{\partial \beta_n} J(\beta_0, \beta_1, .., \beta_n) = \frac{1}{m} \sum_{i=1}^{m} \left(h_\beta\left(\boldsymbol{x}^{(i)}\right) - y^{(i)} \right) x_n^{(i)}$$

Gradient Descent

Previously ($\mathbf{1}$ feature):

$$\frac{\partial}{\partial \beta_0} J(\beta_0, \beta_1) = \frac{1}{m} \sum_{i=1}^{m} \left(h_\beta\left(\boldsymbol{x}^{(i)}\right) - y^{(i)} \right) x_0^{(i)}$$

$$\frac{\partial}{\partial \beta_1} J(\beta_0, \beta_1) = \frac{1}{m} \sum_{i=1}^{m} \left(h_\beta\left(\boldsymbol{x}^{(i)}\right) - y^{(i)} \right) x_1^{(i)}$$

Now ($\boldsymbol{n}$ features):

$$\frac{\partial}{\partial \beta_0} J(\beta_0, \beta_1, .., \beta_n) = \frac{1}{m} \sum_{i=1}^{m} \left(h_\beta\left(\boldsymbol{x}^{(i)}\right) - y^{(i)} \right) x_0^{(i)}$$

$$\frac{\partial}{\partial \beta_1} J(\beta_0, \beta_1, .., \beta_n) = \frac{1}{m} \sum_{i=1}^{m} \left(h_\beta\left(\boldsymbol{x}^{(i)}\right) - y^{(i)} \right) x_1^{(i)}$$

…

$$\frac{\partial}{\partial \beta_n} J(\beta_0, \beta_1, .., \beta_n) = \frac{1}{m} \sum_{i=1}^{m} \left(h_\beta\left(\boldsymbol{x}^{(i)}\right) - y^{(i)} \right) x_n^{(i)}$$

Gradient Descent

Previously ($\mathbf{1}$ feature):

$$\frac{\partial}{\partial \beta_j} J(\beta_0, \beta_1) = \frac{1}{m} \sum_{i=1}^{m} \left(h_\beta\left(\boldsymbol{x}^{(i)}\right) - y^{(i)} \right) x_j^{(i)}$$

for any $0 \leq j \leq 1$

Now ($\boldsymbol{n}$ features):

$$\frac{\partial}{\partial \beta_j} J(\beta_0, \beta_1, .., \beta_n) = \frac{1}{m} \sum_{i=1}^{m} \left(h_\beta\left(\boldsymbol{x}^{(i)}\right) - y^{(i)} \right) x_j^{(i)}$$

for any $0 \leq j \leq n$

Gradient Descent

Previously ($\mathbf{1}$ feature):

Start with some $\beta_0,\ \beta_1$

Repeat until convergence

{

$$\beta_j := \beta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\beta\left(\boldsymbol{x}^{(i)}\right) - y^{(i)}\right) x_j^{(i)}$$

for any $0 \leq j \leq 1$

}

Now ($\boldsymbol{n}$ features):

Start with some $\beta_0,\ \beta_1, \ldots, \beta_n$

Repeat until convergence

{

$$\beta_j := \beta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\beta\left(\boldsymbol{x}^{(i)}\right) - y^{(i)}\right) x_j^{(i)}$$

for any $0 \leq j \leq n$

}

Gradient Descent

Previously ($\mathbf{1}$ feature):

$$\boldsymbol{b} = \boldsymbol{b} - \alpha \frac{1}{m} \boldsymbol{X}^{\mathrm{T}} \times (\boldsymbol{X} \times \boldsymbol{b} - \boldsymbol{y})$$

$\boldsymbol{X}$ is a $m \times 2$ matrix
$\boldsymbol{b}$ is a 2×1 matrix

Now ($\boldsymbol{n}$ features):

$$\boldsymbol{b} = \boldsymbol{b} - \alpha \frac{1}{m} \boldsymbol{X}^{\mathrm{T}} \times (\boldsymbol{X} \times \boldsymbol{b} - \boldsymbol{y})$$

$\boldsymbol{X}$ is a $m \times (n+1)$ matrix
$\boldsymbol{b}$ is a $(n+1) \times 1$ matrix

Linear Regression

- **Simple** Linear Regression
  - Hypothesis: $\hat{\boldsymbol{y}} = \boldsymbol{X} \times \boldsymbol{b}$
  - Cost Function: $J(\beta_0, \beta_1) = \frac{1}{2m} (\hat{\boldsymbol{y}} - \boldsymbol{y})^{\mathrm{T}} \times (\hat{\boldsymbol{y}} - \boldsymbol{y})$
  - Gradient Descent: $\boldsymbol{b} = \boldsymbol{b} - \alpha \frac{1}{m} \boldsymbol{X}^{\mathrm{T}} \times (\boldsymbol{X} \times \boldsymbol{b} - \boldsymbol{y})$

$\boldsymbol{X}$ is a $m \times 2$ matrix
$\boldsymbol{b}$ is a 2×1 matrix

- **Multiple** Linear Regression
  - Hypothesis: $\hat{\boldsymbol{y}} = \boldsymbol{X} \times \boldsymbol{b}$
  - Cost Function: $J(\beta_0, \beta_1, .., \beta_n) = \frac{1}{2m} (\hat{\boldsymbol{y}} - \boldsymbol{y})^{\mathrm{T}} \times (\hat{\boldsymbol{y}} - \boldsymbol{y})$
  - Gradient Descent: $\boldsymbol{b} = \boldsymbol{b} - \alpha \frac{1}{m} \boldsymbol{X}^{\mathrm{T}} \times (\boldsymbol{X} \times \boldsymbol{b} - \boldsymbol{y})$

$\boldsymbol{X}$ is a $m \times (n+1)$ matrix
$\boldsymbol{b}$ is a $(n+1) \times 1$ matrix

Linear Regression

- **Simple** Linear Regression
  - Hypothesis: $\hat{\boldsymbol{y}} = \boldsymbol{X} \times \boldsymbol{b}$
  - Cost Function: $J(\beta_0, \beta_1) = \frac{1}{2m} (\hat{\boldsymbol{y}} - \boldsymbol{y})^{\mathrm{T}} \times (\hat{\boldsymbol{y}} - \boldsymbol{y})$
  - Gradient Descent: $\boldsymbol{b} = \boldsymbol{b} - \alpha \frac{1}{m} \boldsymbol{X}^{\mathrm{T}} \times (\boldsymbol{X} \times \boldsymbol{b} - \boldsymbol{y})$

$\boldsymbol{X}$ is a $m \times (1+1)$ matrix
$\boldsymbol{b}$ is a $(1+1) \times 1$ matrix

- **Multiple** Linear Regression
  - Hypothesis: $\hat{\boldsymbol{y}} = \boldsymbol{X} \times \boldsymbol{b}$
  - Cost Function: $J(\beta_0, \beta_1, .., \beta_n) = \frac{1}{2m} (\hat{\boldsymbol{y}} - \boldsymbol{y})^{\mathrm{T}} \times (\hat{\boldsymbol{y}} - \boldsymbol{y})$
  - Gradient Descent: $\boldsymbol{b} = \boldsymbol{b} - \alpha \frac{1}{m} \boldsymbol{X}^{\mathrm{T}} \times (\boldsymbol{X} \times \boldsymbol{b} - \boldsymbol{y})$

$\boldsymbol{X}$ is a $m \times (n+1)$ matrix
$\boldsymbol{b}$ is a $(n+1) \times 1$ matrix

Linear Regression

- **Simple** Linear Regression
  - Hypothesis: $\hat{\boldsymbol{y}} = \boldsymbol{X} \times \boldsymbol{b}$
  - Cost Function: $J(\beta_0, \beta_1) = \frac{1}{2m} (\hat{\boldsymbol{y}} - \boldsymbol{y})^{\mathrm{T}} \times (\hat{\boldsymbol{y}} - \boldsymbol{y})$
  - Gradient Descent: $\boldsymbol{b} = \boldsymbol{b} - \alpha \frac{1}{m} \boldsymbol{X}^{\mathrm{T}} \times (\boldsymbol{X} \times \boldsymbol{b} - \boldsymbol{y})$

Simple linear regression is a **special case** of multiple linear regression with $n = 1$

- **Multiple** Linear Regression
  - Hypothesis: $\hat{\boldsymbol{y}} = \boldsymbol{X} \times \boldsymbol{b}$
  - Cost Function: $J(\beta_0, \beta_1, .., \beta_n) = \frac{1}{2m} (\hat{\boldsymbol{y}} - \boldsymbol{y})^{\mathrm{T}} \times (\hat{\boldsymbol{y}} - \boldsymbol{y})$
  - Gradient Descent: $\boldsymbol{b} = \boldsymbol{b} - \alpha \frac{1}{m} \boldsymbol{X}^{\mathrm{T}} \times (\boldsymbol{X} \times \boldsymbol{b} - \boldsymbol{y})$

Simple or Multiple Linear Regression

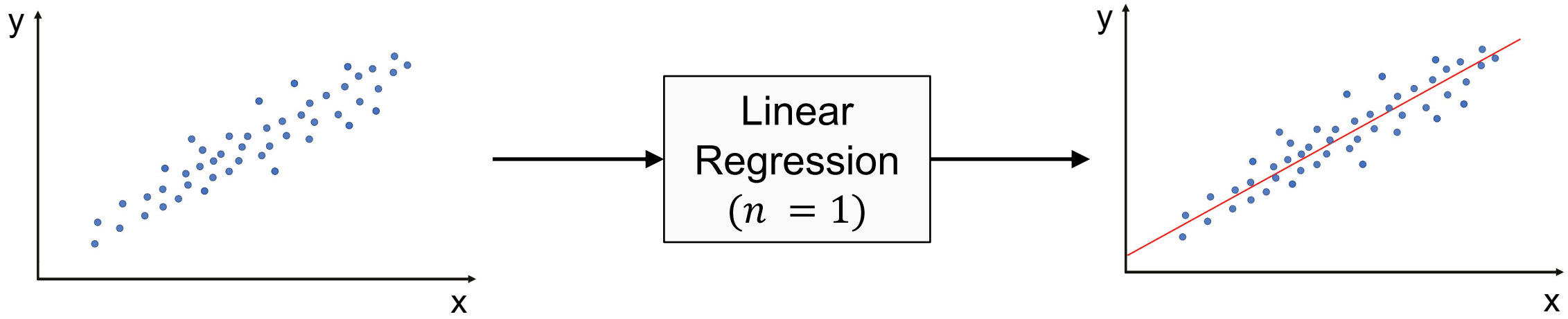
- Training the model
  - Hypothesis: $\hat{\boldsymbol{y}} = \boldsymbol{X} \times \boldsymbol{b}$
  - Cost Function: $J(\beta_0, \beta_1 .., \beta_n) = \frac{1}{2m} (\hat{\boldsymbol{y}} - \boldsymbol{y})^{\mathrm{T}} \times (\hat{\boldsymbol{y}} - \boldsymbol{y})$
  - Gradient Descent: $\boldsymbol{b} = \boldsymbol{b} - \alpha \frac{1}{m} \boldsymbol{X}^{\mathrm{T}} \times (\boldsymbol{X} \times \boldsymbol{b} - \boldsymbol{y})$

- Testing the model
  - Mean Squared Error: $MSE = \frac{1}{n} \sum_{i=1}^{n} \left(y^i - \hat{y}^i \right)^2$
  - R^2 Coefficient of Determination: $R^2 = 1 - \frac{\sum_{i=1}^{n} \left(y^i - \hat{y}^i \right)^2}{\sum_{i=1}^{n} \left(y^i - \bar{y} \right)^2}$

Linear Regression (n=1)

A **simple** linear regression model with $n = 1$ features fits a regression "**line**" in 2 dimensional space.

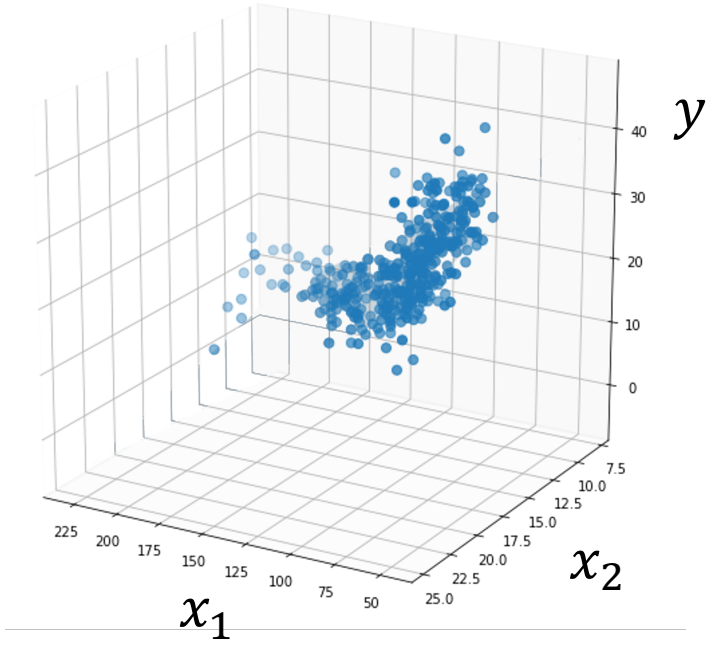
$$y = \hat{\beta}_0 + \hat{\beta}_1 x$$



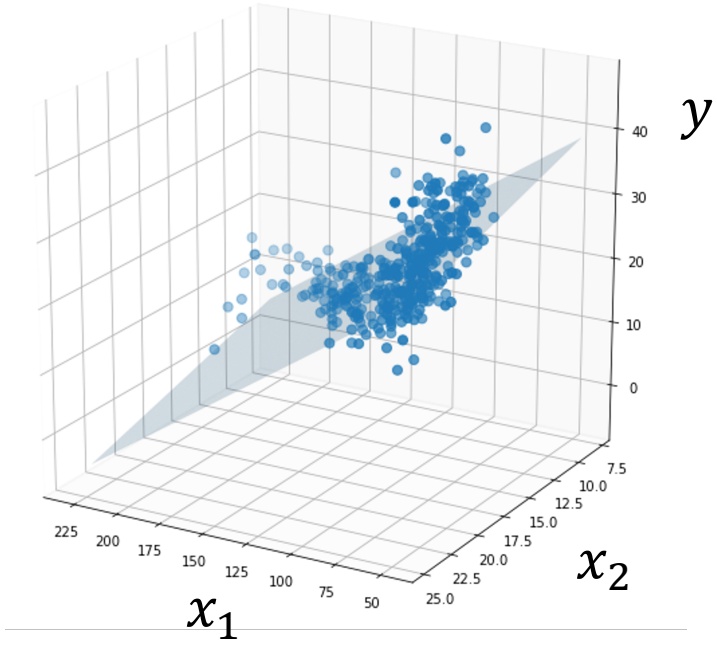
Linear Regression (n=2)

A **multiple** linear regression model with $n = 2$ features fits a regression "**surface**" in 3 dimensional space.

$$y = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2$$

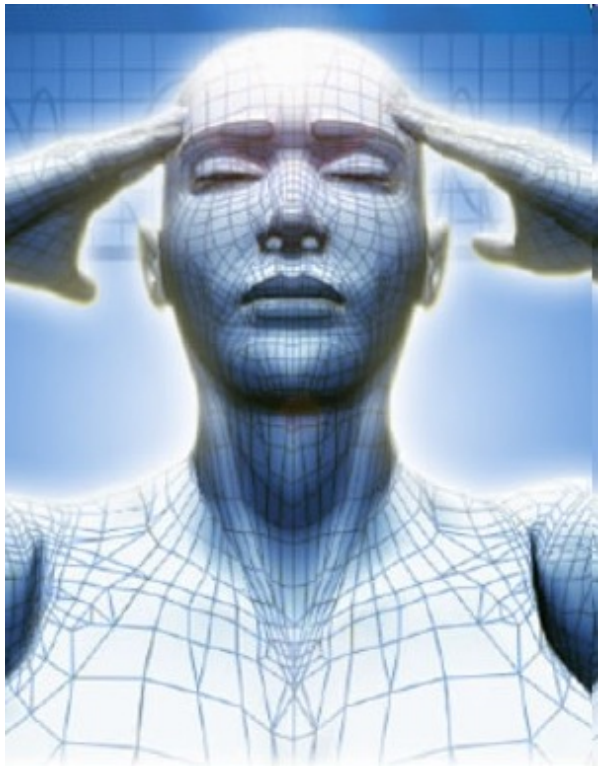


Linear Regression ($n = 2$)



Linear Regression (n>2)

A **multiple** linear regression model with $n > 2$ features fits a regression "**surface**" in $n + 1$ dimensional space.



High dimensional space
cannot be directly visualized.

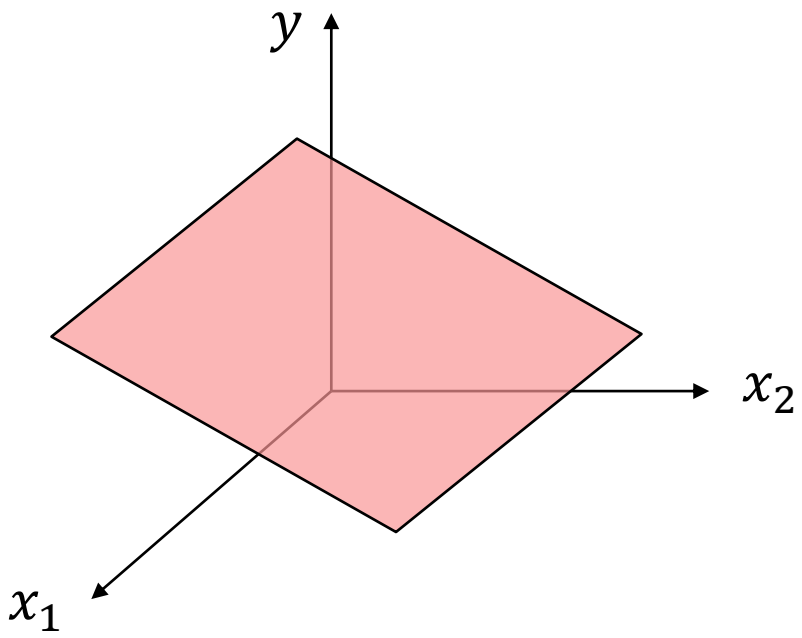
Linear Regression (n=2)

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2$$

set
$x_1 = x$
$x_2 = x^2$

$$y = \beta_0 + \beta_1 x + \beta_2 x^2$$

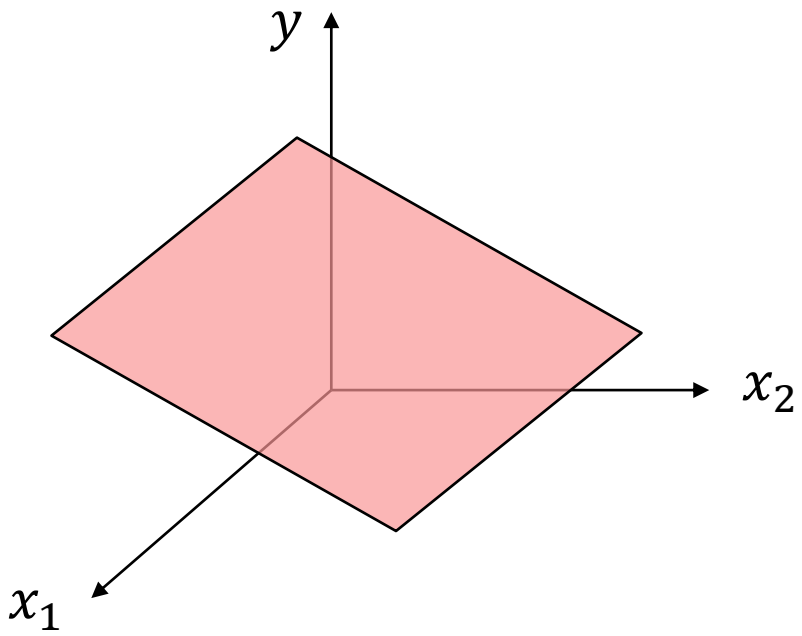
What is the geometric meaning of this equation?



3D plane

Linear Regression (n=2)

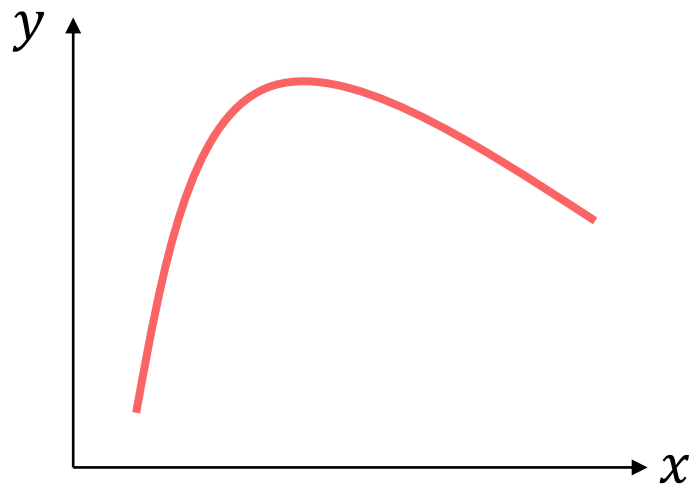
$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2$$



3D plane

Non-linear relation between x and y

$$y = \beta_0 + \beta_1 x + \beta_2 x^2$$



2D polynomial curve

Polynomial Linear Regression

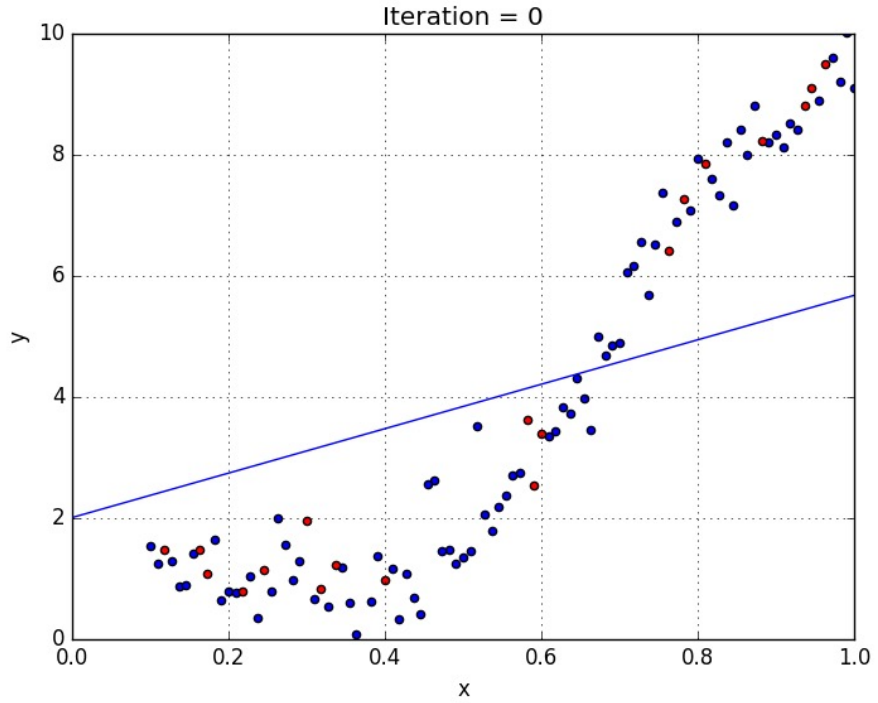
Polynomial Linear Regression is a special case of **Multiple** linear regression which estimates the relationship between a feature x and a target y as an nth degree polynomial.

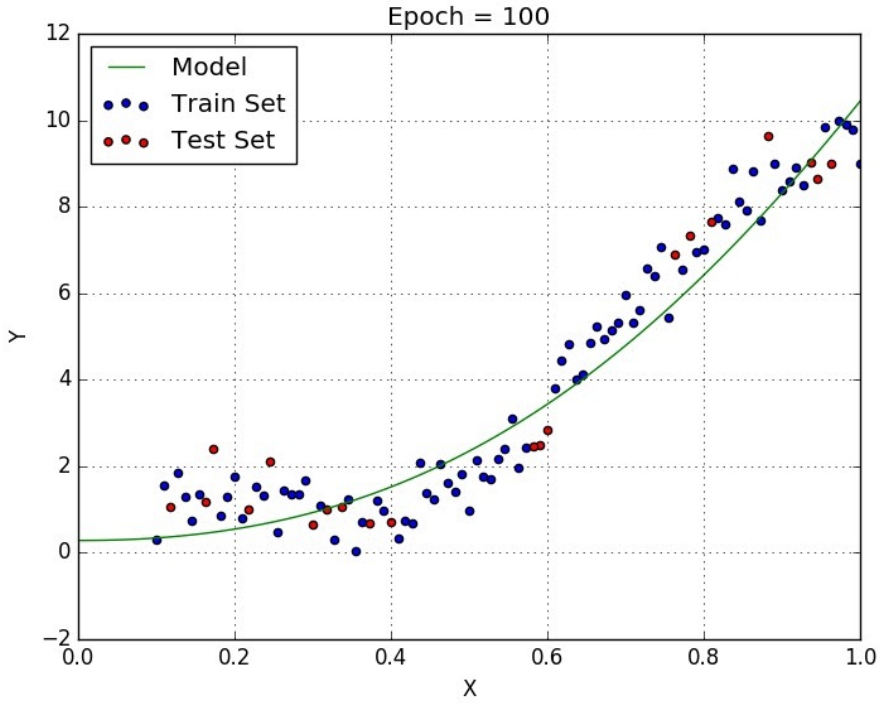
Multiple linear regression : $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + .. + \beta_n x_n$

Polynomial linear regression: $y = \beta_0 + \beta_1 x + \beta_2 x^2 + .. + \beta_n x^n$

Polynomial Linear Regression

Polynomial Linear Regression provides a better approximation of the relationship between the feature and target.





Polynomial Linear Regression

- Training the model
  - Hypothesis: $\hat{\boldsymbol{y}} = \boldsymbol{X} \times \boldsymbol{b}$
  - Cost Function: $J(\beta_0, \beta_1 .., \beta_n) = \frac{1}{2m} (\hat{\boldsymbol{y}} - \boldsymbol{y})^{\mathrm{T}} \times (\hat{\boldsymbol{y}} - \boldsymbol{y})$
  - Gradient Descent: $\boldsymbol{b} = \boldsymbol{b} - \alpha \frac{1}{m} \boldsymbol{X}^{\mathrm{T}} \times (\boldsymbol{X} \times \boldsymbol{b} - \boldsymbol{y})$

What is the difference between **multiple** linear regression and **polynomial** linear regression?

Polynomial Linear Regression

- Training the model
  - Hypothesis: $\hat{\boldsymbol{y}} = \boldsymbol{X} \times \boldsymbol{b}$
  - Cost Function: $J(\beta_0, \beta_1 .., \beta_n) = \frac{1}{2m} (\hat{\boldsymbol{y}} - \boldsymbol{y})^{\mathrm{T}} \times (\hat{\boldsymbol{y}} - \boldsymbol{y})$
  - Gradient Descent: $\boldsymbol{b} = \boldsymbol{b} - \alpha \frac{1}{m} \boldsymbol{X}^{\mathrm{T}} \times (\boldsymbol{X} \times \boldsymbol{b} - \boldsymbol{y})$

Multiple linear regression

$$\boldsymbol{X} = \begin{bmatrix} x_0^{(1)} & x_1^{(1)} & \vdots & x_n^{(1)} \\ x_0^{(2)} & x_1^{(2)} & \vdots & x_n^{(2)} \\ \vdots & \vdots & \vdots & \vdots \\ x_0^{(m)} & x_1^{(m)} & & x_n^{(m)} \end{bmatrix}$$

Polynomial linear regression

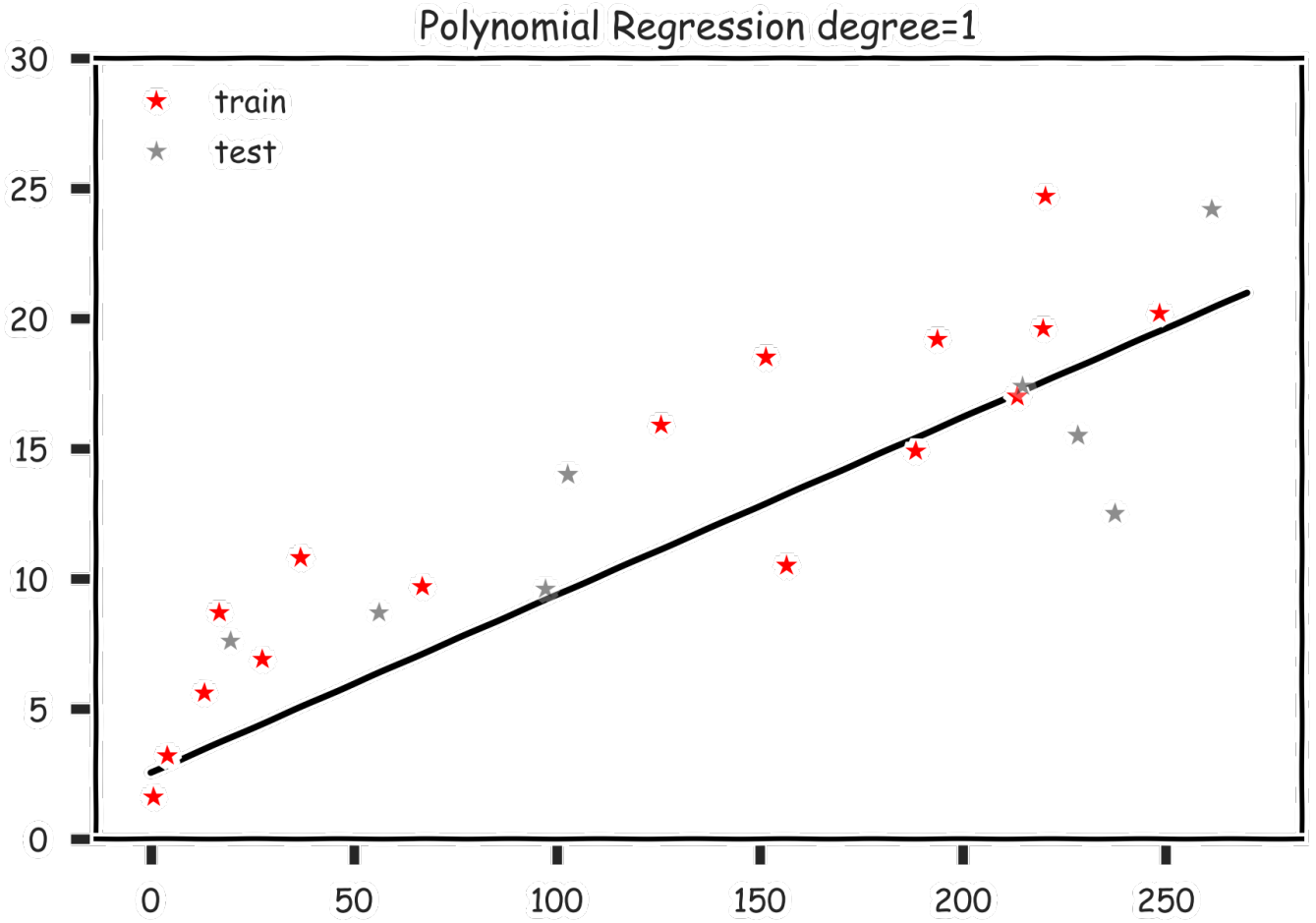
$$\boldsymbol{X} = \begin{bmatrix} 1 & x^{(1)} & \vdots & (x^n)^{(1)} \\ 1 & x^{(2)} & \vdots & (x^n)^{(2)} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x^{(m)} & & (x^n)^{(m)} \end{bmatrix}$$

Polynomial Linear Regression

- Training the model
  - Hypothesis: $\hat{\boldsymbol{y}} = \boldsymbol{X} \times \boldsymbol{b}$
  - Cost Function: $J(\beta_0, \beta_1 .., \beta_n) = \frac{1}{2m} (\hat{\boldsymbol{y}} - \boldsymbol{y})^{\mathrm{T}} \times (\hat{\boldsymbol{y}} - \boldsymbol{y})$
  - Gradient Descent: $\boldsymbol{b} = \boldsymbol{b} - \alpha \frac{1}{m} \boldsymbol{X}^{\mathrm{T}} \times (\boldsymbol{X} \times \boldsymbol{b} - \boldsymbol{y})$

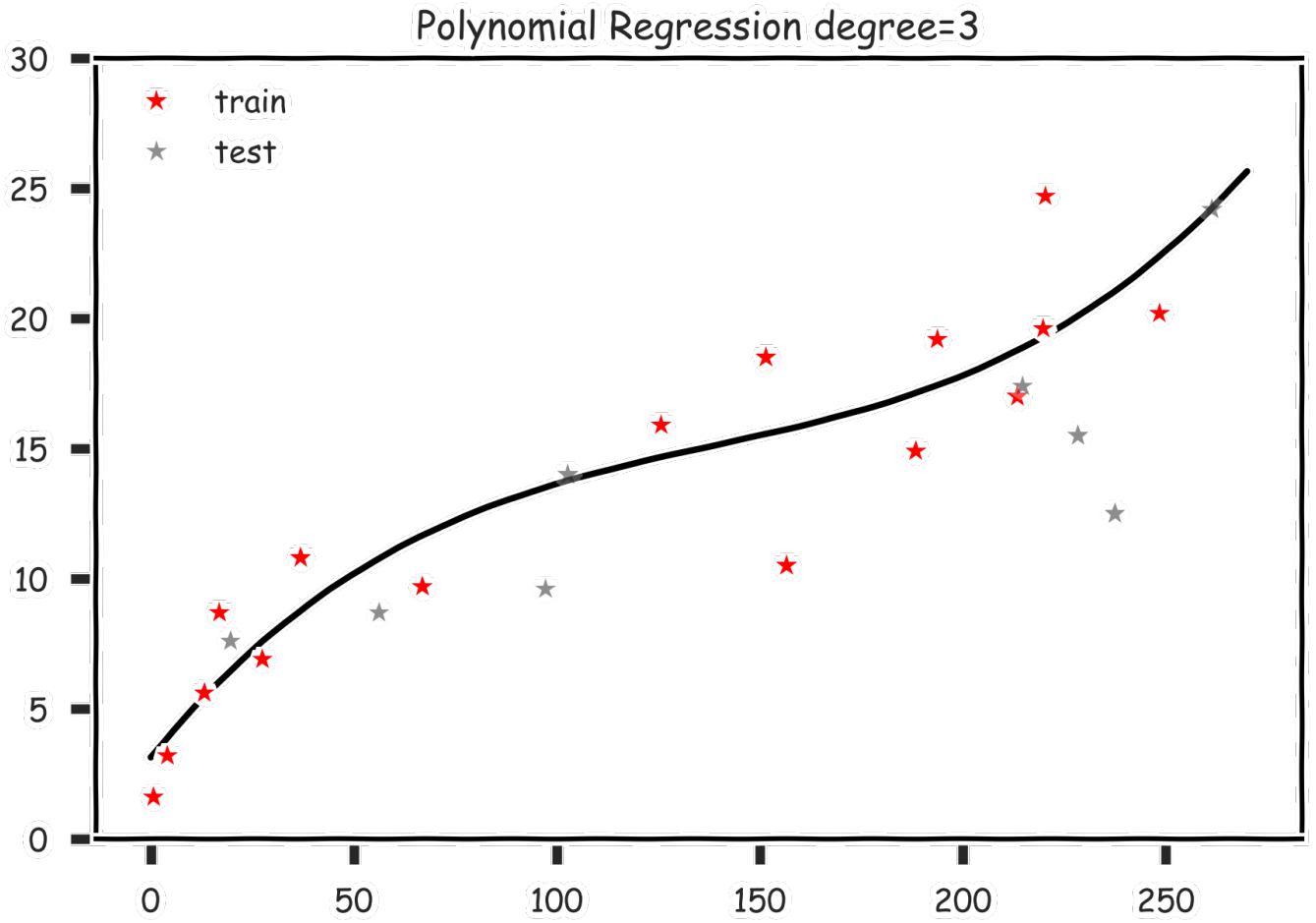
- Testing the model
  - Mean Squared Error: $MSE = \frac{1}{n} \sum_{i=1}^{n} \left(y^i - \hat{y}^i \right)^2$
  - R^2 Coefficient of Determination: $R^2 = 1 - \frac{\sum_{i=1}^{n} \left(y^i - \hat{y}^i \right)^2}{\sum_{i=1}^{n} \left(y^i - \bar{y} \right)^2}$

Overfitting



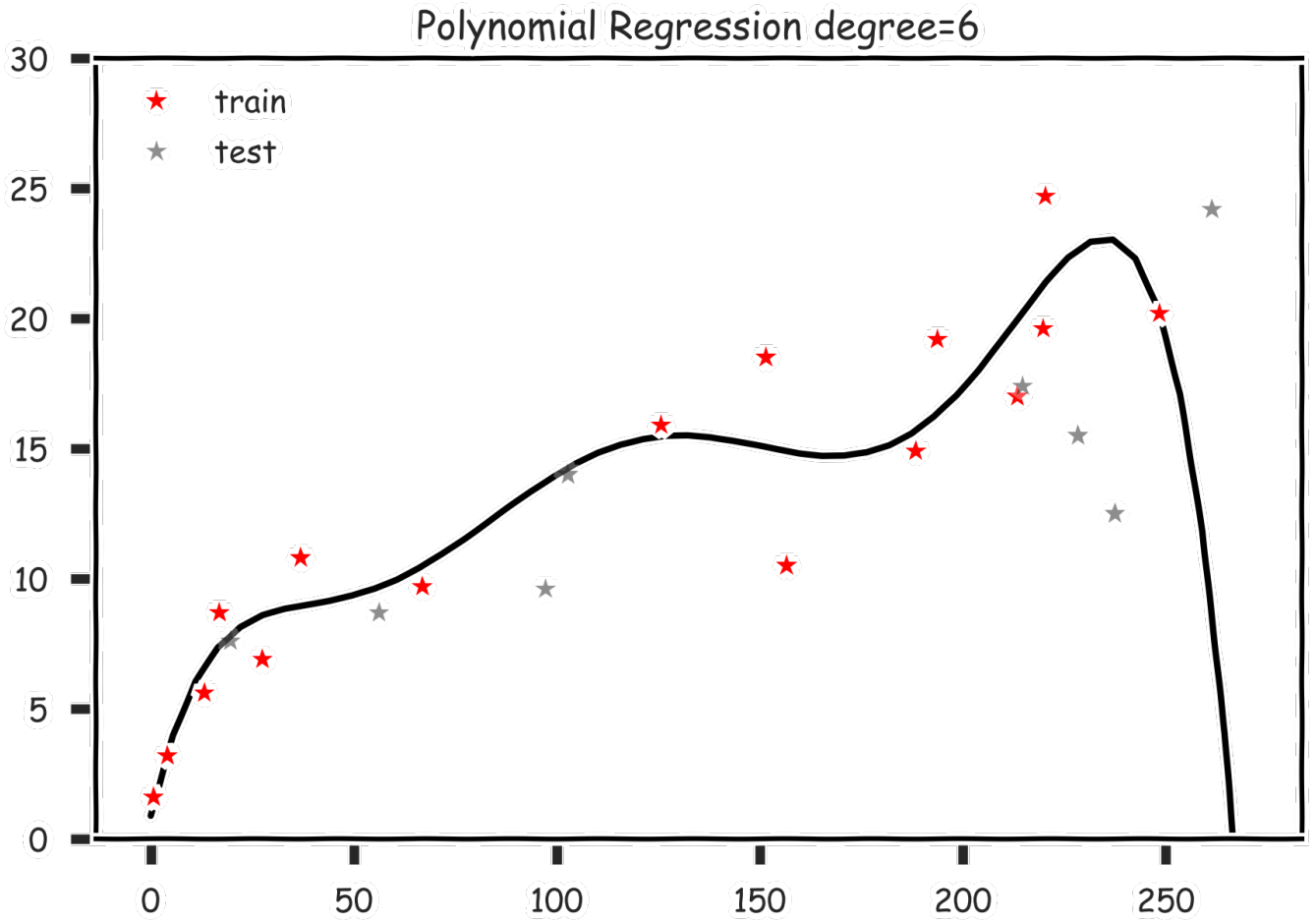
source: Pavlos Protopapas and Kevin Rader

Overfitting



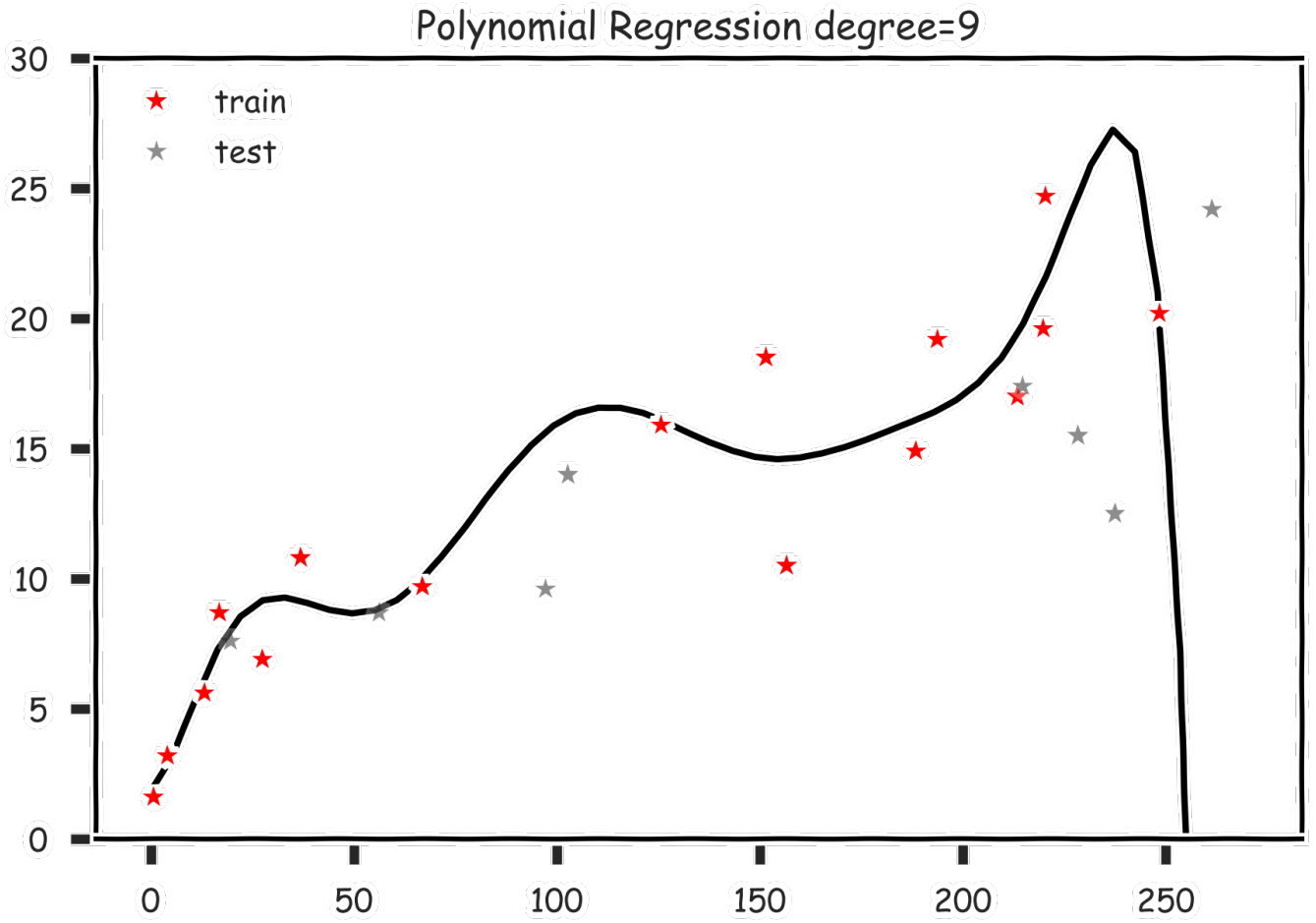
source: Pavlos Protopapas and Kevin Rader

Overfitting



source: Pavlos Protopapas and Kevin Rader

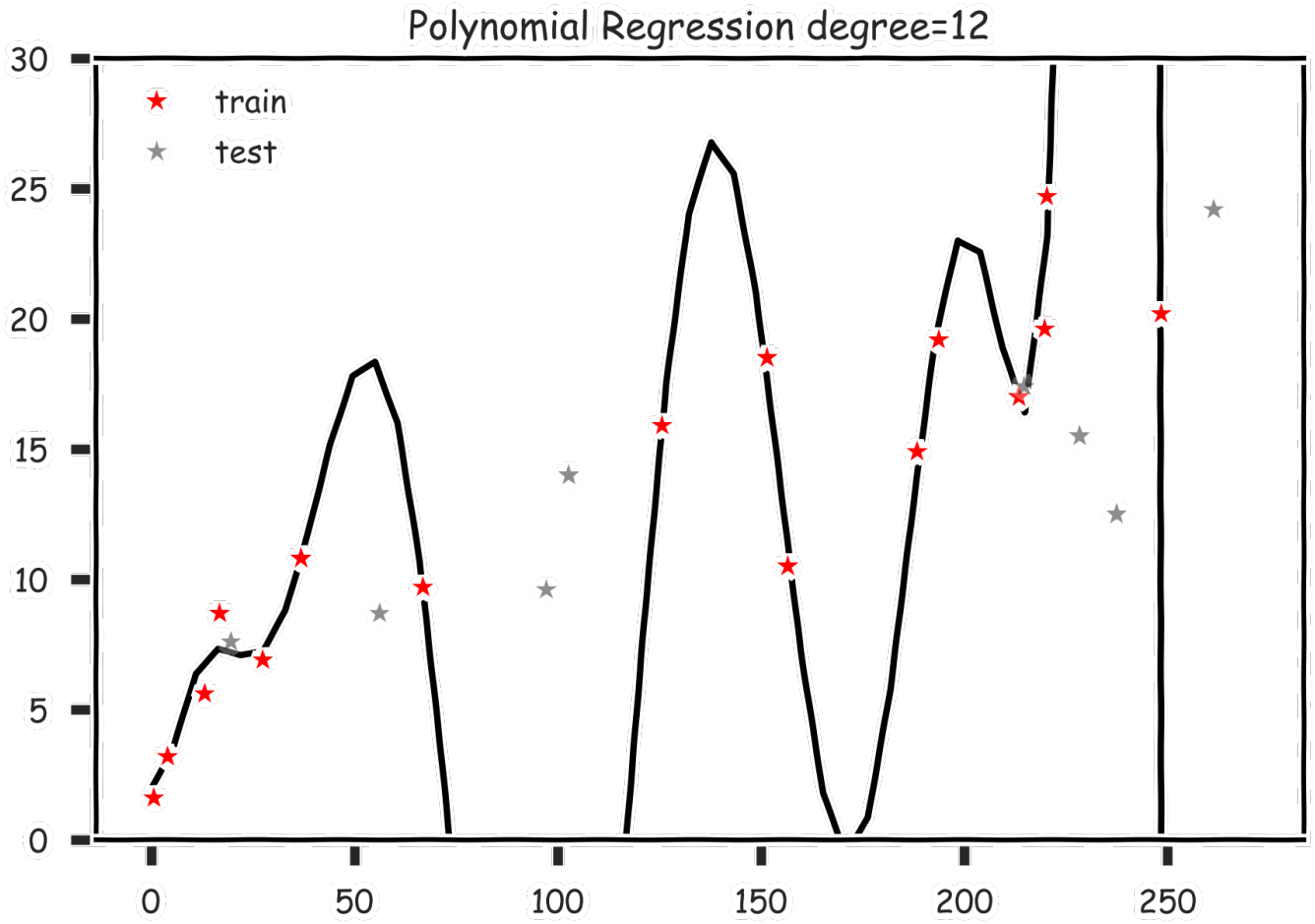
Overfitting



source: Pavlos Protopapas and Kevin Rader

Overfitting

Sign of overfitting: high training R^2 or low MSE and unexpectedly poor testing performance.



source: Pavlos Protopapas and Kevin Rader

Overfitting

- **Overfitting** is the phenomenon where the model is unnecessarily complex, in the sense that portions of the model captures the random noise in the observation, rather than the relationship between feature(s) and target.

- Overfitting causes the model to lose predictive power on new data.

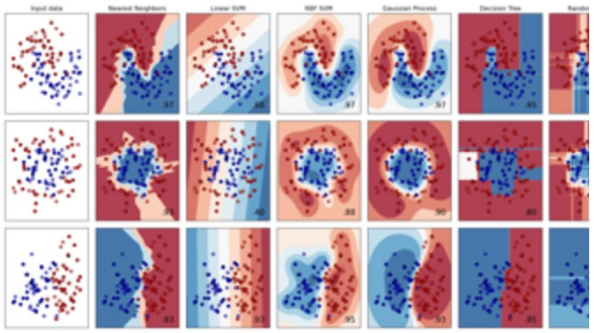
Scikit-learn Library

Scikit-learn is a free software machine learning library for the Python programming language.

Classification

Identifying which category an object belongs to.

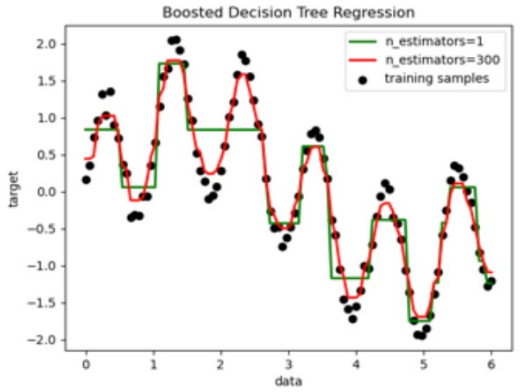
Applications: Spam detection, image recognition.
Algorithms: SVM, nearest neighbors, random forest, and more...



Regression

Predicting a continuous-valued attribute associated with an object.

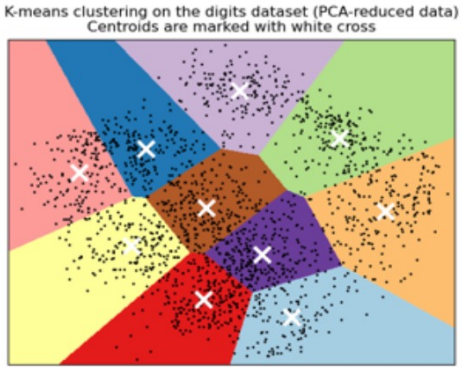
Applications: Drug response, Stock prices.
Algorithms: SVR, nearest neighbors, random forest, and more...



Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes
Algorithms: k-Means, spectral clustering, mean-shift, and more...



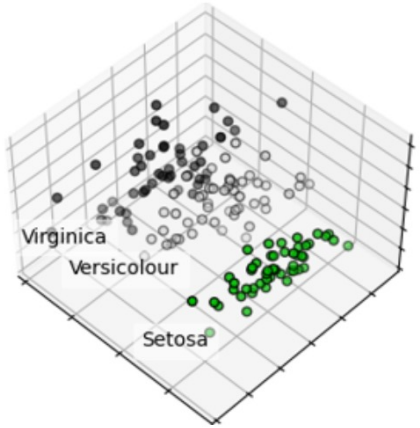
Scikit-learn Library

Scikit-learn is a free software machine learning library for the Python programming language.

Dimensionality reduction

Reducing the number of random variables to consider.

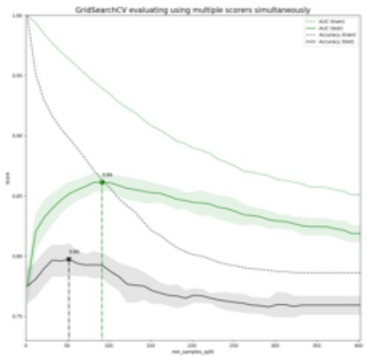
Applications: Visualization, Increased efficiency
Algorithms: k-Means, feature selection, non-negative matrix factorization, and more...



Model selection

Comparing, validating and choosing parameters and models.

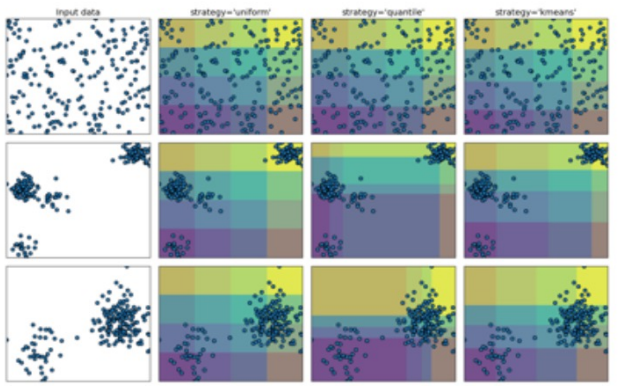
Applications: Improved accuracy via parameter tuning
Algorithms: grid search, cross validation, metrics, and more...



Preprocessing

Feature extraction and normalization.

Applications: Transforming input data such as text for use with machine learning algorithms.
Algorithms: preprocessing, feature extraction, and more...



Scikit-learn: Data Splitting

```
import numpy as np
from sklearn.model_selection import train_test_split

X, y = np.arange(10).reshape((5, 2)), range(5)
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.33, random_state=42)

print( X )
print( list(y) )
print( X_train)
print( X_test)
print( y_train )
print( y_test )
```

source

Scikit-learn: Linear Regression

```
import numpy as np
from sklearn.linear_model import LinearRegression

X = np.array([[1, 1], [1, 2], [2, 2], [2, 3]])
y = np.dot(X, np.array([1, 2])) + 3
model = LinearRegression()
model.fit(X, y)

print( model.score(X, y)   )
print( model.coef_ )
print( model.intercept_ )
print( model.predict(np.array([[3, 5]])) )
```

R^2 coefficient of determination

$[\beta_1 .., \beta_n]$

β_0

source

Scikit-learn: Linear Regression

```
import numpy as np
from sklearn.metrics import r2_score, mean_squared_error

r2 = r2_score(df_target_test, pred)
mse = mean_squared_error(df_target_test, pred)

print(r2)
print(mse)
```

source

Cohort Problem CS7

CS7. *Optional:* Redo the above tasks using Sci-kit learn libraries.

Thank You!