Dong-A Univ. (ISPL)

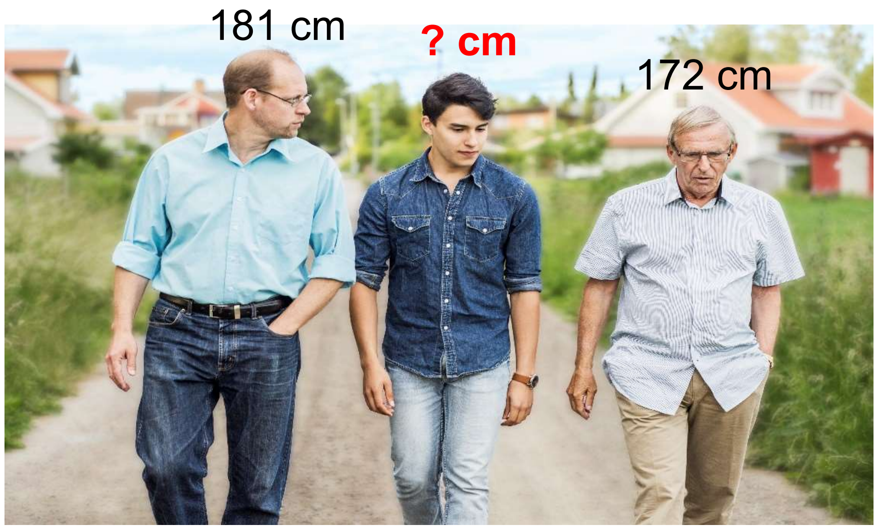
Linear Regression - 실습

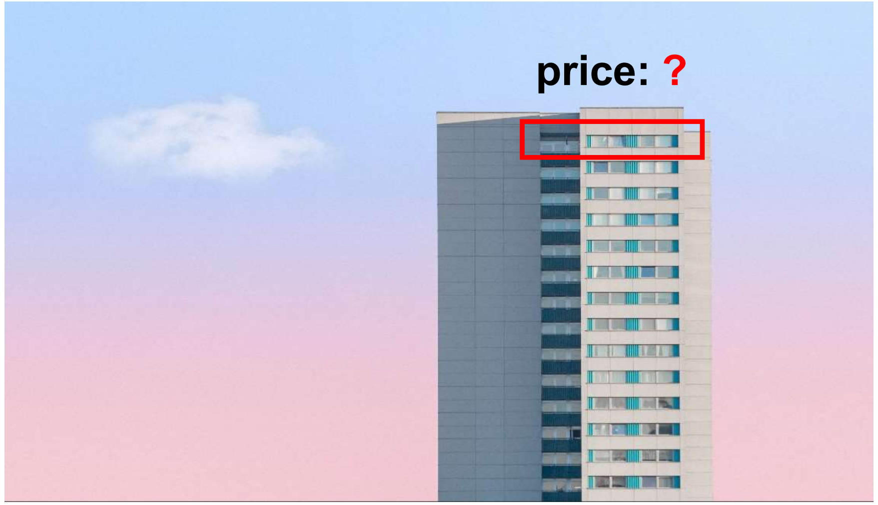
컴퓨터AI공학부

2025년 1학기 머신러닝

실습 개요

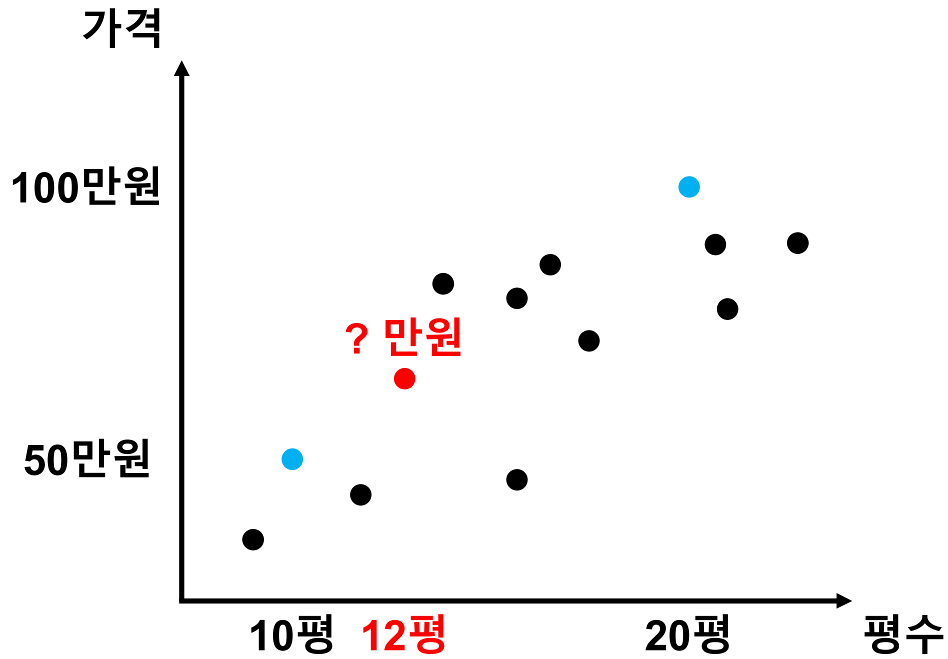
- **실습 목적:** 선형회귀 기법을 통해 주어진 데이터셋에서 데이터 간의 관계를 분석하고 예측하는 모델을 구현





실습 개요

- **Linear Regression (선형회귀)**
  - Single Linear Regression (단일 선형회귀)



단일 선형회귀 예시

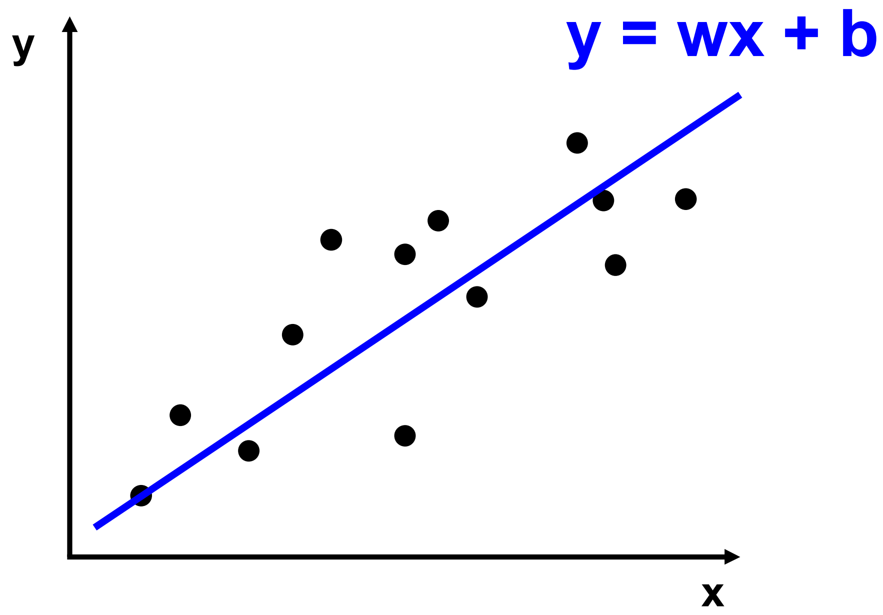
실습 개요

- **Linear Regression (선형회귀)**
  - Single Linear Regression (단일 선형회귀)

- **Solutions**
  - Ordinary Least Squares (OLS)

    = Least Squares Method (LSM)

    = Normal Equation
  - Gradient Descent Method



단일 선형회귀 예시

실습 개요

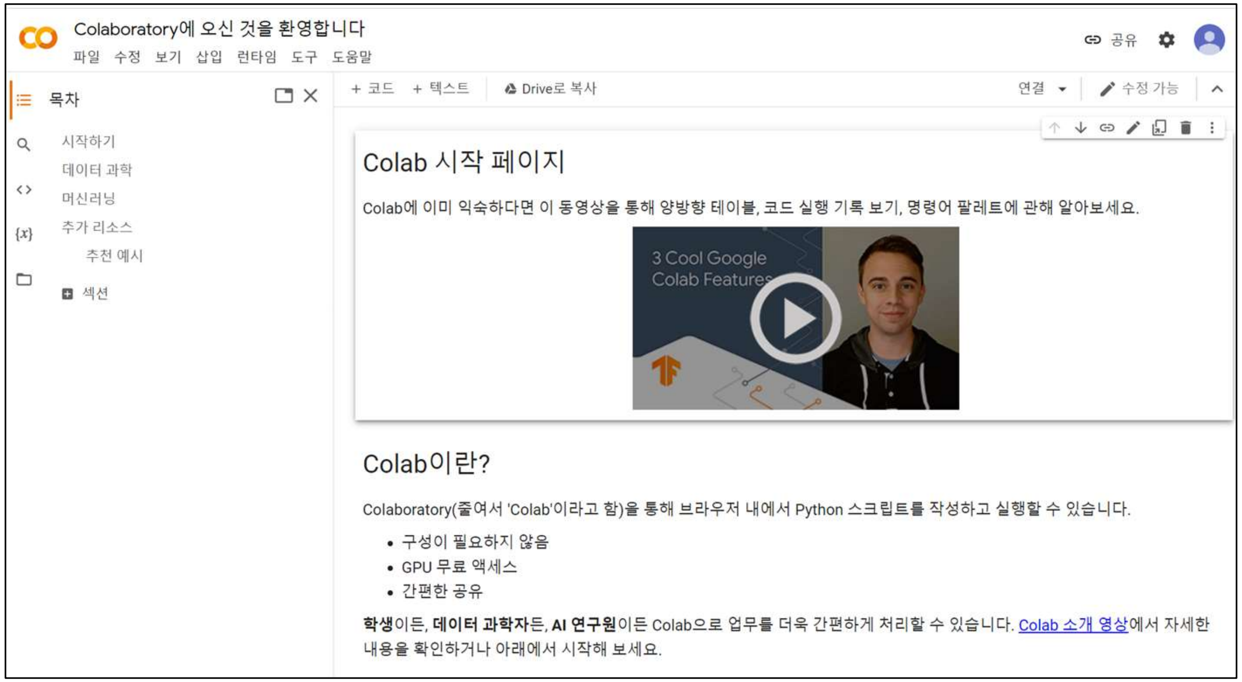
- **Dataset: kc_house_data**
  - 2014 ~ 2015년 사이 판매된 주택 가격 데이터셋
  - 21개 변수, 21,613개 데이터로 구성됨

가격	침실	화장실	크기	…
# price	# bedrooms	# bathrooms	# sqft_living	
75k – 7.7m	0 – 33	0 – 8	290	
221900	3	1	1180	
538000	3	2.25	2570	
180000	2	1	770	
604000	4	3	1960	
510000	3	2	1680	…
1.225E+06	4	4.5	5420	
257500	3	2.25	1715	
291850	3	1.5	1060	
229500	3	1	1780	
323000	3	2.5	1890	
662500	3	2.5	3560	

https://www.kaggle.com/datasets/shivachandel/kc-house-data/

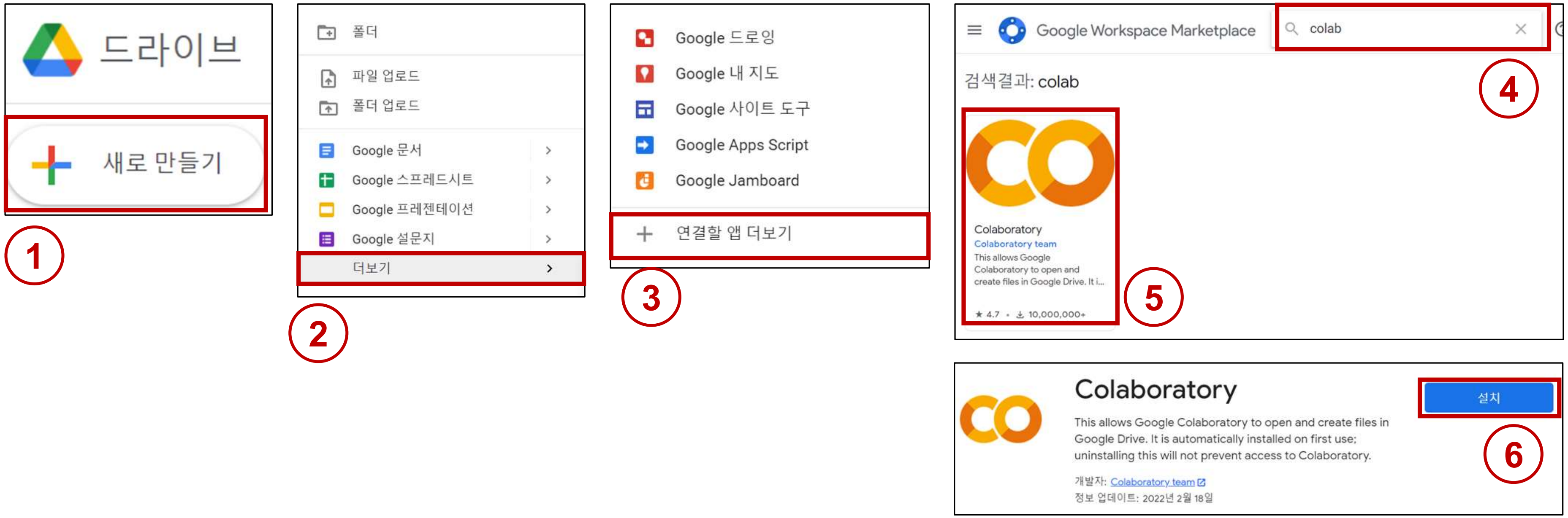
실습 환경 구성

- **Google Colaboratory (Colab)**
  - 딥러닝, 머신러닝 모델 등을 실행할 수 있는 무료 클라우드 서비스
  - 모델 학습을 위해 GPU를 일정 시간동안 무료로 사용할 수 있음
  - 구글 계정으로 로그인해 사용 가능 (colab.research.google.com)



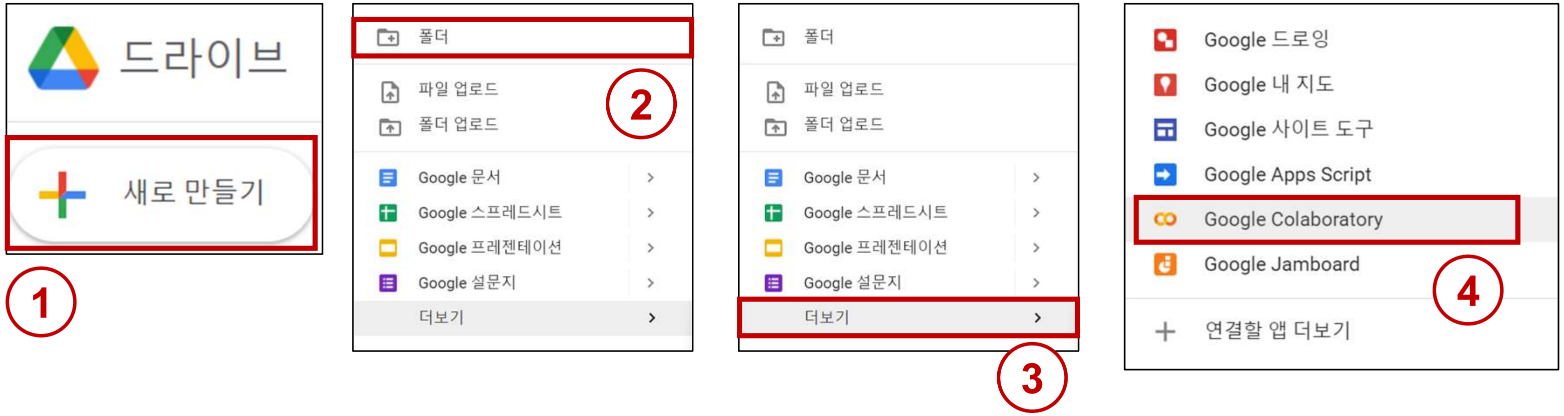
실습 환경 구성

- [1] 웹 브라우저에서 Google 로그인
- [2] 구글 드라이브 접속 (drive.google.com)
- [3] 구글 colab 설치 (아래 사진 참고)



실습 환경 구성

- [4] 구글 드라이브 내 실습 코드를 보관할 폴더 생성 (띄어쓰기, 한글 사용 X)
- [5] 구글 Colab 실행



Colab 실행 화면

실습 환경 구성

- **LMS 강의콘텐츠 5주차 1차시 Base code 및 데이터셋 다운로드**

파일

Gemini가 작성한 코드로 파일 분석 업로드

..

.config

sample_data

kc_house_data.csv

2 데이터셋 저장확인

데이터셋 로딩 및 전처리

- kc_house_data: 미국 워싱턴주 시애틀 지역의 주택 가격 데이터를 포함한 공개 데이터셋
- **price: 주택의 판매 가격 (종속 변수, 목표 값).**
- **sqft_living: 주택의 실내 면적 (평방 피트).**

1 실행

```
# Download dataset file
!wget "https://dongaackr-my.sharepoint.com/:x:/g/personal/sjkim_donga_ac_kr/ET2udIQfsxRAsvnIE

# Load dataset file
data = pd.read_csv('kc_house_data.csv')

# Single linear regression 실습에 사용할 데이터 열만 수집 (price (정답), sqft_living (입력))
X, Y = data['sqft_living'], data['price']

# 데이터 값 확인
df = data[['sqft_living', 'price']]
print(df)
```

실습 환경 구성

- **Single Linear Regression을 위한 데이터셋 확인**

Simple Linear Regression

Import packages

```
[1]  import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt

     from sklearn.model_selection import train_test_split
```

데이터셋 로딩 및 전처리

- kc_house_data: 미국 워싱턴주 시애틀 지역의 주택 가격 데이터를 포함한 공개 데이터셋
- **price: 주택의 판매 가격 (종속 변수, 목표 값).**
- **sqft_living: 주택의 실내 면적 (평방 피트).**

```
[2]  # Download dataset file
     !wget "https://dongaackr-my.sharepoint.com/:x:/g/personal/sjkim_donga_ac_kr/ET2udIQfsxRAsvnIEtg

     # Load dataset file
     data = pd.read_csv('kc_house_data.csv')

     # Single linear regression 실습에 사용할 데이터 열만 수집 (price (정답), sqft_living (입력))
     X, Y = data['sqft_living'], data['price']
     print(X)
     print(Y)
```

- Y: 정답 데이터 (price)
- X: 입력 데이터 (sqft_living)

실습 환경 구성

- **Single Linear Regression을 위한 데이터셋 확인**

```
# Numpy 배열로 전환
X = np.array(X) # sqft_living
Y = np.array(Y) # price

# X, Y 각각에 대한 평균과 표준편차 계산
X_mean = np.mean(X)
Y_mean = np.mean(Y)
X_std = np.std(X)
Y_std = np.std(Y)

# 평균, 표준편차를 이용한 Gaussian 정규화 수행
X = (X - X_mean) / X_std
Y = (Y - Y_mean) / Y_std

# 2차원 행렬 변환
X = np.expand_dims(X, 1)
Y = np.expand_dims(Y, 1)
```

	sqft_living	price
0	1180	221900.0
1	2570	538000.0
2	770	180000.0
3	1960	604000.0
4	1680	510000.0
...	...	...
21608	1530	360000.0
21609	2310	400000.0
21610	1020	402101.0
21611	1600	400000.0
21612	1020	325000.0

X, Y 값의 scale이 너무 큰 경우 학습이 잘 안될 수 있음
→ Gaussian 정규화 수행

실습 환경 구성

- **Single Linear Regression을 위한 데이터셋 확인**

```
# Train dataset / Test dataset 분할 (8:2 비율)
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2,

# Test dataset 시각화
fig = plt.figure()
plt.scatter(X_test, Y_test, color='b', marker='o', s=15)
plt.xlabel("X_test (sqft_living)")
plt.ylabel("Y_test (price)")
plt.show()
```

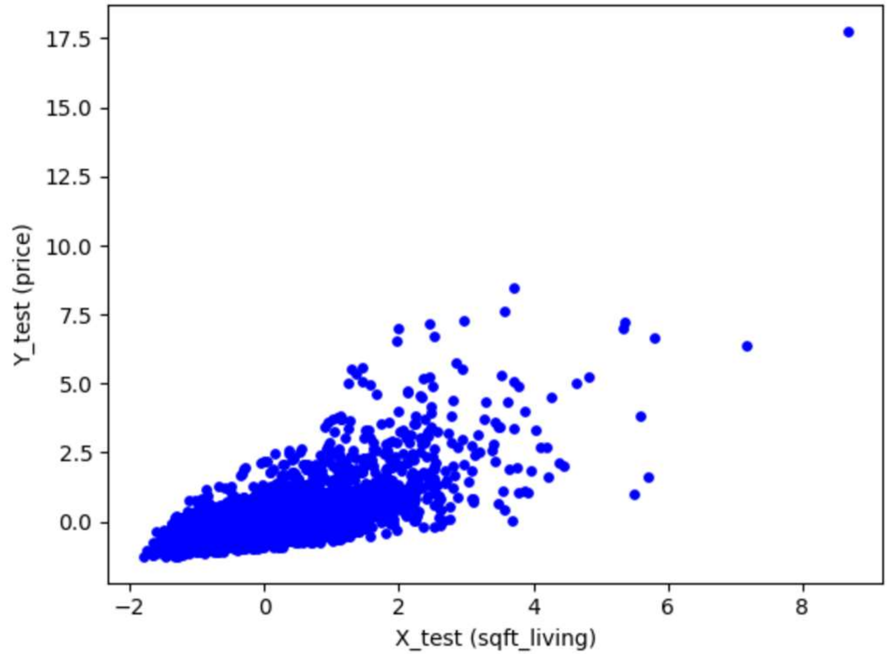
→ Train dataset : Test dataset = 8 : 2 로 분할

실습 환경 구성

- **Single Linear Regression을 위한 데이터셋 확인**

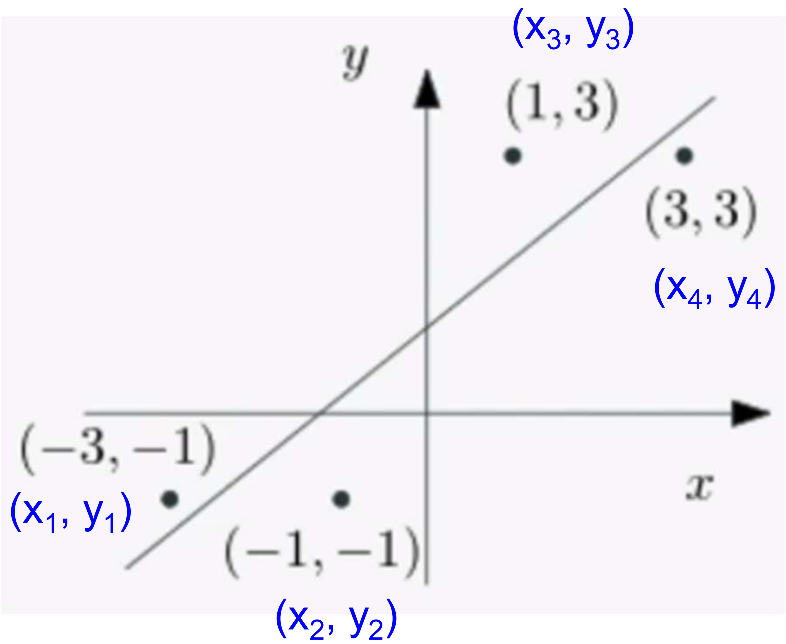
```python
# Train dataset / Test dataset 분할 (8:2 비율)
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2,

# Test dataset 시각화
fig = plt.figure()
plt.scatter(X_test, Y_test, color='b', marker='o', s=15)
plt.xlabel("X_test (sqft_living)")
plt.ylabel("Y_test (price)")
plt.show()
```



Review – Least Squares Method

- **Solutions**
  - Ordinary Least Squares (OLS) = Least Squares Method (LSM) = Normal Equation
  - Gradient Descent Method

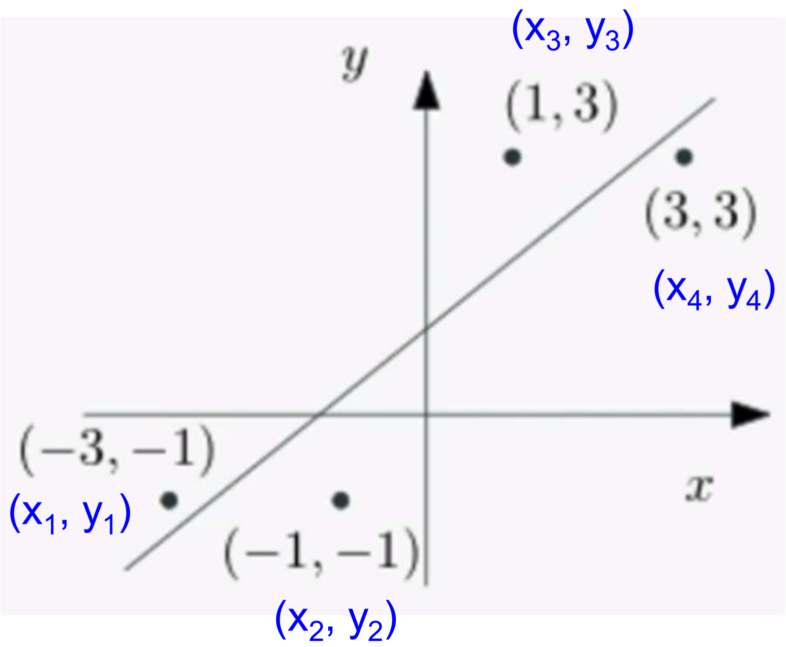


$$X = \begin{bmatrix} -3 \\ -1 \\ 1 \\ 3 \end{bmatrix} \qquad Y = \begin{bmatrix} -1 \\ -1 \\ 3 \\ 3 \end{bmatrix} \qquad \theta = \begin{bmatrix} w \\ b \end{bmatrix}$$

4x1 4x1 2x1

Review – Least Squares Method

- **Solutions**
  - Ordinary Least Squares (OLS) = Least Squares Method (LSM) = Normal Equation
  - Gradient Descent Method

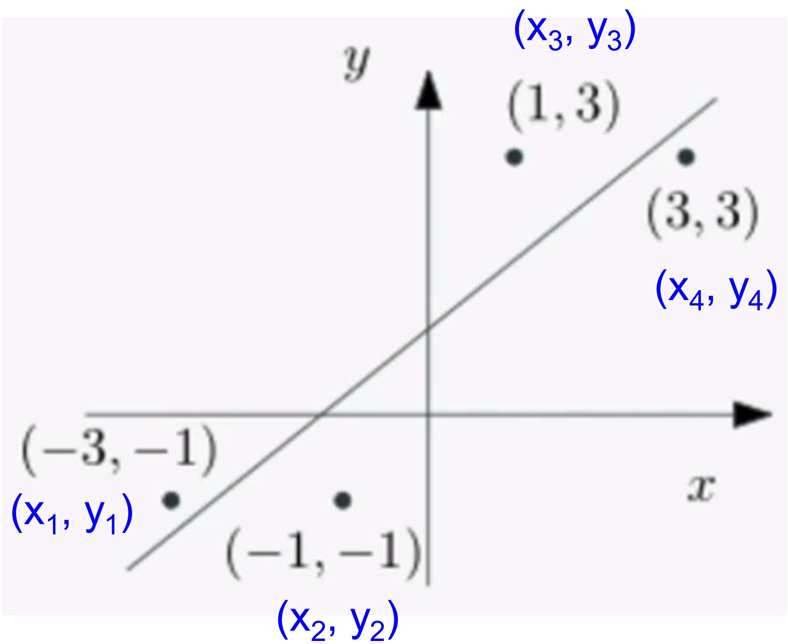


$$X = \begin{array}{c} \color{red}{\text{bias}} \\ \begin{bmatrix} -3 & 1 \\ -1 & 1 \\ 1 & 1 \\ 3 & 1 \end{bmatrix} \end{array} \quad Y = \begin{bmatrix} -1 \\ -1 \\ 3 \\ 3 \end{bmatrix} \quad \theta = \begin{bmatrix} w \\ b \end{bmatrix}$$

4x2 4x1 2x1

$X\theta = Y$ ※최종목표: $Y = \hat{Y}$ 가 되는 θ 탐색

- **Solutions**
  - Ordinary Least Squares (OLS) = Least Squares Method (LSM) = Normal Equation
  - Gradient Descent Method



bias

$$X = \begin{bmatrix} -3 & 1 \\ -1 & 1 \\ 1 & 1 \\ 3 & 1 \end{bmatrix} \quad Y = \begin{bmatrix} -1 \\ -1 \\ 3 \\ 3 \end{bmatrix} \quad \theta = \begin{bmatrix} w \\ b \end{bmatrix}$$

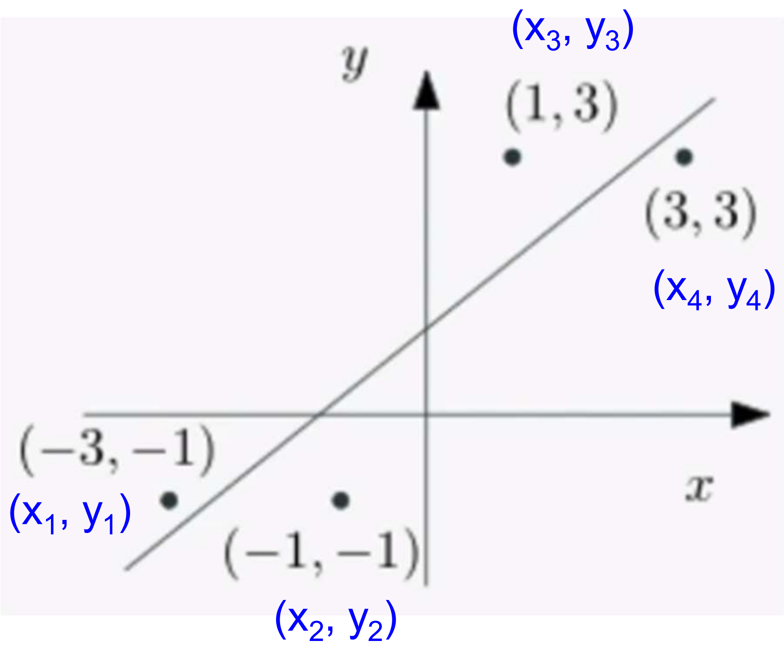
4x2 4x1 2x1

$$X\theta = Y$$

$$\theta = X^{-1} \cdot Y$$ → X가 정방행렬이 아니기 때문에 불가능

Review – Least Squares Method

- **Solutions**
  - Ordinary Least Squares (OLS) = Least Squares Method (LSM) = Normal Equation
  - Gradient Descent Method



$$X = \begin{array}{c} \begin{array}{cc} & \text{bias} \end{array} \\ \begin{bmatrix} -3 & 1 \\ -1 & 1 \\ 1 & 1 \\ 3 & 1 \end{bmatrix} \end{array} \quad Y = \begin{bmatrix} -1 \\ -1 \\ 3 \\ 3 \end{bmatrix} \quad \theta = \begin{bmatrix} w \\ b \end{bmatrix}$$

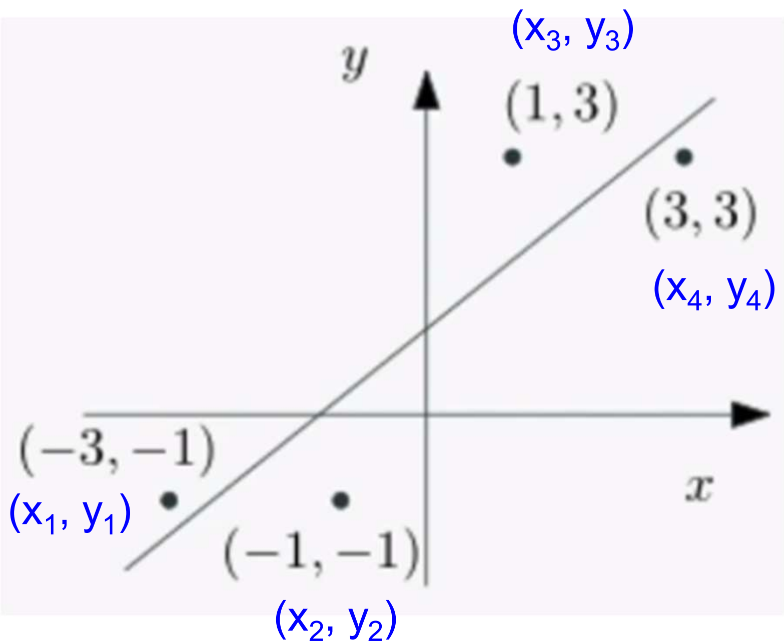
4x2 4x1 2x1

$$X\theta = Y$$

$$(X^T \cdot X)\theta = X^T \cdot Y$$

Review – Least Squares Method

- **Solutions**
  - Ordinary Least Squares (OLS) = Least Squares Method (LSM) = Normal Equation
  - Gradient Descent Method



$$X = \begin{array}{cc} & \text{bias} \\ \left[\begin{array}{c} -3 \\ -1 \\ 1 \\ 3 \end{array}\right. & \left.\begin{array}{c} 1 \\ 1 \\ 1 \\ 1 \end{array}\right] \end{array} \quad Y = \begin{bmatrix} -1 \\ -1 \\ 3 \\ 3 \end{bmatrix} \quad \theta = \begin{bmatrix} w \\ b \end{bmatrix}$$

4x2 4x1 2x1

$$X\theta = Y$$

$$(X^T \cdot X)\theta = X^T \cdot Y$$

$$\cancel{(X^T \cdot X)^{-1} \cdot (X^T \cdot X)}\theta = (X^T \cdot X)^{-1} \cdot (X^T \cdot Y)$$

실습 – Least Squares Method

- **Solutions**
  - Ordinary Least Squares (OLS) = Least Squares Method (LSM) = Normal Equation
  - Gradient Descent Method

```python
# --------- Least Squares Method 수식 구현 ---------
# Normal Equation: θ = (XᵀX)^(-1) XᵀY
# X의 전치 행렬


# XᵀX 계산

# 역행렬 계산: (XᵀX)^(-1)

# XᵀY 계산

# 최종 파라미터 θ 계산: θ = (XᵀX)^(-1) XᵀY

return self.theta
```

$$\theta = \begin{bmatrix} W \\ b \end{bmatrix}$$

$$\theta = (X^T \cdot X)^{-1} \cdot (X^T \cdot Y)$$

실습 – Least Squares Method

- **Solutions**
  - Ordinary Least Squares (OLS) = Least Squares Method (LSM) = Normal Equation
  - Gradient Descent Method

- a 행렬과 똑같은 크기의 1로 채워진 행렬 생성:

```
arr = np.ones_like(a)
```

- 행렬 가로 쌓기:

```
arr = np.hstack([a, b])
```

- 행렬 곱:

```
arr = np.dot(a, b)
```

- 전치 행렬:

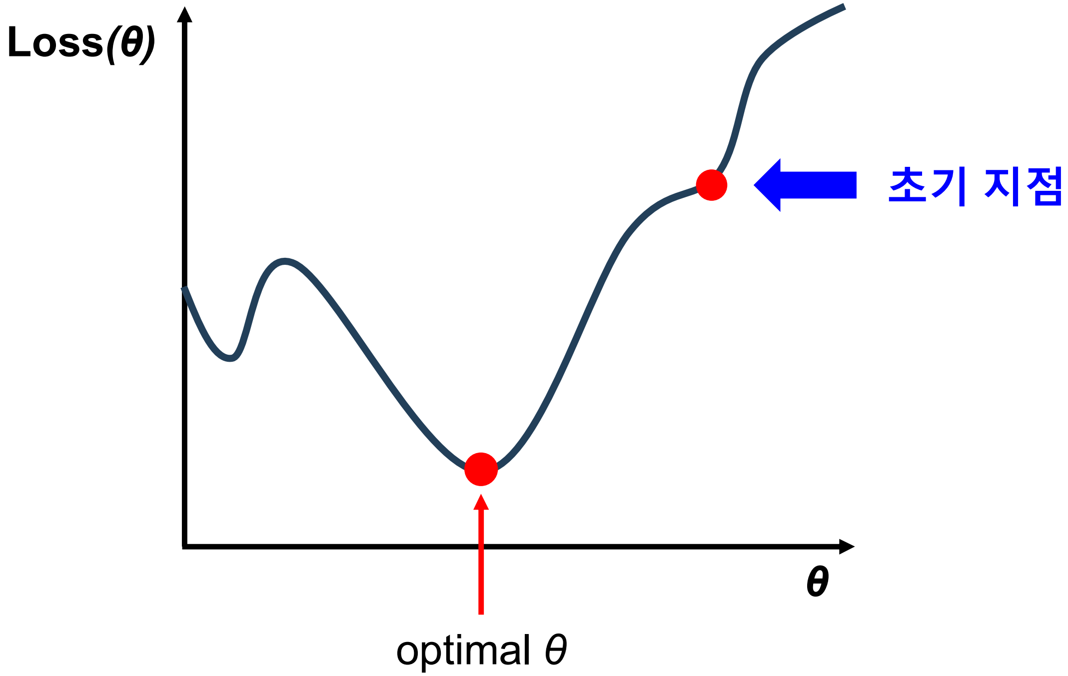
```
arr = a.T
```

- 역 행렬:

```
arr = np.linalg.inv(a)
```

Review – Gradient Descent

- **Solutions**
  - Ordinary Least Squares (OLS) = Least Squares Method (LSM) = Normal Equation
  - Gradient Descent Method



bias

$$X = \begin{bmatrix} -3 & 1 \\ -1 & 1 \\ 1 & 1 \\ 3 & 1 \end{bmatrix} \quad Y = \begin{bmatrix} -1 \\ -1 \\ 3 \\ 3 \end{bmatrix} \quad \theta = \begin{bmatrix} w \\ b \end{bmatrix}$$

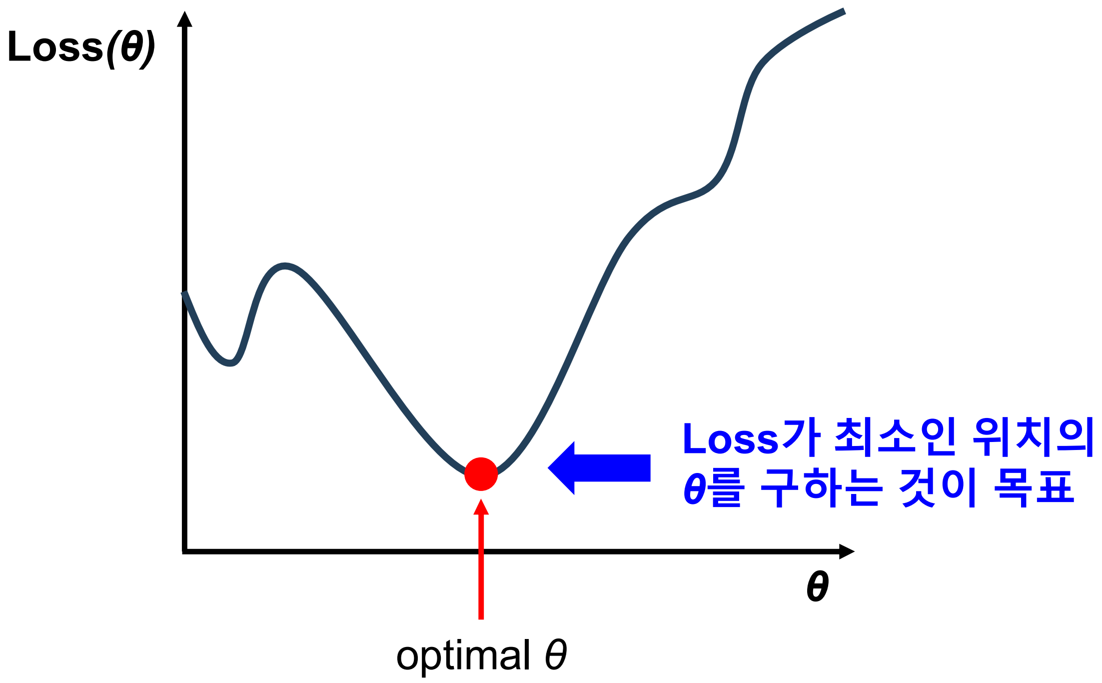
4x2 4x1 2x1

$$X\theta = \hat{Y}$$

$$Loss = \sum Y - \hat{Y}$$

Review – Gradient Descent

- **Solutions**
  - Ordinary Least Squares (OLS) = Least Squares Method (LSM) = Normal Equation
  - Gradient Descent Method



$$
\begin{array}{cc} & \text{bias} \\ X = & \begin{bmatrix} -3 & 1 \\ -1 & 1 \\ 1 & 1 \\ 3 & 1 \end{bmatrix} \end{array}
\qquad
Y = \begin{bmatrix} -1 \\ -1 \\ 3 \\ 3 \end{bmatrix}
\qquad
\theta = \begin{bmatrix} w \\ b \end{bmatrix}
$$

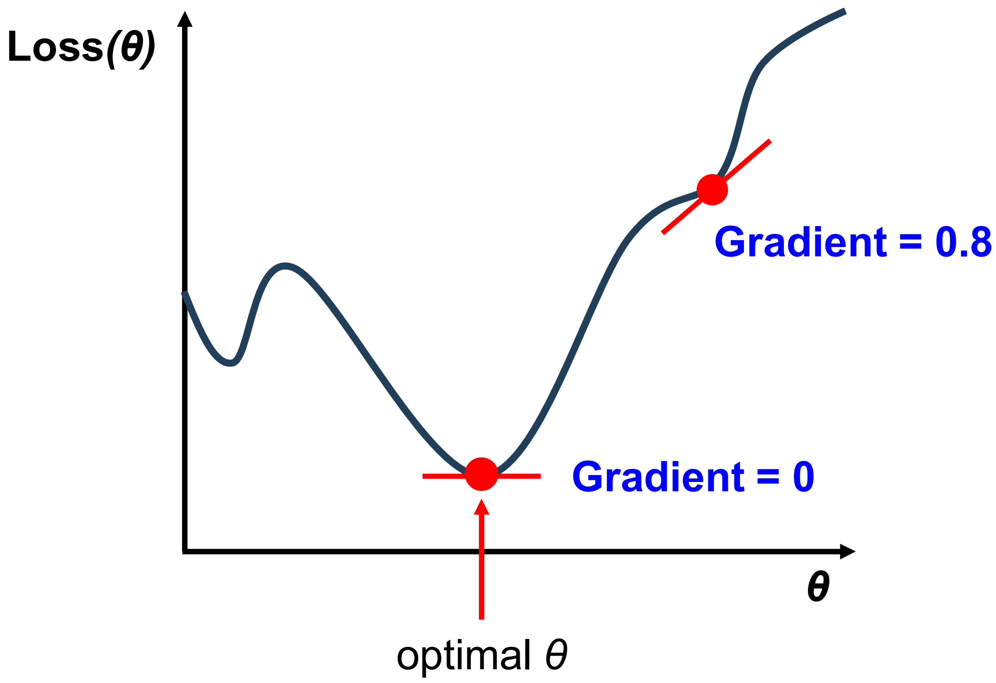
4x2 4x1 2x1

$$X\theta = \hat{Y}$$

$$Loss = \sum Y - \hat{Y}$$

Review – Gradient Descent

- **Solutions**
  - Ordinary Least Squares (OLS) = Least Squares Method (LSM) = Normal Equation
  - Gradient Descent Method



Gradient Descent algorithm

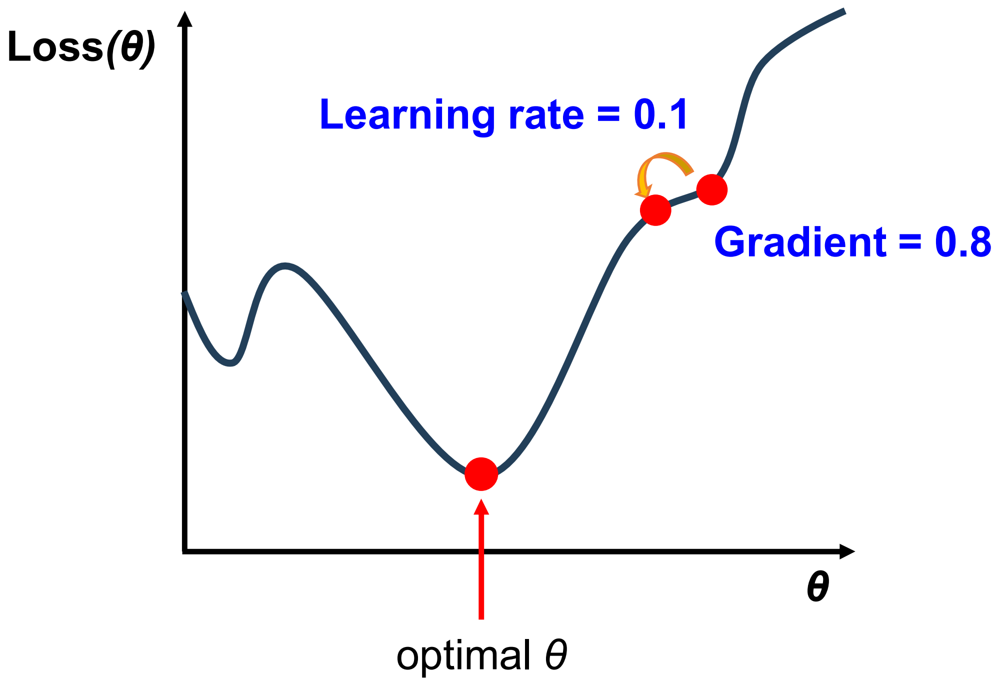
① 현재 지점에서 미분을 이용해 gradient 계산

② Gradient에 learning rate를 곱하고
반대 방향으로 weight update

$$\theta_{t+1} = \theta_t - \underbrace{\boxed{\alpha}}_{\text{Learning rate}} \underbrace{\boxed{\frac{\partial L}{\partial \theta_t}}}_{\text{Gradient}}$$

Review – Gradient Descent

- **Solutions**
  - Ordinary Least Squares (OLS) = Least Squares Method (LSM) = Normal Equation
  - Gradient Descent Method



Gradient Descent algorithm

① 현재 지점에서 미분을 이용해 gradient 계산

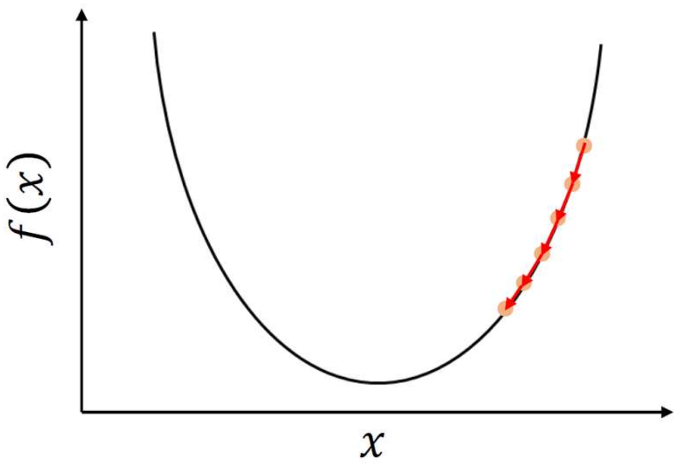
② Gradient에 learning rate를 곱하고
반대 방향으로 weight update

$$\theta_{t+1} = \theta_t - \alpha \frac{\partial L}{\partial \theta_t}$$

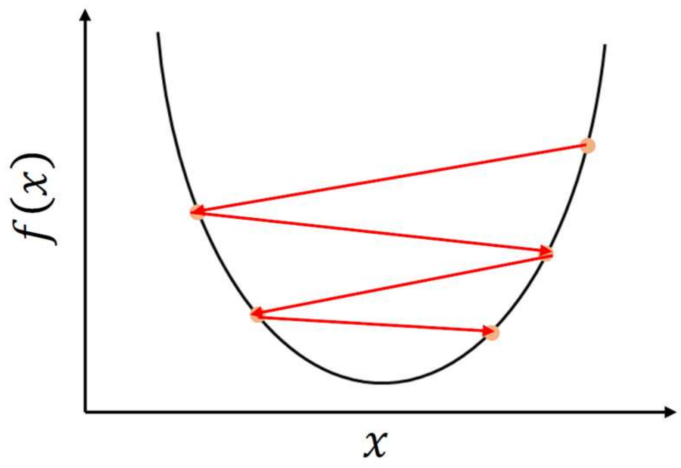
$$\boxed{= \theta_t - 0.08}$$

Review – Gradient Descent

- **Solutions**
  - Ordinary Least Squares (OLS) = Least Squares Method (LSM) = Normal Equation
  - Gradient Descent Method
    - ✓ Learning rate: 파라미터를 얼마나 업데이트할 지 정하는 **하이퍼파라미터**



α가 너무 작은 경우



α가 너무 큰 경우

❖ α: Learning rate

Review – Gradient Descent

- **Gradient Descent Method (Parameter update)**

$$\hat{Y} = wx + b$$

$$Loss = \frac{1}{N}\sum\left(Y - \hat{Y}\right)^2$$

$$= \frac{1}{N}\sum(Y - wx_i - b)^2$$

Review – Gradient Descent

- **Gradient Descent Method (Parameter update)**

$$\widehat{Y} = wx + b$$

$$Loss = \frac{1}{N}\sum\left(Y - \widehat{Y}\right)^2$$

$$= \frac{1}{N}\sum(Y - wx_i - b)^2$$

$$\frac{\partial L}{\partial w} = \frac{1}{N} \times 2 \times \sum(Y - wx_i - b) \times -x_i$$

$$\approx \frac{2}{N}\sum\left(Y - \widehat{Y}\right) \times -X$$

Review – Gradient Descent

- **Gradient Descent Method (Parameter update)**

$$\hat{Y} = wx + b$$

$$Loss = \frac{1}{N}\sum\left(Y - \hat{Y}\right)^2$$

$$= \frac{1}{N}\sum(Y - wx_i - b)^2$$

$$\frac{\partial L}{\partial w} = \frac{1}{N} \times 2 \times \sum(Y - wx_i - b) \times -x_i$$

$$\approx \frac{2}{N}\sum\left(Y - \hat{Y}\right) \times -X$$

$$\frac{\partial L}{\partial b} = \frac{1}{N} \times 2 \times \sum(Y - wx_i - b) \times -1$$

$$\approx \frac{2}{N}\sum\left(Y - \hat{Y}\right) \times -1$$

동아대학교 DONG-A UNIVERSITY

Review – Gradient Descent

- **Gradient Descent Method (Parameter update)**

$$\widehat{Y} = wx + b$$

$$Loss = \frac{1}{N}\sum(Y - \widehat{Y})^2$$

$$= \frac{1}{N}\sum(Y - wx_i - b)^2$$

$$w_{t+1} = w_t - \alpha \times \frac{\partial \mathrm{L}}{\partial \mathrm{w}}$$

$$b_{t+1} = b_t - \alpha \times \frac{\partial \mathrm{L}}{\partial \mathrm{b}}$$

$$\frac{\partial L}{\partial w} = \frac{1}{N} \times 2 \times \sum(Y - wx_i - b) \times -x_i$$

$$\approx \frac{2}{N}\sum(Y - \widehat{Y}) \times -X$$

$$\frac{\partial L}{\partial b} = \frac{1}{N} \times 2 \times \sum(Y - wx_i - b) \times -1$$

$$\approx \frac{2}{N}\sum(Y - \widehat{Y}) \times -1$$

실습 – Gradient Descent

- **Single Linear Regression 모델 작성**

```
def __init__(self, iteration=1000, learning_rate=0.1):
  self.iteration = iteration              # 반복 횟수 설정
  self.learning_rate = learning_rate      # 학습률 설정
  self.theta = None                       # 학습된 파라미터 저장 변수

def fit(self, X, Y):
  N = X.shape[0]                          # 데이터 개수

  # 행렬 X에 bias 열 추가
  bias = np.ones((N, 1))                  # (N x 1)
  X = np.hstack([X, bias])                # (N x 2)

  # w, b 초기값 설정
  w = 0.0
  b = 0.0
```

실습 – Gradient Descent

- **Single Linear Regression 모델 작성**

```
def __init__(self, iteration=1000, learning_rate=0.1):
  self.iteration = iteration                 # 반복 횟수 설정
  self.learning_rate = learning_rate         # 학습률 설정
  self.theta = None                          # 학습된 파라미터 저장 변수

def fit(self, X, Y):
  N = X.shape[0]                             # 데이터 개수

  # 행렬 X에 bias 열 추가
  bias = np.ones((N, 1))                     # (N x 1)
  X = np.hstack([X, bias])                   # (N x 2)

  # w, b 초기값 설정
  w = 0.0
  b = 0.0
```

bias

$$X = \begin{bmatrix} -3 & 1 \\ -1 & 1 \\ 1 & 1 \\ 3 & 1 \end{bmatrix}$$

4x2

실습 – Gradient Descent

- **Single Linear Regression 모델 작성**

```
for i in range(self.iteration):
  # [[w],
  #  [b]] 형태로 theta 행렬 생성


  # y_hat 계산: 예측값 = X @ theta


  # dw, db 계산
  # dw = (2/N) * sum((y - y_hat) * -X)

  # db = (2/N) * sum((y - y_hat) * -1)


  # w, b 업데이트
  # w_t+1 = w_t - learning_rate * dw

  # b_t+1 = b_t - learning_rate * db
```

$$\theta = \begin{bmatrix} w \\ b \end{bmatrix}$$

$$\hat{Y} = X\theta$$

실습 – Gradient Descent

- **Single Linear Regression 모델 작성**

```
for i in range(self.iteration):
  # [[w],
  #  [b]] 형태로 theta 행렬 생성


  # y_hat 계산: 예측값 = X @ theta


  # dw, db 계산
  # dw = (2/N) * sum((y - y_hat) * -X)

  # db = (2/N) * sum((y - y_hat) * -1)


  # w, b 업데이트
  # w_t+1 = w_t - learning_rate * dw

  # b_t+1 = b_t - learning_rate * db

```

$$\frac{\partial L}{\partial w} = \frac{1}{N}\sum(Y - \hat{Y}) \times -X$$

$$\frac{\partial L}{\partial b} = \frac{1}{N}\sum(Y - \hat{Y}) \times -1$$

$$w_{t+1} = w_t - \alpha \times \frac{\partial L}{\partial w}$$

$$b_{t+1} = b_t - \alpha \times \frac{\partial L}{\partial b}$$

Questions & Answers

Dongsan Jun (dsjun@dau.ac.kr)
Image Signal Processing Laboratory (www.donga-ispl.kr)
Dept. of Computer Engineering
Dong-A University, Busan, Rep. of Korea