Machine Learning

동아대학교 컴퓨터공학과/AI학과 한정규

Linear Regression

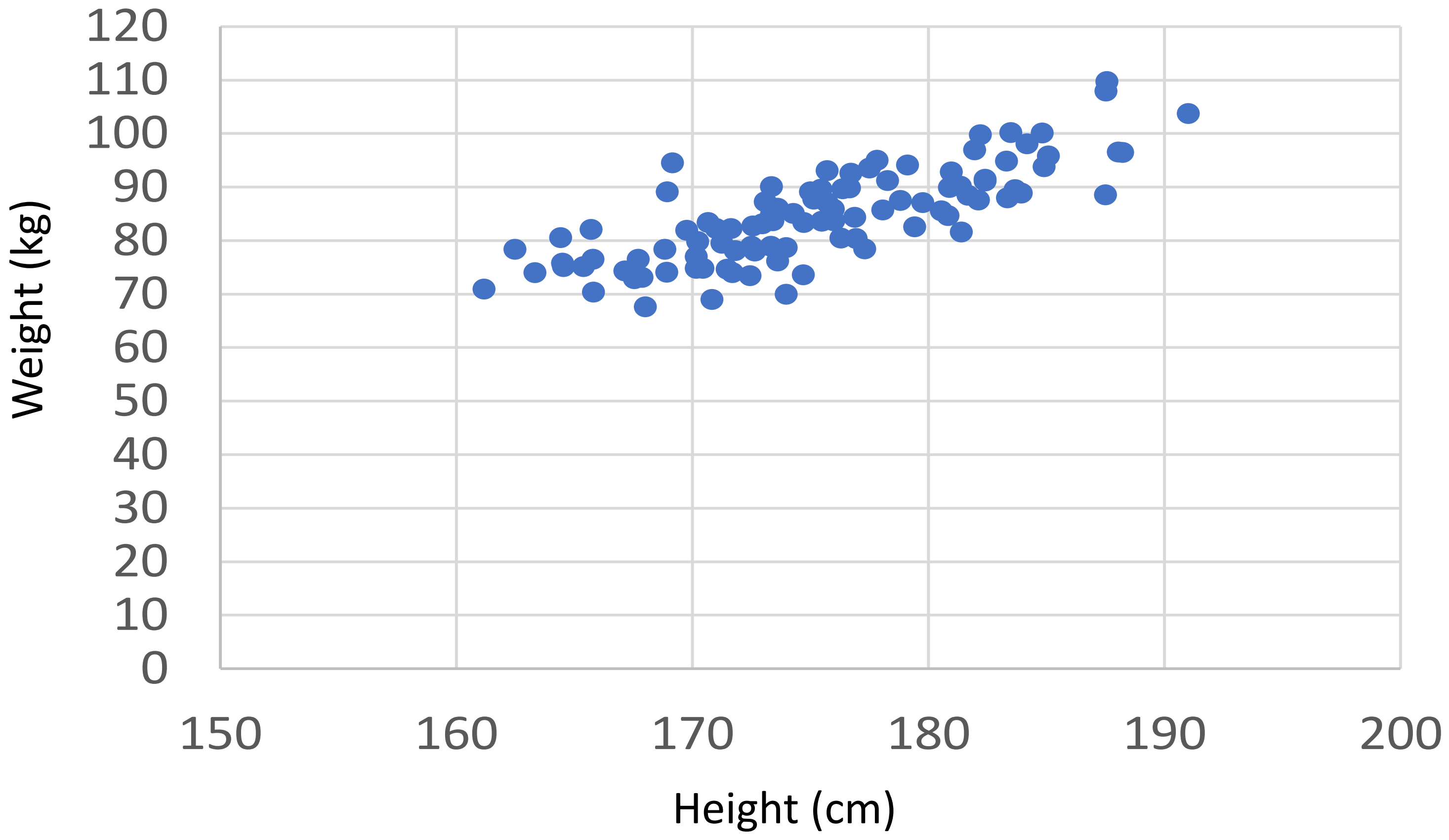
Contents

- What is Regression?
  - Simple Linear Regression
- Training : Gradient Descent
- Usage 1: Prediction
  - Apply Linear Regression on Real Dataset
- Usage 2: Correlation Analysis

What is Regression?

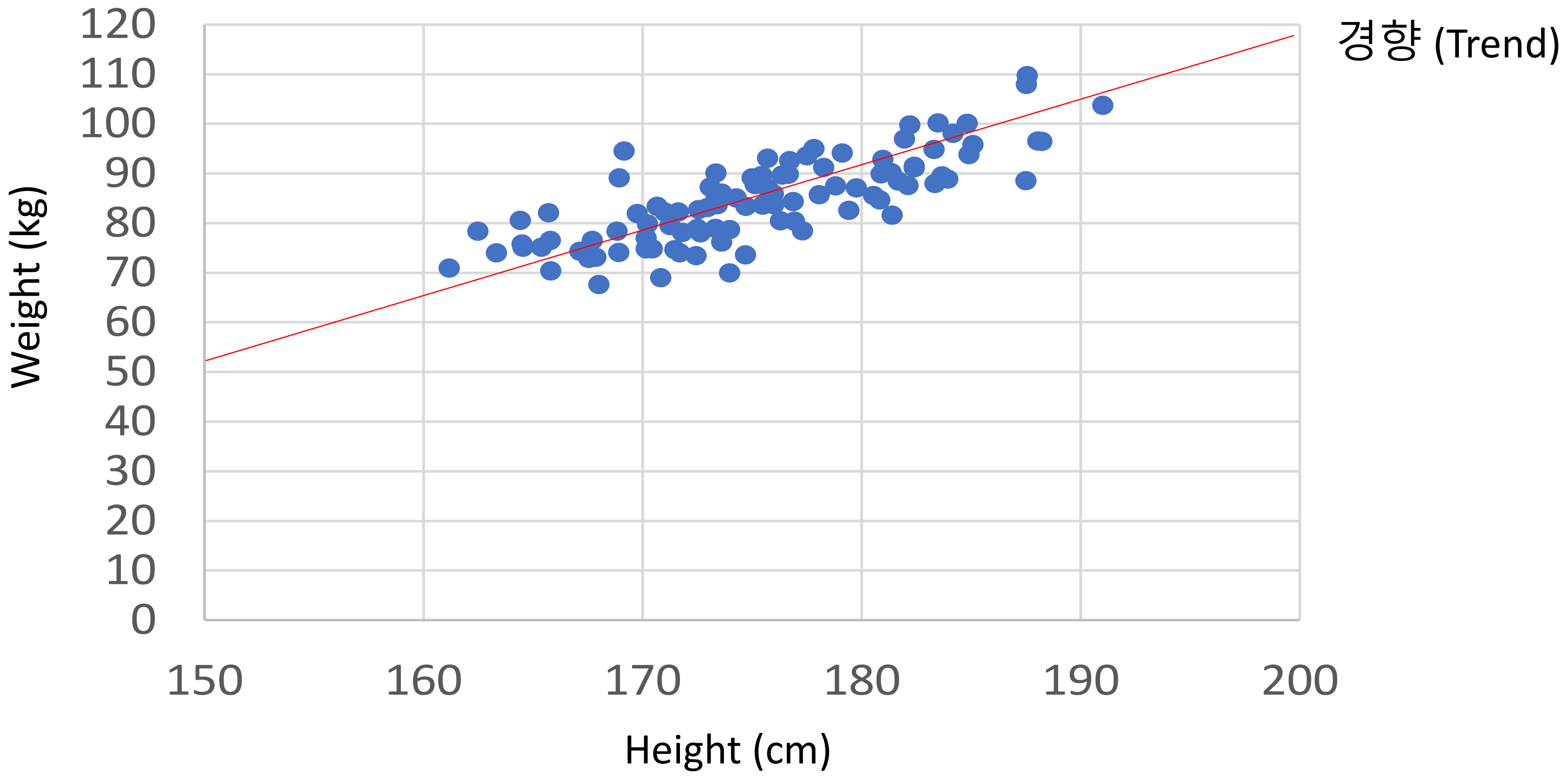
Regression

문제: 아래 그래프를 토대로 어느 사람의 키가 주어졌을 때,
우리는 그 사람의 몸무게를 어느정도는 예측할 수 있을까?



Regression (Cont'd)

만약 예측할 수 있다면 어떻게?



What is Regression? : From users' perspective.

- 주어진 데이터 인스턴스(Data Instance)를 나타내는, 이미 알고 있는 하나 혹은 여러 개의 특징(Feature, 혹은 independent variables)으로부터 목표한 특징(target variable, dependent variable)을 예측하는 것.
  - 예
    - 주어진 데이터 인스턴스: 개인 (개별적인 사람).
    - 이미 알고 있는 하나 혹은 여러 개의 특징 : 키, 운동량, 식사 칼로리 등.
    - 예측하는 특징 : 몸무게

- 예측되는 특징이 가질 수 있는 값은 연속된 실수(Real number)범위의 값이다.
  - 값의 거리 차이가 의미가 있음
  - 예) 몸무게는 60, 78.9, 109.3 등 실수 값으로 표현된다.
  - Categorical value(범주형 값)가 아님
    - Categorical value: 값이 하나의 카테고리를 나타내는 값.
    - 예 1) 성별 표현: 남자 -> 1, 여자 -> 2
    - 예 2) 영화 장르: 액션 -> 1, 로맨스 -> 2, 스릴러 -> 3, SF -> 4

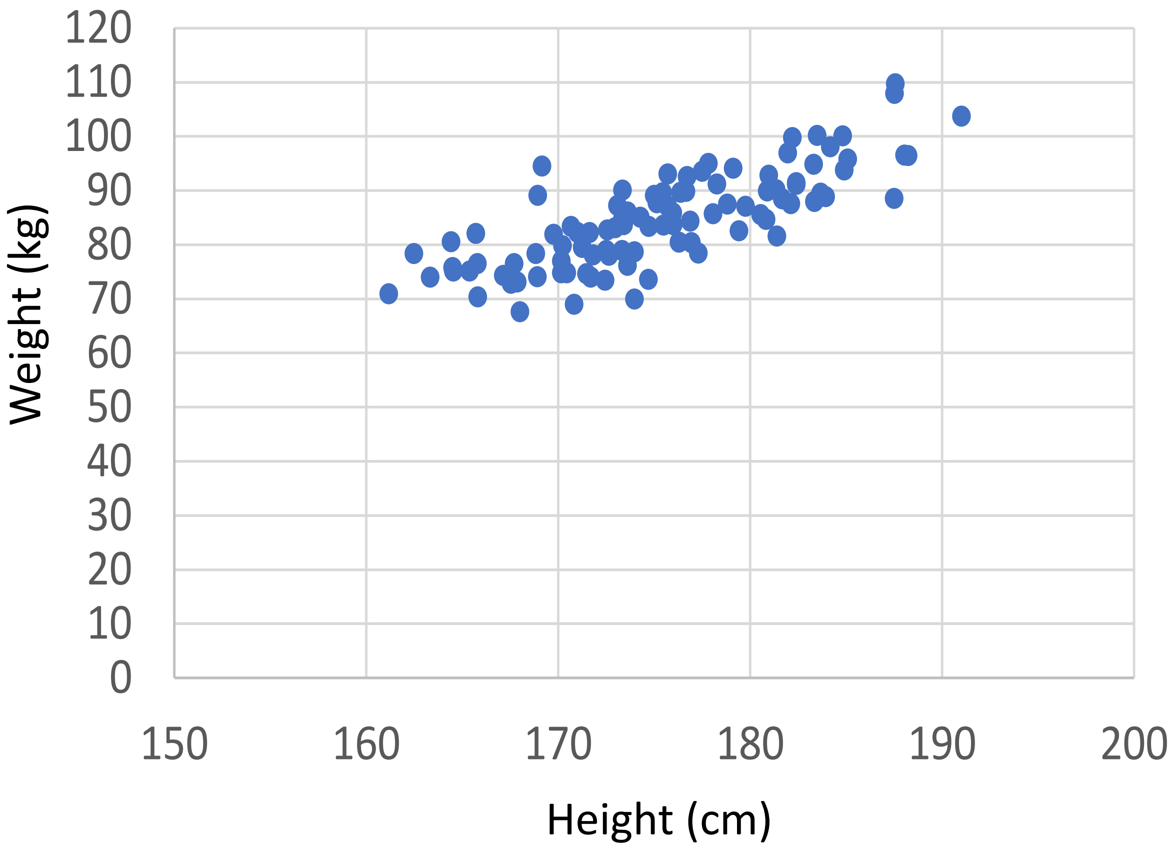
What is regression? : More formal definition

In statistical modeling, regression analysis is a set of statistical processes for estimating the relationships **between a dependent variable** (**Output variable:** often called the 'outcome' or 'response' variable) **and one or more independent variables** (**Input variables:** often called 'predictors', 'covariates', 'explanatory variables' or 'features').

(Excerpted from Wikipedia)

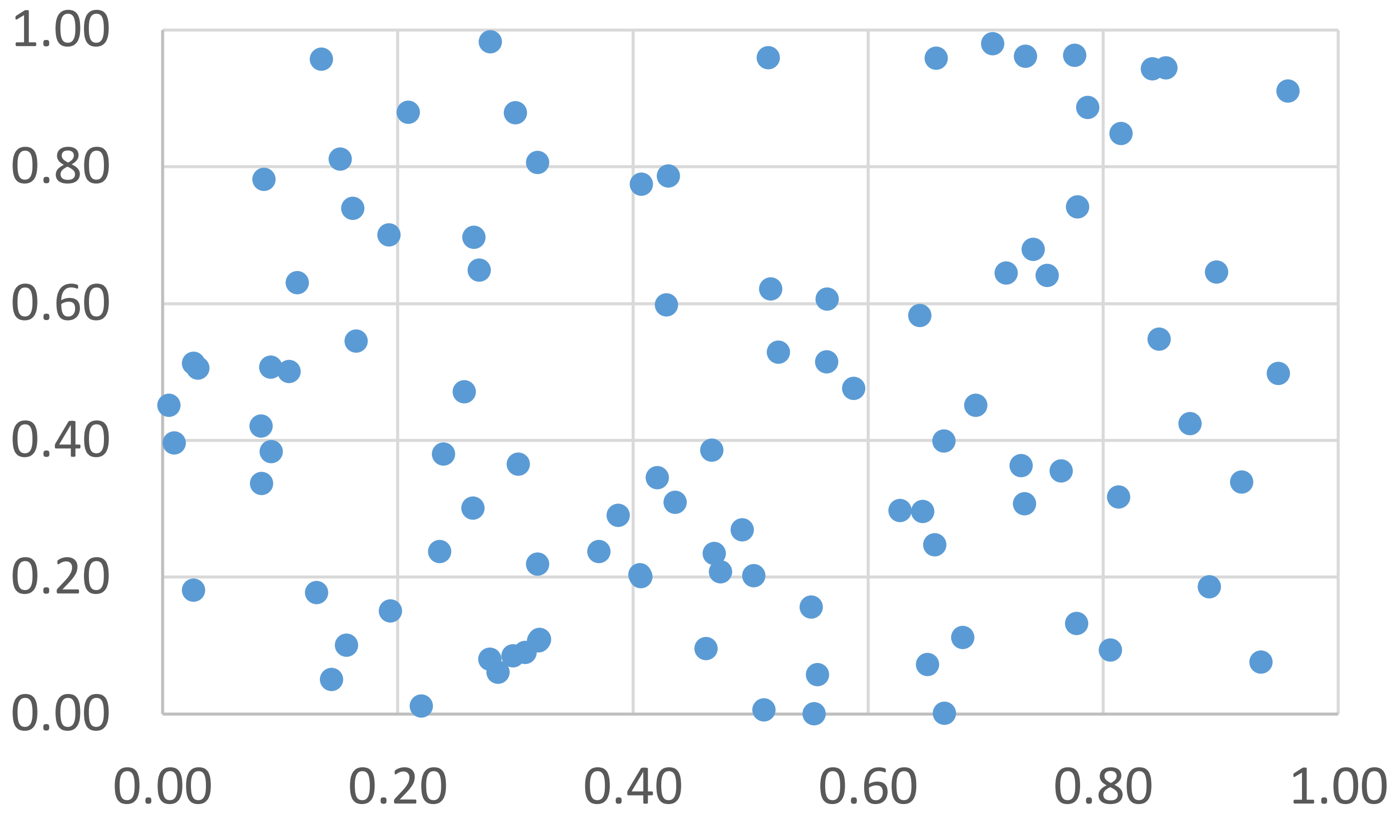
Relationships between variables

다음 그림에서 변수 Height와 Weight는 어떤 관계성이 있어 보인다?
Height가 주어지면 Weight를 어느정도 예측해 볼 수 있는가?

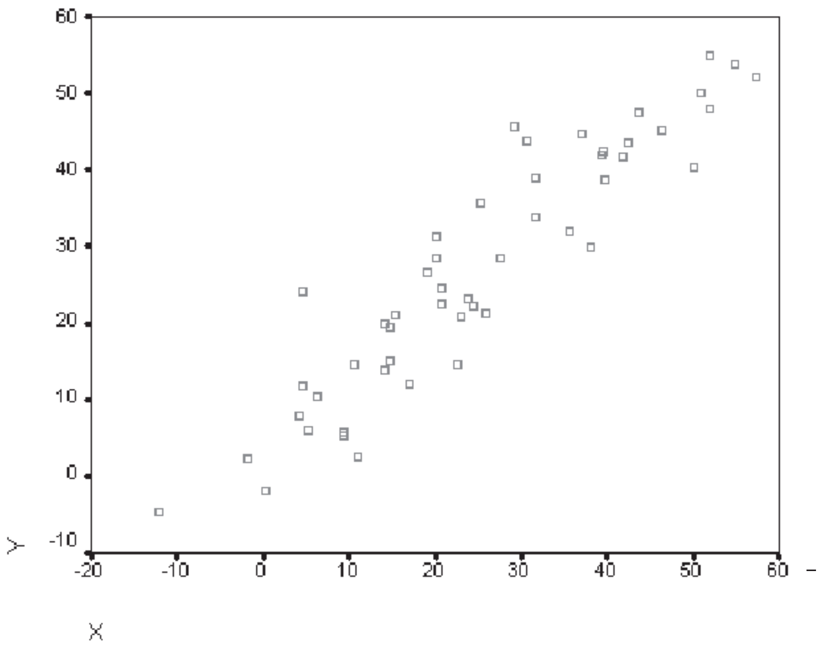


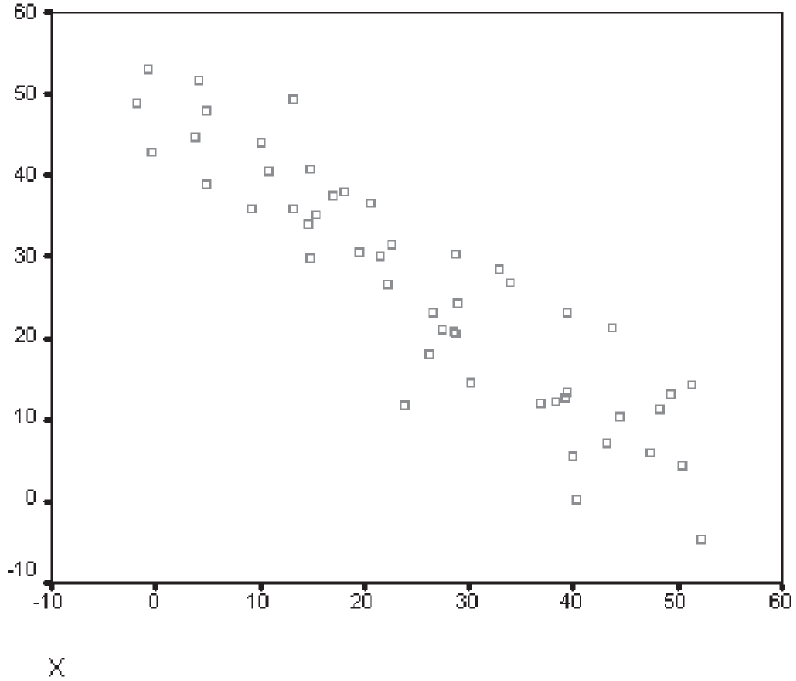
Relationships between variables (Cont'd)

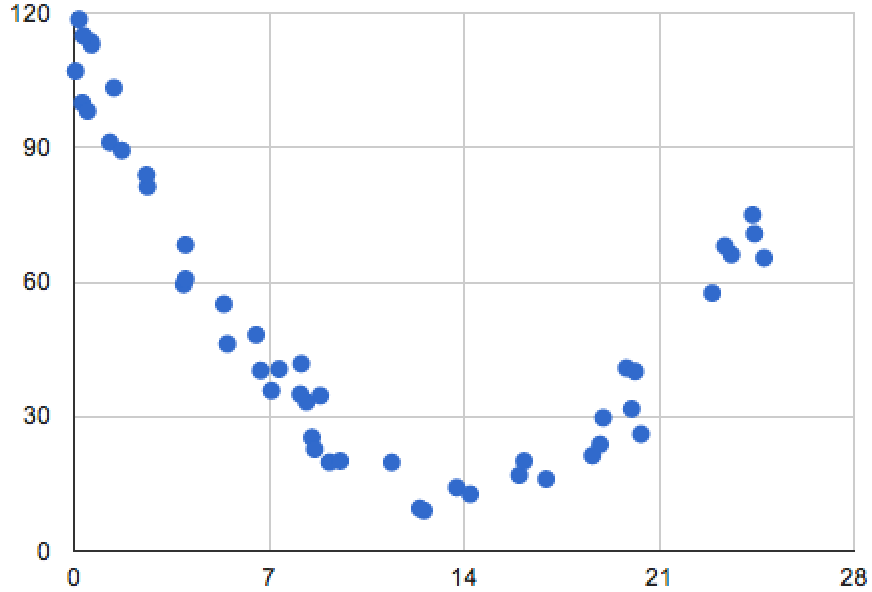
다음 그림에서 변수 Height와 Weight는 어떤 관계성이 있어 보인다?
x 값이 주어지면 y값을 어느정도 예측해 볼 수 있는가?



Relationships between variables (Cont'd)





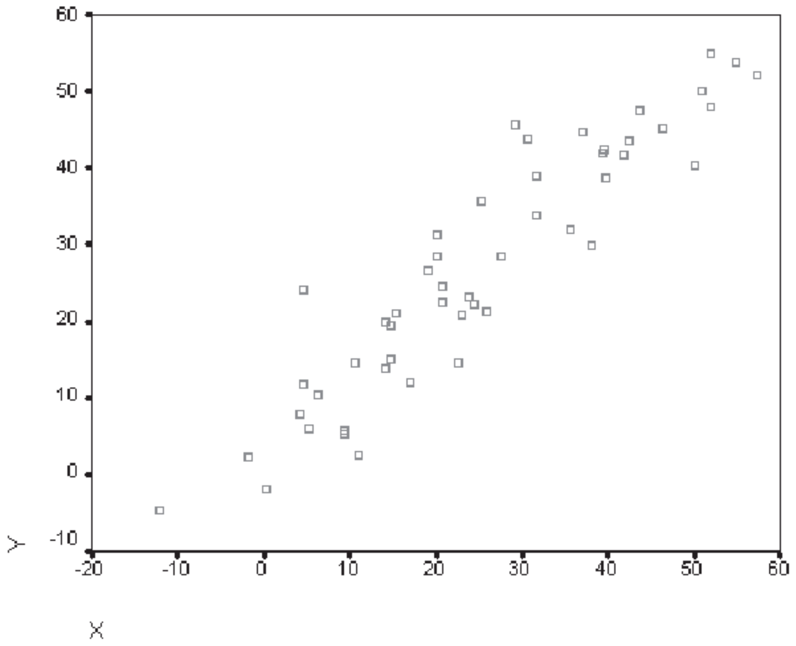


Randolph, Justus. (2007). Multidisciplinary methods in educational technology research and development.

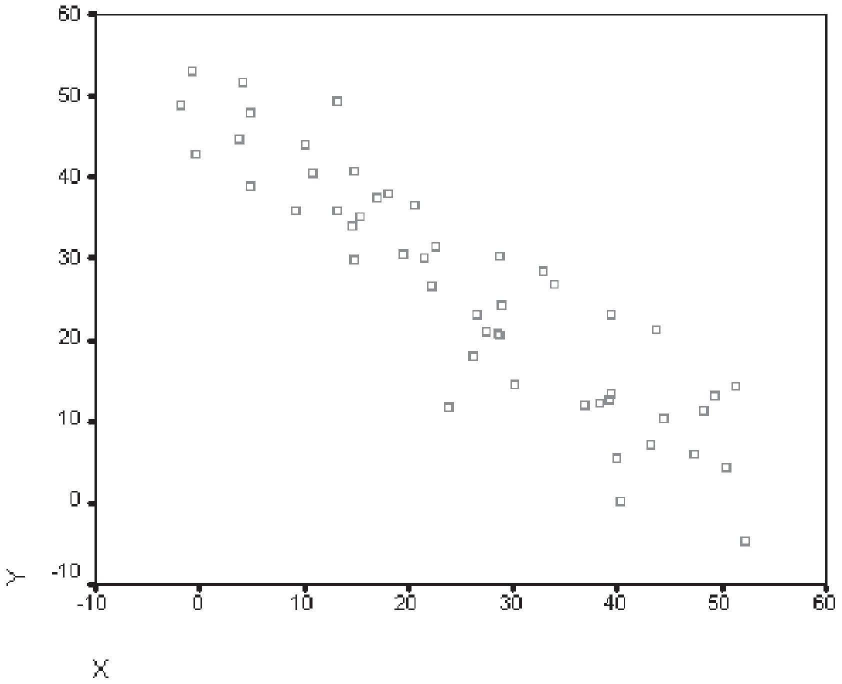
https://www.statisticshowto.com/quadratic-regression/

Linear Relation

Linear Relation : 변수 사이의 관계가 직선 (n 차원 공간에서는 hyper-plane으로 표현되는 관계)



Positive Linear Relation between x and y.



Negative Linear Relation between x and y.

What is regression? : More formal definition (Again)

In statistical modeling, regression analysis is a set of statistical processes for **estimating the relationships** **between a dependent variable** (**Output variable:** often called the **'outcome'** or 'response' variable) **and one or more independent variables** (**Input variables:** often called 'predictors', 'covariates', 'explanatory variables' or **'features**').

(Excerpted from Wikipedia)

Linear Regression

- **Linear Regression이란**
  - **Outcome variable 을 하나 이상의 feature variable들의 선형 결합으로 예측하는 Regression.**

$$y = \sum_{i=1}^{k} w_i x_i + b$$

y : Outcome variable (예측하려는 변수).
$\boldsymbol{x_1}, \boldsymbol{x_2},\ldots, \boldsymbol{x_k}$: Feature variables. (이하 줄여서 feature라 한다.)
$\boldsymbol{w_1}, \boldsymbol{w_2},\ldots, \boldsymbol{w_k}$: Weights for each feature variable.
b : bias

Feature 3개인 Linear Regression 예)

$$y = 1.5 * x_1 - 0.1 * x_2 + 0.5\, x_3 + b$$

y : 몸무게
$\boldsymbol{x_1}$: 평균 식사량
$\boldsymbol{x_2}$: 운동량
$\boldsymbol{x_3}$: 복부 둘레

Why linear regression?

- Imagine we want to **predict the outcome variable y by function f of k input features** such as $\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_k}$.
  - $y = f(\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_k})$

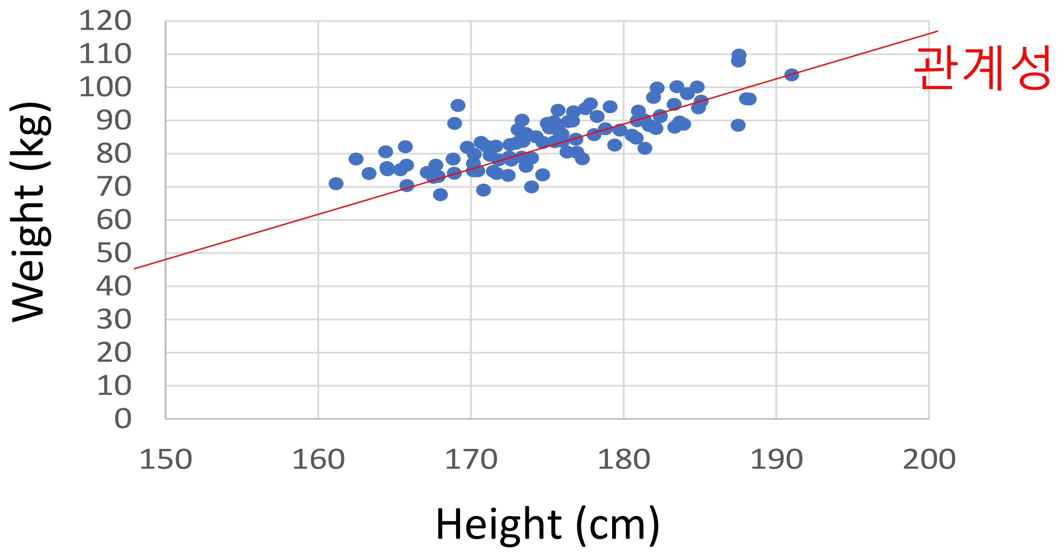
- However, in most cases, we **do not have enough data to directly estimate f**.

- Therefore, we usually have to **assume that it has some restricted form (Assumption, or model)**, such as linear.
  - $y = \sum_{i=1}^{k} w_i x_i + b = \boldsymbol{w} \cdot \boldsymbol{x} + \boldsymbol{b}$
  - $\boldsymbol{w} = <\boldsymbol{w_1}, \boldsymbol{w_2}, \ldots, \boldsymbol{w_k}>$
  - $\boldsymbol{x} = <\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_k}>$

Simple Linear Regression

- Feature 가 하나밖에 없는 다음 Linear regression을 생각해 보자.
  - $y = \sum_{i=1}^{1} w_i x_i + b$
  - <=> $y = w_1 x_1 + b$
  - <=> $y = wx + b$ (하나 밖에 없으므로 w, x에서 subscript 1을 삭제)

- 즉, 위와 같은 형태의 단순한 Linear Regression은 하나의 feature와 예측하려는 outcome variable와의 선형 관계(Linear Relationship)를 학습하는 것
- 예) Height (Feature)와 Weight (Outcome variable)의 관계성



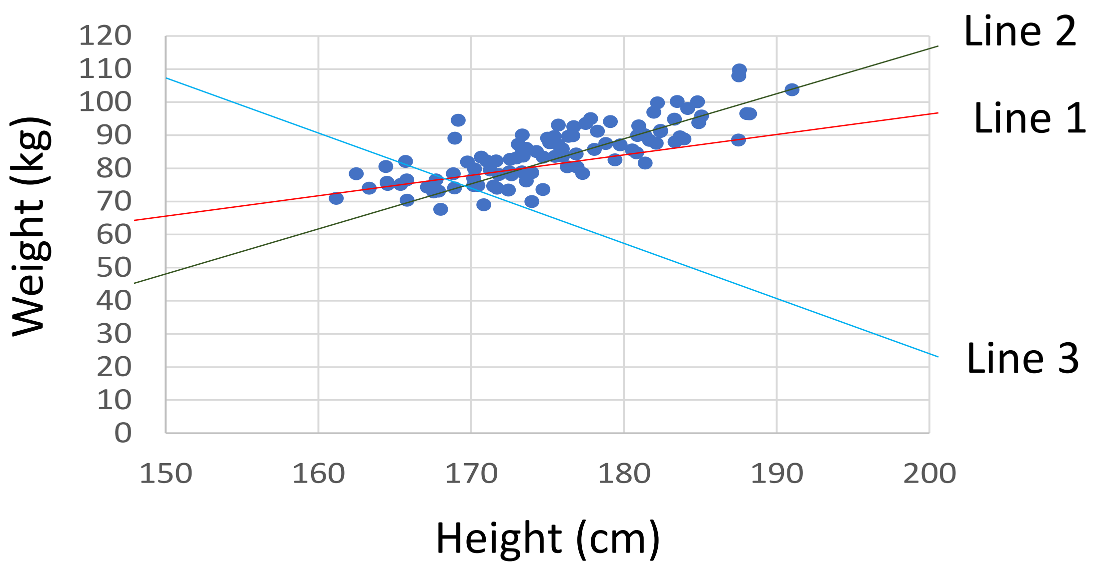
Training :
Gradient Descent

Simple Linear Regression

- Simple Linear regression에서 y 와 x 과의 관계는 아래 식으로 표현된다.
  - $y = wx + b$

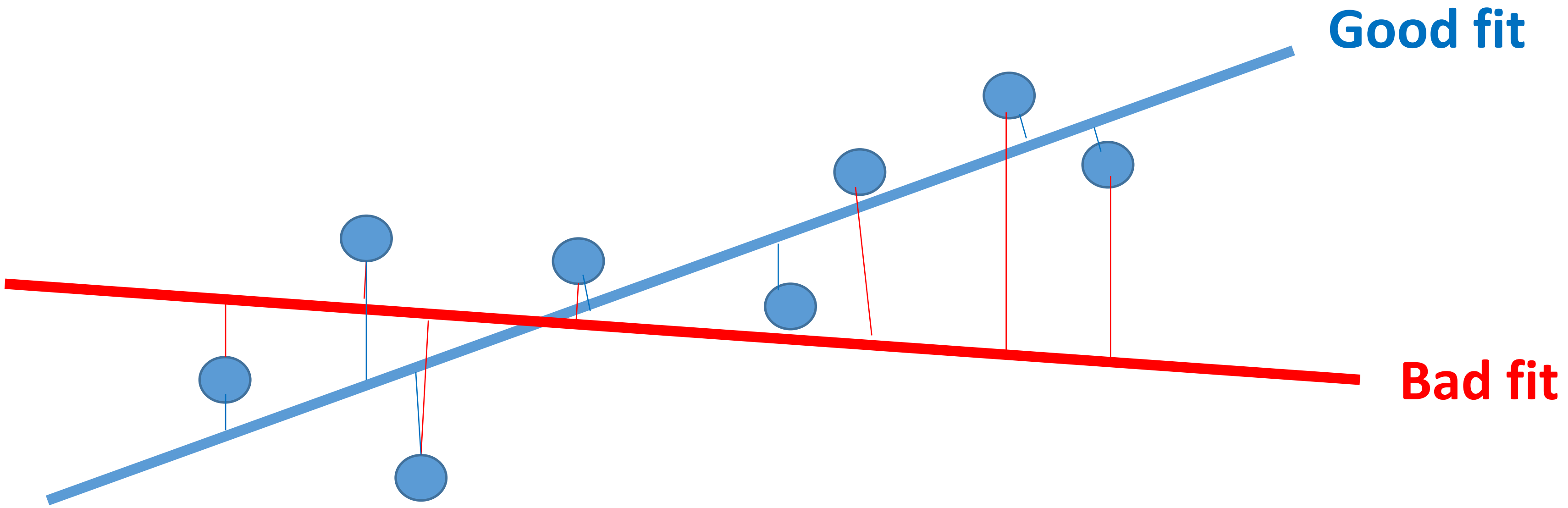
- 선형 관계(Linear Relationship)를 학습하는 것은 두 변수(y 와 x) 사이의 관계성을 가장 잘 표현하는 직선을 찾는 것이 된다.
  - 다르게 말하면 위 식에서 w, b 의 값을 찾는 것이 된다.

- 그렇다면 어떤 직선이 관계성을 가장 잘 표현하는 직선인가?



How to fit (Train)? : Minimum Error

각각의 Training data와 예측한 선 사이의 거리의 합이 적으면 적을수록 좋다.



Objective Function & Model Parameter

- **Objective Function (목적함수)** : 모델학습을 위한 목표(방향, 가이드라인)을 제시하는 함수.
  - 일반적으로 모델 학습은 **목적함수의 값을 Minimization(최소화)** 혹은 Maximization(최대화)하는 **Model parameter 값을 학습하는** 것이다.
  - 학습 잘 된 모델이란 목적함수의 최소값 혹은 최대값에 최대한 근접한 parameter를 학습한 모델이다.

- **Model parameter**
  - 식으로 나타낸 모델에서 입력과 출력 이외의, 그 값을 학습해야 할 변수.
  - 모델의 실제 형태(instance)를 결정하는 parameter
  - Programming의 Class 및 instance의 관계와 유사
    - Line => class, 실제 그려진 직선 => instance
  - 예) 다음 simple linear regression 식 $y = wx + b$ 에서 학습해야 할 Model parameter는?

Objective Function : MSE (Mean Squared Error)

- **MSE:**
- **각각의 Training data에 대해, “모델을 사용하여 예측한 값”과 “실제 값”의 차이의 제곱의 합.**

Training data 가 $< x_1, y_1 >, < x_2, y_2 > \ldots, < x_n, y_n >$ 의 n쌍 있을 때, Simple linear regression “$y = wx + b$” 의 MSE는 다음과 같다.

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$
$$= \frac{1}{n}\sum_{i=1}^{n}(y_i - (w \cdot x_i + b))^2$$

$\boldsymbol{y_i}$: i 번째 데이터에 대한 정답값.
$\boldsymbol{\hat{y}_i}$: i 번째 데이터에 대한 예측값.
$\boldsymbol{n}$: data의 개수
$\boldsymbol{x_i}$: i 번째 데이터에 대한 입력값.

Training

- MSE의 값을 최소화하기 위한 w과 b의 값은 어떻게 구하는가?

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - (w \cdot x_i + b))^2$$

를 잘 살펴보면, MSE는 w, b 를 입력 변수로 하는 함수라 볼 수 있음

$$\text{MSE}(w, b) = \frac{1}{n}\sum_{i=1}^{n}(y_i - (w \cdot x_i + b))^2$$

그렇다면 n, y_i, x_i는? 변수?

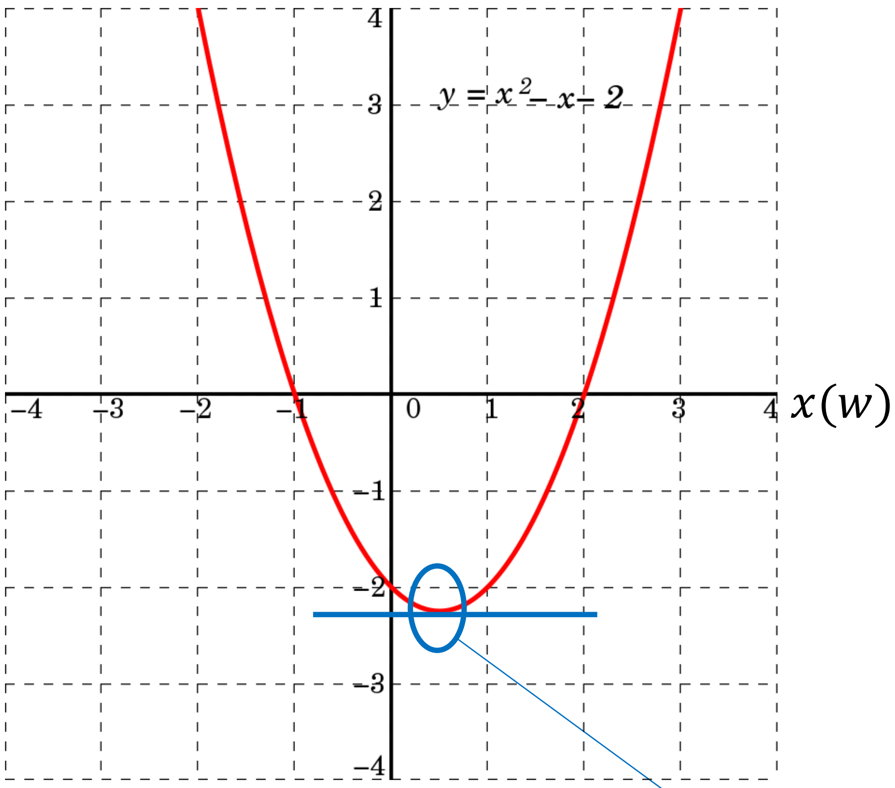
직접 해보기: $n = 2$라고 할 경우 위 MSE의 식을 전개하여 w에 대해 정리하라.

Training (Cont'd)

- $l_i = x_i^2 \cdot w^2 + 2(x_i \cdot b - x_i \cdot y_i) \cdot w + y_i^2 - 2 \cdot b \cdot y_i + b^2$
- 에서 MSE를 w 에 관한 함수로 보면 MSE는 **w의 2차 함수다**.
- **또는 b 의 2차함수이다.**

2차 함수 (Convex function or Concave Function)의 특징.

y(MSE)



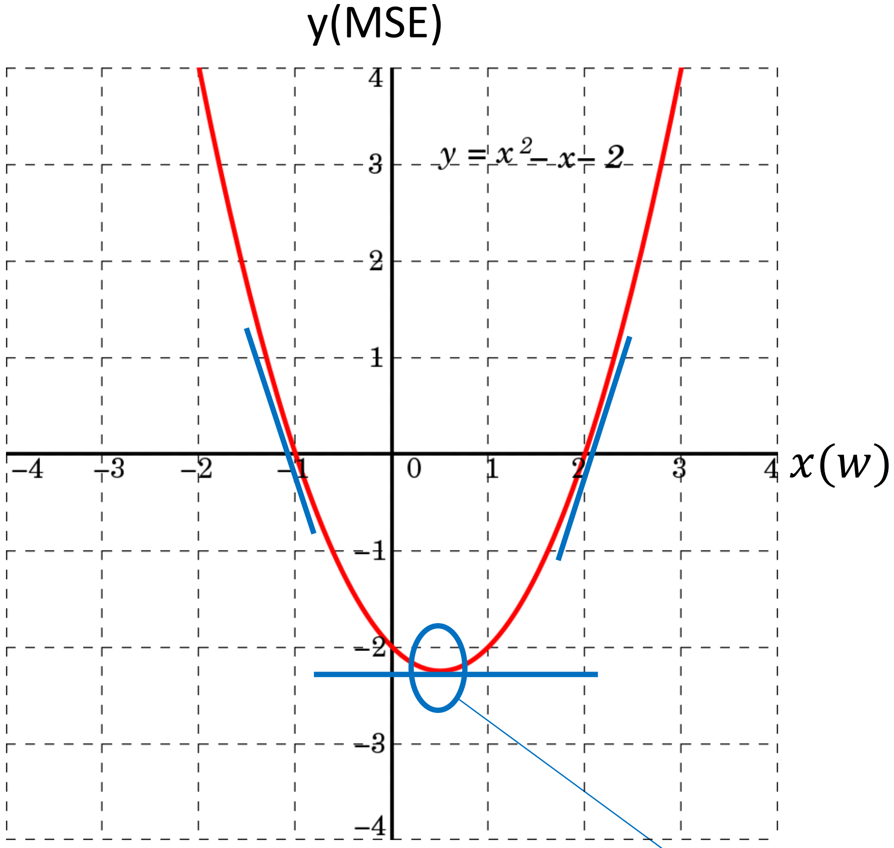
2차 함수 I이 가질 수 있는 최대, 혹은 최소점은 하나이다.

혹은 최소점에서의 Gradient(미분값)은 0이다.

Gradient가 0인 점은 최소(Convex Function일 경우) 혹은 최대(Concave Function일 경우) 점 뿐이다.

Convex Function의 최소점: 기울기 0.

2차 함수 (Convex function or Concave Function)의 특징 (Cont'd)

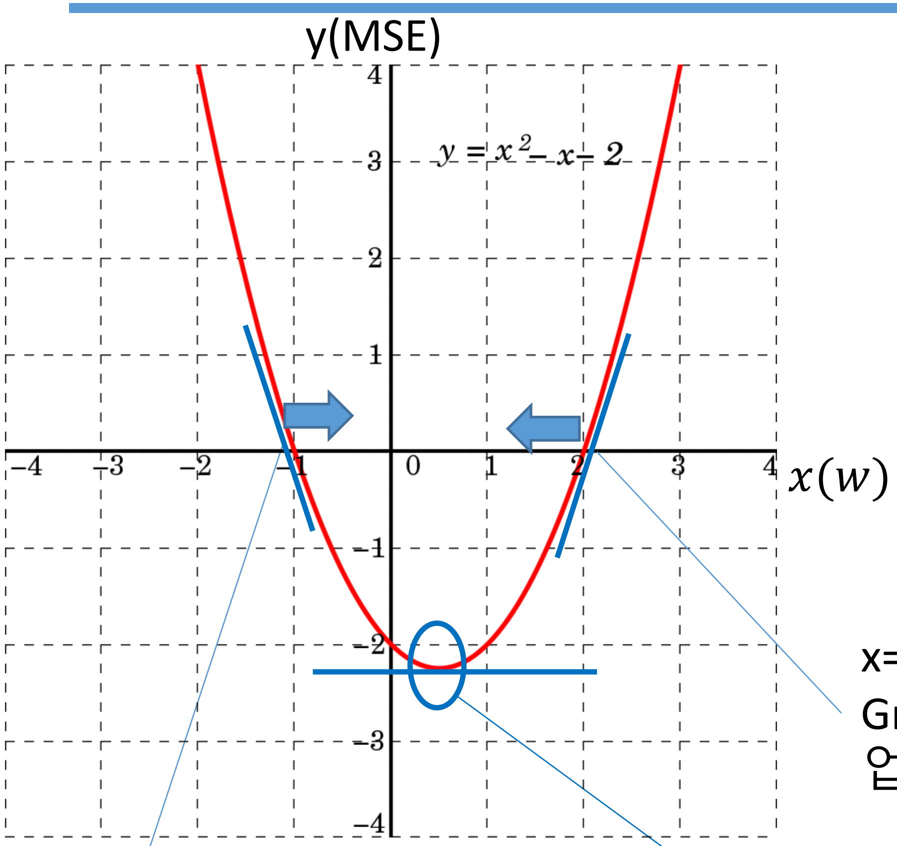


Convex Function에서
최소점보다 x (parameter)가 큰 점은 Gradient가 항상 +이다.
최소점보다 x (parameter)가 작은 점은 Gradient가 항상 –이다.

(Concave Function에서
최대점보다 x (parameter)가 큰 점은 Gradient가 항상 –이다.
최대점보다 x (parameter)가 작은 점은 Gradient가 항상 +이다.
)

Convex Function의 최소점: 기울기 0.

Stochastic Gradient Descent (Cont'd)



Convex Function에서
최소점보다 x (parameter)가 큰 점은 Gradient가 항상 +이다.
최소점보다 x (parameter)가 작은 점은 Gradient가 항상 –이다.

따라서 현재 x 값을 바탕으로 함수의 최소점을 찾고 싶다면,
현재의 Gradient의 반대 방향으로 x 값을 움직여서 (update)해서
Gradient가 0인 x 값을 찾아 나간다.

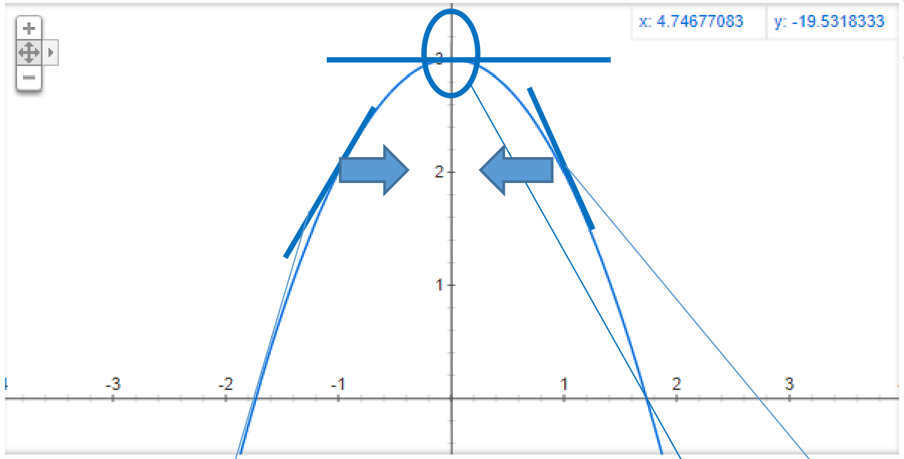
x=2 에서 최소 점을 찾기 위해서는 x=2 를
Gradient의 반대방향 (음의 방향)으로
업데이트.

Convex Function의 최소점: 기울기 0.

x=-1 에서 최소 점을 찾기 위해서는 x=-1
를 Gradient의 반대방향 (양의 방향)으로
업데이트.

Stochastic Gradient Descent (Cont'd)

Concave Function에서
최대점보다 x (parameter)가 큰 점은 Gradient가 항상 -이다.
최대점보다 x (parameter)가 작은 점은 Gradient가 항상 +이다.



따라서 현재 x 값을 바탕으로 함수의 최소점을 찾고 싶다면, 현재의 **Gradient의 방향**으로 x 값을 움직여서 (update)해서 Gradient가 0인 x 값을 찾아 나간다.

x=1 에서 최대점을 찾기 위해서는 x=1 를 Gradient의 방향 (음의 방향)으로 업데이트.

Concave Function의 최대점: 기울기 0.

x=-1 에서 최소 점을 찾기 위해서는 x=-1 를 Gradient의 방향 (양의 방향)으로 업데이트.

Gradient Descent 알고리즘

Objective Function f의 값을 최소화하는 입력값 w에 대한 최적값 (Optimal value) w^*은,
최신 w값에 다음 update를 반복하여 계산할 수 있음

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \boldsymbol{\alpha}\frac{\partial f}{\partial w}$$

α: **learning rate**

$\mathrm{MSE}(w,b) = \frac{1}{n}\sum_{i=1}^{n}(y_i - (w \cdot x_i + b))^2$ 에 대한 $\boldsymbol{w}$의 update는 다음과 같이 쓸 수 있음

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \boldsymbol{\alpha}\frac{\partial \mathrm{MSE}(w,b)}{\partial w}$$

$$\frac{\partial \mathrm{MSE}(w,b)}{\partial w} = \frac{\partial \frac{1}{n}\sum_{i=1}^{n}\left(y_i-(w\cdot x_i+b)\right)^2}{\partial w} = \frac{1}{n}\sum_{i=1}^{n}\frac{\partial\left(y_i-(w\cdot x_i+b)\right)^2}{\partial w}$$ (by 미분의 합규칙)

Gradient Descent Algorithm for Linear Regression

Input : Training Data ***D***: ($< x_1, y_1 >, < x_2, y_2 > \ldots, < x_n, y_n >$),
Objective Function: f, learning rate: $\boldsymbol{\alpha}$, max_iterations: $\boldsymbol{i_{max}}$, epsilon : ε
Output : w, b.
{
Randomly initialize the elements of w and b.
l_{old} = 0
for 1 to $\boldsymbol{i_{max}}$:
    $\boldsymbol{w} \leftarrow \boldsymbol{w} - \boldsymbol{\alpha}\frac{\partial f}{\partial w}$
    $\boldsymbol{b} \leftarrow \boldsymbol{b} - \boldsymbol{\alpha}\frac{\partial f}{\partial b}$
    calculate l = $\sum_{i=0}^{n}(y_i - (w \cdot x_i + b))^2$

    if |l - l_{old} | < ε :
        break;
    else
        l_{old} = l
endfor
Return $\boldsymbol{w}$, $\boldsymbol{b}$.
}

Gradient Descent Algorithm의 문제

- 계산 속도가 느리다.
  - 왜?

Stochastic Gradient Descent

$$\frac{\partial \mathrm{MSE}(w,b)}{\partial w} = \frac{\partial \frac{1}{n}\sum_{i=1}^{n}\left(y_i-(w\cdot x_i+b)\right)^2}{\partial w} = \frac{1}{n}\sum_{i=1}^{n}\frac{\partial\left(y_i-(w\cdot x_i+b)\right)^2}{\partial w}$$ (by 미분의 합규칙)

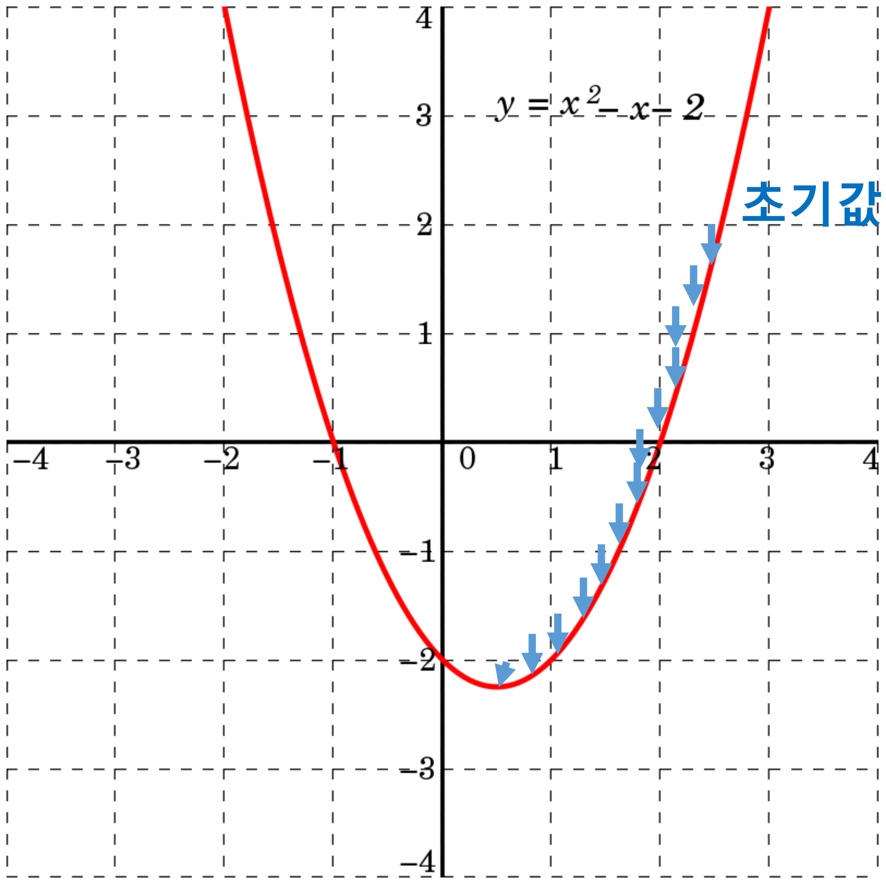
$\left(y_i-(w\cdot x_i+b)\right)^2 = l_i$ 라 표현하면

Stochastic Gradient Descent for linear regression

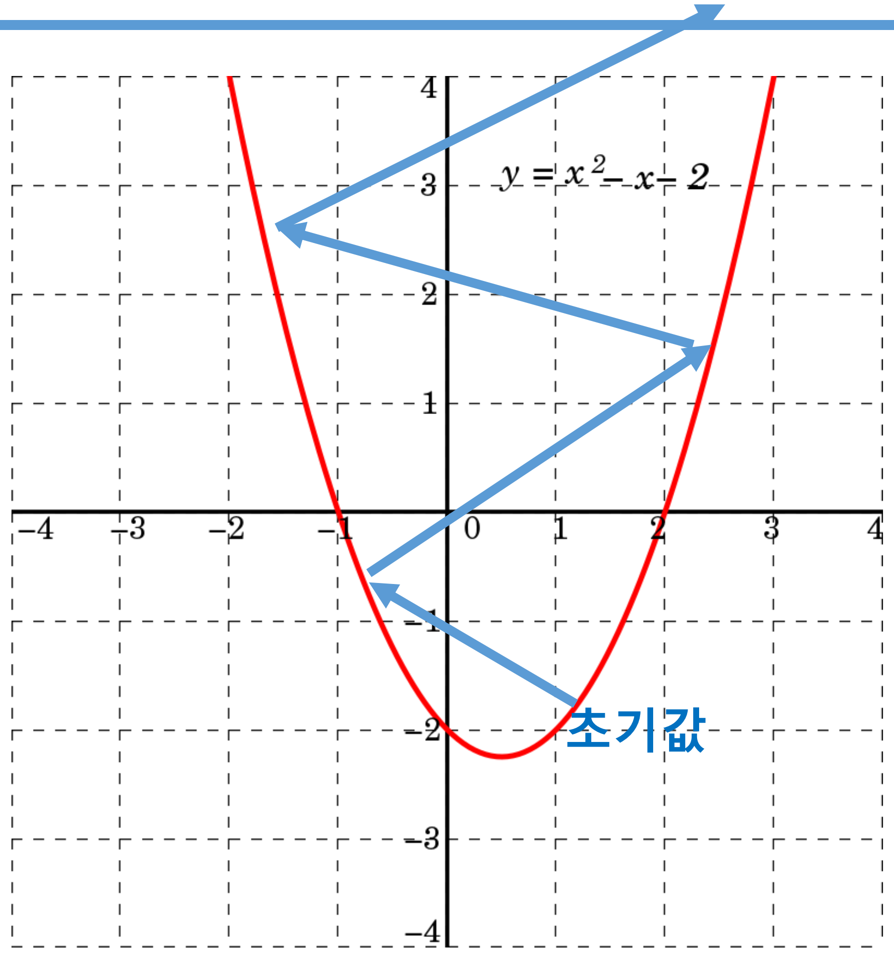
Input : Training Data ***D***: $(< x_1, y_1 >, < x_2, y_2 > \ldots, < x_n, y_n >)$,
Objective Function: f, learning rate: $\boldsymbol{\alpha}$, max_iterations: $\boldsymbol{i_{max}}$, epsilon : ε
Output : w, b.
{
Randomly initialize the elements of w and b.
l_{old} = 0
for 1 to $\boldsymbol{i_{max}}$:
    for each $< x_i, y_i > \in D$
        $\boldsymbol{w} \leftarrow \boldsymbol{w} - \boldsymbol{\alpha}\frac{\partial l_i}{\partial w}$
        $\boldsymbol{b} \leftarrow \boldsymbol{b} - \boldsymbol{\alpha}\frac{\partial l_i}{\partial b}$
    endfor
    calculate l = $\sum_{i=0}^{n}(y_i - (w \cdot x_i + b))^2$

    if |l - l_{old} | < ε :
        break;
    else
        l_{old} = l
endfor
Return $\boldsymbol{w}$, $\boldsymbol{b}$.
}

기타: 적절한 learning rate의 중요성



Learning rate α가 작은 경우:
Training이 매우 느리게 진행.



Learning rate α가 큰 경우:
발산 (Divergence)함.

Local minimum (maximum), Global minimum (maximum)

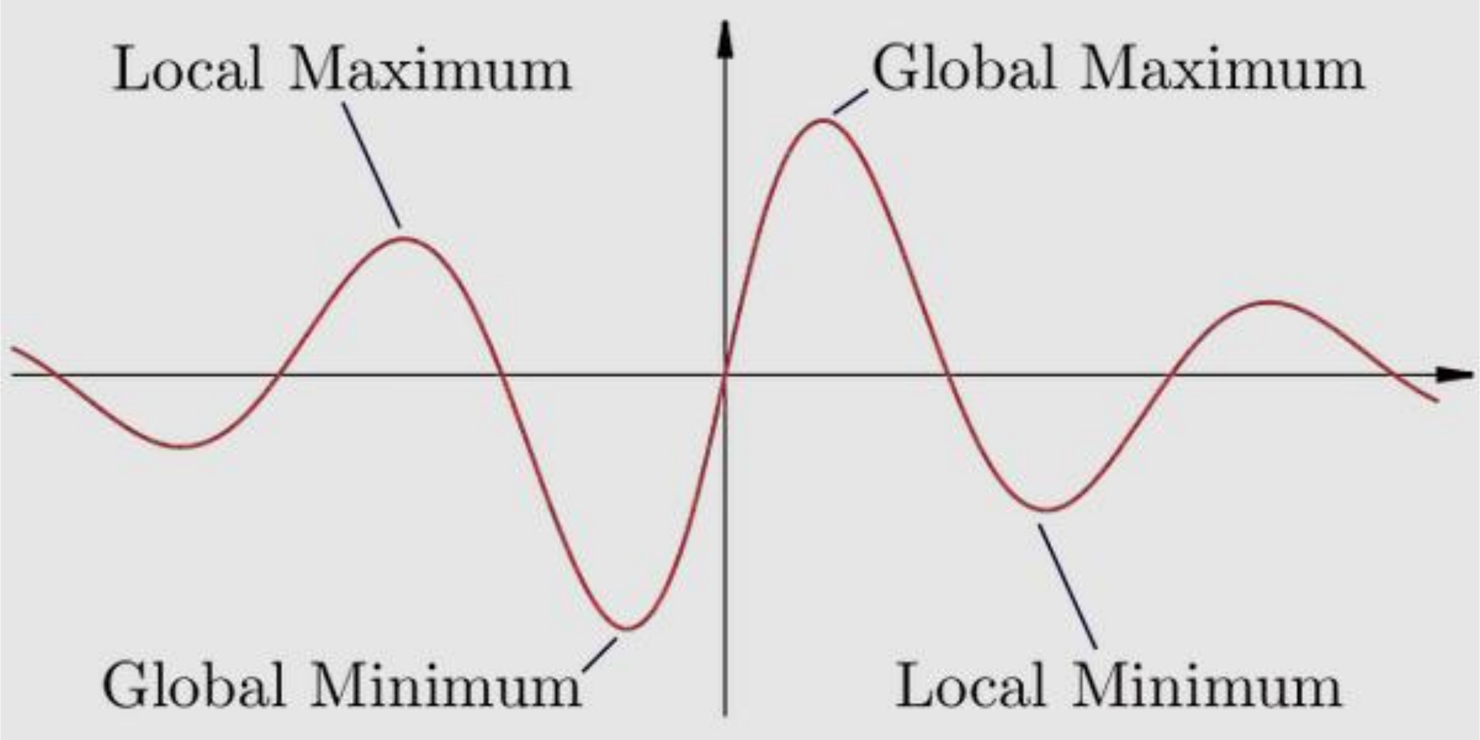


Image source : https://deepai.org/machine-learning-glossary-and-terms/stochastic-gradient-descent

Gradient descent for multivariate Linear Regression

Multivariate Linear Regression (일반 Linear regression)

- **Simple Linear Regression에서 feature x의 값이 scalar가 아닌, 벡터 (값이 여러 개 존재 (Multivariate))인 일반화된 버전을 생각해보자.**
  - **Outcome variable 을 하나 이상의 feature variable들의 선형 결합으로 예측하는 Regression.**

$$y = \sum_{i=1}^{k} w_i x_i + b$$

y : Outcome variable (예측하려는 변수).
$\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_k}$: Feature variables. (이하 줄여서 feature라 한다.)
$\boldsymbol{w_1}, \boldsymbol{w_2}, \ldots, \boldsymbol{w_k}$: Weights for each feature variable.
b : bias

Feature 3개인 Linear Regression 예)

$$y = 1.5 * x_1 - 0.1 * x_2 + 0.5\, x_3 + b$$

y : 복부 둘레
$\boldsymbol{x_1}$: 몸무게
$\boldsymbol{x_2}$: 운동량
$\boldsymbol{x_3}$: 식사량

Multivariate Linear Regression : Objective Function

- **다음과 같은 MSE를 Objective Function으로 사용한다.**

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(y_j - \left(\sum_{i=1}^{k} w \cdot x_i + b\right)\right)^2$$

x_i = <$x_{i1}, x_{i2}, \dots, x_{ik}$>,
w = <$w_1, w_2, \dots, w_k$>,

Gradient Descent Update:

$$w = w - a\frac{\partial \text{MSE}(w,b)}{\partial w}$$

$$\frac{\partial \text{MSE}(w,b)}{\partial w} = < \frac{\partial \text{MSE}(w,b)}{\partial w_1}, \frac{\partial \text{MSE}(w,b)}{\partial w_2}, \dots, \frac{\partial \text{MSE}(w,b)}{\partial w_k} >$$

연습문제

- $y_i = w \cdot x_i + b$
- $\text{MSE}(w) = \frac{1}{n}\sum_{i=1}^{n}\left(y_j - (\sum_{i=1}^{k} w \cdot x_i + b\ )\right)^2$
- 데이터가 $n = 1, x_1\ = < 1, 2 >, y_1 = -1$
- w의 현재값 $w^{(0)}$ 이 <0.5, 0.5>, $b = 1$, learning rate $\alpha = 0.1$의 경우
- Stochastic gradient decent를 활용하여 1회 update한 $w^{(1)}$값을 구하라.

Usage 1 : Prediction

Prediction by using linear regression.

- **사용 예 1. Prediction:**
  - Linear Regression을 학습하였다면, 학습한 모델을 활용할 수 있는 방법 중 하나로, Training data셋에 없던 새 데이터의 feature x에 대해, 학습한 Linear Regression 모델을 사용하여 outcome variable y를 예측할 수 있다.
  - 간단히 말하면 y 값을 알 수 없는 새 데이터에 대해 y를 예측할 수 있다.

Apply simple linear regression on real data by using scikit-learn

- This code is based on the code presented in
  - https://scikit-learn.org/stable/auto_examples/linear_model/plot_ols.html

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn import linear_model
from sklearn.metrics import mean_squared_error, r2_score
from DataLoader import DataLoader

def simple_linear_regression_example():
    # Load the height, weight dataset
    height_x, weight_y = DataLoader.load_height_weight("../data/weight-height.csv")

    num_data = len(height_x)
    train_ratio = 0.8
    num_train = int(num_data * train_ratio)
    num_test = num_data - num_train
```

Apply simple linear regression on real data by using scikit-learn (Cont'd)

```
# Split the data into training/testing sets
height_x_train = height_x[:-num_test].reshape(-1, 1)
height_x_test = height_x[num_train:].reshape(-1, 1)

weight_y_train = weight_y[:-num_test].reshape(-1, 1)
weight_y_test = weight_y[num_train:].reshape(-1, 1)

# Create linear regression object
regr = linear_model.LinearRegression()

# Train the model using the training sets
regr.fit(height_x_train, weight_y_train)

# Make predictions using the testing set
weight_y_pred = regr.predict(height_x_test)
```

Apply simple linear regression on real data by using scikit-learn (Cont'd)

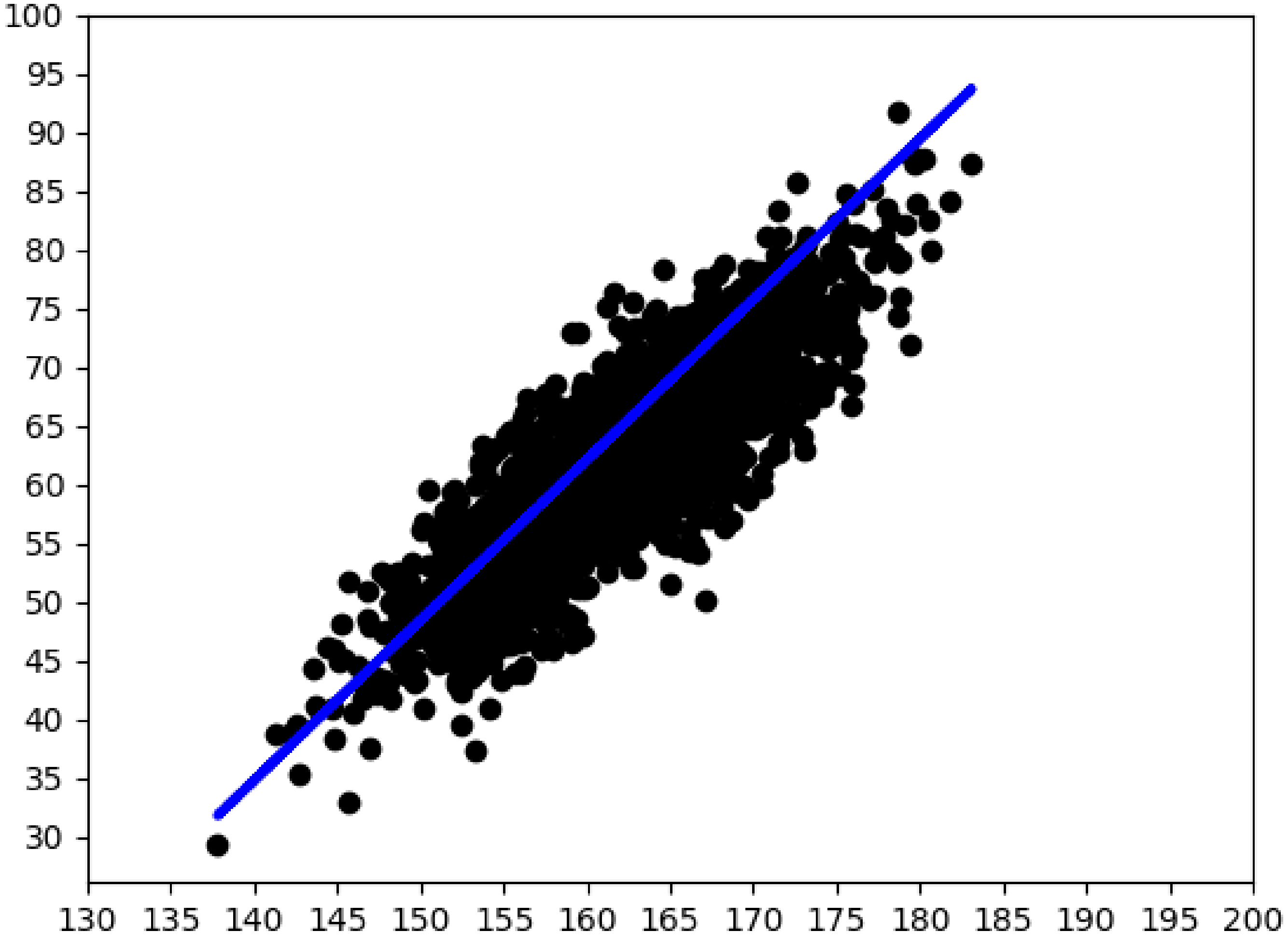
```
    # The coefficients
    print('Coefficients: \n', regr.coef_)
    # The mean squared error
    print('Mean squared error: %.2f'
          % mean_squared_error(weight_y_test, weight_y_pred))
    # The coefficient of determination: 1 is perfect prediction
    print('Coefficient of determination: %.2f'
          % r2_score(weight_y_test, weight_y_pred))
    # Plot outputs
    plt.scatter(height_x_test, weight_y_test,  color='black')
    plt.plot(height_x_test, weight_y_pred, color='blue', linewidth=3)

    plt.xticks(np.arange(130, 205, step=5))
    plt.yticks(np.arange(30, 105, step=5))

    plt.show()


if __name__ == '__main__':
    simple_linear_regression_example()
```

Apply simple linear regression on real data by using scikit-learn (Cont'd)



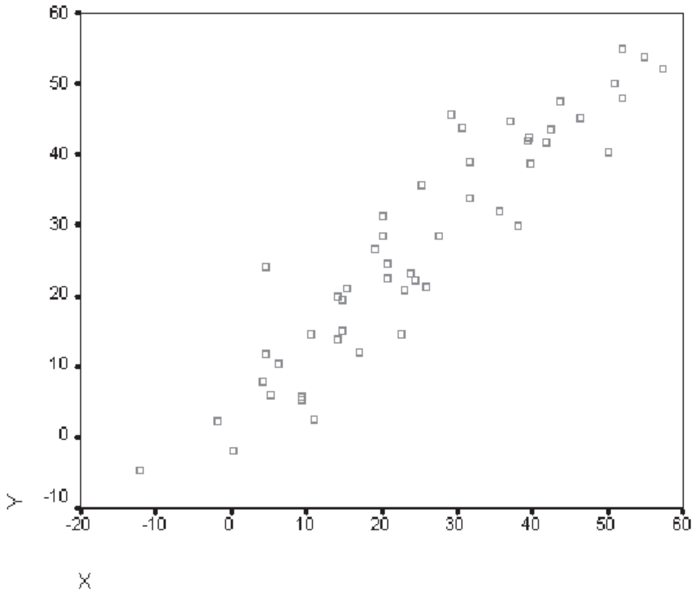
Prediction 결과를 해석할 때 주의점.

주의해야할 점

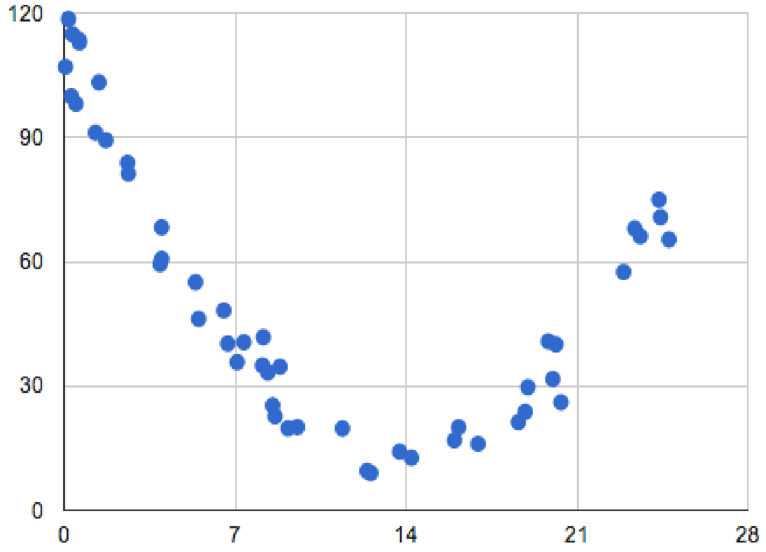
- 데이터가 어떻게 분포하고 있건 Linear Regression으로 model parameter(w, b)는 계산된다.
- 달리 말하면 어떤 데이터도 Linear Regression Model을 학습할 수 있고, 이를 사용해서 예측할 수 있다.
- 다만, Model을 사용한 예측이 유용한지 유용하지 않은지의 차이가 있을 뿐이다.

Linear Regression으로 예측이 가능한 데이터 분포 vs 가능하지 않은 분포

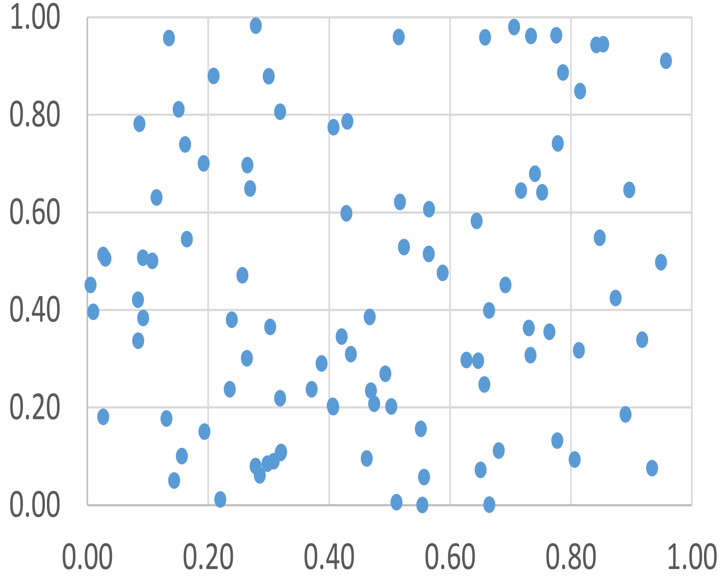
Linear Regression으로 예측 가능할지 그렇지 않을지의 Point는 무엇일까?



VS



VS



Randolph, Justus. (2007). Multidisciplinary methods in educational technology research and development.

https://www.statisticshowto.com/quadratic-regression/

Linear Regression으로 예측이 가능한 데이터 분포 vs 가능하지 않은 분포 (Cont'd)

Anscombe's quartet

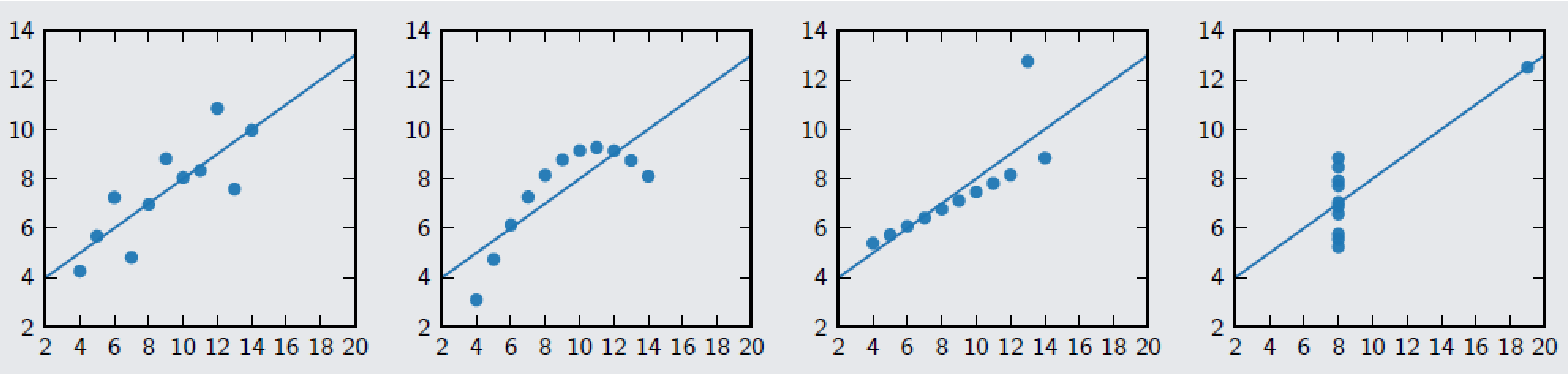


Image source : https://www.mit.edu/~6.s085/notes/lecture3.pdf

Anscombe's quartet은 통계 지표는 유사하지만 실제 데이터 분포는 매우 다른 4개의 데이터셋.
각 데이터셋은 11개의 (x, y) 좌표로 이루어진다.
1973년, 통계학자인 프란시스 앤스컴(Francis Anscombe)이 데이터 분석 전 1) 시각화의 중요성과 2) 특이치 및 주 영향 관측값(influential observation)의 영향을 보여주기 위해 만들었다. (Wikipedia 발췌)
* 선의 w = 0.5, b = 0.1

참고 : Anscombe's quartet data.

Anscombe's quartet

I		II		III		IV	
x	*y*	*x*	*y*	*x*	*y*	*x*	*y*
10.0	8.04	10.0	9.14	10.0	7.46	8.0	6.58
8.0	6.95	8.0	8.14	8.0	6.77	8.0	5.76
13.0	7.58	13.0	8.74	13.0	12.74	8.0	7.71
9.0	8.81	9.0	8.77	9.0	7.11	8.0	8.84
11.0	8.33	11.0	9.26	11.0	7.81	8.0	8.47
14.0	9.96	14.0	8.10	14.0	8.84	8.0	7.04
6.0	7.24	6.0	6.13	6.0	6.08	8.0	5.25
4.0	4.26	4.0	3.10	4.0	5.39	19.0	12.50
12.0	10.84	12.0	9.13	12.0	8.15	8.0	5.56
7.0	4.82	7.0	7.26	7.0	6.42	8.0	7.91
5.0	5.68	5.0	4.74	5.0	5.73	8.0	6.89

Model Evaluation :

- 학습한 Linear Regression 모델이 데이터의 경향을 제대로 학습하였는지 어떻게 판단할 수 있는가?
- 예측 결과를 신뢰할 수 있는지 판단할 수 있어야 한다.

- **방법 1 : Evaluation Data에 대해 MSE를 계산하여 예측을 믿을 수 있는지 판단한다.**
  - **MSE의 크기가 (사용자가 생각하는) 오차 범위내면 믿는다.**
  - **아니면? 다른 모델을 찾아야한다.**
- **방법 2 : Residual을 사용.**

Model Evaluation : r^2 (Cont'd)

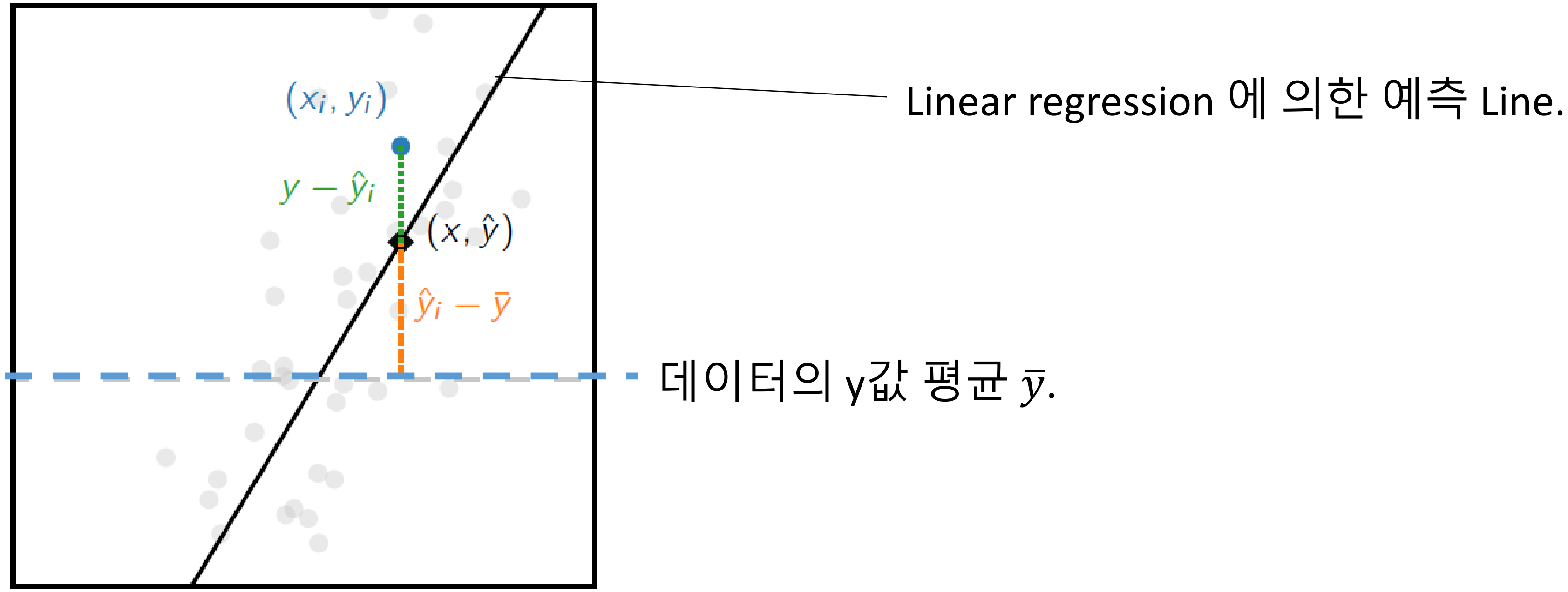


Figure 3.5: An illustration of the components contributing to the difference between the average y-value $\bar{y}$ and a particular point (x_i, y_i) (blue). Some of the difference, $\hat{y}_i - \bar{y}$, can be explained by the model (orange), and the remainder, $y_i - \hat{y}_i$, is known as the residual (green).

Image source : https://www.mit.edu/~6.s085/notes/lecture3.pdf

Model Evaluation : r^2

- 우리가 feature x와 output y의 그래프를 그렸을 때 x에 관계 없이 모든 y 값이 y의 평균값에 근접해 있다면 우리는 x와 y는 별 관련 없다 할 수 있다.

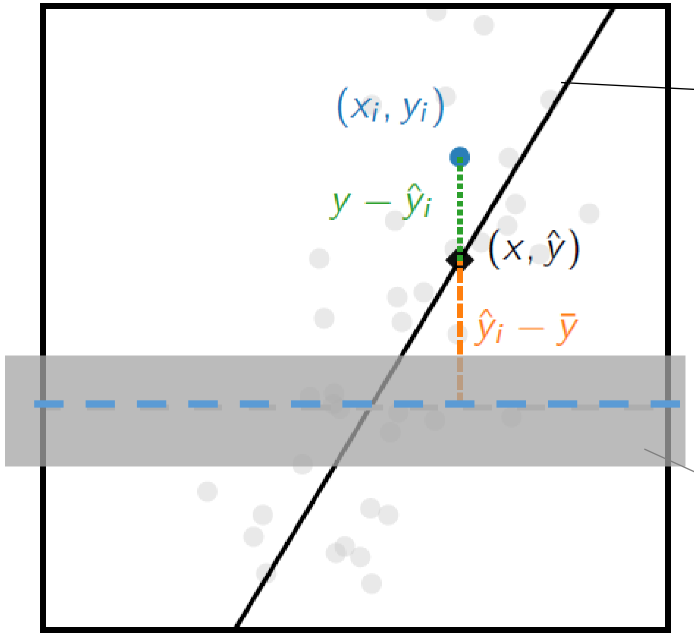
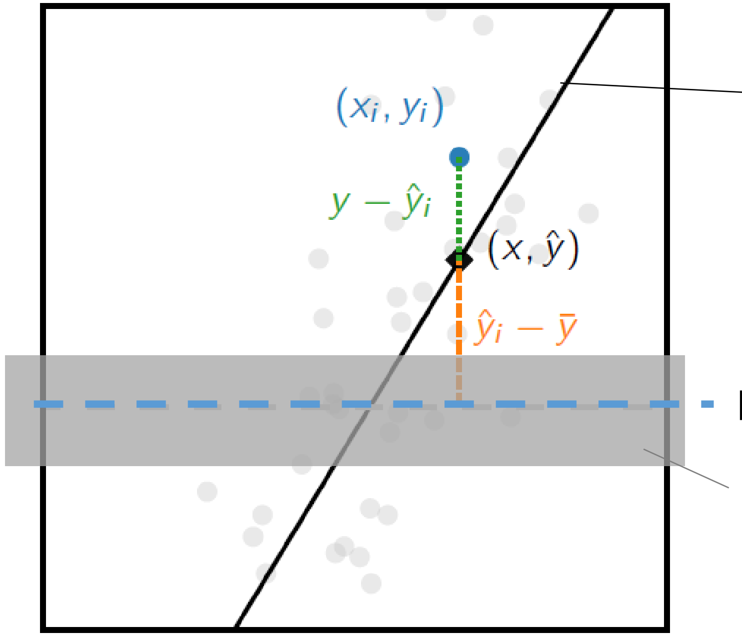


Figure 3.5: An illustration of the components contributing to the difference between the average y-value $\bar{y}$ and a particular point (x_i, y_i) (blue). Some of the difference, $\hat{y}_i - \bar{y}$, can be explained by the model (orange), and the remainder, $y_i - \hat{y}_i$, is known as the residual (green).

Image source : https://www.mit.edu/~6.s085/notes/lecture3.pdf

Model Evaluation : r^2 (Cont'd)



Linear regression 에 의한 예측 Line.

데이터의 y값 평균 $\bar{y}$.

데이터가 평균 주위에 늘어서 있다면?

- 따라서 어떤 데이터 i의 y값을 y_i 라 할 때 y_i 와 y값 평균 $\bar{y}$ 의 차이 $y_i - \bar{y}$ 를 계산한다고 생각해 보자.
- 위 논의에 근거하면, x와 y가 관련이 크려면 최소한 $y_i - \bar{y}$ 차이가 커야한다. ($|y_i - \bar{y}|$ 이 커야 한다.)

Model Evaluation : $\boldsymbol{r^2}$ (Cont'd)

- 그런데 우리가 관심있는 것은 Linear Regression model을 사용한 예측이 얼마나 잘되는가 이므로,

- $y_i - \bar{y}$ 계산 과정에 model을 사용하여 예측한 예측값 $\hat{y}_i$ 도 끼워 넣는다.
  - $\boldsymbol{y_i - \bar{y} = (\hat{y}_i - \bar{y}) + (y_i - \hat{y}_i)}$

- 위 식 오른쪽의 $(\hat{y}_i - \bar{y})$ **의 절대값이 클수록** Model의 y 예측값과 y 평균값의 차이가 크므로, x와 y의 관계가 있다면 이를 **모델에서 잘 예측**한다는 것이 된다.

- 위 식 오른쪽의 $(y_i - \hat{y}_i)$**의 절대값이 클수록** Model의 y 예측값과 실제 y값의 차이가 크므로 **모델이 예측을 잘 못한다**는 것이 된다.

Model Evaluation : $\boldsymbol{r^2}$ (Cont'd)

- $y_i - \bar{y} = (\hat{y}_i - \bar{y}) + (y_i - \hat{y}_i)$ 을 차이의 절대값을 고려할 수 있도록 변경하고 모든 데이터에 대해 일반화하면, 아래와 같은 식이 된다.
  - $\sum_{i=1}^{n}(y_i - \bar{y})^2 = \sum_{i=1}^{n}(\hat{y}_i - \bar{y})^2 + \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$
  - 유도 과정은 생략. 관심 있으면 https://rpubs.com/beane/n3_1b 를 참조하라.

- $\sum_{i=1}^{n}(y_i - \bar{y})^2 = \sum_{i=1}^{n}(\hat{y}_i - \bar{y})^2 + \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$

- 에서 $\sum_{i=1}^{n}(\hat{y}_i - \bar{y})^2$ 클수록 Model이 잘 예측하고, $\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$ 이 작을수록Model이 잘못 예측한다.

- 그런데 아무리 Model이 잘 예측해도 $\sum_{i=1}^{n}(y_i - \bar{y})^2$를 넘을 수는 없으므로, $\sum_{i=1}^{n}(y_i - \bar{y})^2$ 로 위 식의 좌/우변을 나누어 모델 예측 성능을 Normalization할 수 있다.

Model Evaluation : r^2 (Cont'd)

- $1 = \boxed{\frac{\sum_{i=1}^{n}(\widehat{y_i}-\bar{y})^2}{\sum_{i=1}^{n}(y_i-\bar{y})^2}} + \boxed{\frac{\sum_{i=1}^{n}(y_i-\widehat{y_i})^2}{\sum_{i=1}^{n}(y_i-\bar{y})^2}}$

모델의 예측력

실제 데이터에 대한 예측 오차

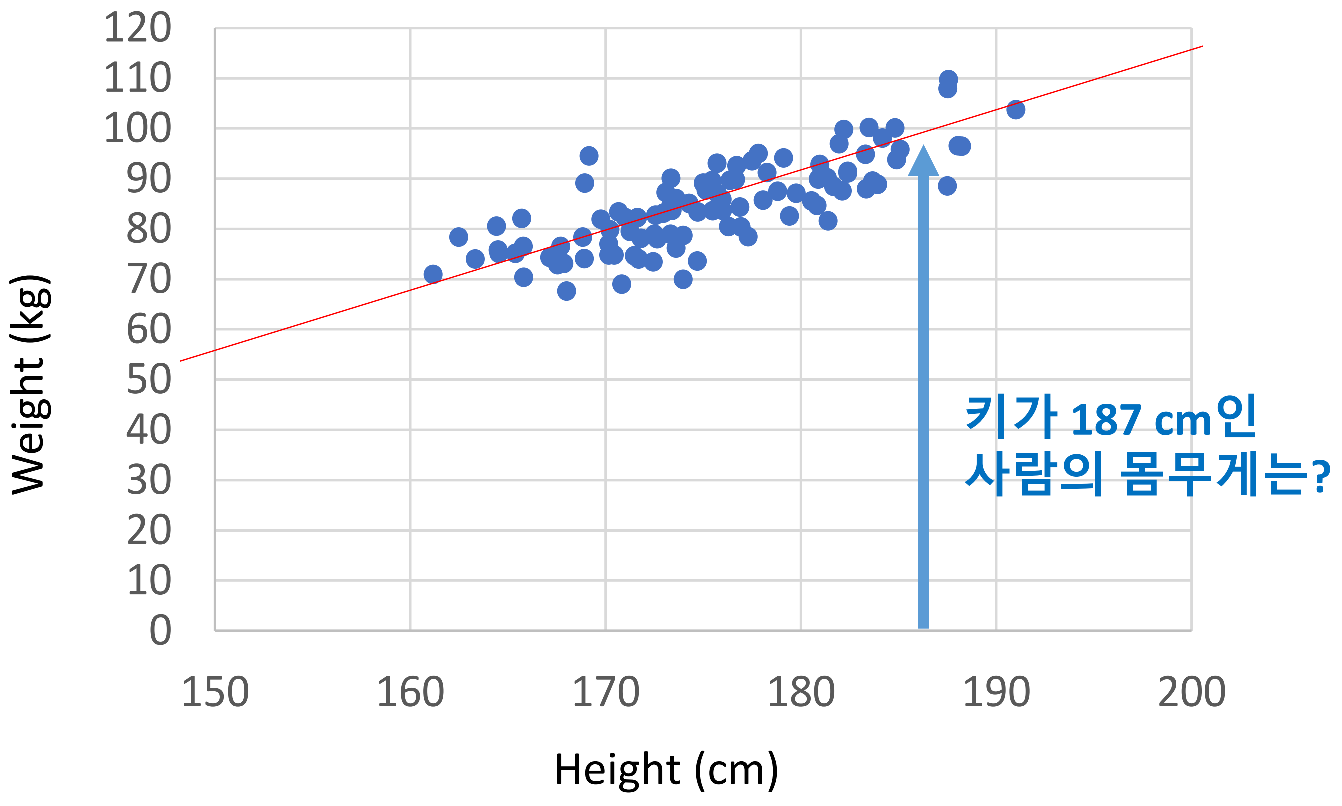
$r^2 = \frac{\sum_{i=1}^{n}(\widehat{y_i}-\bar{y})^2}{\sum_{i=1}^{n}(y_i-\bar{y})^2}$ 이라 하면, $\frac{\sum_{i=1}^{n}(y_i-\widehat{y_i})^2}{\sum_{i=1}^{n}(y_i-\bar{y})^2}$ 는 $\mathbf{1 - r^2}$ 로 표현됨.

0 <= r^2 <=1 , r^2 이 1에 근접할 수록 모델 표현력이 좋다.

Training data 뿐만 아니라 Evaluation data에 대해서도 r^2 를 계산해서 판단해야 한다.

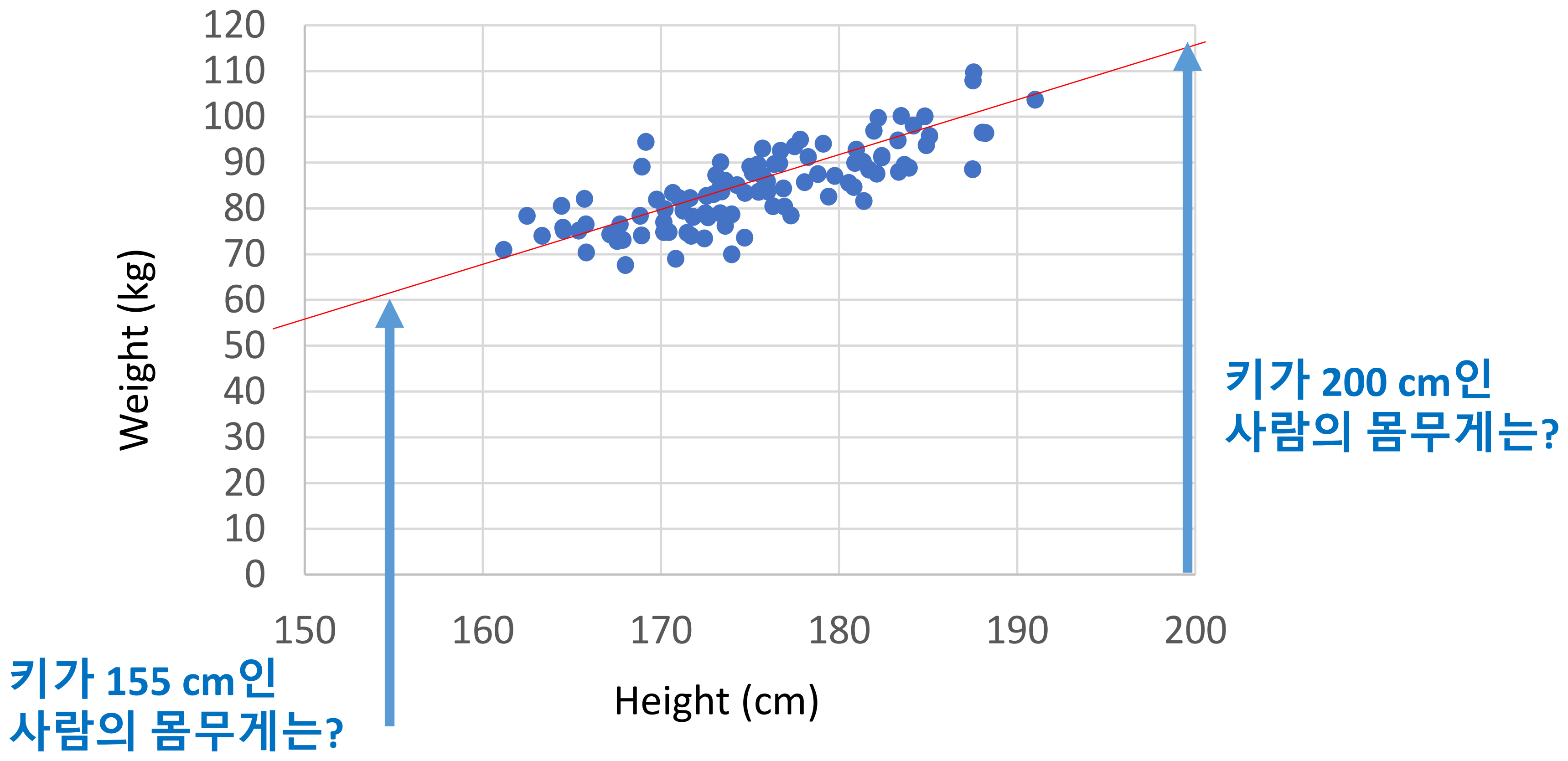
Regression을 사용한 예측에 있어서의 주의점:
Interpolation Vs. Extrapolation : Interpolation

Interpolation (보간 혹은 내삽) : 알고 있는 두 개 이상의 데이터 사이의 데이터를 추측하는 것.
직선에서 추측하려는 데이터의 좌측과 우측의 데이터를 **모두 알고 있는 경우.**

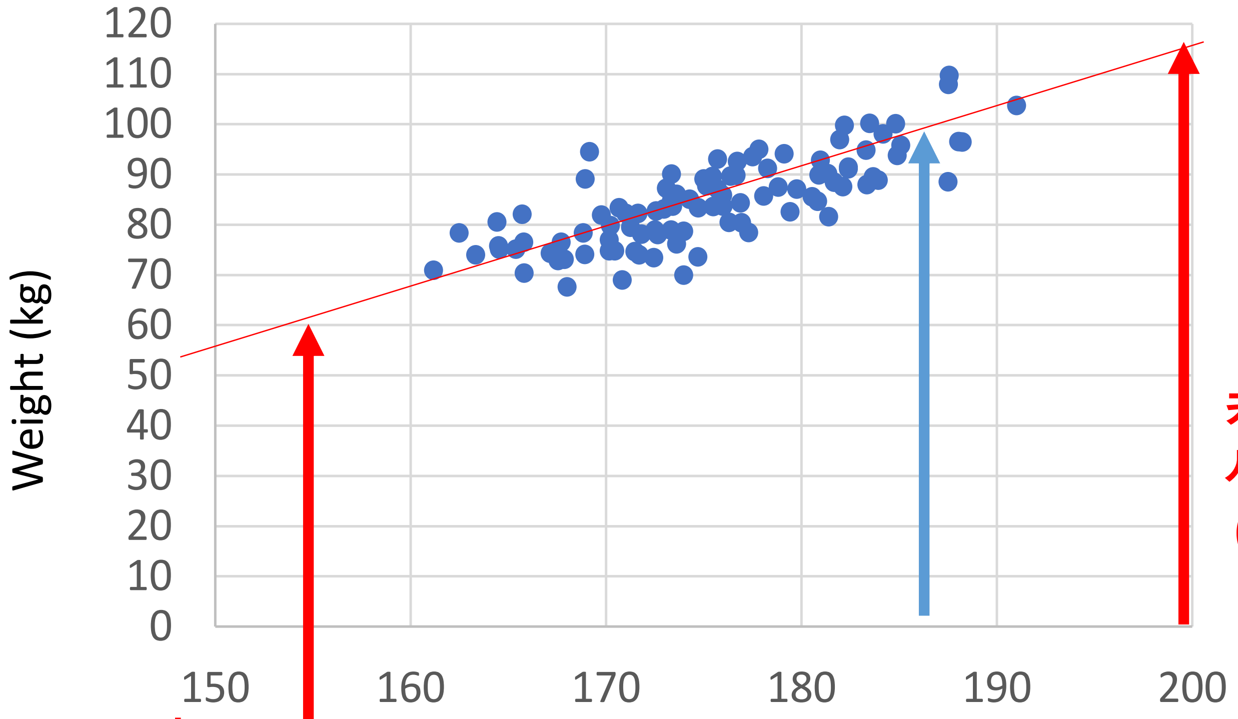


Regression을 사용한 예측에 있어서의 주의점:
Interpolation Vs. Extrapolation : Extrapolation

Extrapolation (외삽) : 알고 있는 데이터 범위 밖의 데이터를 추측하는 것.
직선에서 추측하려는 데이터의 좌측 혹은 우측의 데이터를 **한쪽만 알고 있는 경우**.



Regression을 사용한 예측에 있어서의 주의점: Exploration 할 경우는 예측 결과가 실제와 다를 확률이 더 높다.



키가 200 cm인
사람의 몸무게는?
(105 kg 전후?)

키가 155 cm인
사람의 몸무게는?
(60 kg 전후?)

Height (cm)

키가 187 cm인
사람의 몸무게는?

주위 데이터를 보면
어느정도 정답이 예측이
된다.

귀납추론의 문제점

러셀의 닭

The man who has fed the chicken every day throughout its life at last wrings its neck instead.

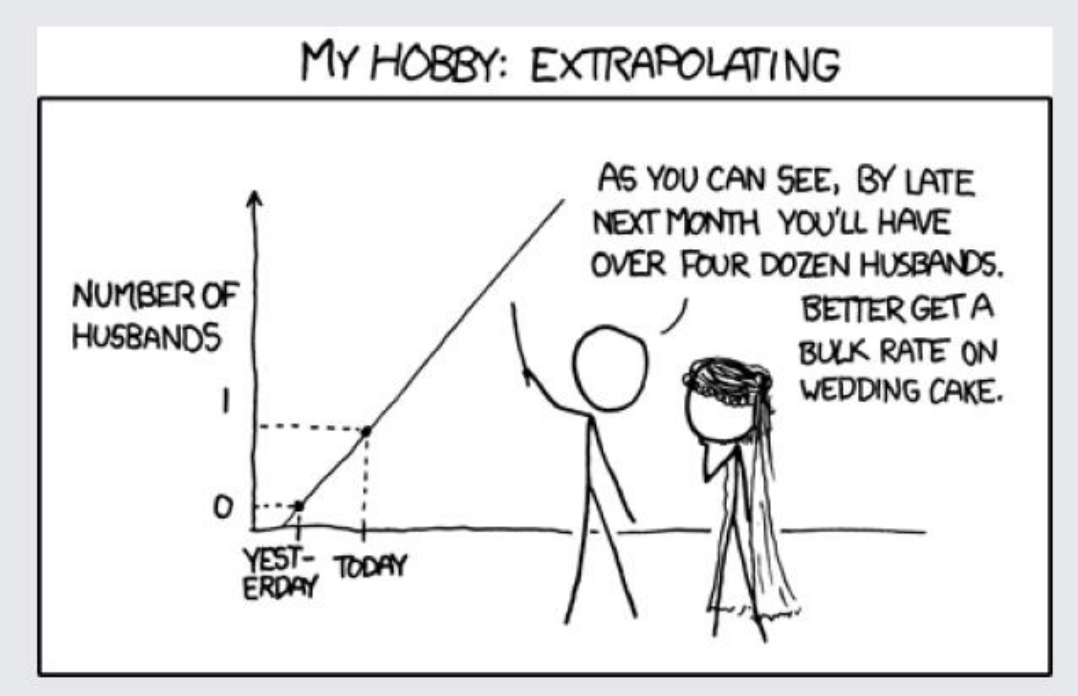


Image source : https://www.mit.edu/~6.s085/notes/lecture3.pdf

Linear Regression

Training : Regularization

Overfitting

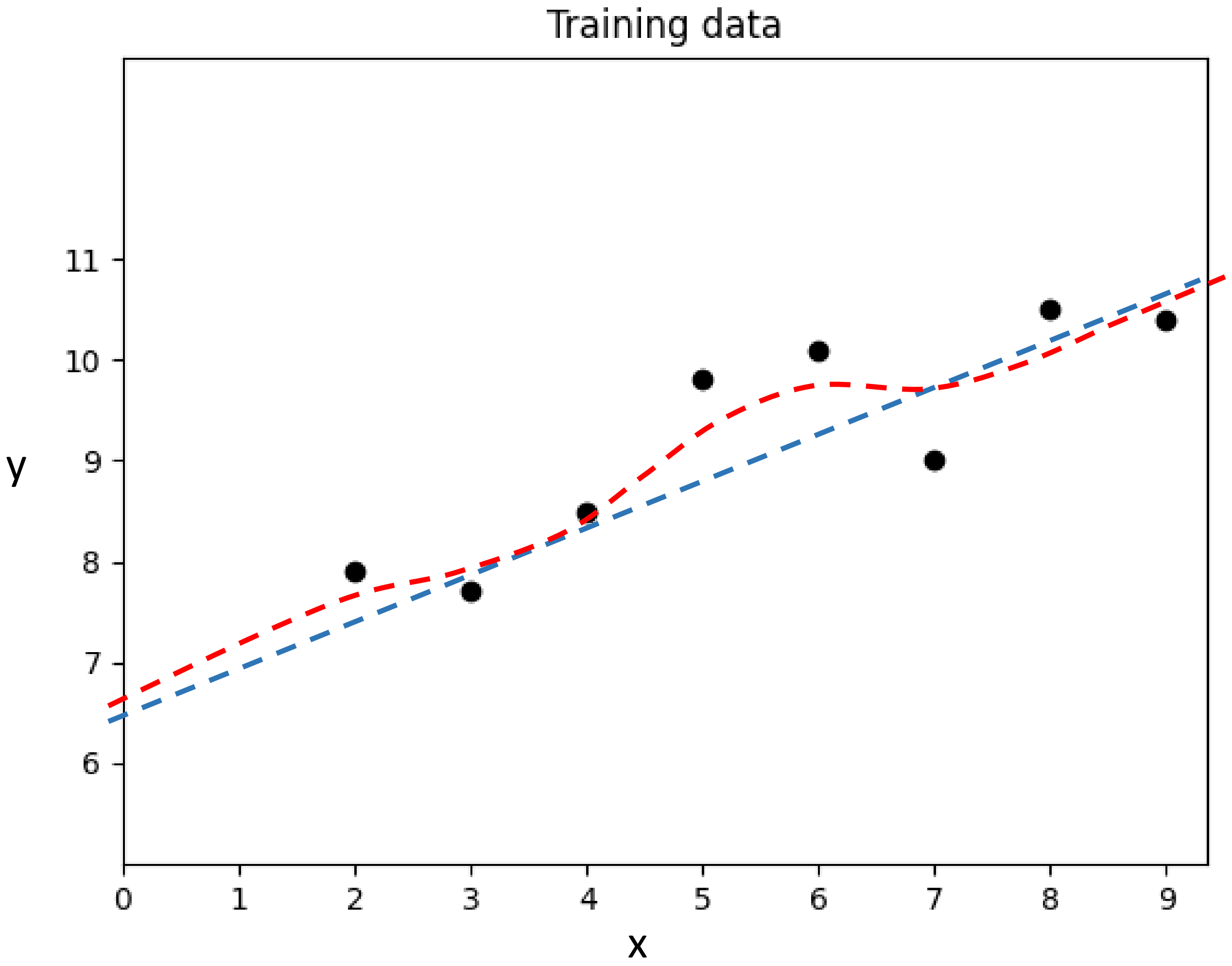
- **Overfitting**
  - 모델이 Training data를 필요이상으로 정확하게 예측하도록 학습되어 모델의 Generalization 능력이 저하되는 것.
    - Training data에 대해서는 완벽한 예측성능을 보이나 Test data에 대해서는 예측 성능이 낮다.
  - 일반적으로
  - **(1) Training data의 규모가 작거**나,
  - **(2) 모델의 표현력이 data의 분포 경향에 비해 뛰어난 경우**에 일어난다.

Overfitting : Model의 표현력 (Representative Power)

- **Model's Representative Power:**
  - **Model이 표현할 수 있는 복잡도.**
    - **Model이 기반으로 하는 이론에 따라 표현할 수 있는 최대 복잡도가 정해진다.**
      - 예 1) Linear Regression : 선형 모델, (제한된) 비선형 모델 표현
      - 예 2) Multi-Layer Perceptron : 거의 대부분의 비선형 모델 표현
    - **같은 Model에서는 일반적으로 model의 parameter 수가 늘어날 수록 표현력이 증가한다.**
      - 예) Simple Linear Regression VS. Multi-variate Linear Regression
      - 문제) Linear Regression의 model parameter는?

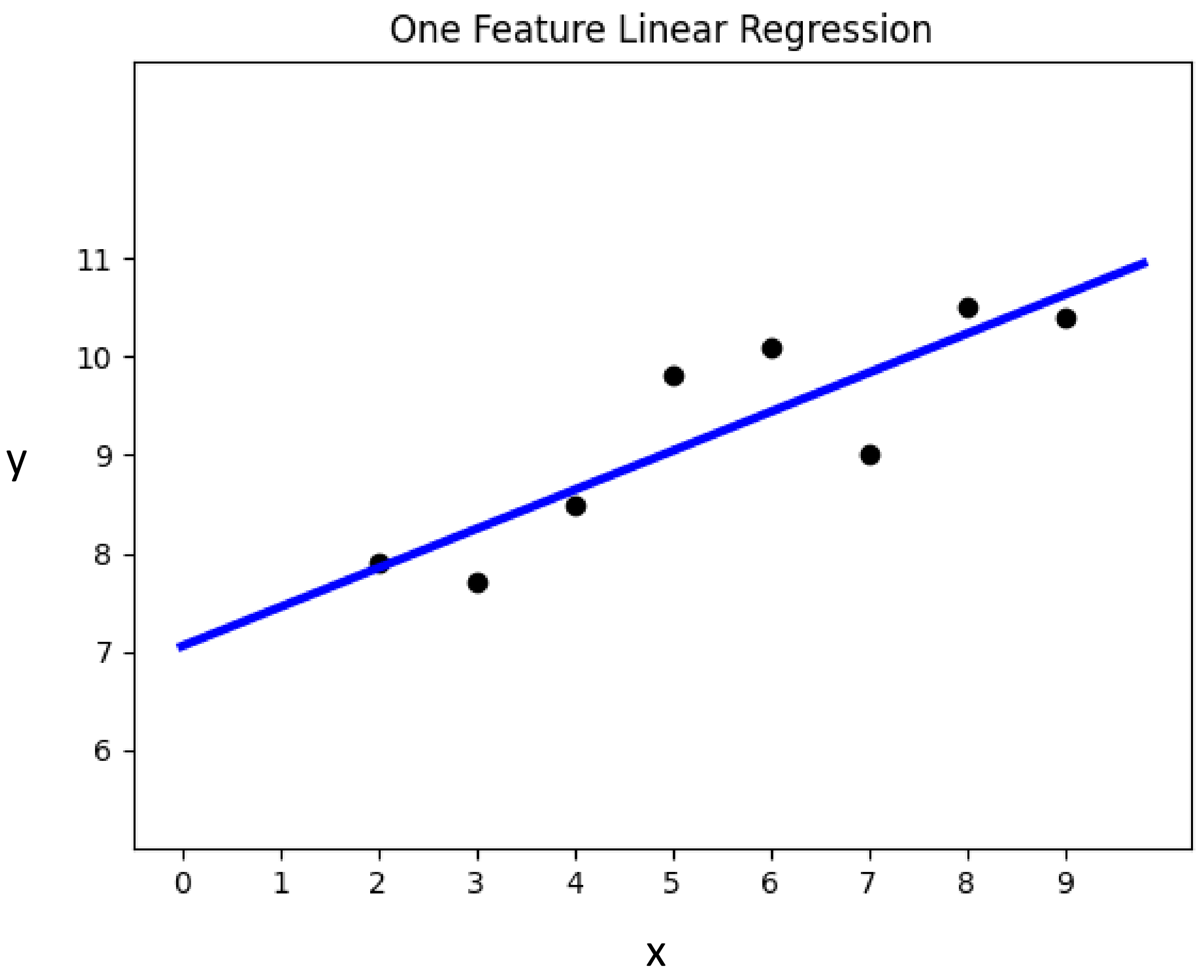
Overfitting 예

Training data로 부터 예측되는 데이터의 분포는?



Overfitting 예 (Cont'd)

Simple Linear Regression : y = wx + b



Overfitting – Example

```
"""
================================================================
Linear Regression Overfit Example
================================================================
"""
import matplotlib.pyplot as plt
from sklearn import linear_model
from sklearn.metrics import mean_squared_error, r2_score
from DataLoader import DataLoader
import numpy as np


def print_coef(coef):
    coef_array = coef.flatten()
    for val in coef_array:
        print("%.4f" % val, end="\t")
    print("")
```

```
def make_6_order_feature_vecs(x):
    x1 = x
    x2 = x * x
    x3 = x2 * x
    x4 = x3 * x
    x5 = x4 * x
    x6 = x5 * x

    x1 = x1.reshape(-1, 1)
    x2 = x2.reshape(-1, 1)
    x3 = x3.reshape(-1, 1)
    x4 = x4.reshape(-1, 1)
    x5 = x5.reshape(-1, 1)
    x6 = x6.reshape(-1, 1)
    features = np.concatenate((x6, x5, x4, x3, x2, x1), axis=1)
    return features
```

```
def linear_regression_overfit_example():

    # Load the wine features, wine quality dataset
    x, y = DataLoader.load_overfit_example()

    x_train = x.reshape(-1, 1)
    y_train = y.reshape(-1, 1)

    regr = linear_model.LinearRegression()
    # Train the model using the training sets
    regr.fit(x_train, y_train)

    # Make predictions using the testing set
    x_min = 0
    x_max = 10
    step = 0.2
    test_x = np.arange(x_min, x_max, step).reshape(-1, 1)
    y_pred = regr.predict(test_x)
```

```
# Plot outputs
    plt.scatter(x_train, y_train, color='black')
    plt.xticks(np.arange(0, 10, step=1))
    plt.yticks(np.arange(6, 12, step=1))
    plt.ylim([5, 13])
    plt.title('Training data')
    plt.show()

    # Plot outputs
    plt.scatter(x_train, y_train, color='black')
    plt.plot(test_x, y_pred, color='blue', linewidth=3)
    plt.xticks(np.arange(0, 10, step=1))
    plt.yticks(np.arange(6, 12, step=1))
    plt.ylim([5, 13])
    plt.title('One Feature Linear Regression')
    plt.show()
```

```
features = make_6_order_feature_vecs(x)
features = features.reshape(-1, 6)
regr_multi = linear_model.LinearRegression()

regr_multi.fit(features, y_train)

# Make predictions using the testing set
x_min = 0
x_max = 10
step = 0.2
test_x = np.arange(x_min, x_max, step).reshape(-1, 1)
test_features = make_6_order_feature_vecs(test_x)

y_pred = regr_multi.predict(test_features)
print_coef(regr_multi.coef_)
# Plot outputs
plt.scatter(x_train, y_train, color='black')
plt.plot(test_x, y_pred, color='blue', linewidth=3)
plt.xticks(np.arange(0, 10, step=1))
plt.yticks(np.arange(6, 12, step=1))
plt.ylim([5, 13])
plt.title('Multi-Feature (6 order x ) with Regularization W = 0')
plt.show()
```

```
# L2 Regularization W = 0.1
ridge = linear_model.Ridge(alpha=0.1)
ridge.fit(features, y_train)
y_pred = ridge.predict(test_features)
print_coef(ridge.coef_)
# Plot outputs
plt.scatter(x_train, y_train, color='black')
plt.plot(test_x, y_pred, color='blue', linewidth=3)
plt.xticks(np.arange(0, 10, step=1))
plt.yticks(np.arange(6, 12, step=1))
plt.ylim([5, 13])
plt.title('Multi-Feature (6 order x ) with L2 Reg. Lambda = 0.1')
plt.show()

# L2 Regularization W = 1
ridge = linear_model.Ridge(alpha=1.0)
ridge.fit(features, y_train)
y_pred = ridge.predict(test_features)
print_coef(ridge.coef_)
# Plot outputs
plt.scatter(x_train, y_train, color='black')
plt.plot(test_x, y_pred, color='blue', linewidth=3)
plt.xticks(np.arange(0, 10, step=1))
plt.yticks(np.arange(6, 12, step=1))
plt.ylim([5, 13])
plt.title('Multi-Feature (6 order x ) with L2 Reg. Lambda = 1')
plt.show()
```

```python
# L1 Regularization W = 0.1
    lasso = linear_model.Lasso(alpha=0.1)
    lasso.fit(features, y_train)
    y_pred = lasso.predict(test_features)
    print_coef(lasso.coef_)
    # Plot outputs
    plt.scatter(x_train, y_train, color='black')
    plt.plot(test_x, y_pred, color='blue', linewidth=3)
    plt.xticks(np.arange(0, 10, step=1))
    plt.yticks(np.arange(6, 12, step=1))
    plt.ylim([5, 13])
    plt.title('Multi-Feature (6 order x ) with L1 Reg. Lambda = 0.1')
    plt.show()
```

```python
# L1 Regularization W = 1
    lasso = linear_model.Lasso(alpha=1.0)
    lasso.fit(features, y_train)
    y_pred = lasso.predict(test_features)
    print_coef(lasso.coef_)
    # Plot outputs
    plt.scatter(x_train, y_train, color='black')
    plt.plot(test_x, y_pred, color='blue', linewidth=3)
    plt.xticks(np.arange(0, 10, step=1))
    plt.yticks(np.arange(6, 12, step=1))
    plt.ylim([5, 13])
    plt.title('Multi-Feature (6 order x ) with L1 Reg. Lambda = 1')
    plt.show()

if __name__ == '__main__':
    linear_regression_overfit_example()
```

Overfitting 예 (Cont'd) – Representative Power가 증가된 Linear Regression

- **6 feature Multi-variable linear regression :**
- **다음 Linear Regression에서**
  - $\boldsymbol{y} = \boldsymbol{w_6 x_6} + \boldsymbol{w_5 x_5} + \boldsymbol{w_4 x_4} + \boldsymbol{w_3 x_3} + \boldsymbol{w_2 x_2} + \boldsymbol{w_1 x_1} + \boldsymbol{b}$
- 입력 vector $\vec{x} =< x_6, x_5, x_4, x_3, x_2, x_1 >$ 이 다음과 같다고 하자.
  - $x_6 = x^6$
  - $x_5 = x^5$
  - $x_4 = x^4$
  - $x_3 = x^3$
  - $x_2 = x^2$
  - $x_1 = x$

Overfitting 예 (Cont'd) – Representative Power가 증가된 Linear Regression

6-order Linear Regression (Non-Linear Function):

$$\boldsymbol{y} = \boldsymbol{w_6}x^6 + \boldsymbol{w_5}x^5 + \boldsymbol{w_4}x^4 + \boldsymbol{w_3}x^3 + \boldsymbol{w_2}x^2 + \boldsymbol{w_1}x + \boldsymbol{b}$$

Training Data에 대해서는 거의 완벽히 y를 예측할 수 있다.

그러나 이 Model을 사용하여 예측하면?

왜 이런 일이? Model이 필요 이상으로 머리가 좋기 때문

- 머리가 좋다
  - = Model Representative Power가 뛰어나다.
  - = Model Parameter가 필요 이상으로 많다.

- Model Parameter에 하나하나의 Training data의 특징을 외울 수 있다.
  - 주어진 말(training data)을 단순하게 설명하려면, 말의 주요 특징만을 추출하여 설명한다.
  - **=> Generalization**
    - 목이 길고, 굽 있는 다리가 네 개 이고, 목 위에 갈기가 있고, 뒤에 꼬리가 있다. 배는 볼록하다.
    - 대부분의 말 (training data에 나타나지 않은 data)은 위의 특징을 갖고 있다.

  - 주어진 말(training data) 하나하나를 외울 수 있다면, 해당 말의 모든 특징을 세세하게 설명할 수 있다.
  - **=> Memorization (Overfitting)**
    - 목이 길고, 굽 있는 다리가 네 개 이고, 목 위에 갈기가 있고, 뒤에 꼬리가 있다. 배는 볼록한데다가 왼쪽 귀에 점이 두 개 있고, 배에 줄무늬가 있으며, 이가 하나 없고,...
    - 다른 말 (training data에 나타나지 않은 data) 은 위의 특징을 가지지 않는 경우가 많다.

Occam's Razor : Simpler one is better

Image source: https://examples.yourdictionary.com/examples-of-occam-s-razor.html

Two scientific theories which explain the same situation equally well, **the simpler one is preferred**.

Overfitting을 피하기 위해서는 Training set에 대한 Error가 비슷한 경우**(explain the same situation equally well)**, parameter 수가 더 적은 Model이 더 좋다. **(the simpler one is preferred)**

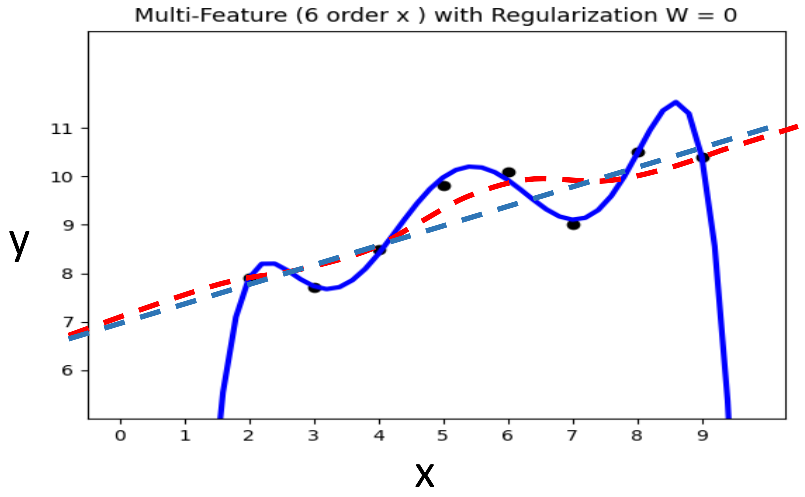
어려운 점: Model Parameter 수를 줄이는 것은 때때로 쉽지 않다.

- 예) 복부 둘레를 예측하는데 있어서 <몸무게, 운동량, 식사량, 키> 를 Feature로 사용하는데, 이 중 두 개를 줄여야 한다고 하자. 무엇을 줄일까?
  - 남길 중요한 Feature 2개를 알기 어렵다.
    - 잘 모르니까 Machine Learning으로 학습한다.
  - 다 복부 둘레와 연관이 있어 보인다.
    - 모든 feature를 사용하고 싶다.

- Feature 수를 줄이지 않고 Overfitting을 방지할 수 있는 방법이 없을까?

Regularization

- Model Parameter의 값이 극단적인 분포를 같지 않도록 하여 Overfitting을 방지하는 방법.
  - 일반적으로 Overfitting이 일어날 경우 model parameter가 극단적인 값을 가지는 경향이 있다.



6-order Linear Regression (Non-Linear Function):

$$\boldsymbol{y} = \boldsymbol{w_6}x^6 + \boldsymbol{w_5}x^5 + \boldsymbol{w_4}x^4 + \boldsymbol{w_3}x^3 + \boldsymbol{w_2}x^2 + \boldsymbol{w_1}x + \boldsymbol{b}$$

w1	w2	w3	w4	w5	w6
-0.0102	0.326	-4.1293	26.3653	-88.8855	149.7337

- 주로 아래 두 Regularization 방법이 주로 사용된다.
  - **L2 Regularization (Ridge Regularization),**
  - **L1 Regularization (Lasso Regularization)**

Regularization

- Objective Function 에 Regularization 식을 추가한다.
  - Objective Function : $l_{old} = \sum_{i=1}^{n}\left(y_j - (w \cdot x_i + b)\right)^2$
  - New Objective Function: $l_{new} = \sum_{i=1}^{n}\left(y_j - (w \cdot x_i + b)\right)^2 + \boldsymbol{\lambda} \cdot \boldsymbol{R}$
    - λ : Regularization Weight
    - R : Regularization Equation (Sigma 밖에 있음에 유의)

L2 Regularization

- L2 Regularization (Ridge Regularization)
  - Model Parameter 를 θ 라 표기하면, L2 Regularization은 Model이 $\|\theta\|_2^2$ 를 최소화 하는 θ를 학습하도록 한다.
  - $\|\theta\|_2^2$: θ 의 Frobenius norm 의 제곱 : θ가 vector라면 vector의 크기의 제곱
  - Linear regression의 식이 $\hat{y}_i = w \cdot x_i + b$ 이고, 우리가 학습해야 할 variable이 w, b 이므로,
  - Linear regression의 L2 Norm은 $|w|^2 + b^2 = w \cdot w + b^2 = w^2 + b^2$ 이 된다.
  - 따라서,
    - New Objective Function: $l_{new} = \sum_{i=1}^{n}(y_i - (w \cdot x_i + b))^2 + \frac{\lambda}{2} \cdot (w^2 + b^2)$
      - $\frac{\lambda}{2}$ 의 2는 SGD를 쉽게 하기 위한 Trick. 이론상으로는 λ만 표기해도 됨.

Stochastic Gradient Descent (SGD) with L2 Regularization

주어진 $< x_i, y_i >$ **에 대해 SE (**l_i**)를** w **로 미분할 경우.**
아래 식에서 vector 사이의 곱은 dot product를 뜻한다.)

$$
\begin{aligned}
l_i &= (y_i - (w \cdot x_i + b))^2 + \frac{\boldsymbol{\lambda}}{\mathbf{2}} \cdot (w^2 + b^2) \\
&= y_i^2 \ - 2y_i \cdot (w \cdot x_i + b) + (w \cdot x_i + b)^2 + \frac{\boldsymbol{\lambda}}{\mathbf{2}} \cdot (w^2 + b^2) \\
&= y_i^2 \ - 2y_i \cdot w \cdot x_i - 2y_i b + w^2 \cdot x_i^2 + 2b \cdot w \cdot x_i + b^2 + \frac{\boldsymbol{\lambda}}{\mathbf{2}} \cdot w^2 + \frac{\boldsymbol{\lambda}}{\mathbf{2}} \cdot \boldsymbol{b^2} \\
&= (x_i^2 + \frac{\boldsymbol{\lambda}}{\mathbf{2}}) \cdot w^2 + (2b \cdot x_i - 2y_i \cdot x_i) \cdot w \cdot + (1 + \frac{\boldsymbol{\lambda}}{\mathbf{2}}) b^2 - 2y_i b + y_i^2 \quad (w \text{ 에 대해 정리})
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial l_i}{\partial w} &= 2 \cdot (x_i^2 + \frac{\boldsymbol{\lambda}}{\mathbf{2}}) \cdot w - 2y_i \cdot x_i + 2b \cdot x_i \\
&= -2 \cdot x_i \cdot (y_i - w \cdot x_i - b) + \boldsymbol{\lambda} \cdot w \\
&= -2 \cdot x_i \cdot (y_i - (w \cdot x_i + b)) + \boldsymbol{\lambda} \cdot w \\
&= -2 \cdot x_i \cdot (y_i - \hat{y}_i) + \boldsymbol{\lambda} \cdot w
\end{aligned}
$$

Stochastic Gradient Descent (SGD) with L2 Regularization (Cont'd)

주어진 $< x_i, y_i >$ **에 대해 SE** (l_i)**를** w **으로 미분할 경우.**
아래 식에서 vector 사이의 곱은 dot product를 뜻한다.)

$$l_i = (y_i - (w \cdot x_i + b))^2 + \frac{\boldsymbol{\lambda}}{\mathbf{2}} \cdot (w^2 + b^2)$$

$$= y_i^2 - 2y_i \cdot (w \cdot x_i + b) + (w \cdot x_i + b)^2 + \frac{\boldsymbol{\lambda}}{\mathbf{2}} \cdot (w^2 + b^2)$$

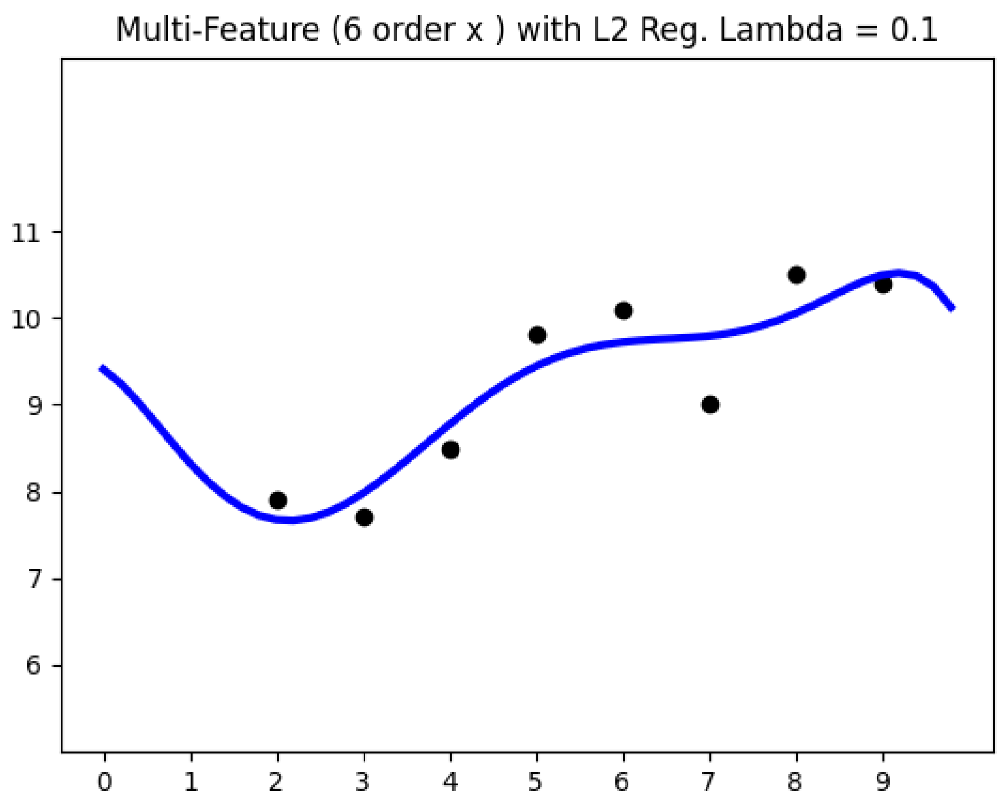
$$= y_i^2 - 2y_i \cdot w \cdot x_i - 2y_i b + w^2 \cdot x_i^2 + 2b \cdot w \cdot x_i + b^2 + \frac{\boldsymbol{\lambda}}{\mathbf{2}} \cdot w^2 + \frac{\boldsymbol{\lambda}}{\mathbf{2}} \cdot \boldsymbol{b^2}$$

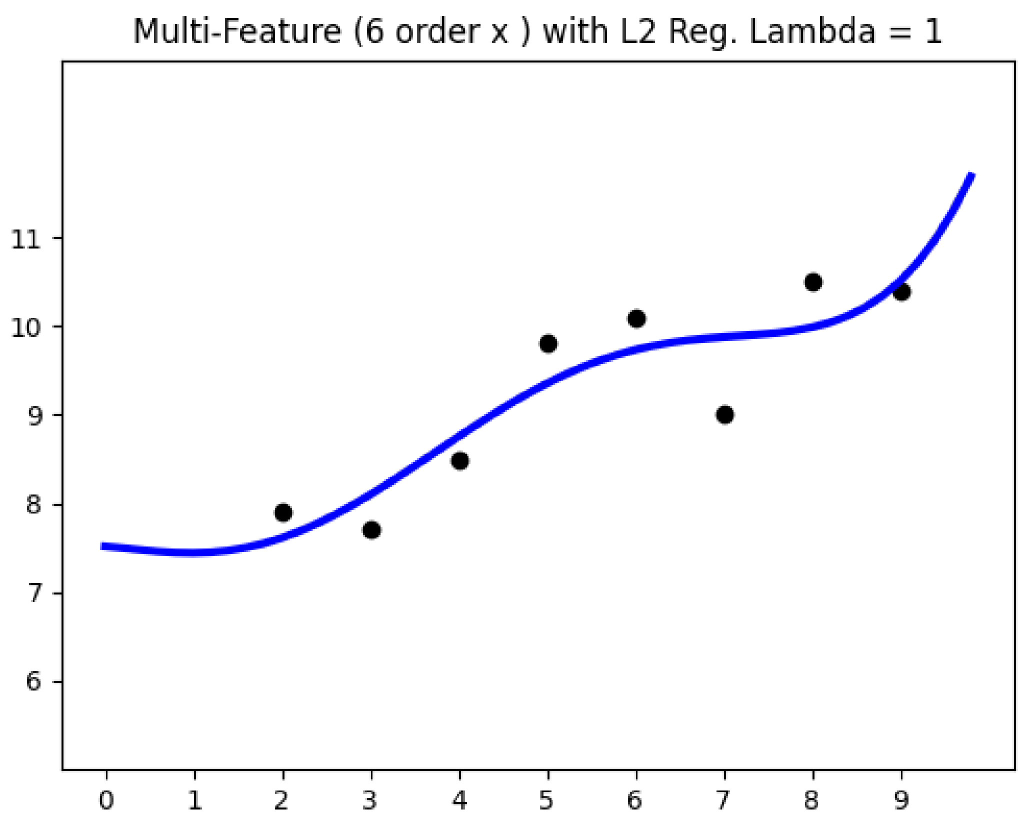
$= ?$ (b 에 대해 정리)

$\frac{\partial l_i}{\partial b}$ = ?

L2 Regularization의 효과

- Overfitting 방지
- Model parameter이 Mean $\mu = 0$, Standard Deviation σ 의 분포를 가진다.
- = Model parameter 값이 극단적으로 차이나지 않고 비슷하지만 조금씩 다른 값을 가진다.





L2 Regularization의 효과 (Cont'd)

- Overfitting 방지
- Model parameter이 Mean $\mu = 0$, Standard Deviation σ 의 분포를 가진다.
- = Model parameter 값이 극단적으로 차이나지 않고 비슷하지만 조금씩 다른 값을 가진다.

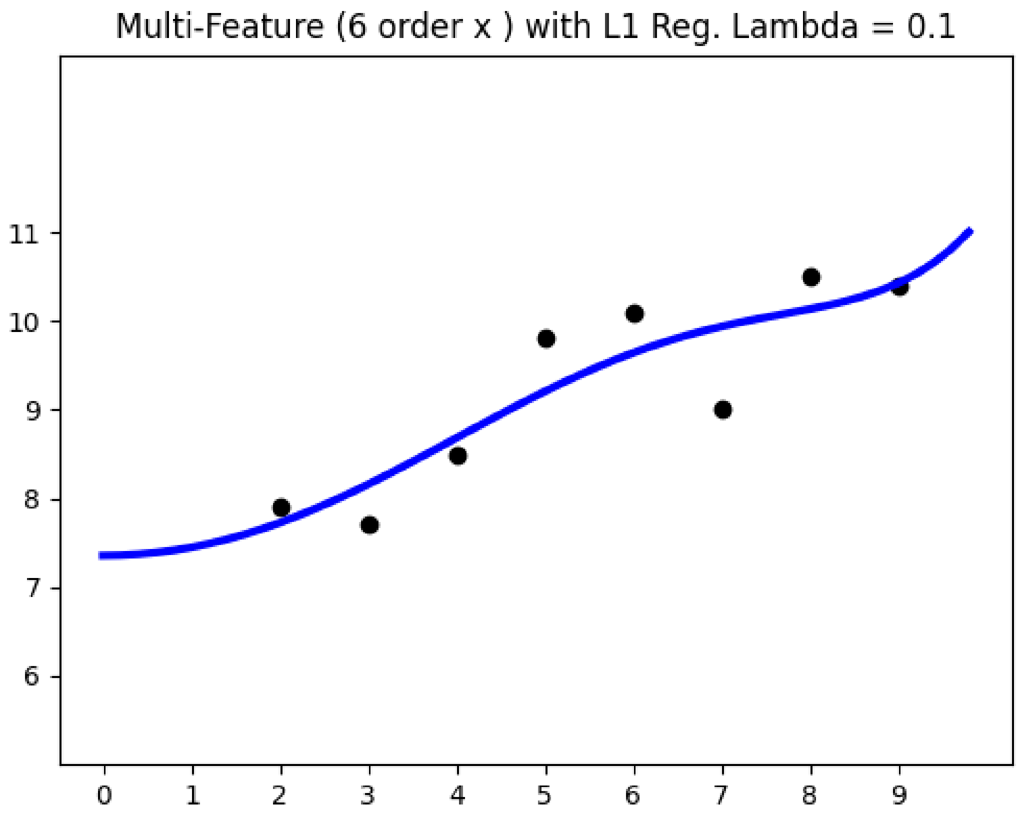
L2 Lambda	w1	w2	w3	w4	w5	w6	**Mean**	**SD**
No Reg.	-0.0102	0.326	-4.1293	26.3653	-88.8855	149.7337	**13.900**	**43.968**
0.1	-0.0005	0.0147	-0.1559	0.7017	-1.0072	-0.6312	**-0.180**	**0.611**
1	0	0.002	-0.0263	0.1258	-0.1013	-0.0721	**-0.012**	**0.082**

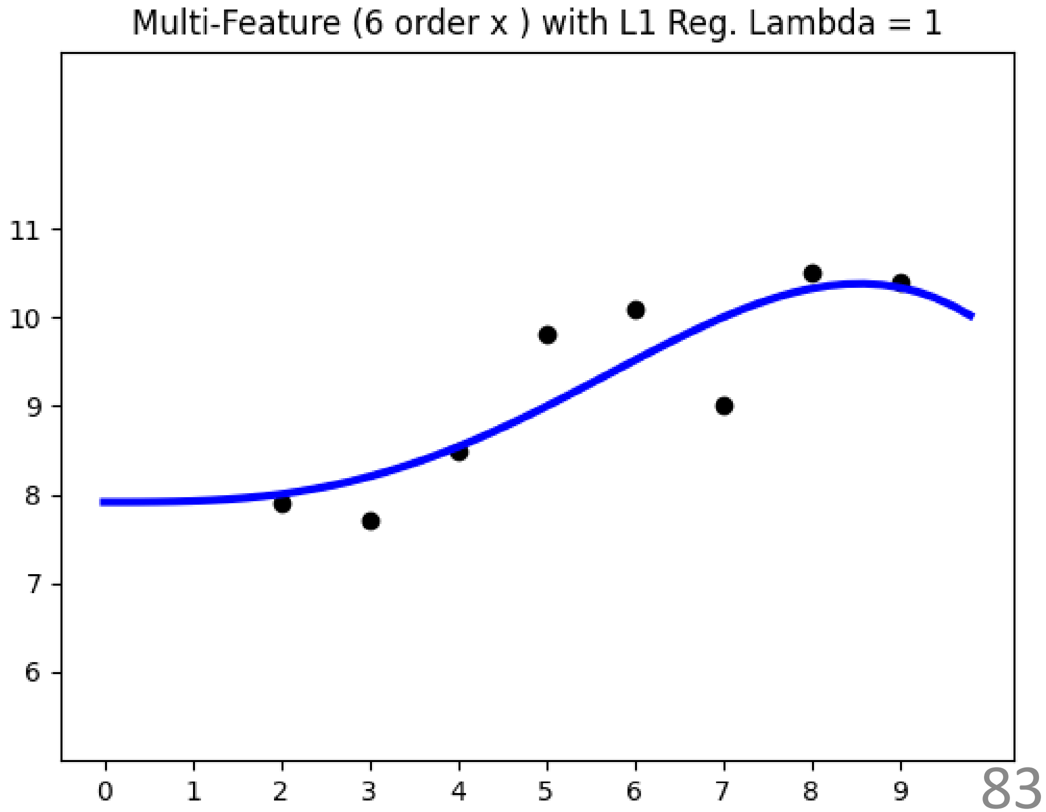
L1 Regularization

- L1 Regularization (Lasso Regularization)
  - Model Parameter 를 ϴ 라 표기하면, L1 Regularization은 Model이 $\|\theta\|_1^1$를 최소화 하는 ϴ를 학습하도록 한다.
  - $\|\theta\|_1^1$: ϴ Matrix element의 절대값의 합
  - Linear regression의 식이 $\hat{y}_i = w \cdot x_i + b$ 이고, 우리가 학습해야 할 variable이 w, b 이므로,
  - Linear regression의 L1 Norm은 $\sum_{i=1}^{k} |w_i| + |b|$ 이 된다. ($w =< w_1, w_2, \ldots, w_k >$)
  - 따라서,
    - New Objective Function: $l_{new} = \sum_{i=1}^{n}(y_i - (w \cdot x_i + b))^2$ **+λ(** $\sum_{i=1}^{k} |w_i| + |b|$)

L1 Regularization의 효과

- Overfitting 방지
- Sparse한 model parameter가 학습됨.
    - 몇몇 중요한 weight에만 값이 있고 중요하지 않은 parameter는 값이 0에 가까워진다.





L1 Regularization의 효과 (Cont'd)

- Overfitting 방지
- Sparse한 model parameter가 학습됨.
  - 몇몇 중요한 weight에만 값이 있고 중요하지 않은 parameter는 값이 0에 가까워진다.

L1 Lambda	w1	w2	w3	w4	w5	w6
No Reg.	-0.0102	0.326	-4.1293	26.3653	-88.8855	149.7337
0.1	0	-0.0002	-0.0009	0.0033	0.0904	0
1	0	-0.0002	0	0.012	0	0

Scikit-learn Classes

- linear_model.Ridge :
  - Linear Regression with L2 Regularization
- linear_model.Lasso :
  - Linear Regression with L1 Regularization

L2, L1 Regularization: 무엇을 언제 사용하는가?

- Overfitting을 방지하기 위해 Regularization 을 항상 사용한다.
  - 기본적으로는 L2를 사용하도 무방함.
  - 최소한의 주요 Feature을 추출하고 싶을 경우 L1을 사용하면 좋음.
- Regularization Weight **λ** 는 어떻게 정하는가?
  - Grid search
    - 여러 **λ** 값을, Cross validation을 사용하여 테스트하여 evaluation set에 대한 성능이 가장 높은 **λ** 값을 사용.

Usage 2 :
Feature Importance analysis

Feature Importance Analysis

- 여러 Feature들 중에 어느 feature가 결론 (y)를 예측하는데 얼마만큼 중요한지 분석할 수 있다.

  예) 평균 식사량, 운동량, 복부 둘레 중 몸무게에 가장 영향을 미치는 feature는?

Wine Quality Data

Input Feature

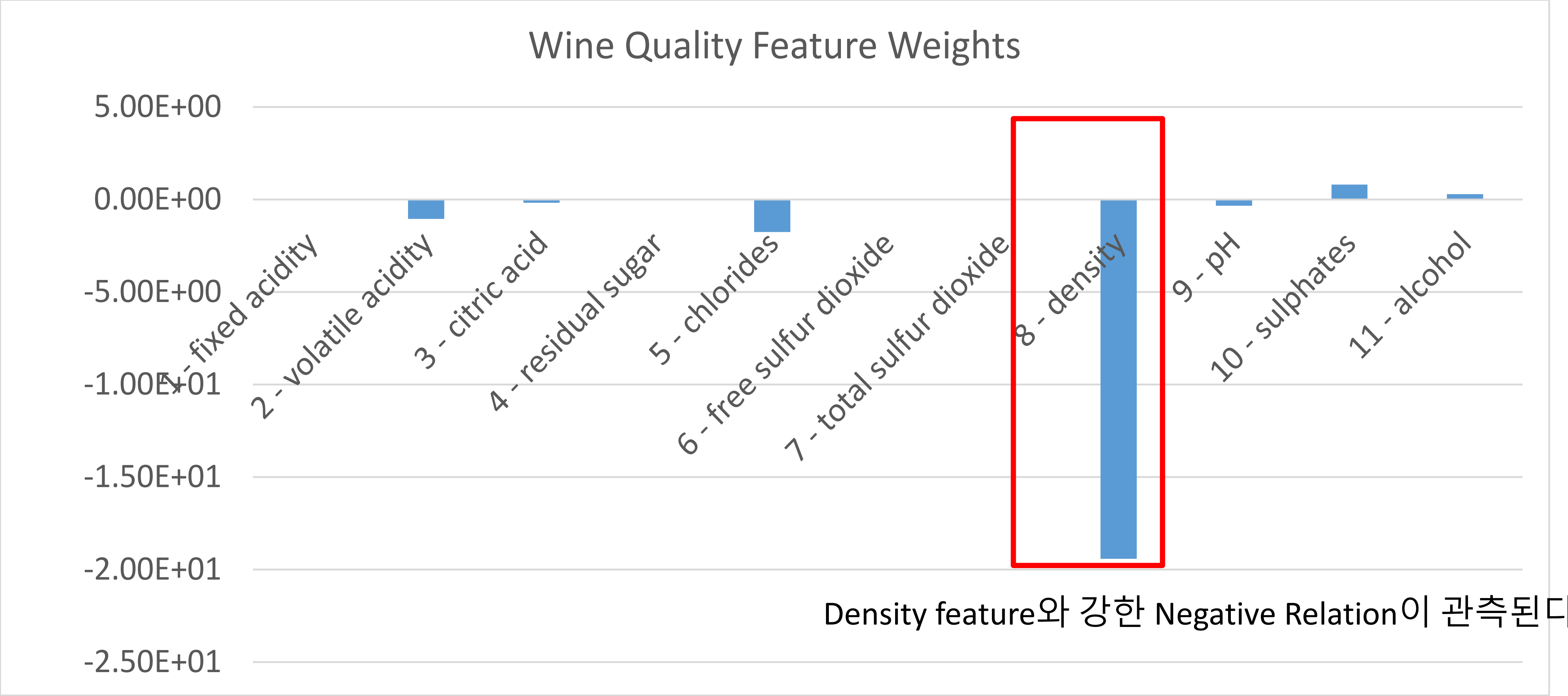
1 - fixed acidity
2 - volatile acidity
3 - citric acid
4 - residual sugar
5 - chlorides
6 - free sulfur dioxide
7 - total sulfur dioxide
8 - density
9 - pH
10 - sulphates
11 – alcohol

Output variable (based on sensory data):
12 - quality (score between 0 and 10)

P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis. Modeling wine preferences by data mining from physicochemical properties. In Decision Support Systems, Elsevier, 47(4):547-553, 2009.

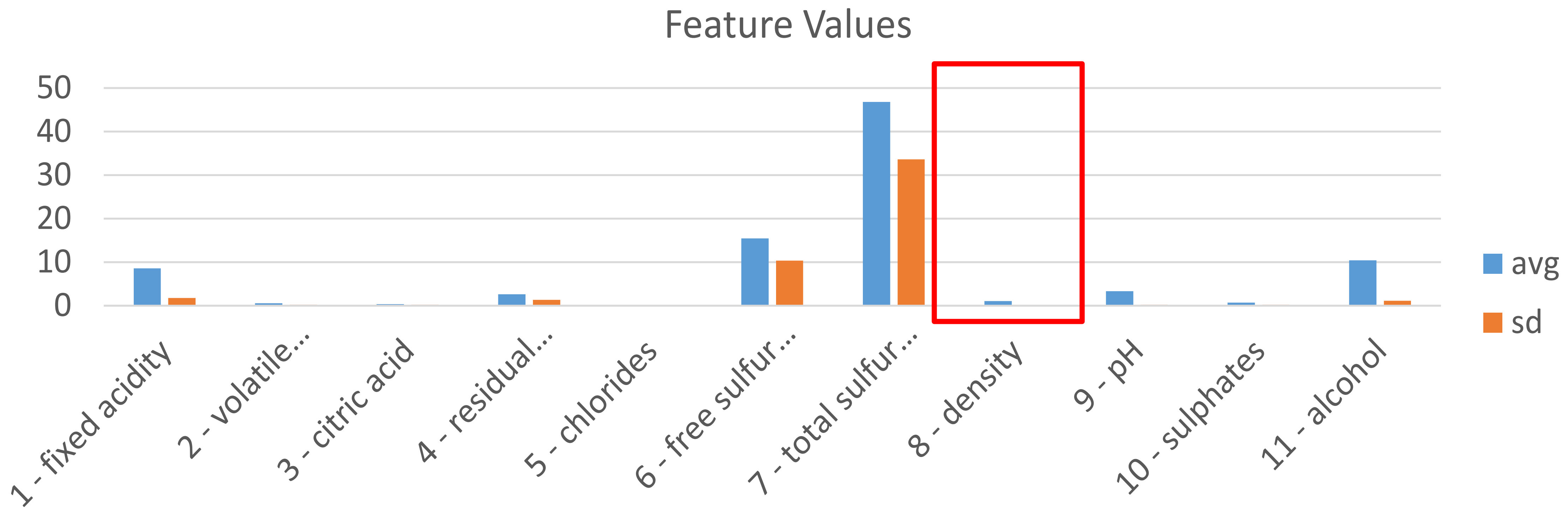
https://archive.ics.uci.edu/ml/datasets/Wine+Quality

Wine Quality – Feature Weights



Density feature와 강한 Negative Relation이 관측된다.

Wine Quality – Feature value average and standard deviation



다만 실제 Density feature의 값 평균을 보면, 실제 값이 매우 작다.
Total sulfur dioxide 가 feature 값 평균이 제일 큰데, Total sulfur dioxide가 더 중요하지 않을까?

Normalization :
제대로 된 해석을 위해서는 서로 다른 Feature 사이의 값의 규모(Scale)에 공통된 기준이 필요

- Feature 값의 규모가 다를 경우 Feature Weight값의 단순비교는 힘들다.
  - 예) 다음과 같은 경우
    - Density 평균이 0.5 이고 sd가 0.1 에서 Density가 0.2 오르면(0.7이되면) wine quality가 1 내려간다.
    - Total sulfur dioxide 평균이 50 이고 sd가 10 에서 Total sulfur dioxide 값이 5 오르면(55가 되면) 1 내려간다.

    - **Question 1 :** Density와 Total sulfur dioxide 중 어느 feature이 wine quality에 더 부정적인 영향을 미치는가?
    - **Question 2 :** Linear Regression의 weight의 절대값은 Density의 Weight이 더 큰가? Total sulfur dioxide의 weight가 더 큰가?

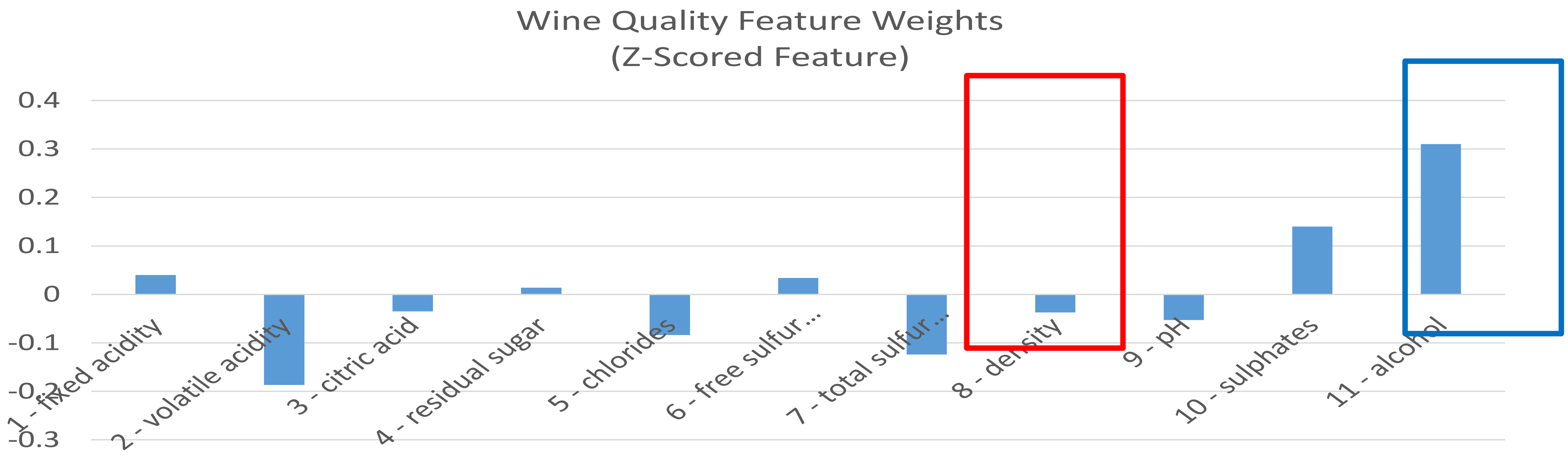
Normalization 예 : Z-Score

$$Z = \frac{\overset{\text{Score}}{x} - \overset{\text{Mean}}{\mu}}{\underset{\text{SD}}{\sigma}}$$

Z-score가 의미하는 내용을 풀어 쓰면?

Image Source: https://www.simplypsychology.org/z-score.html

Wine Quality – Feature Weights (Again)

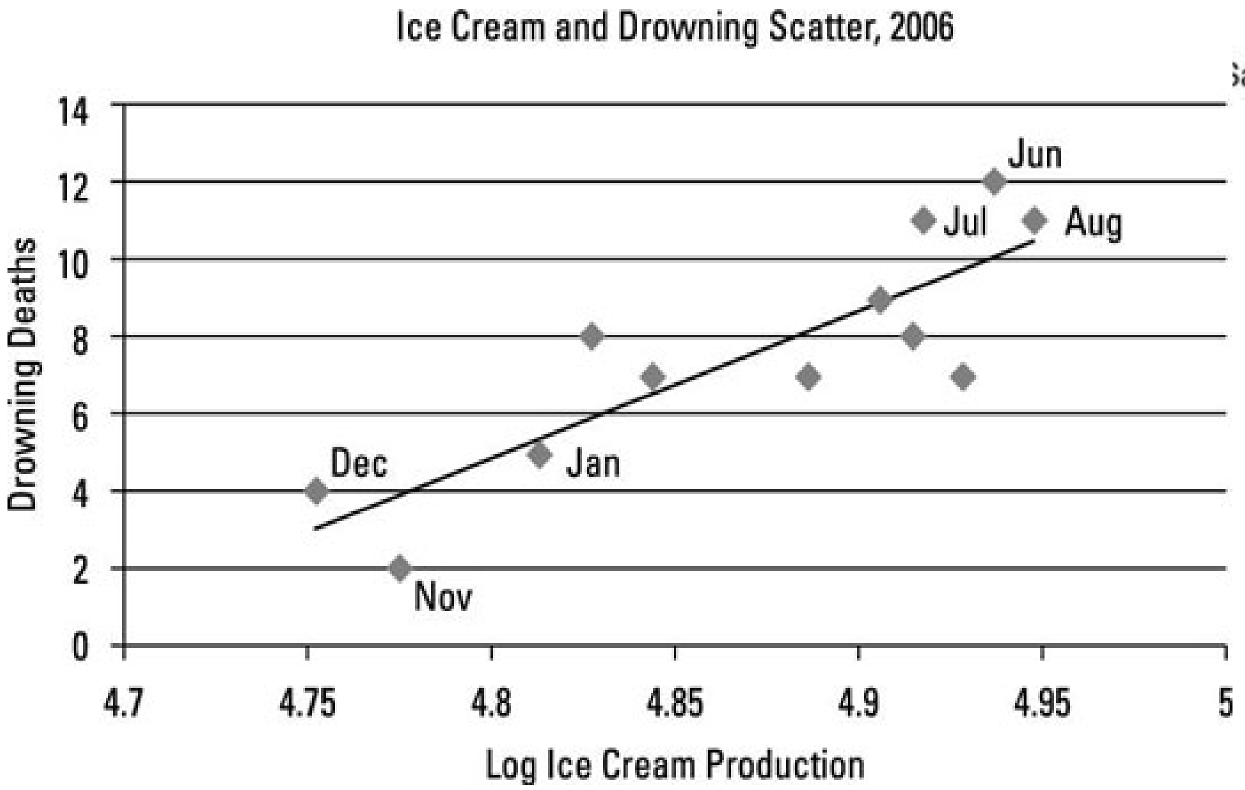


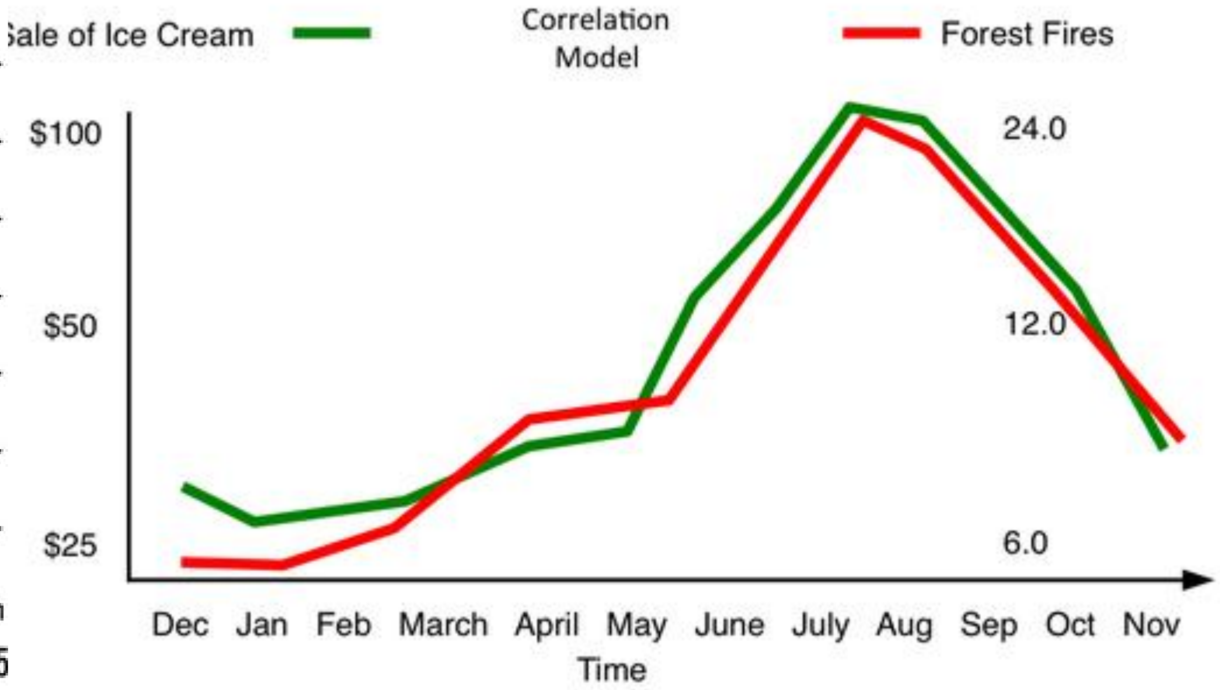
Feature Importance **해석할 때 유의점**: Correlation VS. Causation

- **Causation :** 원인-결과 관계
  - 예)
    - **원인:** 직선거리 300 km를 평균시속 100km/h로 달렸다.
    - **결과 :** 3 시간만에 시작점에서 끝점에 도착했다.
- **Correlation** : 연관되어서 패턴이 나타나지만 원인-결과관계가 있는지 없는지는 모른다.
  - 예) Drowning deaths and ice-cream sales are **strongly correlated**.

- Just because there's a strong correlation between two variables, there isn't necessarily a causal relationship between them.

Correlation VS. Causation





https://www.decisionskills.com/blog/how-ice-cream-kills-understanding-cause-and-effect

Summary

- Feature Importance Analysis : 학습된 Linear Regression의 weight vector를 사용하여 어떤 feature가 예측 값에 어느정도 영향을 주는지 분석할 수 있다.
- Linear Regression을 사용한 Feature Importance Analysis를 수행할 경우, z-score 등의 기법을 사용하여 training/test data의 값을 normalization한 다음 Model을 학습하여야 한다.
- 분석에 있어서 feature와 결과 (예측 값)와의 관계는 Causation (원인-결과) 관계가 아닌, Correlation (연관) 관계임을 항상 유의하라.