Spring 2023

SWCON253: Machine Learning

# Lecture 04
# Linear Regression

Jinwoo Choi
Assistant Professor
CSE, Kyung Hee University

# Contents

1. **Linear Regression**
2. **Normal Equation**
3. **Polynomial Regression**
4. ***For Better Results***
   - [illegible]
   - Learning Rate Tuning

**References**

- ***Machine Learning*** by Andrew Ng, Coursera (https://www.coursera.org/learn/machine-learning)

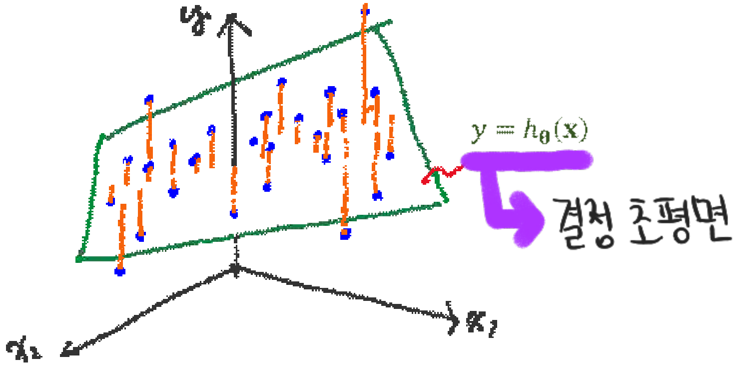
# 1. Linear Regression

1. Multivariate Linear Regression
2. Data Representation
3. Linear Model Representation
4. Cost Function (& Gradient)
5. Parameter Update (by Gradient Descent)

# 0. Multivariate Linear Regression

- Find the best *linear* function $h_\theta$ for the given training dataset $\mathbb{D}$ with *multiple*($n$)-features
  - A *feature vector*: $\mathbf{x} = [x_1, \dots, x_n]^T$
  - *Training dataset*: $\mathbb{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{m} = \{(\mathbf{x}^{(1)}, y^{(1)}), \cdots, (\mathbf{x}^{(m)}, y^{(m)})\}$
  - *Linear model*:

$$\boxed{\begin{aligned} h_\theta(\mathbf{x}) &= h_\theta(x_1, \dots, x_n) \\ &= \theta_0 + \theta_1 x_1 + \cdots + \theta_n x_n \end{aligned}}$$




### Ex) Housing Price Prediction

- *Multiple features*: size, # bedrooms, # floors, age
- *Single output*: the price of a house : y

| Size (feet²) $x_1$ | Number of bedrooms $x_2$ | Number of floors $x_3$ | Age of home (years) $x_4$ | Price ($1000) $y$ |
|---|---|---|---|---|
| 2104 | 5 | 1 | 45 | 460 → y' |
| 1416 | 3 | 2 | 40 | 232 |
| 1534 | 3 | 2 | 30 | 315 |
| 852 | 2 | 1 | 36 | 178 |
| ... | ... | ... | ... | ... |

m

$n$ = number of features    n = 4  4차원의 특징 벡터

$x^{(i)}$ = input (features) of $i^{th}$ training example. i번째 훈련 샘플

$x_j^{(i)}$ = value of feature $j$ in $i^{th}$ training example. // 의 j번째 특징 값

# 1. *Data Representation*

## Data with *Multiple* Features

- *A feature vector:* $\mathbf{x} = [x_1, \ldots, x_n]^T$
- *Training dataset:* $\mathbb{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{m} = \{(\mathbf{x}^{(1)}, y^{(1)}), \cdots, (\mathbf{x}^{(m)}, y^{(m)})\}$

$$X = \begin{bmatrix} x_1^1 & x_2^1 & \cdots & x_n^1 \\ x_1^2 & x_2^2 & \cdots & x_n^2 \\ \vdots & & & \\ x_1^m & x_2^m & \cdots & x_n^m \end{bmatrix}$$

## For a batch processing

design matrix
(input)

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}^{(1)T} \\ \mathbf{x}^{(2)T} \\ \vdots \\ \mathbf{x}^{(m)T} \end{bmatrix} \longrightarrow \boxed{\mathbf{h}_\theta} \longrightarrow$$

target vector
(label)

$$\mathbf{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix}$$

비교를 통해 학습

output vector
(prediction)

$$\hat{\mathbf{y}} = \mathbf{h}_\theta(\mathbf{X}) = \begin{bmatrix} h_\theta(\mathbf{x}^{(1)}) \\ h_\theta(\mathbf{x}^{(2)}) \\ \vdots \\ h_\theta(\mathbf{x}^{(m)}) \end{bmatrix}$$

# 2. *Linear Model Representation*

같은 얘기

◆ Representation 1

- Let $\mathbf{x} \triangleq [x_0, x_1, \dots, x_n]^T$, $\boldsymbol{\theta} \triangleq [\theta_0, \theta_1, \dots, \theta_n]^T$

  ★ where $x_0 = 1$. 한 차원 추가 (상수항 $\theta_0$를 위해 추가)

- Then, for a ***single*** training example (i.e., $\mathbf{x}^{(i)}$):

$$\boxed{h_\theta(\underline{x})} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n$$

$$= [\theta_0 \quad \theta_1 \quad \ldots \quad \theta_n] \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix} \boxed{= \underline{\theta}^T \underline{x}}$$

- and for a ***batch*** of training examples (i.e., $\mathbf{X}$):

$$\boxed{\mathbf{h}_\theta(\mathbf{X}) = \begin{bmatrix} \boldsymbol{\theta}^T\mathbf{x}^{(1)} \\ \boldsymbol{\theta}^T\mathbf{x}^{(2)} \\ \vdots \\ \boldsymbol{\theta}^T\mathbf{x}^{(m)} \end{bmatrix} = \mathbf{X}\boldsymbol{\theta}}$$

◆ Representation 2 (weight & bias)

- Let $\mathbf{x} \triangleq [x_1, \dots, x_n]^T$, $\boldsymbol{\theta} \triangleq [\theta_1, \dots, \theta_n]^T$

- Then, for a ***single*** training example

$$h_\theta(\underline{x}) = \underline{\theta}^T \underline{x} + \theta_0$$

$$( h_w(x) = w^T x + b )$$

- and, for a ***batch*** of training example

$$h_\theta(X) = X\underline{\theta} + \theta_0 \begin{bmatrix} \vdots \\ \vdots \end{bmatrix}$$

$$( h_w(X) = X\underline{w} + \underline{b} )$$

# 2. Linear Model Representation (cont'd)

## Summary of Input, Output, & Model

$$\boldsymbol{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix}$$

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}^{(1)T} \\ \mathbf{x}^{(2)T} \\ \vdots \\ \mathbf{x}^{(m)T} \end{bmatrix} \longrightarrow \boxed{\mathbf{h}_\theta} \longrightarrow$$

$$\mathbf{h}_\theta(\mathbf{X}) = \begin{bmatrix} h_\theta(\mathbf{x}^{(1)}) \\ h_\theta(\mathbf{x}^{(2)}) \\ \vdots \\ h_\theta(\mathbf{x}^{(2)}) \end{bmatrix} = \begin{bmatrix} \boldsymbol{\theta}^T\mathbf{x}^{(1)} \\ \boldsymbol{\theta}^T\mathbf{x}^{(2)} \\ \vdots \\ \boldsymbol{\theta}^T\mathbf{x}^{(m)} \end{bmatrix} = \mathbf{X}\boldsymbol{\theta}$$

벡터 ↑

↓ 행렬

Linear Model

# 3. MSE Cost for Linear Model

## MSE Cost

- Classic form:

$$J(\boldsymbol{\theta}) = \frac{1}{2m}\sum_{i=1}^{m}\Big(\underbrace{h_\theta(\mathbf{x}^{(i)})}_{=\hat{y}^{(i)}} - y^{(i)}\Big)^2$$

오차 : Error

linear model

$$= \frac{1}{2m}\sum_{i=1}^{m}\left(\boldsymbol{\theta}^T\mathbf{x}^{(i)} - y^{(i)}\right)^2$$

$\frac{\partial}{\partial\theta_j}$

- Vector form

벡터

$$J(\boldsymbol{\theta}) = \frac{1}{2m}\|\mathbf{h}_\theta(\mathbf{X}) - \mathbf{y}\|_2^2$$

linear model

$$= \frac{1}{2m}\|\mathbf{X}\boldsymbol{\theta} - \mathbf{y}\|_2^2$$

$$= \frac{1}{2m}(\mathbf{X}\boldsymbol{\theta} - \mathbf{y})^T(\mathbf{X}\boldsymbol{\theta} - \mathbf{y})$$

cf) $\|x\|_2^2 = x^T x$ 이용

유도과정:
다음 슬라이드

## Gradient of the MSE Cost

- Classic form:

For $j$=0,…,$n$ :

$$\frac{\partial J(\boldsymbol{\theta})}{\partial\theta_j} = \frac{1}{m}\sum_{i=1}^{m}\left(\boldsymbol{\theta}^T\mathbf{x}^{(i)} - y^{(i)}\right)x_j^{(i)}$$

- Vector form

$$\nabla J(\boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m}\left(\boldsymbol{\theta}^T\mathbf{x}^{(i)} - y^{(i)}\right)\mathbf{x}^{(i)}$$

$$= \frac{1}{m}\mathbf{X}^T(\mathbf{X}\boldsymbol{\theta} - \mathbf{y})$$

# 3. MSE Cost for Linear Model (cont'd)

- $\nabla_{\theta} \|X\theta - y\|_2^2$ 구하기 :

$$\|X\theta - y\|_2^2 = (X\theta - y)^T (X\theta - y) \overset{①}{=} (\theta^T X^T - y^T)(X\theta - y)$$

$$= \theta^T X^T X \theta - y^T X \theta - \theta^T X^T y + y^T y$$

$$\overset{②}{=} \underbrace{\theta^T X^T X \theta}_{\text{Quadratic Form of } \theta} - 2 y^T X \theta + y^T y$$

① $(a - b)^T = a^T - b^T$

$(Ax)^T = x^T A^T$

② $\theta^T X^T y = (X\theta)^T y = y^T X \theta$

$$\Rightarrow \nabla_{\theta} \|X\theta - y\|_2^2 = \nabla_{\theta}(\theta^T X^T X \theta) - 2\nabla(y^T X \theta)$$

$$\overset{③}{=} 2X^T X \theta - 2X^T y$$

$$= 2X^T(X\theta - y) \quad //$$

③ $\nabla_x (x^T A^T A x) = 2A^T A x$ — symmetric

$\nabla_x (b^T A x) = \nabla_x c^T x = c$

$= (b^T A)^T$

$= A^T b$

a row vector $\equiv c^T$

# 4. Parameter Update by GD (for Linear Model)

## Gradient Descent for *Linear Regression with MSE Cost*

- Classic form: $\theta = \theta + \Delta\theta = \theta - \alpha \nabla_\theta$

  *Repeat until convergence* {

  Update $\forall\theta_j$'s simultaneously:

  $$\theta_j := \theta_j - \alpha \frac{\partial J(\boldsymbol{\theta})}{\partial \theta_j} = \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{\cdots} \left(\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}\right) x_j^{(i)}$$

  }

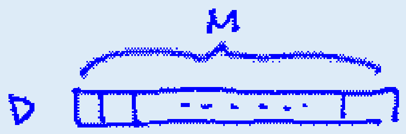
- Vector form

  *Repeat until convergence* {

  $$\boldsymbol{\theta} := \boldsymbol{\theta} - \alpha \nabla J(\boldsymbol{\theta}) = \boldsymbol{\theta} - \alpha \frac{1}{m} \mathbf{X}^T (\mathbf{X}\boldsymbol{\theta} - \mathbf{y})$$

  }

Learning Modes (Recap.)




- m = 1 : online mode
- m = M : batch mode
- 1 < m < M : mini-batch mode
  - Shuffle the training example then apply minibatch mode
  - Or randomly select training examples for each minibatch.

# 2. Normal Equation

해석적인 방법

1. Normal Equation
2. Gradient Descent vs. Normal Equation

# Normal Equations

◆ Analytic Solution to *Linear Regression with MSE Loss*

$$\nabla J(\boldsymbol{\theta}) = \frac{1}{m}\mathbf{X}^T(\mathbf{X}\boldsymbol{\theta} - \mathbf{y}) = \mathbf{0}$$

$X^TX\theta - X^TY = 0$

$X^TX\theta = X^TY$

$$\boxed{\boldsymbol{\theta}^* = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}}$$

Examples: $m = 4.$

| $x_0$ | Size (feet²) $x_1$ | Number of bedrooms $x_2$ | Number of floors $x_3$ | Age of home (years) $x_4$ | Price ($1000) $y$ |
|---|---|---|---|---|---|
| 1 | 2104 | 5 | 1 | 45 | 460 |
| 1 | 1416 | 3 | 2 | 40 | 232 |
| 1 | 1534 | 3 | 2 | 30 | 315 |
| 1 | 852 | 2 | 1 | 36 | 178 |

$$X = \underbrace{\begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 36 \end{bmatrix}}_{m \times (n+1)} \qquad y = \underbrace{\begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \end{bmatrix}}_{m\text{-dimensional vector}}$$

$$\boxed{\theta = (X^TX)^{-1}X^Ty}$$

상수항

$X\ \theta - y$ =0 (즉, $\theta^* = X^{-1}\ y$ )로 구하지 않는 이유는?

일반적으로 역행렬이 없음

주어진 문제에서 X는 design matrix이고 차원이 대략 m x n (m은 training example 개수, n은 feature 차원)이 되므로 square 행렬이 아닐 수 있습니다.

역행렬은 square 행렬의 경우만 존재하므로 일반적인 m x n 행렬은 역행렬을 구할 수 없습니다.

반면에, $X^TX$나 $XX^T$ 형태로 만들면 square가 되어 역행렬을 구할 수 있습니다.

# Gradient Descent vs. Normal Equation

$$\nabla J(\theta) = \frac{1}{m} X^T (X\theta - y) \qquad \theta^* = (X^T X)^{-1} X^T y$$

하이퍼 파라미터

| Gradient Descent | Normal Equation |
|---|---|
| Need to choose alpha | No need to choose alpha |
| Needs many iterations | No need to iterate |
| O ($kn^2$) 복잡도 | O ($n^3$), need to calculate inverse of $X^T X$ |
| Works well when n is large | Slow if n is very large |

특징 벡터의 차원이 커지면

$$X = m \overbrace{\begin{bmatrix} \text{---} \\ \text{---} \\ \vdots \\ \text{---} \end{bmatrix}}^{n}$$

$$\underset{(n \times m)\cdot(m \times n)}{X^T X} \to (n \times n)$$

- With the normal equation, computing the inversion has complexity $O(n^3)$.
  - ★ So if we have a very large *number of features* (i.e., large $n$), the normal equation will be slow.
  - ★ In practice, when $n$ exceeds 10,000 it might be a good time *to go to an iterative process*.

# 3. Polynomial Regression

곡선

# Polynomial Regression

## ◆ Polynomial Regression can be solved by Linear Regression

- Linear Regression 문제로 환원하여 풀 수 있음:

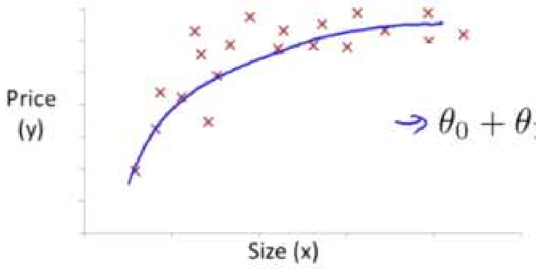
$$x_1 = (size)$$
$$x_2 = (size)^2$$
$$x_3 = (size)^3$$

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_1^3 + \theta_3 x_1^3$$

- 이 때 feature scaling이 중요해 짐

변수마다 가질 수 있는 범위가 상이하게 달라짐

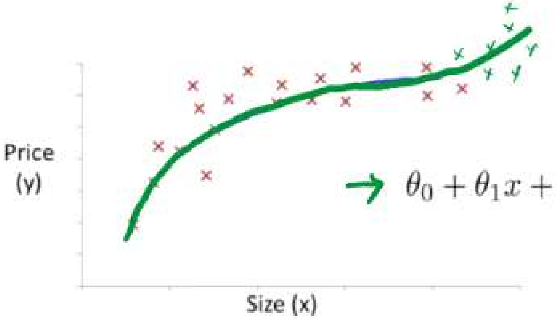
● Ex) Housing Price Prediction




size

$\rightarrow \theta_0 + \theta_1 x + \theta_2 x^2$

은 $x_1$ 은 $x_2$




$\rightarrow \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3$

## ◆ Polynomial이외의 Nonlinear Function의 경우는?

● 마찬가지 방법(변수 치환)을 통해 선형회귀로 바꾸어 풀 수 있음!

# 4. For Better Results

1. Feature Normalization
2. Learning Rate

# Feature Normalization

R

- **Feature Scaling (Range Normalization)**
  - *Make sure features are on a similar scale*

E.g. $x_1$ = size (0-2000 feet²)

$x_2$ = number of bedrooms (1-5)

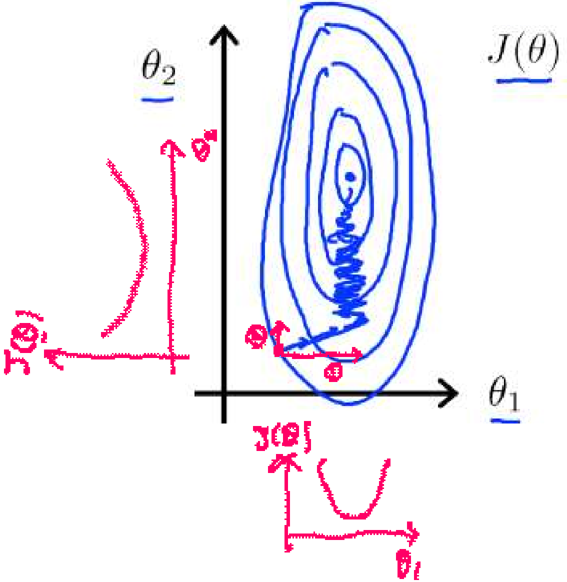
$x_1$이 $x_2$보다 학습에 큰 영향을 미침

Scaling

$$x_1 = \frac{\text{size (feet}^2)}{2000}$$

$$x_2 = \frac{\text{number of bedrooms}}{5}$$

$0 \le x_1 \le 1 \qquad 0 \le x_2 \le 1$

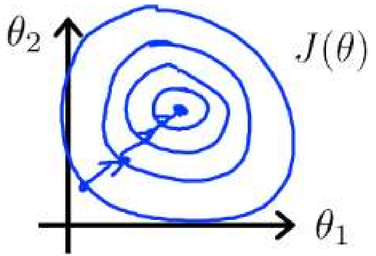



$h_\theta(x) = \theta_1 x_1 + \theta_2 x_2$

$J(\theta) = \frac{1}{2m}\sum (h_\theta(x^{(i)}) - y^{(i)})^2$

① $\frac{\partial J(\theta)}{\partial \theta_1} = \frac{1}{m}\sum (h_\theta(x^{(i)}) - y^{(i)}) x_1^{(i)}$ → 훨씬 큼

② $\frac{\partial J(\theta)}{\partial \theta_2} = \frac{1}{m}\sum (h_\theta(x^{(i)}) - y^{(i)}) x_2^{(i)}$




∴ Converge faster!

# Feature Normalization (cont'd)

- **Feature Scaling (cont'd)**
  - Eg., get every feature into approximately a $-1 \leq x_j \leq 1$ range.

$0 \leq x_1 \leq 3$ ✓

$-2 \leq x_2 \leq 0.5$ ✓

$-100 \leq x_3 \leq \boxed{100}$ ✗

$-0.0001 \leq x_4 \leq \boxed{0.0001}$ ✗

$-3$ to $3$ ✓

$-\frac{1}{3}$ to $\frac{1}{3}$ ✓

# Feature Normalization (cont'd)

- **Mean Normalization**
  - Replace $x_j$ with $x_j - \mu_j$ to make features have approximately zero mean.

평균

모든 데이터에서 평균을 뺌 → 평균 0 됨

E.g. → $\boxed{x_1 = \frac{size - 1000}{2000}}$

$\boxed{x_2 = \frac{\#bedrooms - 2}{5}}$ 4

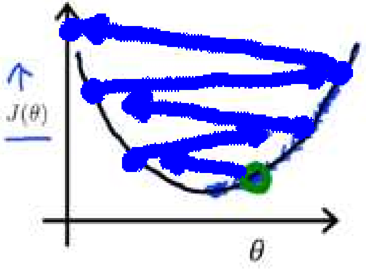
1-5 bed

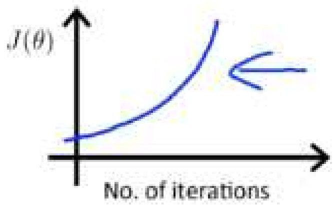
→ $\boxed{-0.5 \leq x_1 \leq 0.5}$ $\boxed{-0.5 \leq x_2 \leq 0.5}$

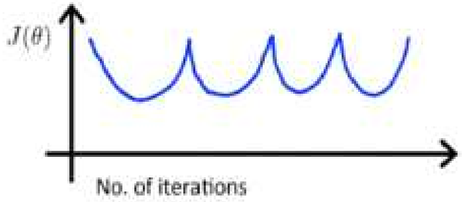
- **Caution:**
  - *Do not normalize* $x_0 = 1$. 상수항을 위한 항
  - There is **no** need to do feature normalization with the ***normal equation.***

# Learning Rate ($\alpha$)

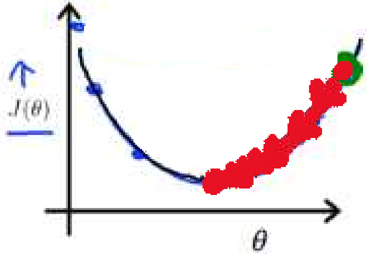
- **How to Choose Learning Rate $\alpha$?**
  - **Too large $\alpha$**: may not converge 발산

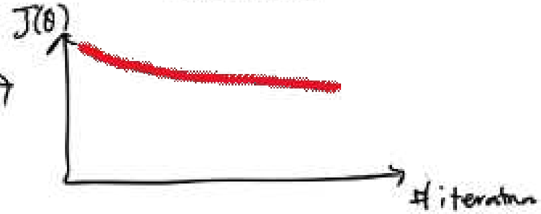



  - **Too small $\alpha$**: slow convergence 오래걸림




  - ***Rule of thumb***:
    - ★ *Try ..., 0.001, ..., 0.01, ..., 0.1, ..., 1, ...*
      *then try the in-betweens if not satisfactory*

1e-1 ~ 1e-5 or -6

처음에 크게 했다가 점점 줄이는 방법

  - ***Learning Rate Scheduling***
    - ★ Starts with some large $\alpha$, and then decrease $\alpha$ according to a schedule