[SWCON253] Machine Learning – Lec.06

Linear Regression + α

Fall 2025

김 휘 용

hykim.v@khu.ac.kr

경희대학교
KYUNG HEE UNIVERSITY

Contents

References

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

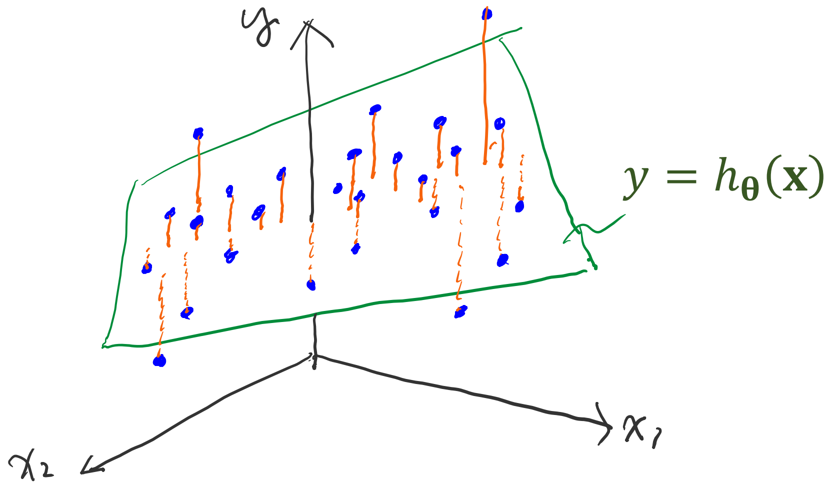
1. Linear Regression

- ✓ Multivariate Linear Regression
- ✓ Data Representation
- ✓ Linear Model Representation
- ✓ Cost Function (& Gradient)
- ✓ Parameter Update (by Gradient Descent)

0. Multivariate Linear Regression

- Find the best ***linear*** function $h_{\boldsymbol{\theta}}$ 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***:

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



Ex) Housing Price Prediction

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

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
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
…	…	…	…	…

M

n = number of features $n=4$

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

$x_j^{(i)}$ = value of feature j in i^{th} training example.

1. *Data Representation*

Data with *Multiple* Features

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

For batch processing

- *(mini-)batch size:* $1 \le m \le M$

design matrix (input)

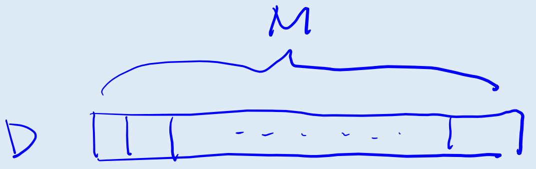
$$\mathbf{X} = \begin{bmatrix} \mathbf{x}^{(1)T} \\ \mathbf{x}^{(2)T} \\ \vdots \\ \mathbf{x}^{(m)T} \end{bmatrix} \longrightarrow \boxed{\mathbf{h}_{\boldsymbol{\theta}}} \longrightarrow$$

target vector (label)

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

output vector (prediction)

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



- $m = 1$: 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. *Linear Model Representation*

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 $x_0 = 1$.

- 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

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

$$(h_w(\underline{x}) = \underline{w}^T \underline{x} + b)$$

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

$$h_\theta(X) = X\underline{\theta} + \theta_0 \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$$

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

2. *Linear Model Representation (cont'd)*

◆ **Summary of Input, Output, & Model**

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

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

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

Linear Model

3. MSE Cost for Linear Model

MSE Cost

- Classic form:

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

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

- Vector form

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

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

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

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 :

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

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

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

① $\begin{cases} (a-b)^T = a^T - b^T \\ (Ax)^T = x^T A^T \end{cases}$

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

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

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

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

③ $\begin{cases} \nabla_x (x^T A^T A x) = 2A^T A x \\ \nabla_x (b^T A x) = \nabla_x c^T x = c \end{cases}$

symmetric

a row vector $\triangleq c^T$

$$= (b^T A)^T$$

$$= A^T b$$

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

◆ Gradient Descent for *Linear Regression with MSE Cost*

- Classic form:

Repeat until convergence {

Update $\forall\theta_j$'s *simultaneously*:

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

}

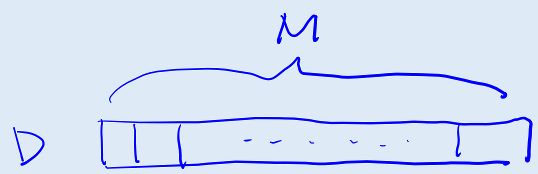
- Vector form

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

- ✓ Normal Equation
- ✓ Gradient Descent vs. Normal Equation

Normal Equations

- **Analytic Solution to *Linear Regression with MSE Loss***

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

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

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

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^TX나 XX^T 형태로 만들면 square가 되어 역행렬을 구할 수 있습니다.

Examples: $m = 4.$

	Size (feet²)	Number of bedrooms	Number of floors	Age of home (years)	Price ($1000)
x_0	x_1	x_2	x_3	x_4	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

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

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

Gradient Descent vs. Normal Equation

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

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

Gradient Descent	Normal Equation
Need to choose alpha Bad	No need to choose alpha Good
Needs many iterations Bad	No need to iterate Good
O (kn^2) Very Good!	O (n^3), need to calculate inverse of X^TX Very Bad!
Works well when n is large	Slow if n is very large

$$X = \begin{array}{cc} & n \\ m & \begin{bmatrix} \text{---} \\ \text{---} \\ \vdots \\ \text{---} \end{bmatrix} \end{array}$$

$X^TX \rightarrow (n \times n)$

$(n \times m) \cdot (m \times n)$

- 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

Polynomial Regression

Polynomial Regression can be solved by Linear Regression

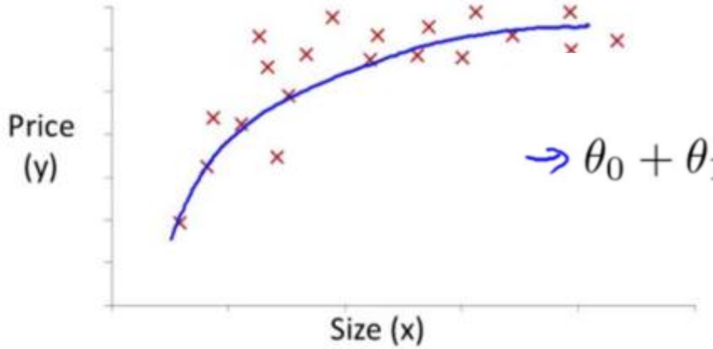
- Linear Regression 문제로 환원하여 풀 수 있음:

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

- 이때 feature scaling이 중요해 짐

$$h_\theta(x) = \theta_0 + \theta_1 \underset{\substack{\| \\ x_1}}{x} + \theta_2 \underset{\substack{\| \\ x_2}}{x^2} + \theta_3 \underset{\substack{\| \\ x_3}}{x^3}$$

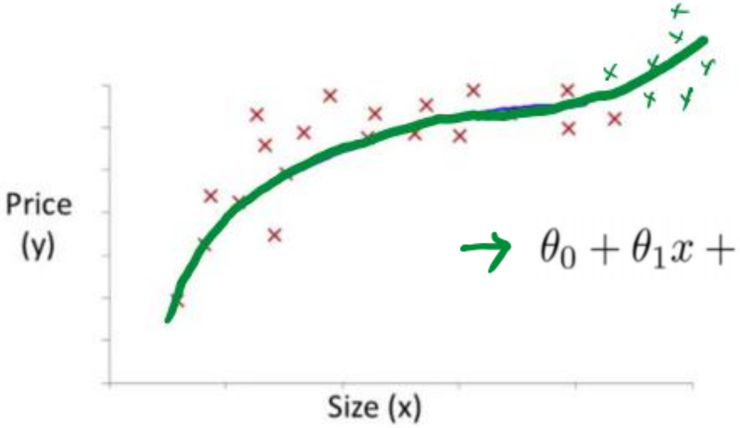
- Ex) Housing Price Prediction



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!

$$h_\theta(x) = \theta_0 + \theta_1 \underset{\substack{\| \\ x_1}}{\sqrt{x}} + \theta_2 \underset{\substack{\| \\ x_2}}{\cos(x+\pi)}$$

4. For Better Results

- ✓ Feature Normalization
- ✓ Learning Rate Selection

Feature Normalization

Feature Scaling (Range Normalization)

- *Make sure features are on a similar scale*

E.g. x_1 = size (0-2000 feet²)

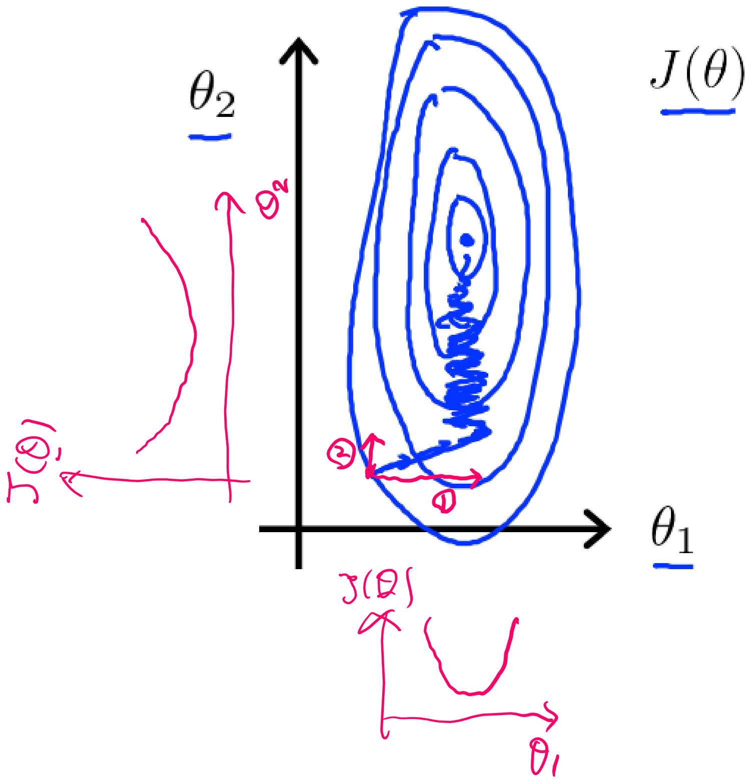
x_2 = number of bedrooms (1-5)

Scaling (Normalization)

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

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

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

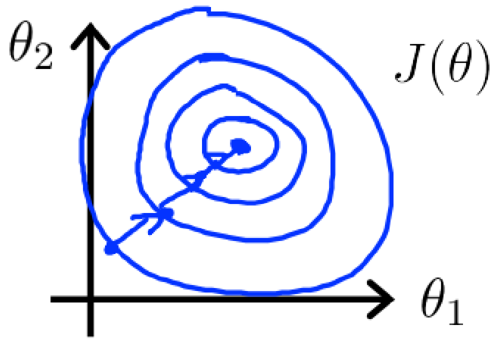


$h_\theta(x) = \theta_1 x_1 + \theta_2 x_2$

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

① $\frac{\partial J(\theta)}{\partial \theta_1} = \frac{1}{m}\sum_i \left(h_\theta(x^{(i)}) - y^{(i)}\right) \cdot x_1^{(i)}$

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



∴ Converge faster!

Feature Normalization (cont'd)

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

- **Feature Normalization** = Range Normalization + Mean Normalization

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

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

$\boxed{-0.5 \le x_1 \le 0.5}$, $\boxed{-0.5 \le x_2 \le 0.5}$

Feature Normalization (cont'd)

Feature Normalization Summary

- Mean shifting:

$$x_j'^{(i)} = x_j^{(i)} - \mu_j \quad \text{(i.e., } \mathbf{x}'^{(i)} = \mathbf{x}^{(i)} - \boldsymbol{\mu}\text{)}$$

- With Scaling Method 1: Use *range* ($R = \max|x_j|$) to normalize

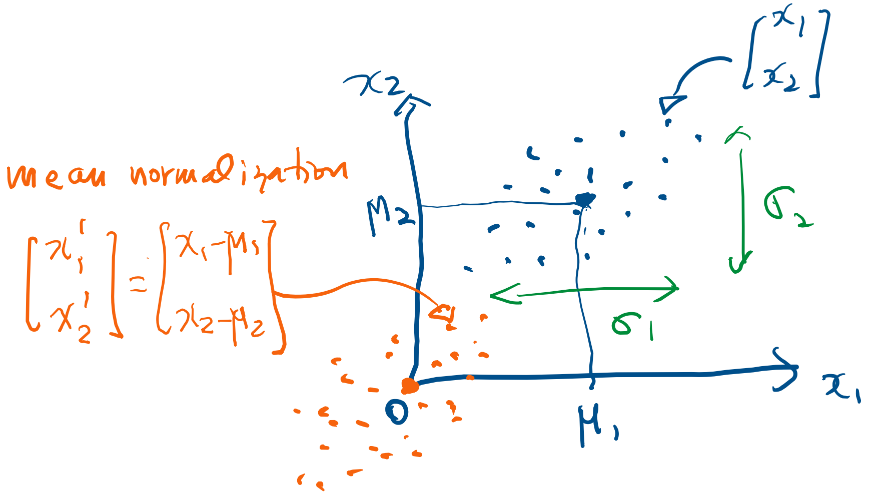
$$x_j'^{(i)} = \frac{x_j^{(i)} - \mu_j}{\max|x_j|} \quad \text{(i.e., } \mathbf{x}'^{(i)} = \frac{\mathbf{x}^{(i)} - \boldsymbol{\mu}}{\max|\mathbf{x}^{(i)}|}\text{)}$$

→ Normalized range: $-1 \le x_j' \le 1$ (strictly)

- With Scaling Method 2: Use *standard deviation* () to normalize

$$x_j'^{(i)} = \frac{x_j^{(i)} - \mu_j}{\sigma_j} \quad \text{(i.e., } \mathbf{x}'^{(i)} = \frac{\mathbf{x}^{(i)} - \boldsymbol{\mu}}{\boldsymbol{\sigma}}\text{)}$$

→ Normalized range: $-1 \le x_j' \le 1$ (roughly)



feature normalization: $x_j' = \dfrac{x_j - \mu_j}{\sigma_j}$

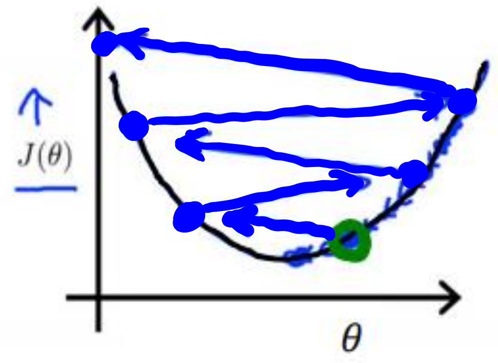
$(x_j = \sigma_j x_j + \mu_j)$

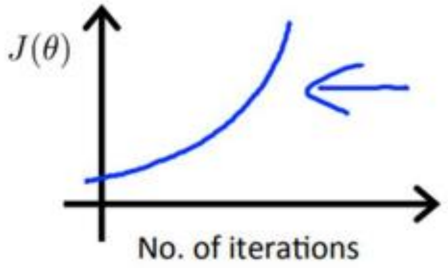
Caution:

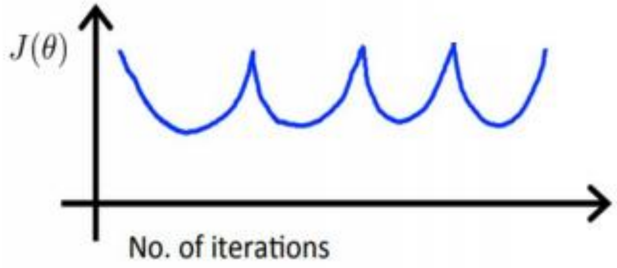
- *Do not normalize* $x_0 = 1$.
- There is **no** need to do feature normalization with the ***normal equation***.

Learning Rate Selection

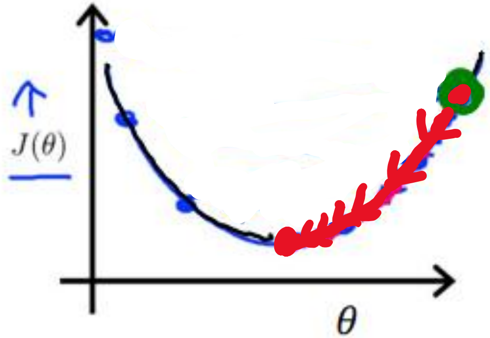
- How to Choose Learning Rate α?
  - **Too large α**: may not converge







  - **Too small α**: slow convergence



  - ***<u>Rule of thumb</u>***:
    - ★ Try …, 0.001, …, 0.01, …, 0.1, …, 1, …
      then try the in-between if not satisfactory
  - ***<u>Learning Rate Scheduling</u>***
    - ★ Starts with some large α, and then decrease α according to a schedule

Q & A