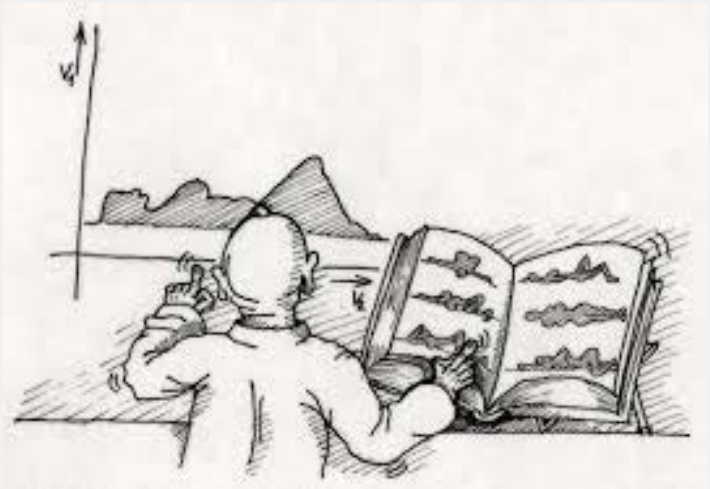
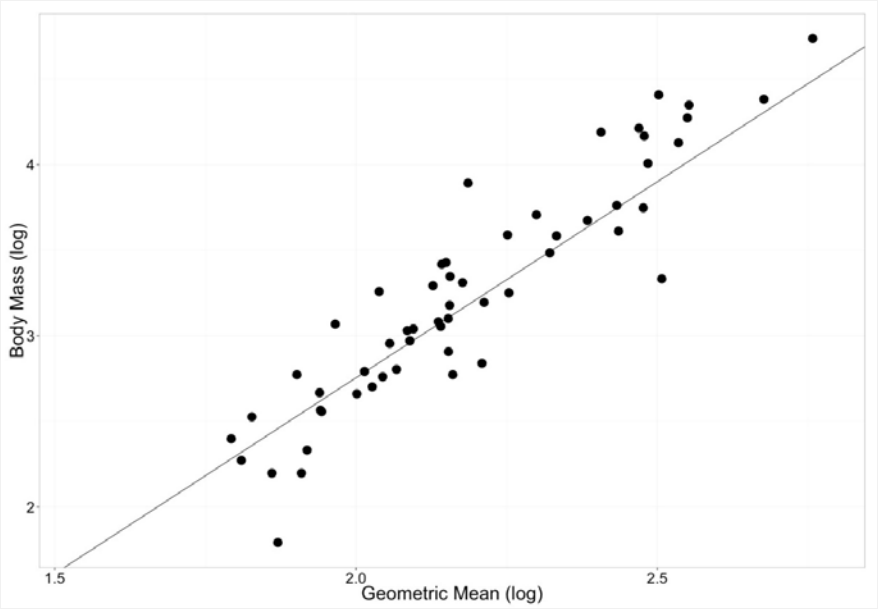
수치해석
(2019학년도 1학기)

[7주/2차시 학습내용]: Ch.14 Curve Fitting (곡선 접합) 중 Linear Least-Squares Regression (선형 최소 제곱 회귀법)에 대해 알아본다

국민대학교 소프트웨어학부

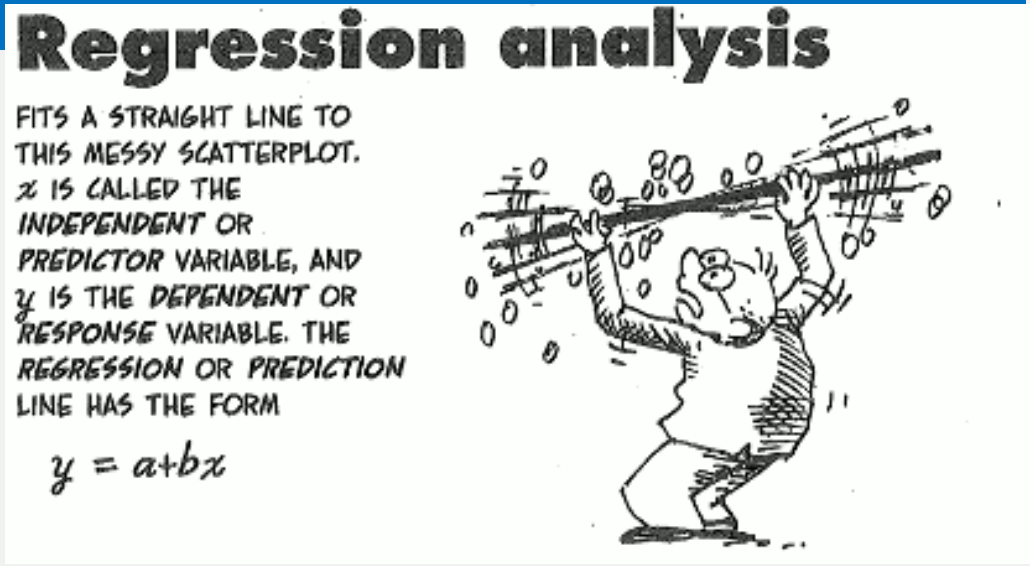
Part 4 Curve Fitting
(곡선 접합)
Ch14. Linear Regression
(선형 회귀)



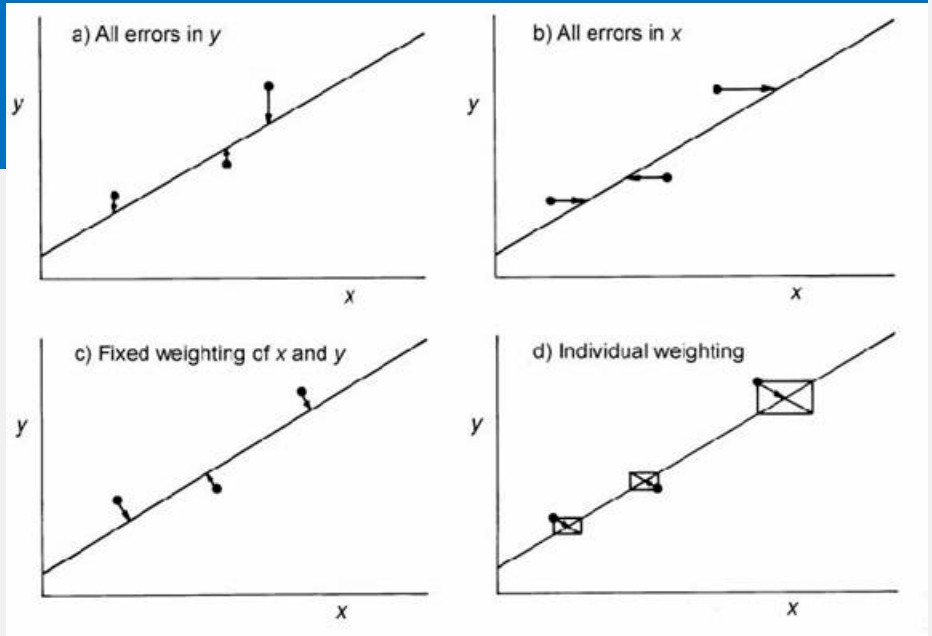


국민대학교 소프트웨어학부

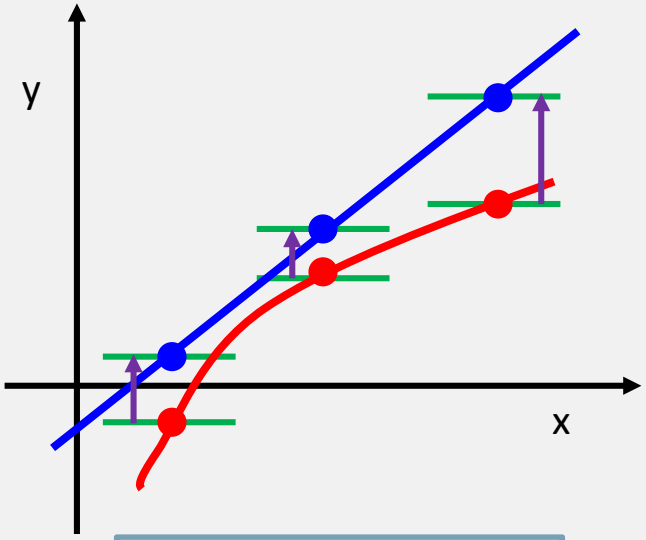
Error Optimization



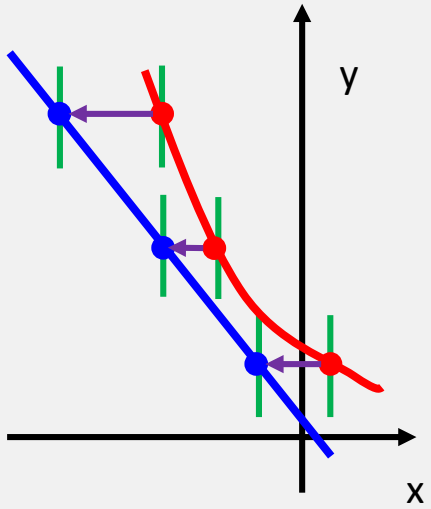
X and Y axis Errors



파란 색 직선 위의 점은 항상 에러를 가지고 있다

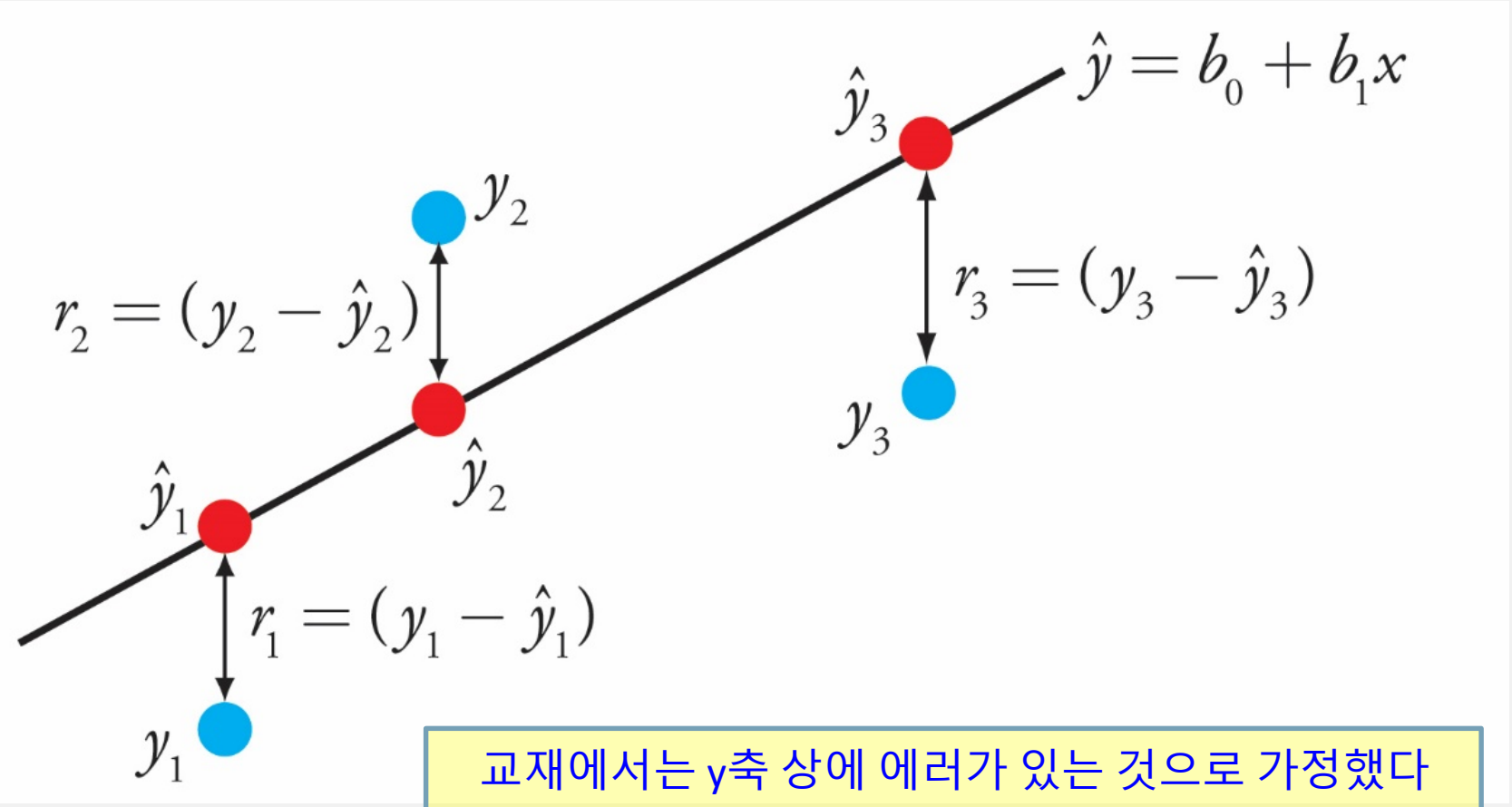


Y 축 상의 에러



X 축 상의 에러

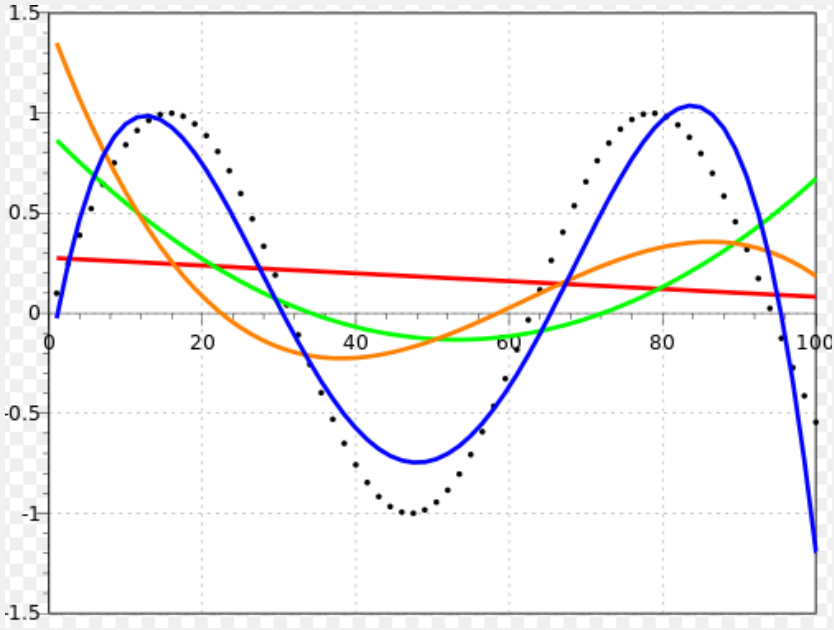
Y axis Error



교재에서는 y축 상에 에러가 있는 것으로 가정했다

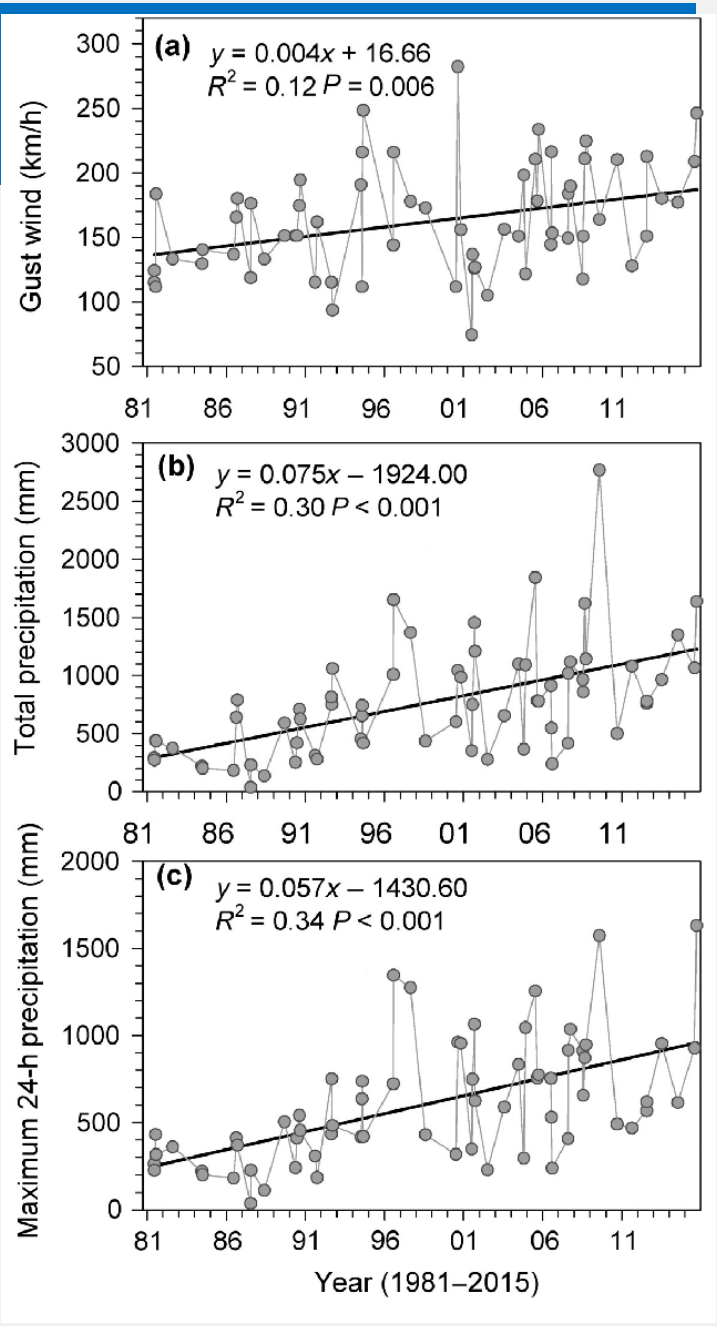
Fitting (입어 보기)

- Fitting room
  - a room in a store in which one can try on clothes before deciding whether to purchase them. (옷 사기 전에 옷을 입어보는 곳)
  - 선으로 데이터를 대신 표현해 보기
  - 선으로 데이터를 입어 보기
- Curve fitting Strategy (곡선 접합 전략)
  - 커브 피팅은 데이터 요소 집합에 가장 잘 맞는 커브를 찾는 프로세스.
  - 최소 제곱(least squares)은 제곱 된 잔차(residuals, 여기서는 에러)의 합을 최소화하는 곡선 맞춤 방법.
  - 잔차(residuals)는 실제 값과 예측 값과의 차이



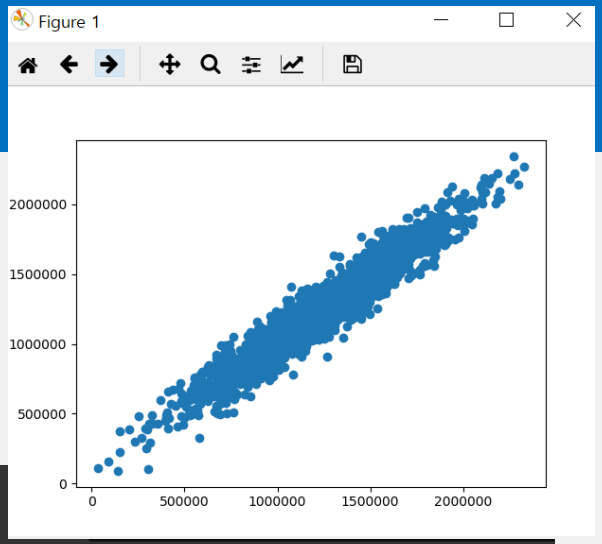
Curve fitting

- 돌풍(gust wind, km/h)의 최소 제곱 분석에 의한 시간적 경향 선형 회귀 접합
- 대만에 상륙하는 태풍의
- (a) 돌풍의 속도 (km / h)
- (b) 총 강수량 (mm)
- (c) 최대 24 시간 강수량 (mm)의 시간 추세
- 타이완 중앙 기상국 타이푼 데이터베이스 (http://rdc28.cwb.gov.tw/)의 데이터.



Curve fitting

- 개인의 연봉 수입과 소유 집값에는 어떤 연관성이 있을까?
- 수입이 높으면, 비싼 집에 사는가?
- 미국 휴스톤의 예를 보자.



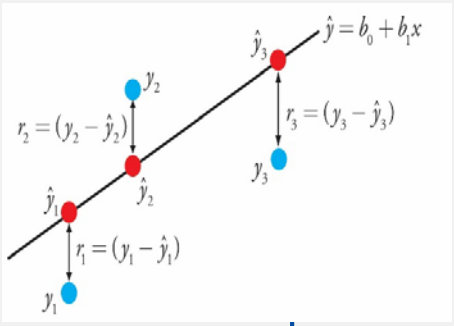
USA_Housing.csv

https://github.com/SCKIMOSU/Numerical-Analysis/blob/master/houston.py

	A	B	C	D	E	F	G
1	Avg. Area Income	Avg. Area House Age	Avg. Area Number of	Avg. Area Number of	Area Population	Price	Address
2	79545.45857	5.682861322	7.009188143	4.09	23086.8005	1059033.558	208 Michael Ferry A Laurabury, NE 3701(
3	79248.64245	6.002899808	6.730821019	3.09	40173.07217	1505890.915	188 Johnson Views S Lake Kathleen, CA 4
4	61287.06718	5.86588984	8.51272743	5.13	36882.1594	1058987.988	9127 Elizabeth Strav Danieltown, WI 0648
5	63345.24005	7.188236095	5.586728665	3.26	34310.24283	1260616.807	USS Barnett FPO AP 44820
6	59982.19723	5.040554523	7.839387785	4.23	26354.10947	630943.4893	USNS Raymond FPO AE 09386
7	80175.75416	4.988407758	6.104512439	4.04	26748.42842	1068138.074	06039 Jennifer Islan Tracyport, KS 16077
8	64698.46343	6.025335907	8.147759585	3.41	60828.24909	1502055.817	4759 Daniel Shoals Nguyenburgh, CO 20
9	78394.33928	6.989779748	6.620477995	2.42	36516.35897	1573936.564	972 Joyce Viaduct Lake William, TN 17
10	59927.66081	5.36212557	6.393120981	2.3	29387.396	798869.5328	USS Gilbert FPO AA 20957

Curve-fitting Strategy (곡선 접합 전략)

Y 축 상의 에러



내가 만든 **직선**이 **실제점 (Real Point)** 3개를 동시에 지나가도록 만들겠다가 선형회귀의 기본 철학임, 그러나 동시에 실제점 3개를 지나가도록 하지는 못한다.

$$y' = a_0 + a_1 x + e$$

Modeled point
(직선 위의 점)
(x_3, y'_3)

Real point
(x_2, y_2)

Error e_3

Real point
(x_1, y_1)

Error e_2

(x_2, y'_2)

Error e_1

(x_1, y'_1) Modeled point

intercept a_0

a_1

8 6 4 2 — 2 4 6 8 10 x

기준점 x는 동일
기준점 x 에 대해
실제점은 y
모델화 된 점은 y'

파란 색 직선 위의 점, 모델화 된 점 y' 가 모델화가 너무 잘 되어 에러를 가지고 있지 않다면, $y' = y$ 가 되는 것이 곡선 접합의 전략. 에러가 0 이 되는 직선을 찾아야 한다. 그 직선은 $y' = y$ 이 되는 직선이며 이것이 최적화

a_0 : intercept

a_1 : Slope

What is Regression ?

numpy 메소드 polyfilt()을 통해 선형 회귀, 다항식 회귀에 대한 개념을 파악한다

회귀 분석은?

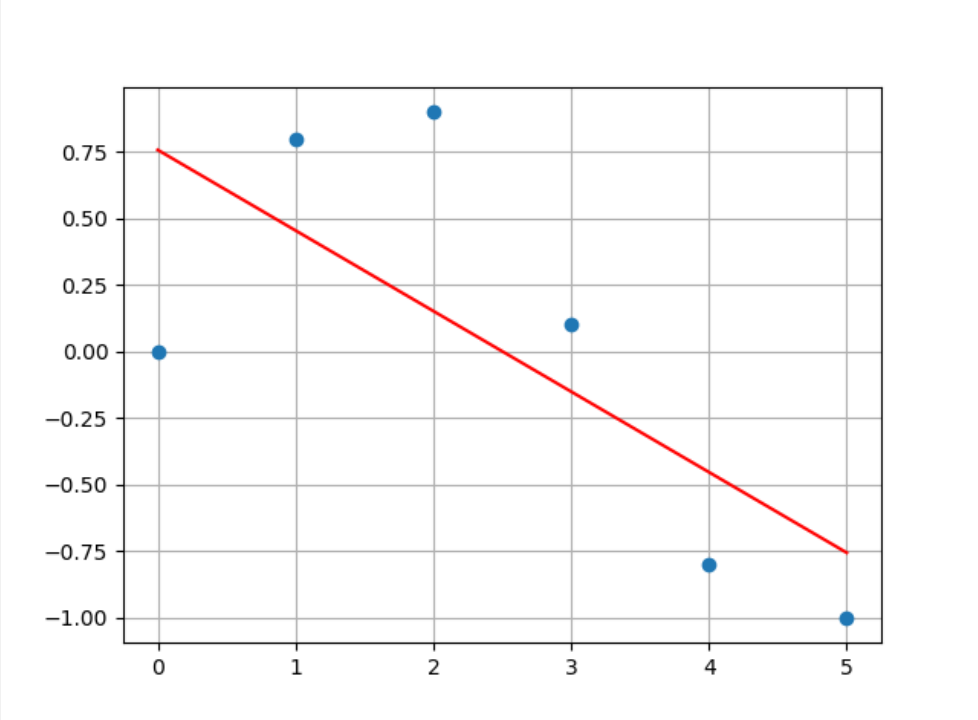
- 회귀 분석은 주어진 X, Y 값을 갖는 인스턴스를 부여 받고
- 알려지지 않은 X에 대해 Y를 예측할 수 있도록 함수를 배워야 하는 감독 학습 문제 (supervised learning problem)이다 .
- 선형 회귀 분석은 입력 x, y가 주어지면, 직선 함수 y’ 을 찾는 것이다.

Polyfit을 통한 간단한 Linear Regression

```
import numpy as np
import matplotlib.pyplot as plt
x=np.array([0,1,2,3,4,5])
y=np.array([0, 0.8, 0.9, 0.1, -0.8, -1])
n=np.size(x)
b=(n*np.sum(x*y)-(np.sum(x)*np.sum(y)))/(n*np.sum(x**2)-(np.sum(x))**2)
# b= -0.30285714285714288: slope
a=(np.sum(y)-b*np.sum(x))/n
# 0.75714285714285723: intercept

p1=np.polyfit(x,y,1)  # 1: linear  array([-0.30285714,  0.75714286])  slope and
intercept
plt.figure(1)
plt.plot(x, y, 'o')
plt.grid()
plt.plot(x, np.polyval(p1,x), 'r-')  # p1 from np.polyfit, plot(x, p1 with polyval

plt.figure(2)
plt.plot(x, y, 'o')
plt.grid()
plt.plot(x, p1[0]*x+p1[1], 'r-')
```

Polyfit을 통한 간단한 Linear Regression

- numpy 메소드 polyfilt()을 통해 **선형 회귀(Linear Regression)**에 대한 개념을 파악한다
- https://github.com/SCKIMOSU/Numerical-Analysis/blob/master/polyfit.py



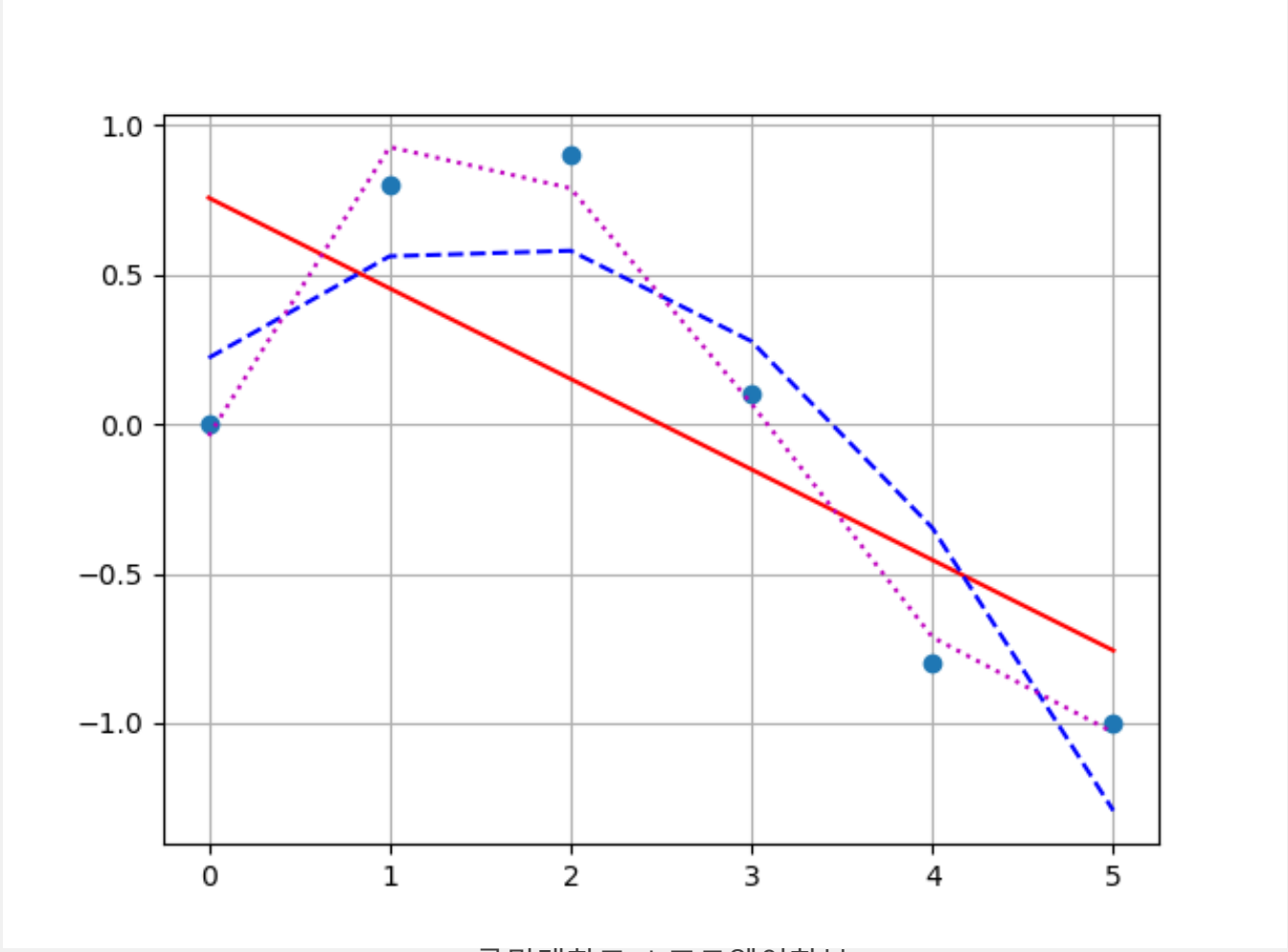
Polyfit을 통한 Polynomial Regression

- numpy 메소드 polyfilt()을 통해 **다항 회귀 (Polynomial Regression)** 에 대한 개념을 파악한다

```
p2=np.polyfit(x, y, 2)  #  quadratic  second-order
#  array([-0.16071429,  0.50071429,  0.22142857]) coefficient of quadratic  second-order

p3=np.polyfit(x, y, 3)  # cubic
#  array([ 0.08703704, -0.81349206,  1.69312169, -0.03968254]) coefficient of cubic third-order

plt.figure(2)
plt.plot(x, y, 'o')
plt.grid()
plt.plot(x, np.polyval(p1,x), 'r-')
# p1 from np.polyfit, plot(x, p1 with polyval

plt.plot(x, np.polyval(p2,x), 'b--') # --: dached line
#np.polyval(p2,x)=   array([ 0.22142857,  0.56142857,  0.58      ,  0.27714286, -0.34714286, -
1.29285714])

plt.plot(x, np.polyval(p3,x), 'm:') # : dotted line
```

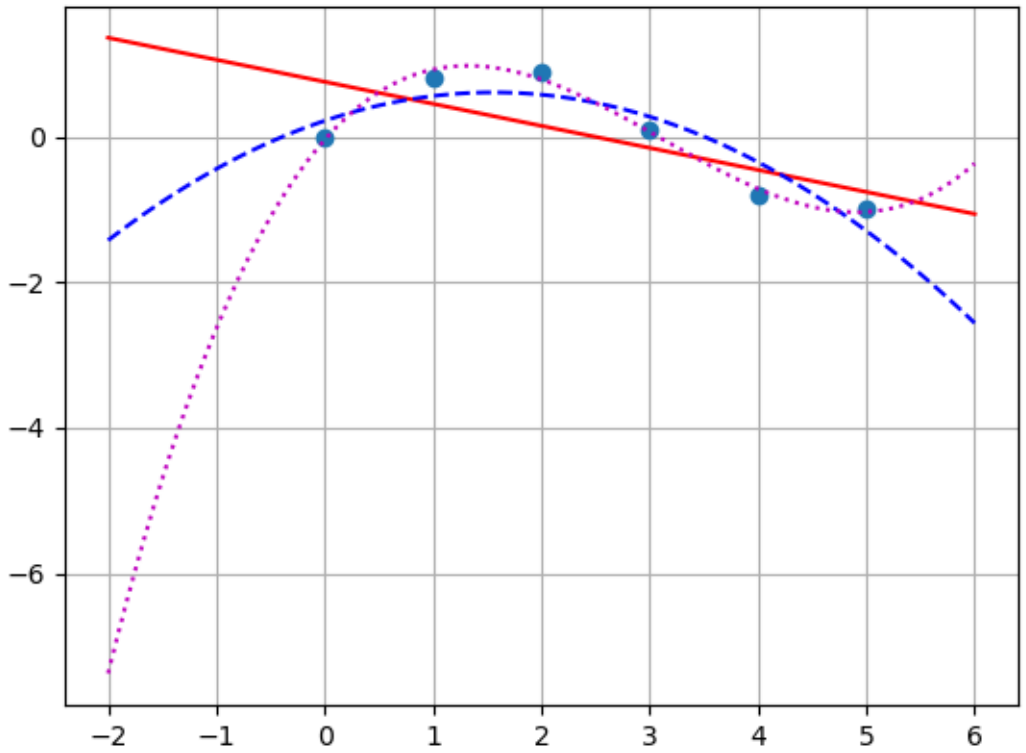
Linear & Polynomial Regression

- 선형 회귀 및 다항 회귀 시각화



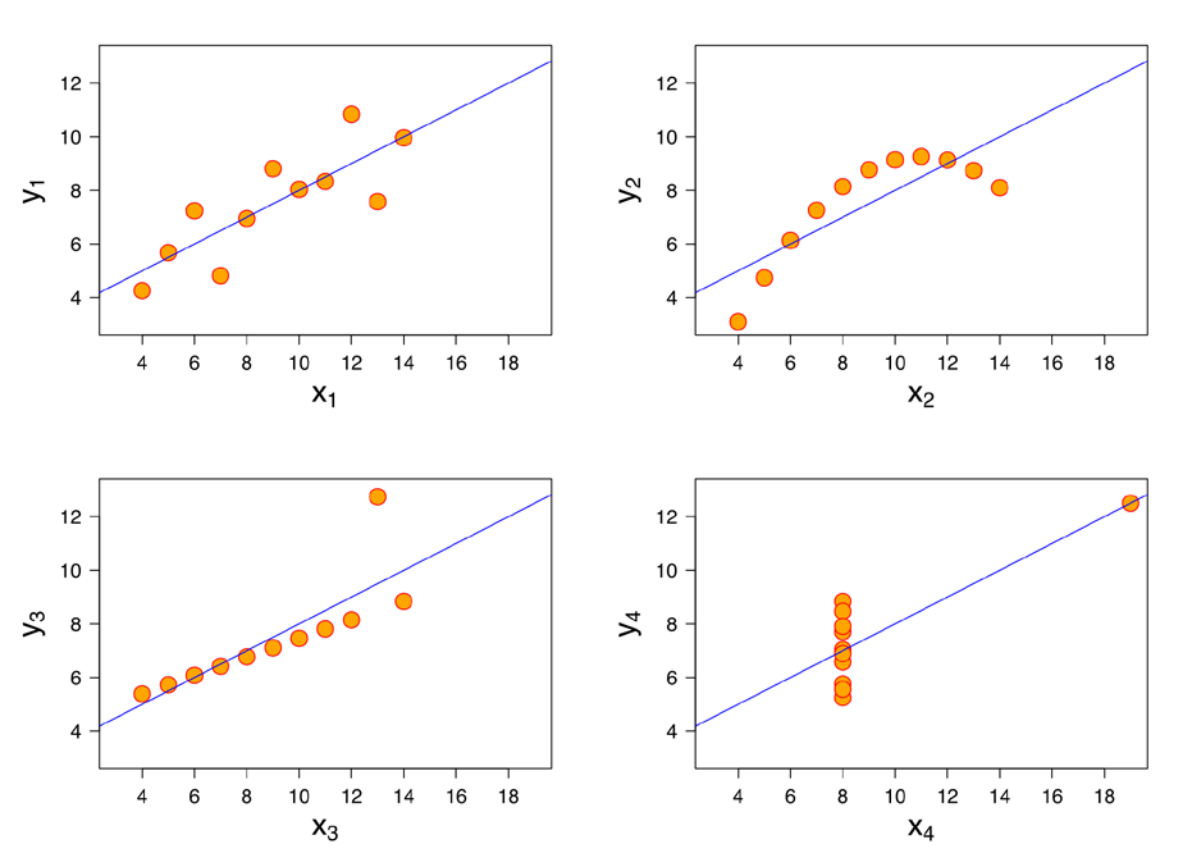
Prediction (when -1, -2, and 6?)

```
plt.figure(3)
plt.plot(x, y, 'o')
plt.grid()
xp=np.linspace(-2, 6, 100)
plt.plot(xp, np.polyval(p1,xp), 'r-')
plt.plot(xp, np.polyval(p2,xp), 'b--')
plt.plot(xp, np.polyval(p3,xp), 'm:')
```



동일한 선형 회귀식을 갖는 다양한 데이터 셋

- 여러 데이터 집합은 동일한 선형 회귀식을 갖지만, 그것의 데이터는 매우 다른 양상을 보인다.



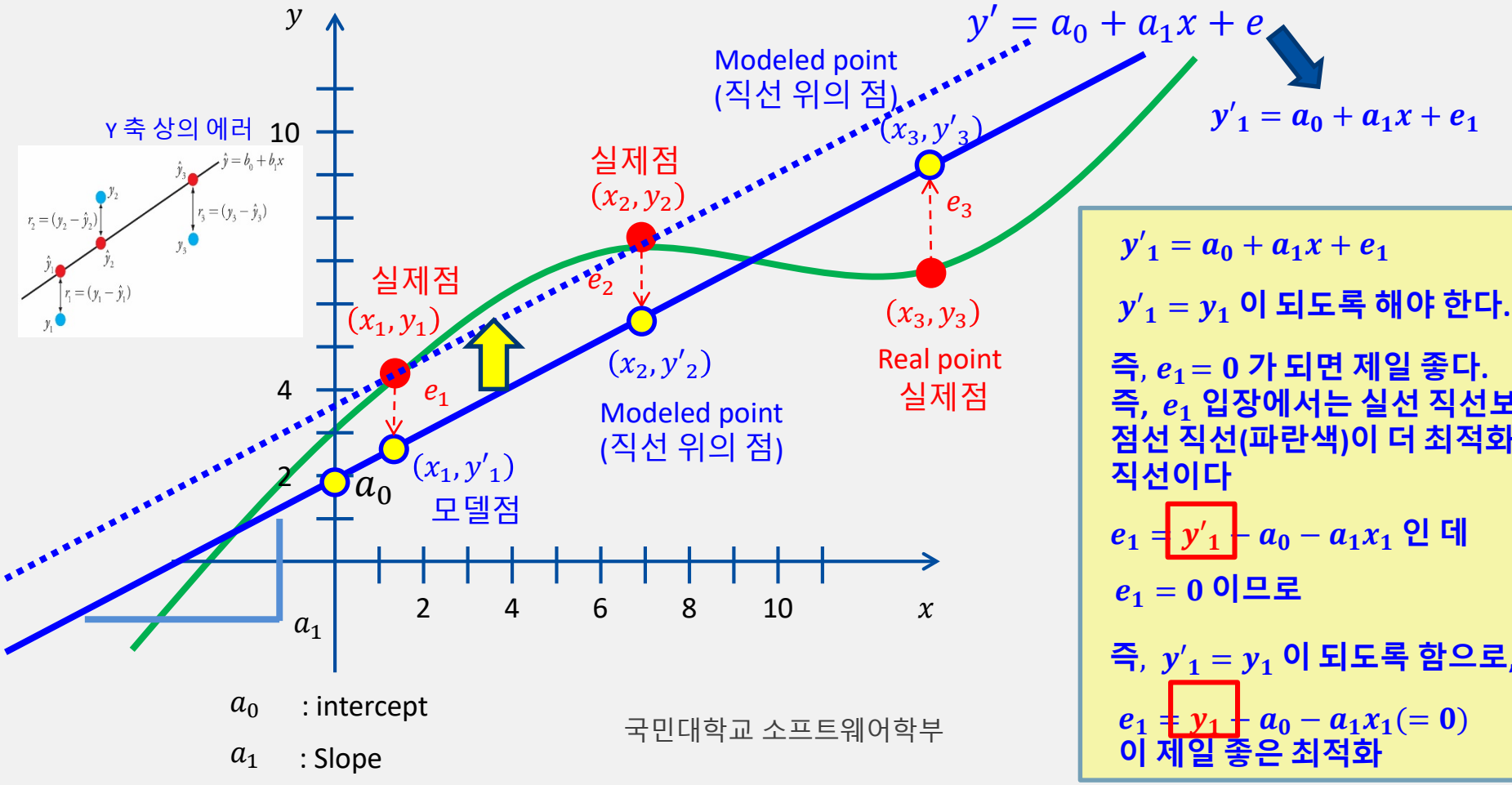
Curve-fitting Strategy
(곡선 접합 전략)

Error Optimization

수치해석은 근사치의 학문

가장 이상적인 Curve-fitting Strategy for e_1

내가 만든 직선이 실제점 3개를 동시에 지나가도록 만들겠다가 선형회귀의 기본 철학임, e_1 의 입장에서는 **직선**이 **실제점** y_1 을 지나가면 가장 이상적이다. 실제로 **직선**이 y_1 을 지나간다고 **가정**한다. 그래서 e_1 은 원래 $y'_1 - a_0 - a_1x_1$로 표현되는 데, 선형회귀의 기본 철학인 내가 만든 직선이 실제점을 지나간다는 희망사항에 따라, e_1 은 $y_1 - a_0 - a_1x_1$ 으로 표현된다.



a_0 : intercept

a_1 : Slope

국민대학교 소프트웨어학부

$y'_1 = a_0 + a_1x + e_1$

$y'_1 = y_1$ 이 되도록 해야 한다.

즉, $e_1 = 0$ 가 되면 제일 좋다.
즉, e_1 입장에서는 실선 직선보다 점선 직선(파란색)이 더 최적화된 직선이다

$e_1 = y'_1 - a_0 - a_1x_1$ 인 데

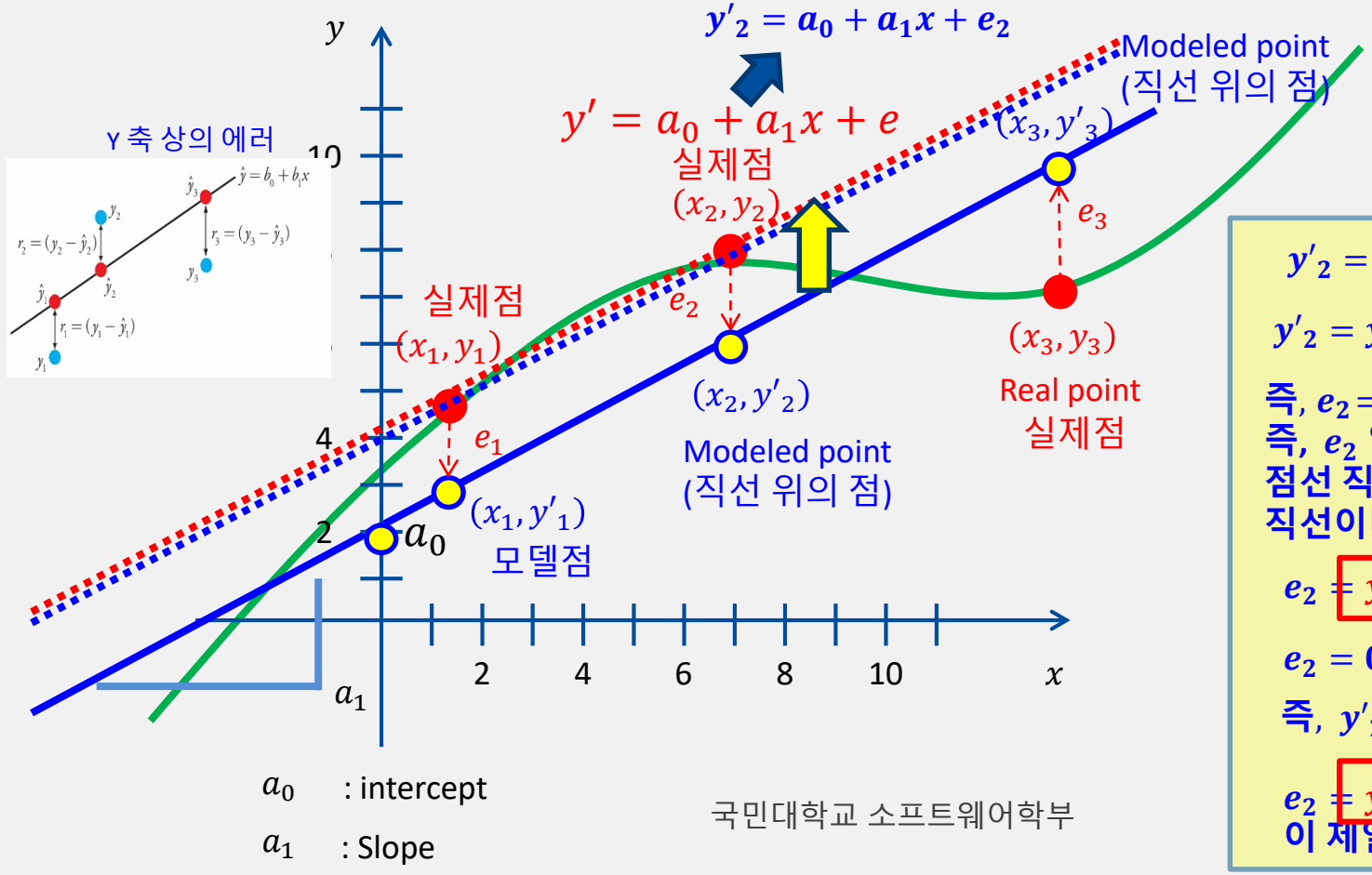
$e_1 = 0$ 이므로

즉, $y'_1 = y_1$ 이 되도록 함으로,

$e_1 = y_1 - a_0 - a_1x_1 (= 0)$
이 제일 좋은 최적화

가장 이상적인 Curve-fitting Strategy for e_2

내가 만든 직선이 실제점 3개를 동시에 지나가도록 만들겠다가 선형회귀의 기본 철학임, e_2 의 입장에서는 **직선**이 실제점 y_2 를 지나가면 가장 이상적이다. 실제로 **직선**이 y_2 를 지나간다고 **가정**한다. 그래서 e_2 은 원래 $y'_2 - a_0 - a_1x_2$로 표현되는 데, 선형회귀의 기본 철학인 내가 만든 직선이 실제점을 지나간다는 희망사항에 따라, e_2 는 $y_2 - a_0 - a_1x_2$ 으로 표현된다.



a_0 : intercept

a_1 : Slope

국민대학교 소프트웨어학부

$y'_2 = a_0 + a_1x + e_2$

$y'_2 = y_2$ 이 되도록 해야 한다.

즉, $e_2 = 0$ 가 되면 제일 좋다.
즉, e_2 입장에서는 실선 직선보다 점선 직선(빨강색)이 더 최적화된 직선이다

$e_2 = y'_2 - a_0 - a_1x_2$ 인 데

$e_2 = 0$ 이므로

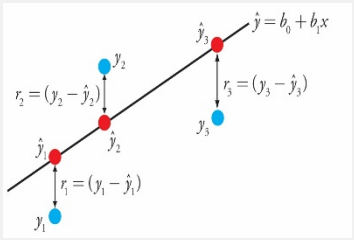
즉, $y'_2 = y_2$ 이 되도록 함으로,

$e_2 = y_2 - a_0 - a_1x_2 (= 0)$
이 제일 좋은 최적화

이상적인 Curve-fitting Strategy for e_1, e_2, e_3

$$y'_1 = a_0 + a_1 x + e_1$$
$$y'_2 = a_0 + a_1 x + e_2$$
$$y'_3 = a_0 + a_1 x + e_3$$

Y 축 상의 에러



a_0 : intercept

a_1 : Slope

$$y' = a_0 + a_1 x + e$$

$e_1 = y_1 - a_0 - a_1 x_1$
즉, $e_1 = 0$ 이 되는 것이 제일 좋은 최적화

$e_2 = y_2 - a_0 - a_1 x_2$
즉, $e_2 = 0$ 이 되는 것이 제일 좋은 최적화

$e_3 = y_3 - a_0 - a_1 x_3$
즉, $e_3 = 0$ 이 되는 것이 제일 좋은 최적화

직선은 하나지만 각기 다른 요구 조건을 수용해야 하고, 수용할 수 있다고 가정한다

곡선 접합 전략

- 에러 입장에서 요구하는 사항을 다 들어주게 되면, 세 개의 다른 직선을 만들어야 한다.
- 다른 직선이란 기울기 (a_1) 와 절편 (a_0) 이 다르다는 얘기이다.
- 우리가 정의한 직선 y'_1, y'_2, y'_3는 같은 기울기 (a_1)와 같은 절편(a_0)으로 구성되어 있어서 하나의 직선이 된다.
- 즉, 기울기 (a_1) 와 절편 (a_0) 이 다른 세 개의 직선을 만들지 않았다.
- 한 개의 직선에 의해 만들어지는 에러 값 (e_1 , e_2, e_3)이 달라졌다.
- 하나의 기울기 (a_1)와 하나의 절편(a_0)으로 구성된 하나의 직선을 만드는 것이 목표이다.

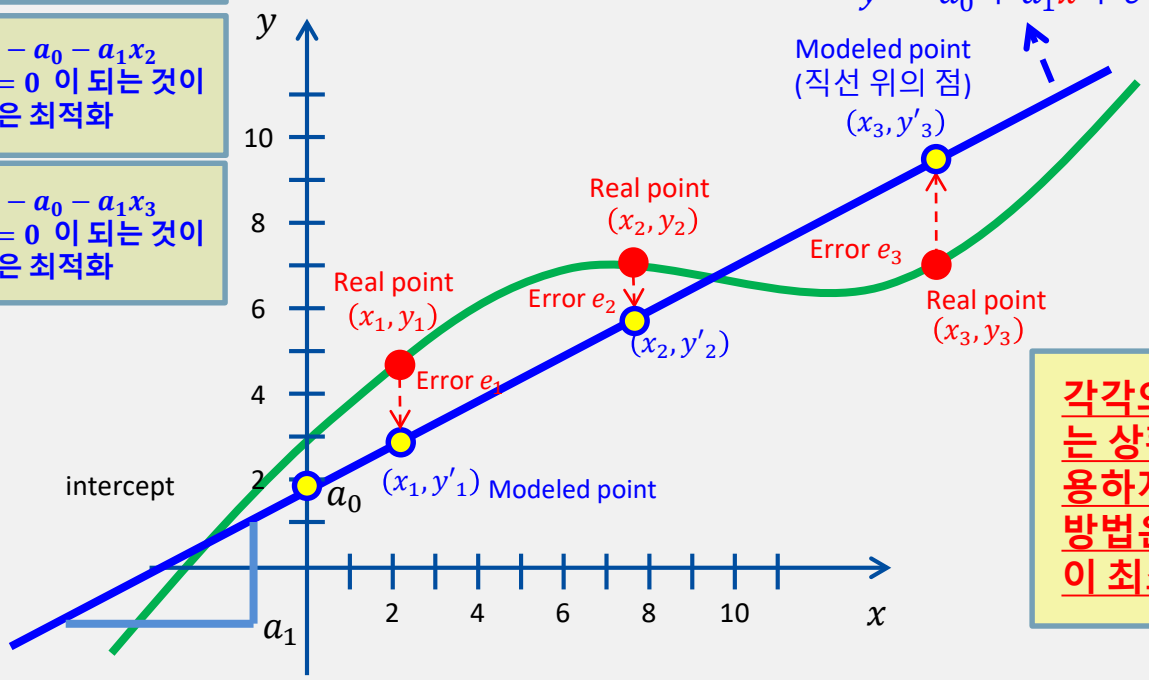
$$y'_1 = a_0 + a_1 x + e_1$$
$$y'_2 = a_0 + a_1 x + e_2$$
$$y'_3 = a_0 + a_1 x + e_3$$

Minimize the sum of errors (차선책)

$e_1 = y_1 - a_0 - a_1 x_1$
즉, $e_1 = 0$ 이 되는 것이 제일 좋은 최적화

$e_2 = y_2 - a_0 - a_1 x_2$
즉, $e_2 = 0$ 이 되는 것이 제일 좋은 최적화

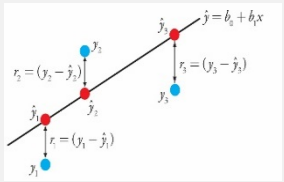
$e_3 = y_3 - a_0 - a_1 x_3$
즉, $e_3 = 0$ 이 되는 것이 제일 좋은 최적화



$$e_1 = y_1 - a_0 - a_1 x_1$$
$$e_2 = y_2 - a_0 - a_1 x_2$$
$$e_3 = y_3 - a_0 - a_1 x_3$$

각각의 모든 요구 조건을 수용해야 하는 상황이지만, 그렇게 하지 못한다. 수용하지 못하는 상황에서 최선을 다하는 방법은 조금씩 양보를 해서 에러의 합이 최소화되도록 하는 것이다.

Y 축 상의 에러



$$minimize\ the\ sum\ of\ error\ (e_1 + e_2 + \cdots + e_n) = min \sum_{i=1}^{n} e_i$$
$$= min \sum_{i=1}^{n} (y_i - a_0 - a_1 x_i)$$

Minimize the sum of errors (차선책)

- e_1 입장에서는 실선 직선보다 점선 직선(파란색)이 더 최적화된 직선이다
- e_2 입장에서는 실선 직선보다 점선 직선(빨강색)이 더 최적화된 직선이다
- e_1, e_2, e_3 각각의 모든 요구 조건을 수용해야 하는 상황이지만, 그렇게 하지 못한다.
- e_1, e_2, e_3 의 모든 요구 조건을 수용하지 못하는 상황에서 최선을 다하는 방법은 e_1, e_2 가 조금씩 양보를 해서 e_1, e_2 입장에서는 절대 에러가 없게 해 달라는 요구를 버리고, 조금의 에러를 수용해야 한다.
- 그리고, 전체적으로 보면, e_1, e_2, e_3 에러의 합이 최소화되도록 하는 것이다.

Least Squares Regression (최소 제곱 회귀)

- **e_1, e_2, e_3 에러의 합이 최소화 되도록 하는 것이다.**
- **최소화를 어떻게 수행할 것인가는 최적화의 문제이다.**
  - Minimize the sum of the squares of the errors → optimization
- **최적화를 어떻게 수행할 것인가?**
  - **두 개의 변수 (Multi variables, a_0, a_1) 가 있을 때, 최대, 최소를 찾는 편미분 (Partial Derivative) 방법 적용이 필요하다.**
    - Find min, max → differentiation
    - Calculate slope (a_1) and intercept (a_0) in order to minimize the sum of error
    - Multi variables (a_0, a_1) → Partial Derivative

Least Error Square Form
최소 에러 제곱 회귀

$$S = min \sum_{i=1}^{n} e_i^2 = min \sum_{i=1}^{n} (y_i - a_0 - a_1 x_i)^2$$

제곱 하는 이유는? 제곱 오차 (Squared Error)

- 에러는 abs(4)=4와 abs(-3)=3과 같이 크기가 에러이다
- 에러의 전체 크기는 7이 되어야 한다
- abs()를 하지 않고, 에러를 더하면 4+(-3)=1이 되고, 에러의 전체 크기를 나타내는 척도가 될 수 없다.
- 그래서 -3 이라는 마이너스 부호를 없애기 위해 (-3)**2, 제곱을 한다

$$S = min \sum_{i=1}^{n} e_i^{\mathbf{2}} = min \sum_{i=1}^{n} (y_i - a_0 - a_1 x_i)^{\mathbf{2}}$$

$$\frac{\partial S}{\partial a_0} = \frac{\partial}{\partial a_0} \sum_{i=1}^{n} (y_i - a_0 - a_1 x_i)^2 = 0$$

$$\frac{\partial S}{\partial a_1} = \frac{\partial}{\partial a_1} \sum_{i=1}^{n} (y_i - a_0 - a_1 x_i)^2 = 0$$

평균 제곱 오차 (Mean Squared Error : MSE)

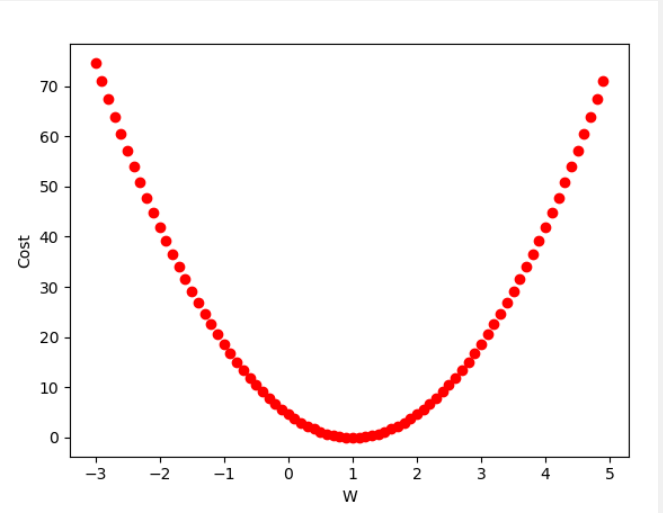
- 가장 많이 사용되는 손실 함수이다.
- 기본적으로 모델의 출력 값과 사용자가 원하는 출력 값 사이의 거리 차이를 오차로 사용한다.
- 그러나 오차를 계산할 때 단순히 거리 차이를 합산하여 평균내게 되면, 거리가 음수로 나왔을 때 합산된 오차가 실제 오차보다 줄어드는 경우가 생긴다.
- 각 거리 차이를 제곱하여 합산한 후에 평균내는 것이다.
  - 전체 데이터의 수를 나누어 1/n을 곱해주어야 한다

손실 함수의 목적

- 머신러닝 모델의 최종적인 목적은 높은 정확도를 끌어내는 매개변수(가중치, 편향)를 찾는 것이다.
- 신경망 학습에서는 최적의 매개변수를 탐색할 때 손실함수의 값을 가능한 한 작게 하는 매개변수 값을 찾는다.
- 이 때, 매개변수의 미분(기울기)을 계산하고, 그 미분 값을 토대로 매개변수 값을 갱신하는 과정을 반복한다.

Least Squares Regression은 Optimization 이다

- $S = min \sum_{i=1}^{n} {e_i}^2$ 의 최소값을 구해야 한다.
- 최소값을 구하는 것은 Optimization이다.
- **2차원** 곡선 방정식 → $min \sum_{i=1}^{n} (y_i - a_0 - a_1 x_i)^2$ 의 **최소값**은 **기울기가 0인 곳에서 발생**한다.
- 즉, $min \sum_{i=1}^{n} (y_i - a_0 - a_1 x_i)^2$ 을 미분해야 한다.
- 그런데, a_0와 a_1두 개의 변수가 있다.
- 편미분을 해야 한다.



Summation 풀어서 미분

$$S = \sum_{i=1}^{n} e_i^2 \rightarrow \frac{\partial S}{\partial a_0} = \frac{\partial}{\partial a_0}(e_1)^2 + \frac{\partial}{\partial a_0}(e_2)^2 + \frac{\partial}{\partial a_0}(e_3)^2 + \cdots$$

$$\frac{\partial S}{\partial a_0} = \frac{\partial}{\partial a_0}(y_1 - a_0 - a_1 x_1)^2 + \frac{\partial}{\partial a_0}(y_2 - a_0 - a_1 x_2)^2 + \frac{\partial}{\partial a_0}(y_3 - a_0 - a_1 x_3)^2 + \cdots$$

$$y = \{f(x)\}^n \qquad y' = n \cdot \{f(x)\}^{n-1} \cdot f'(x)$$

$$\frac{\partial S}{\partial a_0} = 2 \cdot (y_1 - a_0 - a_1 x_1) \cdot (-1) + 2 \cdot (y_2 - a_0 - a_1 x_2) \cdot (-1) + 2 \cdot (y_2 - a_0 - a_1 x_2) \cdot (-1) + \cdots$$

$$\frac{\partial S}{\partial a_0} = (-2) \cdot \sum_{i=1}^{n} (y_i - a_0 - a_1 x_i)$$

$$S = \sum_{i=1}^{n} e_i^2$$

Summation 풀어서 미분하는 것과 Summation 있는 상태로 그대로 미분하는 것의 결과는 같다

$$\frac{\partial S}{\partial a_0} = 2 \cdot \sum_{i=1}^{n} (y_i - a_0 - a_1 x_i)^{2-1} \cdot \frac{\partial (y_i - a_0 - a_1 x_i)}{\partial \boldsymbol{a_0}} = (-2) \cdot \sum_{i=1}^{n} (y_i - a_0 - a_1 x_i)$$

Partial Derivative(편미분) for a_0

$$\frac{\partial S}{\partial a_0} = \frac{\partial}{\partial a_0}\sum_{i=1}^{n}(y_i - a_0 - a_1 x_i)^2 = 0$$

$$y = \{f(x)\}^n$$

$$y' = n \cdot \{f(x)\}^{n-1} \cdot f'(x)$$

$$= 2 \cdot \sum_{i=1}^{n}(y_i - a_0 - a_1 x_i)^{2-1} \cdot \frac{\partial(y_i - a_0 - a_1 x_i)}{\partial \boldsymbol{a_0}}$$

$$= 2 \cdot \sum_{i=1}^{n}(y_i - a_0 - a_1 x_i) \quad \cdot(-1)$$

$$= -2 \cdot \sum_{i=1}^{n}(y_i - a_0 - a_1 x_i) \quad = 0$$

$$= \sum_{i=1}^{n} y_i - \sum_{i=1}^{n} a_0 - \sum_{i=1}^{n} a_1 x_i = 0$$

Least Squares Regression

$$\sum_{i=1}^{n} y_i - \sum_{i=1}^{n} a_0 - \sum_{i=1}^{n} a_1 x_i = 0$$

$$\sum_{i=1}^{n} a_0 = a_0 + a_0 + a_0 + \cdots + a_0 = n a_0$$

$$\sum_{i=1}^{n} y_i - n a_0 - \sum_{i=1}^{n} a_1 x_i = 0$$

$$\boldsymbol{n a_0 + \left(\sum x_i \right) a_1 = \sum y_i} - ①$$

Partial Derivative(편미분) for a_1

$$\frac{\partial S}{\partial a_1} = \frac{\partial}{\partial a_1}\sum_{i=1}^{n}(y_i - a_0 - a_1 x_i)^2 = 0$$

$$y = \{f(x)\}^n$$

$$y' = n \cdot \{f(x)\}^{n-1} \cdot f'(x)$$

$$= 2 \cdot \sum_{i=1}^{n}(y_i - a_0 - a_1 x_i)^{2-1} \cdot \frac{\partial(y_i - a_0 - a_1 x_i)}{\partial \boldsymbol{a_1}}$$

$$= -2x_i \cdot \sum_{i=1}^{n}(y_i - a_0 - a_1 x_i) \quad = 0$$

$$= \sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} a_0 x_i - \sum_{i=1}^{n} a_1 {x_i}^2 = 0$$

Least Squares Regression

$$\sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} a_0 x_i - \sum_{i=1}^{n} a_1 {x_i}^2 = 0$$

$$\sum_{i=1}^{n} x_i \, a_0 + \sum_{i=1}^{n} {x_i}^2 \, a_1 = \sum_{i=1}^{n} x_i y_i$$

$$\left(\sum \boldsymbol{x_i}\right) \boldsymbol{a_0} + \left(\sum \boldsymbol{{x_i}^2}\right) \boldsymbol{a_1} = \sum \boldsymbol{x_i y_i} - ②$$

Least Squares Regression

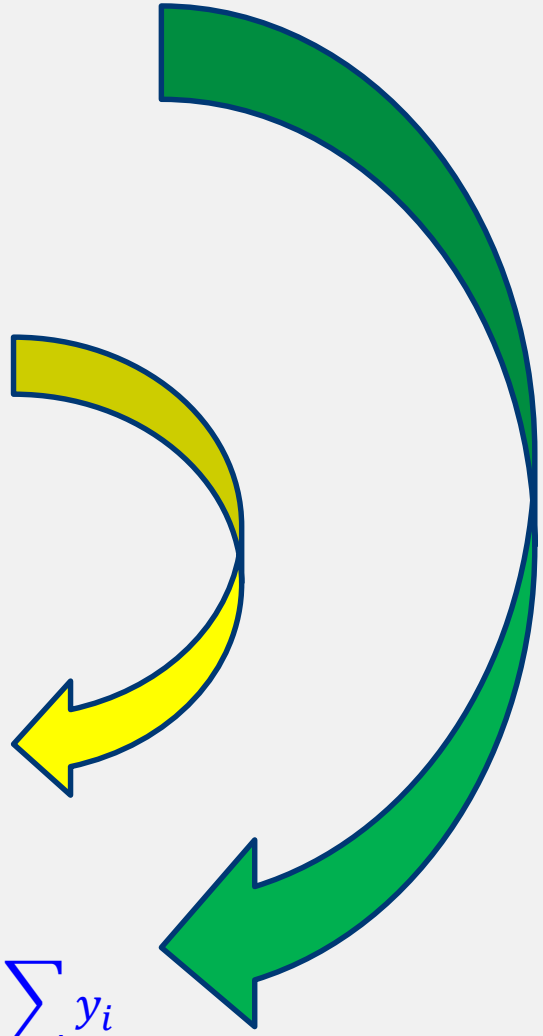
$$n\boldsymbol{a_0} + \left(\sum x_i\right)\boldsymbol{a_1} = \sum y_i \quad ①$$

$$\left(\sum x_i\right)\boldsymbol{a_0} + \left(\sum {x_i}^2\right)\boldsymbol{a_1} = \sum x_i y_i \quad ②$$

$$② \times \boldsymbol{n} - ① \times \left(\sum \boldsymbol{x_i}\right)$$

$$③ \quad \boldsymbol{n}\left(\sum \boldsymbol{x_i}\right)a_0 + \boldsymbol{n}\left(\sum {x_i}^2\right)a_1 = \boldsymbol{n}\sum x_i y_i$$

$$④ \quad n\left(\sum \boldsymbol{x_i}\right)a_0 + \left(\sum \boldsymbol{x_i}\right)\left(\sum x_i\right)a_1 = \sum \boldsymbol{x_i}\sum y_i$$



Slope (a_1) 구하기

모델점 (x , y'= y) 이 실제점 (x , y) 을 대체하는 직선을 찾았다는 가정하에 시작했다.
즉, 직선 위의 모델점 (y') = 실제 곡선 위의 점 (y) 이다.

$$y' = a_0 + a_1 x + e$$

$$n\left(\sum x_i\right)a_0 + n\left(\sum {x_i}^2\right)a_1 = n\sum x_i y_i \quad ③$$

$$n\left(\sum x_i\right)a_0 + \left(\sum x_i\right)^2 a_1 = \sum x_i \sum y_i \quad ④$$

③ − ④ $$\left(n\left(\sum {x_i}^2\right) - \left(\sum x_i\right)^2\right)a_1 = n\sum x_i y_i - \sum x_i \sum y_i$$

$$a_1 = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{n(\sum {x_i}^2) - (\sum x_i)^2}$$

$e_1 = y_1 - a_0 - a_1 x_1$
즉, $e_1 = 0$ 이 되는 것이 제일 좋은 최적화

$e_2 = y_2 - a_0 - a_1 x_2$
즉, $e_2 = 0$ 이 되는 것이 제일 좋은 최적화

$e_3 = y_3 - a_0 - a_1 x_3$
즉, $e_3 = 0$ 이 되는 것이 제일 좋은 최적화

$e_1 = y'_1 - a_0 - a_1 x_1$

$e_2 = y'_2 - a_0 - a_1 x_2$

$e_3 = y'_3 - a_0 - a_1 x_3$

$e_1 = y_1 - a_0 - a_1 x_1$

$e_2 = y_2 - a_0 - a_1 x_2$

$e_3 = y_3 - a_0 - a_1 x_3$

직선은 하나지만 각기 다른 요구 조건을 수용해야 하고, 수용할 수 있다고 가정한다

각각의 모든 요구 조건을 수용해야 하는 상황이지만, 그렇게 하지 못한다. 수용하지 못하는 상황에서 최선을 다하는 방법은 조금씩 양보를 해서 에러의 합이 최소화되도록 하는 것이다.

Intercept(a_0) 구하기

$$y' = a_0 + a_1 x + e$$

$$na_0 + \left(\sum x_i\right) a_1 = \sum y_i \quad ①$$

$$na_0 = \sum y_i - \left(\sum x_i\right) a_1$$

모델화 된 점 y' 가 모델화가 너무 잘 되어 에러를 가지고 있지 않다면, $y' = y$ 가 되는 것이 곡선 접합의 전략임
에러가 0 이 되는 직선을 찾아야 한다.
그 직선은 $y' = y$ 이 되는 직선이며 이것이 최적화

$$a_1 = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{n(\sum {x_i}^2) - (\sum x_i)^2}$$

$$a_0 = \frac{\sum y_i}{n} - \frac{\sum x_i}{n} a_1$$

$$\boldsymbol{a_0 = \bar{y} - \bar{x} a_1}$$

$$\bar{y} = \frac{\sum y_i}{n}$$
$$\bar{x} = \frac{\sum x_i}{n}$$

곡선접합의 전략 $y' = y$

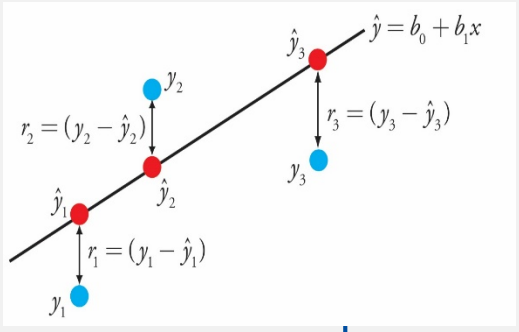
- 모델점 $(x\ ,\ y' = y)$ 이 실제점 $(x\ ,\ y)$ 을 대체하는 직선을 찾았다는 가정하에 시작했다.
- **모델화 된 점 y' 가 모델화가 너무 잘 되어 에러를 가지고 있지 않다면, $y' = y$ 가 되는 것이 곡선 접합의 전략임**
- **즉, 직선 위의 모델점 (y')은 실제 곡선 위의 점 (y) 이다.**
- **직선은 하나지만 각기 다른 요구 조건을 수용해야 하고, 수용할 수 있다고 가정한다**
- **각각의 모든 요구 조건을 수용해야 하는 상황이지만, 그렇게 하지 못한다.**
- **수용하지 못하는 상황에서 최선을 다하는 방법은 조금씩 양보를 해서 에러의 합이 최소화되도록 하는 것이다**

곡선접합의 전략 $y' = y$

- 에러를 최소화하도록, 기울기 a_1 과 절편 a_0 가 구해졌다.
- 에러를 최소화하는 밑거름은 기울기 a_1 과 절편 a_0에 의해 구해지는 직선의 $y' = y$ 가 되는 것이 선형 회귀의 묘미,
- 모델점 y'와 실제점 y 사이에 에러가 실존하지만 유도 시에는 에러가 없다고 보는 것이 선형 회귀의 묘미
- 기울기 a_1 과 절편 a_0은 에러(error)를 $0(zero)$으로 만드는 수식 유도에 의해 구해진 기울기와 절편임으로 기울기값 a_1 과 절편값 a_0를 구하기 위해 사용되는 x 와 y 는 실제점의 x 와 y 값이 사용될 수 있음.

$$a_1 = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{n(\sum x_i{}^2) - (\sum x_i)^2} \qquad a_0 = \bar{y} - \bar{x}a_1$$

곡선접합의 전략 Curve-fitting Strategy



$y' = a_0 + a_1 x + e$

Modeled point
(직선 위의 점)
(x_3, y'_3)

Real point
(x_2, y_2)

Error e_3

Real point
(x_1, y_1)

Error e_2

(x_2, y'_2)

Real point
(x_3, y_3)

Error e_1

(x_1, y'_1) Modeled point

intercept a_0

a_1

6 4 2 0 2 4 6 8 10 x

기준점 x는 동일
기준점 x 에 대해
실제점은 y
모델화 된 점은 y'

파란 색 직선 위의 점, 모델화 된 점 y' 가 모델화가 너무 잘 되어 에러를 가지고 있지 않다면, $y' = y$ 가 되는 것이 곡선 접합의 전략임
에러가 0 이 되는 직선을 찾아야 한다. 그 직선은 $y' = y$ 이 되는 직선이며 이것이 최적화

$$a_1 = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{n(\sum x_i{}^2) - (\sum x_i)^2} \qquad a_0 = \bar{y} - \bar{x} a_1$$

Linear Regression의 묘미 (수치해석은 근사치의 학문)

에러를 최소화하도록, 기울기 a_1 과 절편 a_0 가 구해졌다. 에러를 최소화하는 밑거름은 기울기 a_1 과 절편 a_0에 의해 구해지는 직선의 $y' = y$ 가 되는 것이 선형 회귀의 묘미, 모델점 y'와 실제점 y 사이에 에러가 실존하지만 유도 시에는 에러가 없다고 보는 것이 선형 회귀의 묘미

기울기 a_1 과 절편 a_0은 에러를 $\mathbf{0}(\boldsymbol{zero})$으로 만드는 수식 유도에 의해 구해진 기울기와 절편임으로 기울기값 a_1과 절편값 a_0를 구하기 위해 사용되는 x 와 y 는 실제점의 x 와 y 값이 사용될 수 있음.

$$a_1 = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{n(\sum {x_i}^2) - (\sum x_i)^2}$$
$$a_0 = \bar{y} - \bar{x}a_1$$

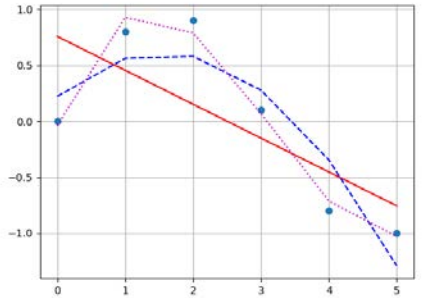
```
import numpy as np
import matplotlib.pyplot as plt
x = np.arange(10, 90, 10.)
y = np.array([25, 70, 380, 550, 610, 1220, 830, 1450])
plt.figure(1)
plt.plot(x, y, 'ro-') # 실제점 y
plt.grid()
```

Python Coding for Linear Regression

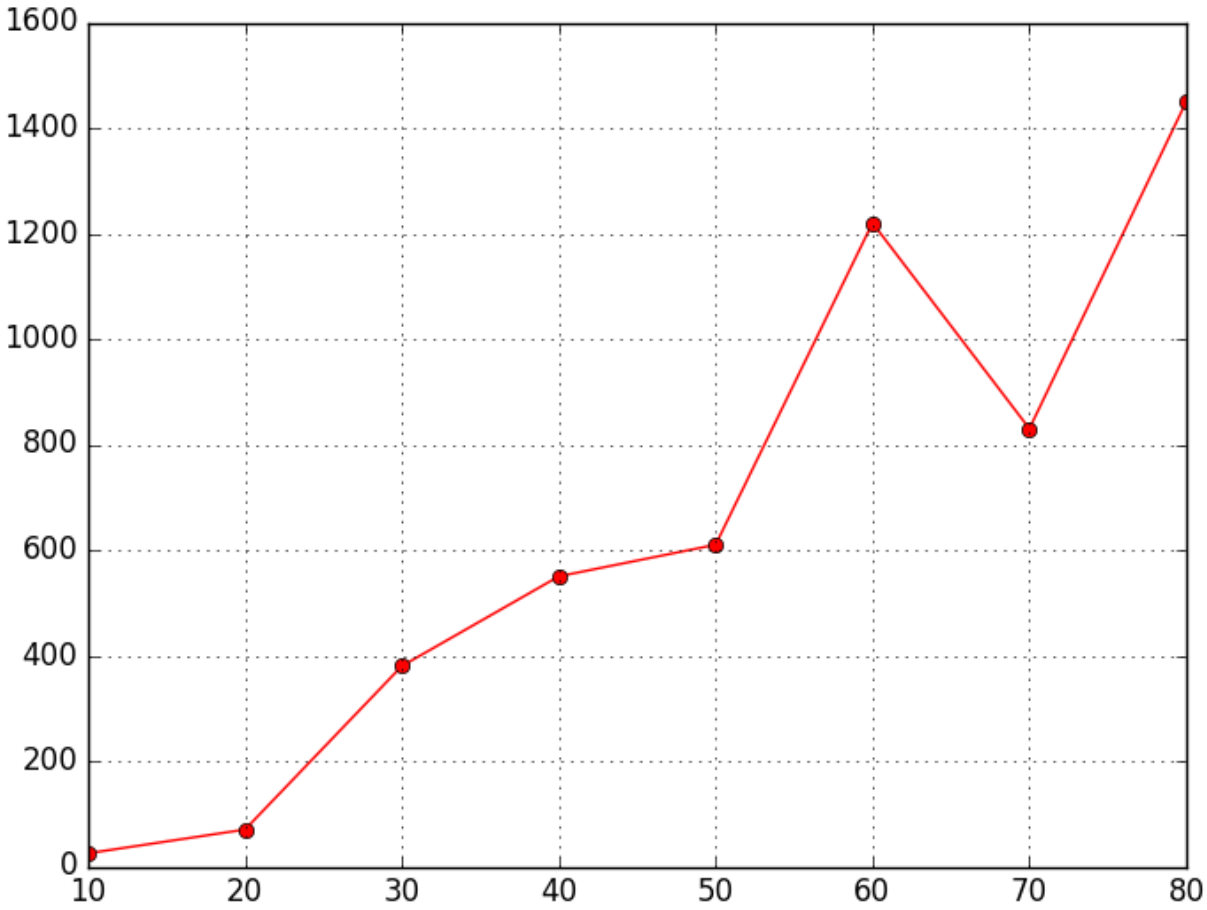
```
xsum=np.sum(x)   # 360.0
ysum=np.sum(y)   # 5135

xysum=sum(x*y)

n=np.size(x)

xavg=xsum/n   # 45.0
yavg=ysum/n # 641.875

a1=(n*xysum-xsum*ysum)/(n*sum(x**2)-xsum**2)
# 19.470238095238095
a0= yavg-xavg*a1
# -234.28571428571422
y1=a1*x+a0  # 가짜점 y1, 모델점, 예측치

#array([-39.58333333, 155.11904762, 349.82142857,
544.52380952,
#       739.22619048, 933.92857143,
1128.63095238, 1323.33333333])

plt.figure(2)
plt.plot(x, y, 'ro-', x, y1, 'b*-')
plt.grid()
```

에러를 최소화하도록, 기울기 a_1 과 절편 a_0 가 구해졌다. 에러를 최소화하는 밑거름은 기울기 a_1 과 절편 a_0에 의해 구해지는 직선의 $y' = y$ 가 되는 것이 선형 회귀의 묘미, 모델점 y'와 실제점 y 사이에 에러가 실존하지만 유도 시에는 에러가 없다고 보는 것이 선형 회귀의 묘미

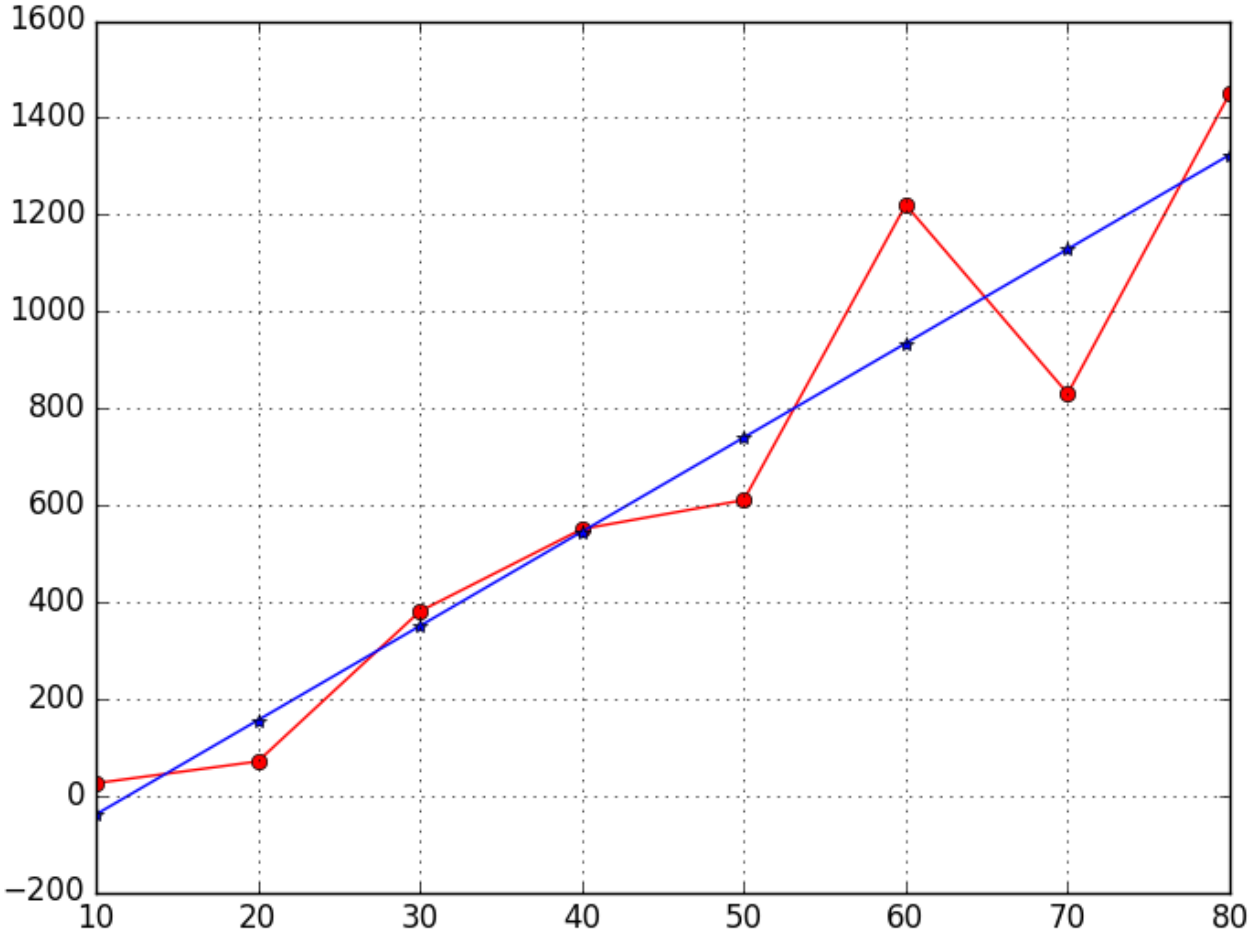
실제 np.polyval(p3,x)에 의해 계산된 계수값들은 거의 실제 곡선과 유사한 곡선을 만들어 낸다 (곡선 접합)



Before Regression



After Regression



Linear Regression Example

Table 1. Example data.

X	Y
1.00	1.00
2.00	2.00
3.00	1.30
4.00	3.75
5.00	2.25

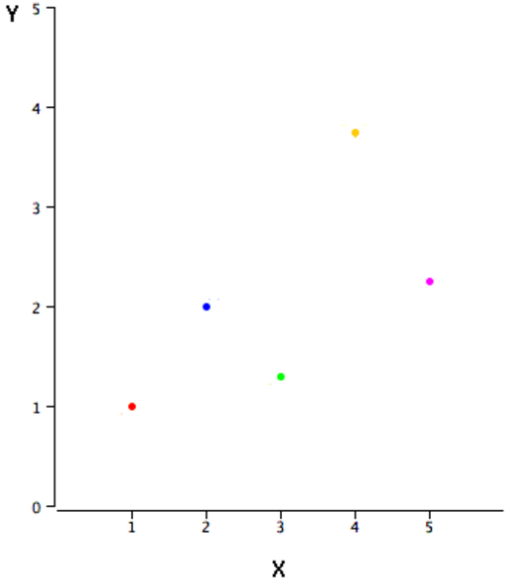


Figure 1. A scatter plot of the example data.

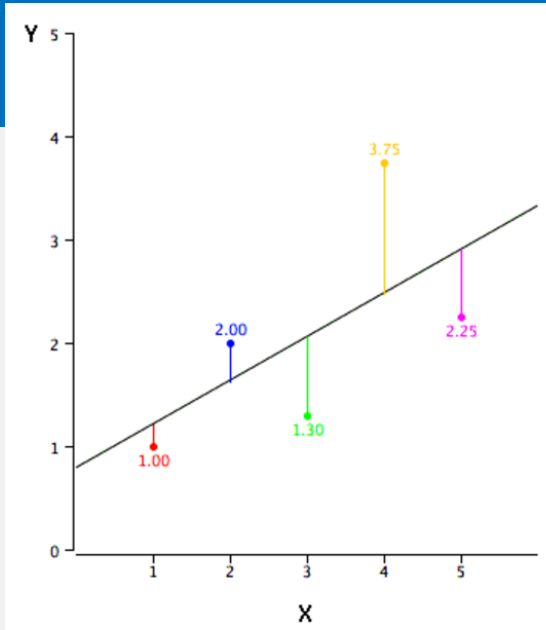


Figure 2. A scatter plot of the example data. The black line consists of the predictions, the points are the actual data, and the vertical lines between the points and the black line represent errors of prediction.

Table 2. Example data.

X	Y	Y'	Y-Y'	$(Y-Y')^2$
1.00	1.00	1.210	-0.210	0.044
2.00	2.00	1.635	0.365	0.133
3.00	1.30	2.060	-0.760	0.578
4.00	3.75	2.485	1.265	1.600
5.00	2.25	2.910	-0.660	0.436

The formula for a regression line is

$$Y' = bX + A$$

where Y' is the predicted score, b is the slope of the line, and A is the Y intercept.

The equation for the line in Figure 2 is

$$Y' = 0.425X + 0.785$$

For X = 1,

$$Y' = (0.425)(1) + 0.785 = 1.21.$$

For X = 2,

$$Y' = (0.425)(2) + 0.785 = 1.64.$$

Linear Least-Squares Regression (선형 최소 제곱 회귀법)

- **Linear least-squares regression is a method to determine the "best" coefficients in a linear model for given data set.**
- **"Best" for least-squares regression means minimizing the sum of the squares of the *estimate* residuals.** For a straight line model, this gives:

$$S_r = \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} (y_i - a_0 - a_1 x_i)^2$$

- This method will yield **a unique line** for a given set of data.

Least-Squares Fit of a Straight Line

- Using the model:

$$y = a_0 + a_1 x$$

the slope and intercept producing **the best fit** can be found using:

$$a_1 = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{n\sum x_i^2 - \left(\sum x_i\right)^2}$$

$$a_0 = \bar{y} - a_1 \bar{x}$$

Example

	V (m/s)	F (N)		
i	x_i	y_i	$(x_i)^2$	$x_i y_i$
1	10	25	100	250
2	20	70	400	1400
3	30	380	900	11400
4	40	550	1600	22000
5	50	610	2500	30500
6	60	1220	3600	73200
7	70	830	4900	58100
8	80	1450	6400	116000
Σ	360	5135	20400	312850

$$a_1 = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{n\sum x_i^2 - \left(\sum x_i\right)^2} = \frac{8(312850) - (360)(5135)}{8(20400) - (360)^2} = 19.47024$$

$$a_0 = \bar{y} - a_1 \bar{x} = 641.875 - 19.47024(45) = -234.2857$$

$$F_{est} = -234.2857 + 19.47024v$$