



SESSION # 7 회귀분석

Team 5_ 정재근, 빈다은, 고병욱

Date _ 2018.05.12

목차

1. 회귀분석의 기본(단순,다중)

- 선형 회귀의 의미
- 기본 가정

2. 로지스틱 회귀분석

- 로지스틱 회귀분석의 의미
- 로지스틱 함수의 도출

2+ 회귀분석의 심화

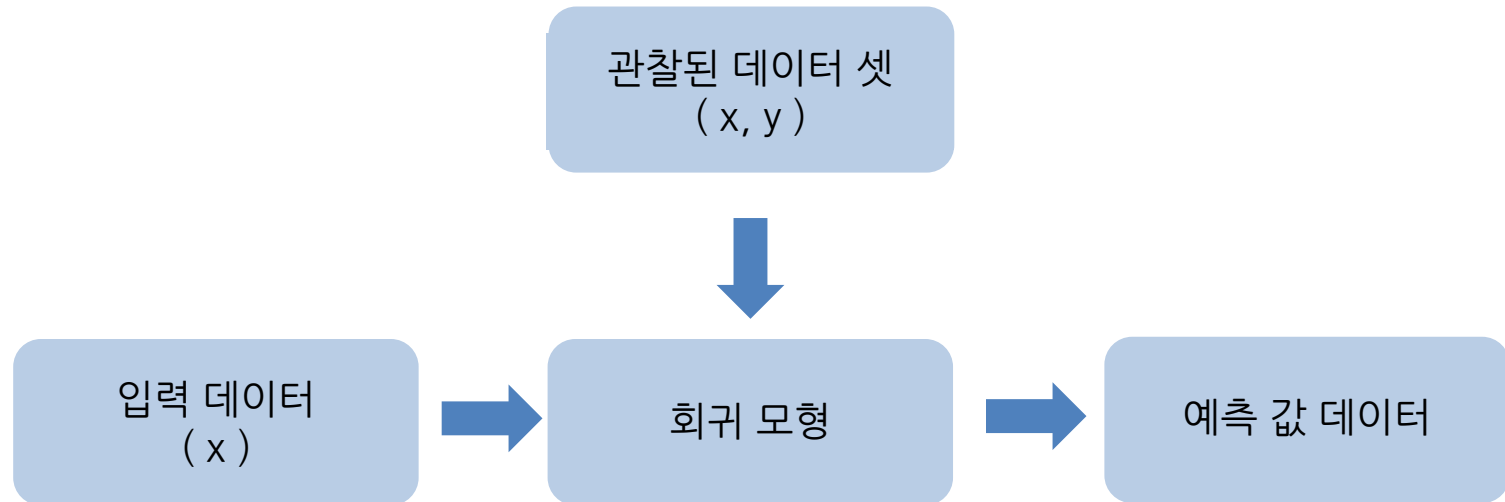
- 더미변수
- 상호작용항
- 위계적 회귀분석

3. 회귀분석 추정 및 검정

4. Code & Quest

O1 회귀분석의 기본

회귀분석이란?



종속변수에 대해 하나 혹은 그 이상의 독립변수가 미치는 영향을 추정하는 통계기법
변수들 간의 함수관계를 **회귀 모형**(회귀 식)으로 표현
과거 데이터를 이용해 미래에 발생할 일을 **예측**할 수 있음

01 회귀분석의 기본

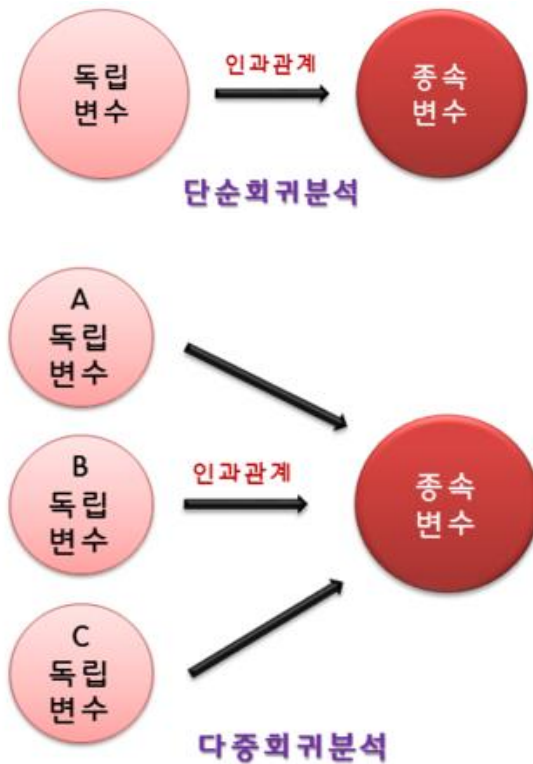
회귀분석의 기본모델

① 단순회귀분석

두 변수 간의 인과관계를 조사하는 방법

② 다중회귀분석

두 개 이상의 독립변수들과 하나의 종속변수의 관계를 분석하는 기법으로
단순 회귀 분석을 확장한 것



O1 회귀분석의 기본

회귀분석의 기본모델

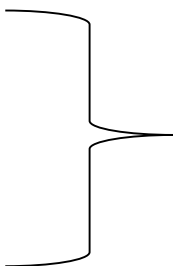
엄밀히 말하면 인과 관계 분석이 아니다.

학문마다 다르지만 통계학에서 말하는 인과관계

1. 상관관계
2. 시간의 전후
3. 제 3의 변수

01 회귀분석의 기본

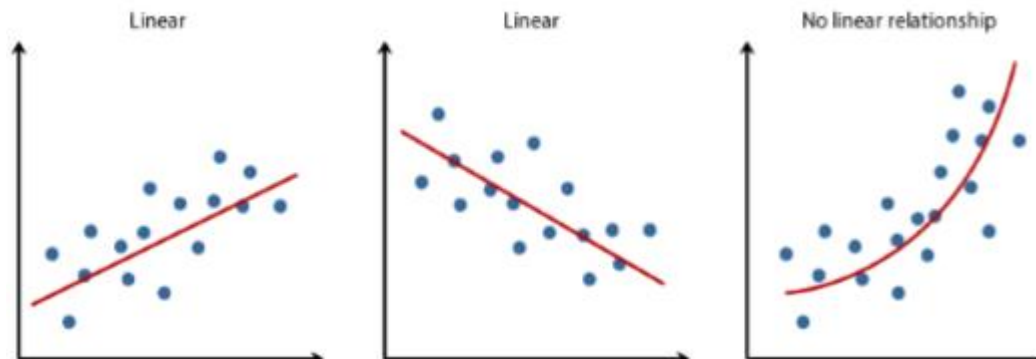
선형 회귀 기본 가정

1. 선형성
 2. 비상관성
 3. 정규성
 4. 등분산성
 5. 독립성
- 
- 오차항

O1 회귀분석의 기본

선형 회귀 기본 가정

1. 선형성



독립변수(x)와 종속변수(y)는 서로 선형적 관계가 있어야 한다.
즉, x 와 y 는 비선형 관계에 있어서는 안 된다.

01 회귀분석의 기본

선형 회귀 기본 가정

2. 비상관성

독립변수(x)들이 서로 상관관계에 있어서는 안 된다.

(다중 공선성 확인이 필요함)

*다중 공선성이란 (multicollinearity)?

- p개의 독립변수(x_1, x_2, \dots, x_p)와 종속변수(y)로 데이터가 구성되어 있을 때 독립변수들 사이에 상관관계가 크게 존재하는 상태를 의미한다.

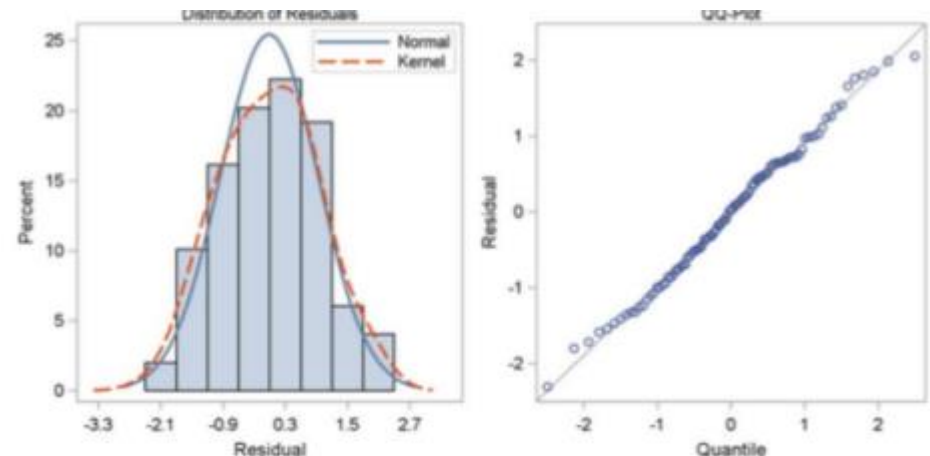
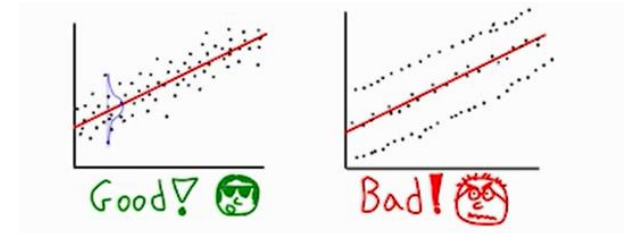
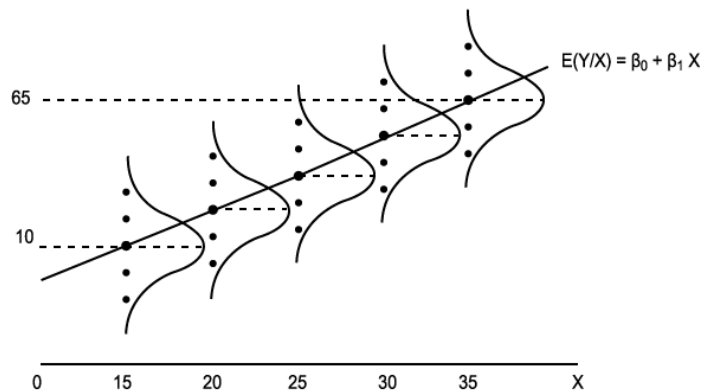
VIF 즉, 분산팽창계수가 10 이상 일 때, 다중공선성이 존재한다고 판단한다.

O1 회귀분석의 기본

선형 회귀 기본 가정

3. 정규성

Y



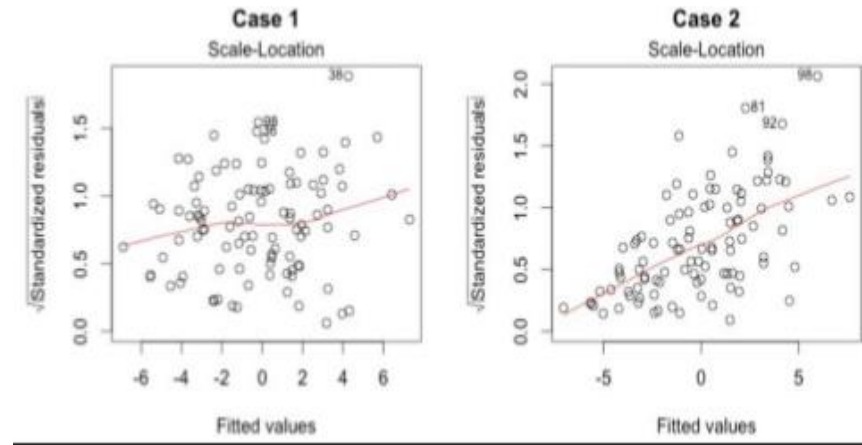
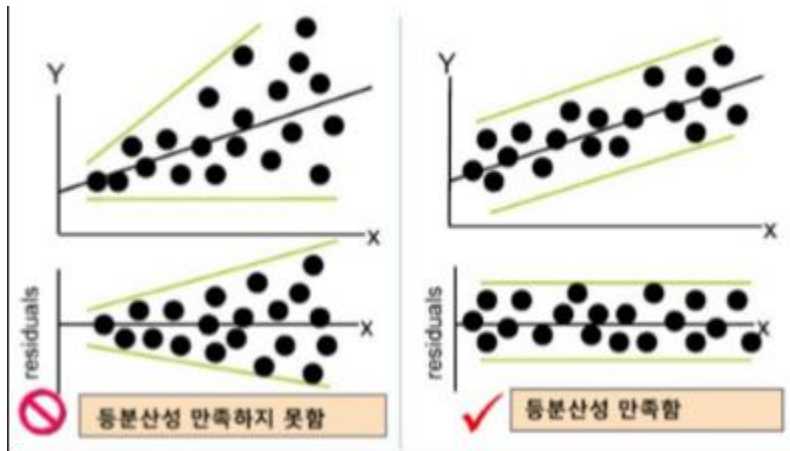
- 오차항이 평균 0, 분산 시그마 제곱인 정규분포를 따른다.
- 정규성을 만족하면,
잔차항의 히스토그램 좌우대칭, 종모양(정규분포)이며 QQ-plot이 직선모양이다.

사진출처: http://www.ssaicstat.com/base/cs/cs_05.php?com_board_basic=read_form&com_board_id=293&topmenu=58left=58&com_board_search_code=8com_board_search_value1=8com_board_search_value2=8com_board_page=1&

O1 회귀분석의 기본

선형 회귀 기본 가정

4. 등분산성

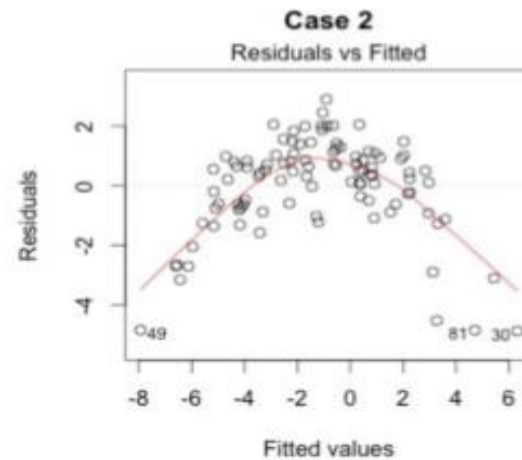
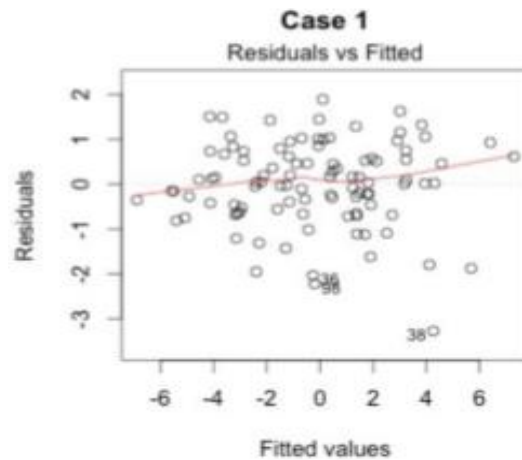


오차항의 분산이 동일하다

01 회귀분석의 기본

선형 회귀 기본 가정

5. 독립성



- 오차항이 서로 독립이다
 - 잔차산점도에서 잔차가 중심선 주위에서 랜덤 패턴으로 변동해야 한다.
 - 가정이 위배되는 경우 자기상관이 서로 존재한다
- 자기상관이 존재하는 경우에는 잔차들 간에 어떠한 함수관계가 존재하게 된다.

01 회귀분석의 기본

다중회귀식의 해석

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	24.7851	1.8309	13.54	<2e-16	***
grades\$study_hour	7.4865	0.3112	24.06	<2e-16	***

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	60.958	4.347	14.021	<2e-16	***
grades\$Private.lesson	5.762	6.086	0.947	0.349	

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	21.9975	1.7914	12.279	4.03e-16	***
grades\$study_hour	7.4805	0.2762	27.083	< 2e-16	***
grades\$Private.lesson	5.5248	1.4946	3.697	0.00058	***

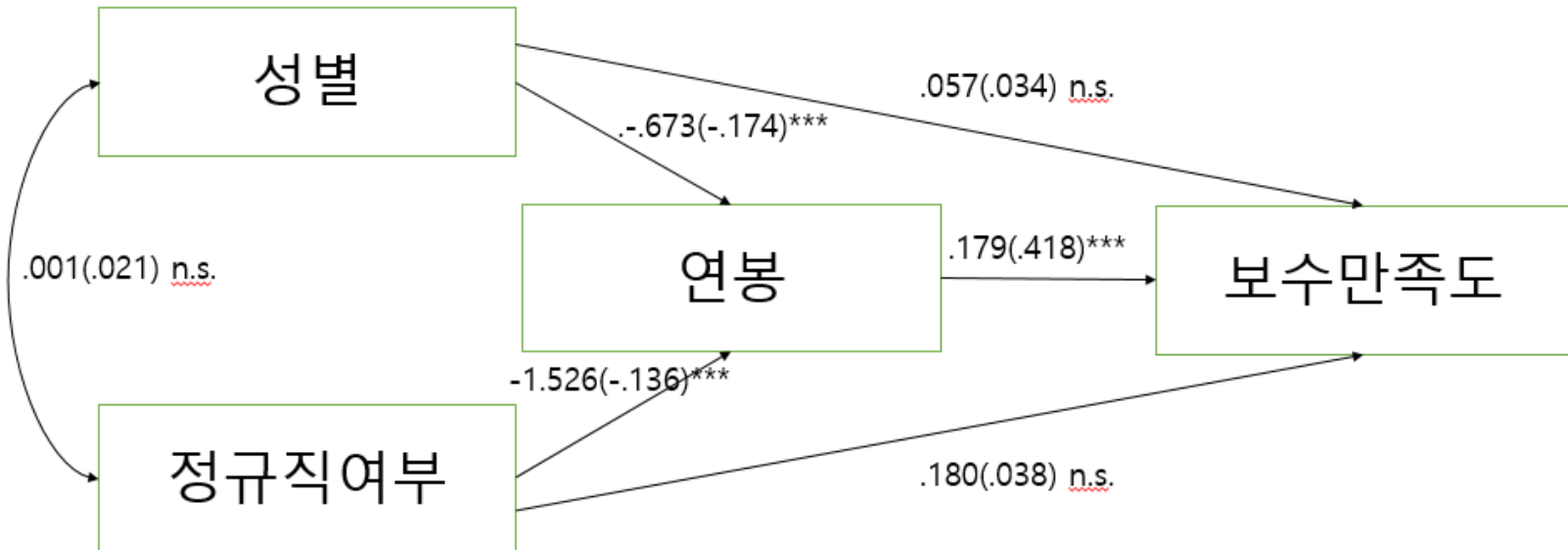
단순과 다중의 차이

공부시간이 통제 되었을 때 과외 유무는 영향을 준다.

01 회귀분석의 기본

경로 분석은 단점을 보완해 준다.

AMOS 경로분석



참고1. 비표준화계수(표준화계수)

참고2. * $p < .05$; ** $p < .01$; *** $p < .001$

01 회귀분석의 기본

회귀분석의 기본 식

단순 : $Y = \alpha_0 + \alpha_1 X_1 + \varepsilon$

다중 : $Y = \alpha_0 + \alpha_1 X_1 + \alpha_2 X_2 + \varepsilon$

$$h_{\theta}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4$$

e.g. $h_{\theta}(x) = \underline{80} + \underline{0.1}x_1 + \underline{0.01}x_2 + \underline{3}x_3 - \underline{2}x_4$

\uparrow \uparrow \uparrow \uparrow
 size bedrooms floors age

Multiple features (variables).

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

Notation:

→ 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.

$$x^{(2)} = \begin{bmatrix} 1416 \\ 3 \\ 2 \\ 40 \end{bmatrix}$$

$$x_3^{(2)} = 2$$

02 로지스틱 회귀분석

영향을 주는 변수
(독립 변수)

영향을 받는 변수
(종속 변수)

분석 방법

연속형 자료

선형관계 파악

단순 회귀분석

다중 회귀분석

연속형 자료

범주형 자료

분류(Classification)

로지스틱 회귀분석

SVM

02 로지스틱 회귀분석

로지스틱 회귀분석이란?

어떤 사건이 발생하는지 (예를 들어, 구매를 하는지 안 하는지)를 직접 예측하는 것이 아니라, 그 사건이 발생할 확률을 예측한다.

따라서 종속 변수 값은 0과 1사이의 값을 갖는다. (일반적으로 이항, 다항도 존재함)

즉, 확률이 0.5보다 크면 그 사건이 일어나며, 0.5보다 작으면 그 사건이 일어나지 않는 것으로 예측한다.

02 로지스틱 회귀분석

선형회귀 분석  로지스틱 회귀분석

1. 선형 회귀식

$$y = ax + b \quad (\text{inf} \leq x, y \leq \text{inf})$$

2. Y를 확률값 p로 바꾼다

$$P = ax + b \quad (0 \leq P \leq 1)$$

* 좌우항의 범위가 맞지 않으므로 P를 변형시켜 좌항의 범위가 무한대로 되도록 한다.

02 로지스틱 회귀분석

3. P대신 Odds를 넣어보자

* Odds란?

$$\frac{\text{성공확률}}{\text{실패확률}} = \frac{\text{성공확률}}{1 - \text{성공확률}} = \frac{P}{1 - P}$$

$$\text{Odds} = \frac{P}{1 - P} = ax + b$$

* $0 < \text{Odds} < \infty$ 값을 가지므로 좌우식의 범위가 여전히 맞지 않다.

4. 따라서, Odds에 로그를 씌어준다. ($-\infty < \text{좌우식} < \infty$)

$$\log_e(\text{Odds}) = \log_e\left(\frac{P}{1 - P}\right) = ax + b$$

02 로지스틱 회귀분석

5. 선형 회귀식을 구한다.

Y는 log Odds이지만 이를 무시하고 최소 제곱법으로 $y = ax + b$ 의 선형 회귀식의 회기 계수 a와 y절편 b를 구한다.

6. P에 관하여 수식을 정리한다.

최종적으로 구하는 것은 P이므로 P에 관하여 수식을 정리한다.

7. 로그 Odds에 관한 선형관계식

$$\log_e\left(\frac{P}{1-P}\right) = ax + b$$

02 로지스틱 회귀분석

8. e를 씌운다.

$$e^{\log_a(\frac{P}{1-P})} = e^{ax + b}$$

$$\frac{P}{1-P} = e^{ax + b}$$

9. 역수를 취한다.

$$\frac{1-P}{P} = \frac{1}{e^{ax + b}}$$

$$\frac{1}{P} - 1 = \frac{1}{e^{ax + b}}$$

02 로지스틱 회귀분석

10. +1을 한다.

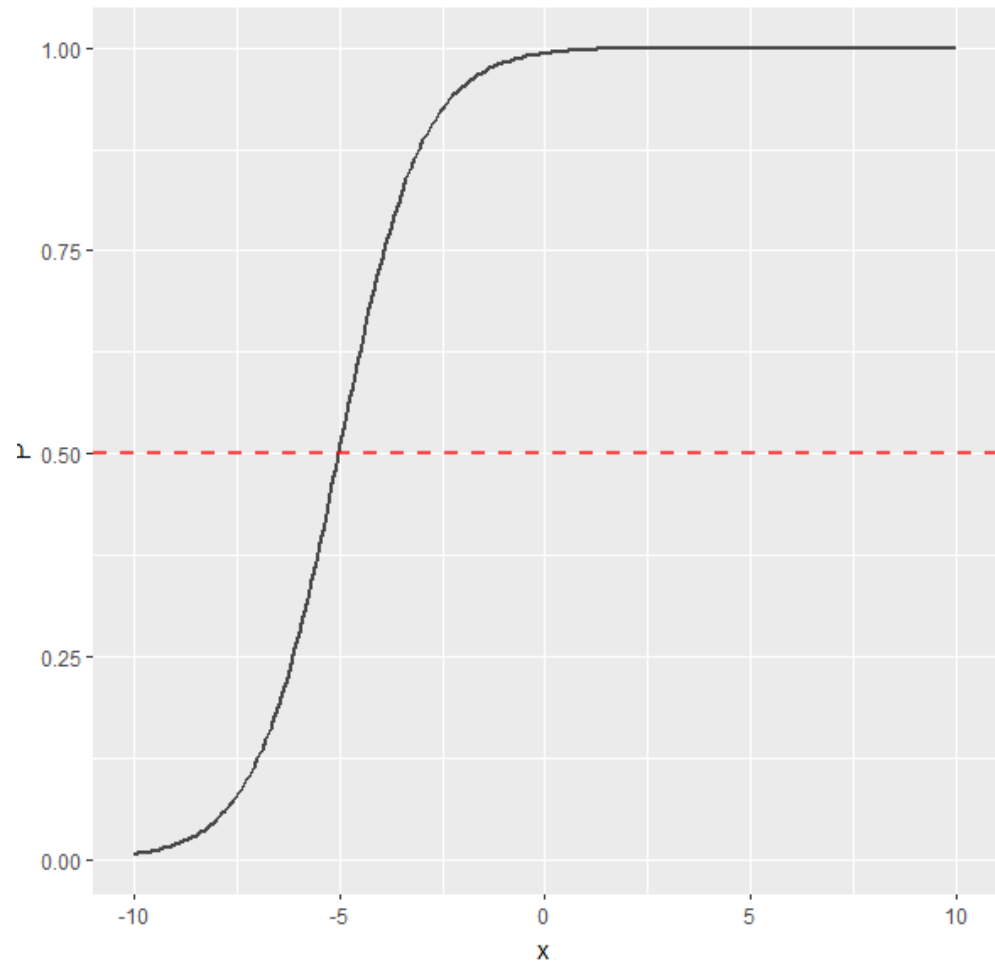
$$\frac{1}{P} = \frac{1}{e^{ax+b}} + 1$$

$$\frac{1}{P} = \frac{1 + e^{ax+b}}{e^{ax+b}}$$

11. 다시 역수를 취한다.

$$P = \frac{e^{ax+b}}{1 + e^{ax+b}}$$

< 로지스틱 곡선 >



임계치 = 0.5

02 로지스틱 회귀분석

< 최적의 임계치 찾기 >

- 일반적으로 $p = 0.5$ 기준으로 0.5보다 크면 A로 분류하고 작으면 B로 분류한다. 이 때, 0.5라는 수치를 임계치라고 부른다.
- 하지만, 이 임계치는 항상 0.5로 정해져 있는 것은 아니고 더 잘 분류할 수 있는 임계치를 분석가가 찾아야 한다.
- 최적의 임계치는 항상 0.5가 아니라 data에 따라 달라질 수 있다.

2+ 회귀분석의 심화

■ 모조 변인(dummy variable) 회귀

범주화된 변인을 변인으로 사용하기.

예) 인종, 지역 등

1 = 강북 2 = 강동 3 = 강서 4 = 강남

기준을 잡고 0과1로 분류

	강북변수	강동변수	강서변수	강남변수(X)
강남	0	0	0	0
강북	1	0	0	0
강동	0	1	0	0
강서	0	0	1	0

2+ 회귀분석의 심화

■ 모조 변인(dummy variable) 회귀

N = 강북
E = 강동
W = 강서
기준 = 강남

지역에 따른 소득 회귀식을 만들어보자

$$Y = 45.9 - 0.6 N - 5.4 E - 4.7 W$$

$$\text{강남} = 45.9 - 0.6 (0) - 5.4 (0) - 4.7 (0) = \mathbf{45.9}$$

$$\text{강북} = 45.9 - \mathbf{0.6} (1) - 5.4 (0) - 4.7 (0) = 45.3$$

$$\text{강동} = 45.9 - 0.6 (0) - \mathbf{5.4} (1) - 4.7 (0) = 40.5$$

$$\text{강서} = 45.9 - 0.6 (0) - 5.4 (0) - \mathbf{4.7} (1) = 41.2$$

2+ 회귀분석의 심화

상호작용 항

상호작용 효과 : 독립변수 X1 와 X2가
독립적으로 갖는 각각의 영향력 이상의 효과

$$Y = a + b_1X_1 + b_2X_2 + \mathbf{b_3X_1X_2}$$

2+ 회귀분석의 심화

상호작용 항

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	21.9975	1.7914	12.279	4.03e-16	***
grades\$study_hour	7.4805	0.2762	27.083	< 2e-16	***
grades\$Private.lesson	5.5248	1.4946	3.697	0.00058	***

Coefficients:

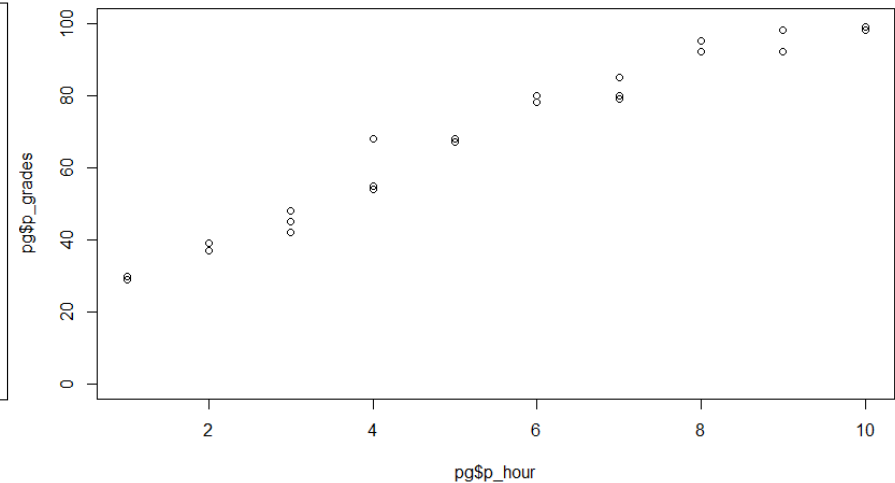
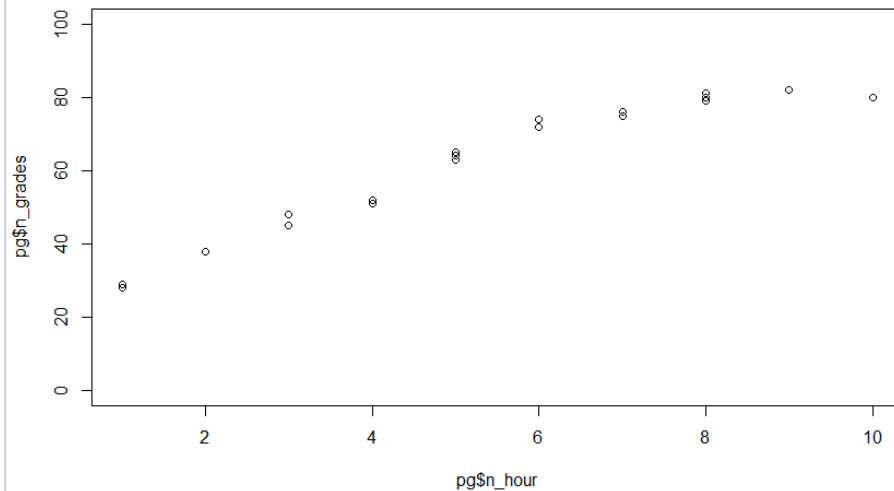
	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	26.613	2.168	12.277	5.79e-16	***
grades\$study_hour	6.594	0.372	17.727	< 2e-16	***
grades\$Private.lesson	-3.003	2.968	-1.012	0.31706	
grades\$interaction	1.633	0.505	3.233	0.00229	**

상호작용항

성적에 대한 공부시간의 효과는 과외 유무에 따라 달라진다.
절편을 바꾸느냐 기울기를 바꾸느냐의 차이

2+ 회귀분석의 심화

상호작용 항



즉 공부 못하는 아이들은 과외로 큰 차이를 만들지 못하지만
공부 잘하는 애들은 과외 유무가 큰 차이를 만든다.

2+ 회귀분석의 심화

위계적 회귀분석

삶의 만족에 영향을 미치는 요소: 위계적 회귀모델

	모델1 b(s.e)	모델2 b(s.e)	모델3 b(s.e)	모델4 b(s.e)
통제변인				
연령	-0.006 (0.014)	-0.000 (0.013)	0.022 (0.015)	0.013 (0.013)
배우자	0.342 (0.249)	0.269 (0.235)	0.101 (0.237)	-0.161 (0.212)
가족과 동거	0.554* (0.251)	0.561** (0.237)	0.686** (0.248)	0.584** (0.222)
질병 및 장애				
만성질환		-0.087 (0.056)	-0.088 (0.056)	-0.071 (0.050)
인지건강		0.513*** (0.110)	0.464*** (0.114)	0.458*** (0.101)
신체 · 인지기능				
IADL			0.069 (0.069)	-0.027 (0.061)
ADL			0.034 (0.039)	-0.014 (0.035)
교육			0.091** (0.030)	0.037* (0.027)
사회활동참여				
직업활동				0.664* (0.388)
교육·복지프로그램				-0.022 (0.094)
종교활동				0.130** (0.050)
사회적 소외				-0.075*** (0.008)
F test	F(3, 306)=5.06**	F(5, 304)=11.50***	F(8, 301)=9.04***	F(12, 297)=15.20***
R square	0.0472	0.1591	0.1937	0.3805

*p<.05; **p<.01; ***p<.001

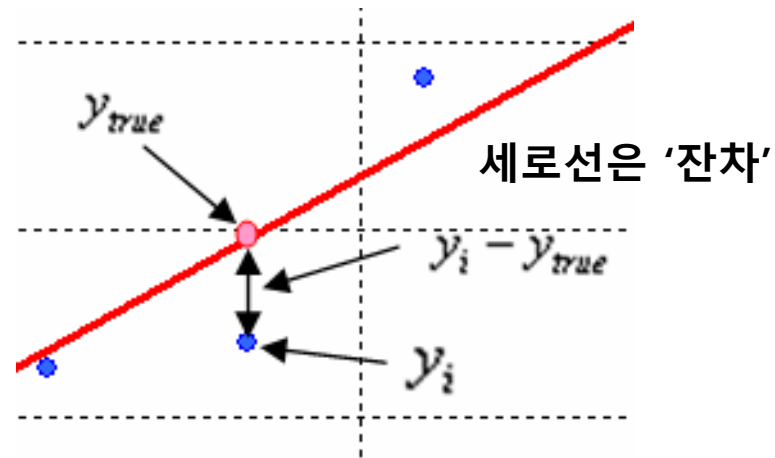
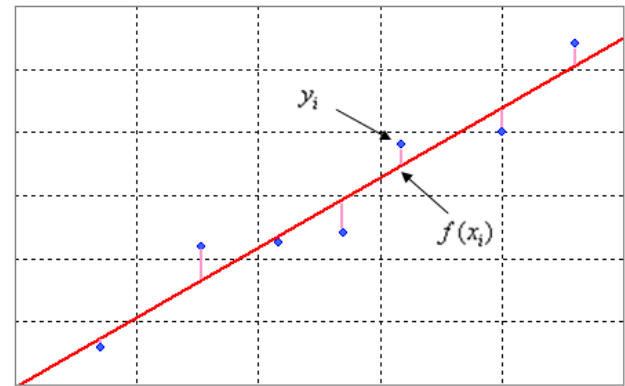
03 단순 · 다중 회귀분석

최소제곱법(LSE)

오차의 제곱(自乘, square)의 총합을
최소화(least)하는 방법(method)

$$\sum_{i=1}^n (y_i - f(x_i))^2 \text{ 가 최소화되도록 하는 함수를 찾는 것}$$

이렇게 해서 구해진 함수 $y=f(x)$ 는 측정값들의
관계에 가장 적합한 함수라고 할 수 있다

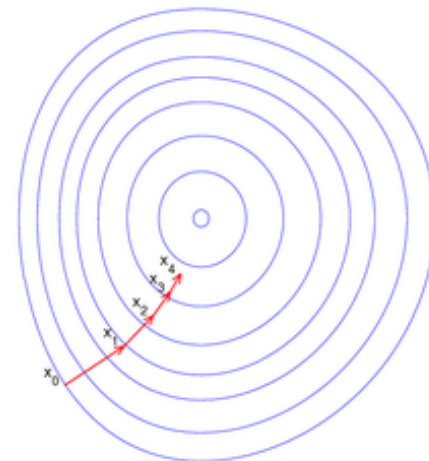


03 단순 · 다중 회귀분석

경사 하강법(Gradient descent)

비용 함수(cost function)를 정의하고 이 함수의 값이 최소화되는 파라미터를 찾는 방법

함수의 최솟값을 찾기 위해 현재 위치(보통 임의의 위치)에서 시작해서
함수의 기울기(경사)를 구하여
기울기가 낮은 쪽으로 계속 이동시켜서
극값에 이를 때까지 반복시키는 것



[장점]

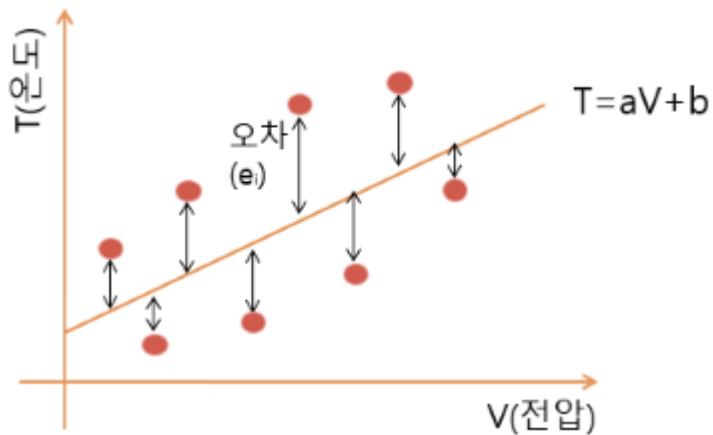
모든 차원과 모든 공간에서의 적용가능
무한 차원상에서도 쓰일 수 있음

[단점]

정확성을 위해서 극값으로 이동함에 있어 매우 많은 단계를 거쳐야함
주어진 함수에서의 곡률에 따라서 거의 같은 위치에서 시작했음에도 불구하고 완전히 다른 결과로 이어질 수도 있음

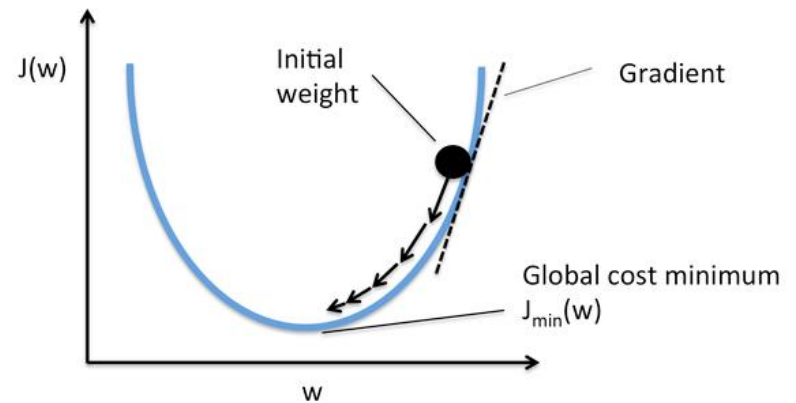
03 단순 · 다중 회귀분석

경사 하강법(Gradient descent)



오차 제곱의 합의 평균이 '비용'

$$\text{Error}_{(m,b)} = \frac{1}{N} \sum_{i=1}^N (y_i - (mx_i + b))^2$$



03 단순 · 다중 회귀분석

경사 하강법(Gradient descent)

Hypothesis: $h_{\theta}(x) = \theta^T x = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_n x_n$

↗ $x_0 = 1$

Parameters: $\theta_0, \theta_1, \dots, \theta_n$ $\underline{\theta}$ n+1-dimensional vector

Cost function:

$$\underline{J(\theta_0, \theta_1, \dots, \theta_n)} = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$$

$J(\theta)$

Gradient descent:

Repeat {

→ $\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} \underline{J(\theta_0, \dots, \theta_n)}$ $J(\theta)$

}

↑
(simultaneously update for every $j = 0, \dots, n$)

03 단순 · 다중 회귀분석

경사 하강법(Gradient descent)

Gradient Descent

Previously ($n=1$):

Repeat {

$$\rightarrow \theta_0 := \theta_0 - \alpha \underbrace{\frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})}_{\frac{\partial}{\partial \theta_0} J(\theta)}$$

$$\rightarrow \theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_1^{(i)}$$

(simultaneously update θ_0, θ_1)

}

New algorithm ($n \geq 1$):

Repeat {

$$\rightarrow \theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

(simultaneously update θ_j for $j = 0, \dots, n$)

}

$$\rightarrow \theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_0^{(i)}$$

$$\rightarrow \theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_1^{(i)}$$

$$\rightarrow \theta_2 := \theta_2 - \alpha \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_2^{(i)}$$

...

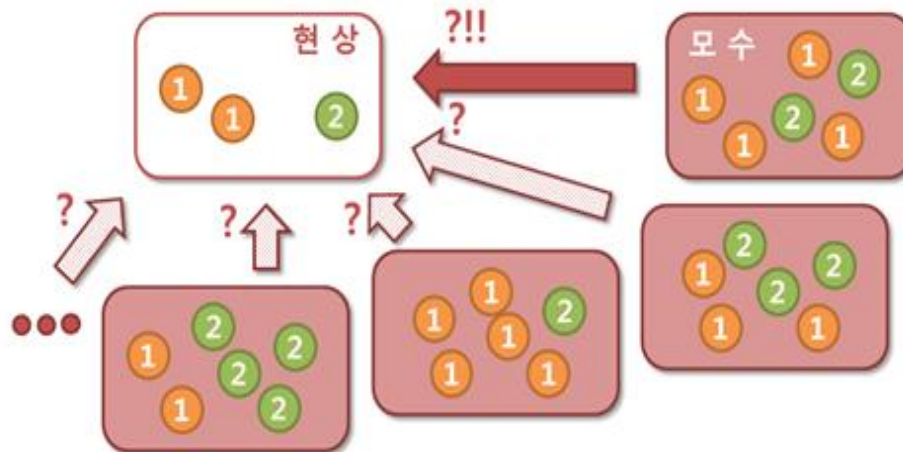
03 단순 · 다중 회귀분석

최대우도추정법(MLE)

A. 확률(probability): 모수로부터 다음과 같이 관찰될 확률은?



B. 우도(likelihood): 현상에 대해 가장 가능성이 높은(우도가 높은) 모수는?



<http://dermabae.tistory.com/188>

03 단순 · 다중 회귀분석

최대우도추정법(MLE)

Likelihood(우도)

주어진 현상을 가지고 이 현상이 추출될 가능성

최대 우도 추정법(MLE)

Maximum Likelihood는 Likelihood를 최대로 하는 파라미터를 찾는 방법

→ Likelihood를 구성한 다음 이를 maximize하면 됨

ex. 동전 던지기 성공 확률 구하기

동전을 4번 던진다고 생각해 보세요. 동전던지기는 베르누이시행(Bernoulli's trial)인데 각각의 사건(event)이 서로 독립이며 발생가능한 경우의 수는 앞, 뒤의 2가지입니다.

Cf. 베르누이 실험 확률실험의 결과가 성공 아니면 실패 즉 두가지 결과로 나타나는 실험

03 단순 · 다중 회귀분석

최대우도추정법(MLE)

4번의 동전던지기에서 (앞면, 앞면, 뒷면, 뒷면)이 나왔다고 하고, 앞면이 나올 확률, 즉 성공확률을 p 라고 표현하면


$$\begin{aligned}\text{Likelihood} &= p * p * (1-p) * (1-p) \\ &= p^2 * (1-p)^2 \\ &= p^2 * (1 - 2p + p^2) \\ &= p^2 - 2p^3 + p^4\end{aligned}$$

위의 Likelihood를 maximize하는 값이 MLE에서 찾는 P

03 단순 · 다중 회귀분석

최대우도추정법(MLE)

UK Population Maximum Likelihood Estimation



$$\Rightarrow f(x_i | p) = p^{x_i} (1-p)^{1-x_i}$$

$$x_i = \begin{cases} 1, & \text{male} \\ 0, & \text{female} \end{cases}$$

$$\Rightarrow f(1 | p) = p^1 (1-p)^{1-1} = p$$

$$\Rightarrow f(0 | p) = p^0 (1-p)^1 = 1-p$$

$$f(x_1, x_2, \dots, x_N | p) = p^{x_1} (1-p)^{1-x_1} \cdot p^{x_2} (1-p)^{1-x_2} \cdot \dots \cdot p^{x_N} (1-p)^{1-x_N}$$

$$P(X_1=x_1, X_2=x_2, \dots, X_N=x_N) = \prod_{i=1}^N p^{x_i} (1-p)^{1-x_i} = L$$

p 는 모집단의 확률 분포 : '전체 국민 중 male일 확률'

$f(x)$: sample 집단에서 관측될 확률

Likelihood의 최대화 \rightarrow 최대한 모집단과 같아질 확률

https://www.youtube.com/watch?v=I_dhPETvll8

03 단순 · 다중 회귀분석

최대우도추정법(MLE)

Likelihood(우도)

- 확률의 반대 개념이나, 비슷하게 볼 수 있음. (해석하면 '가능성')
- sample이 발생하는 것은 모집단의 확률분포에 따라 발생하는 것으로, 회귀분석 (regression)에서는 Normal분포에 따라 사건, 즉 sample이 발생하는 것으로 파악
- 가장 큰 차이는, Likelihood에서 모수(parameter)는 변수이지만, probability에서 모수(parameter)는 fixed되어 있다는 것.
 - Likelihood에서는 모수가 변수이므로 Likelihood를 maximize하는 모수를 찾을 수 있으나 probability에서는 fixed되어 있으므로 maximize하는 값을 찾는 것은 개념적으로 불가함.

03 단순 · 다중 회귀분석

최대우도추정법(MLE)

로그 우도

$$\begin{aligned} L(\theta|x) &= \prod_{i=1}^n f(x_i|\theta) \\ &= \prod_{i=1}^n L(\theta|x_i) \end{aligned}$$

$$\begin{aligned} l(\theta|x) &= \log \prod_{i=1}^n f(x_i|\theta) \\ &= \sum_{i=1}^n \log f(x_i|\theta) = \sum_{i=1}^n l(\theta|x_i). \end{aligned}$$

독립변수와 종속변수가 선형 관계에 있지 않은 일반화 선형 모형에서 회귀식을 추정할 때 자주 사용.

우도비 검정 (likelihood ratio test) :
변수를 추가하면서 모델 간의 우도를 비교하여 우도를 높이는 유의한 변수를 추가하는 주요한 방법

03 단순 · 다중 회귀분석

모델 적합도 판단 - 결정 계수(r^2)

회귀 직선이 실제 데이터를 얼마나 잘 설명하고 있는가를 나타내는 지수

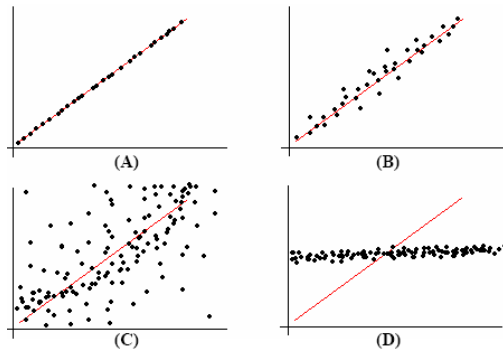
$$R^2 = \frac{SSR}{SS_{total}} = \frac{\text{회귀식으로 설명된 변화량}}{\text{총 변화량}}$$

총 변동(SS_{total})에 대한 회귀의 변동(SSR) 비율
범위는 $0 \leq r^2 \leq 1$

r^2 이 1에 가까울수록 모델이 데이터에 적합함

Cf

변수가 많아지면 r^2 값이 자연스레 증가할 수 밖에 없음



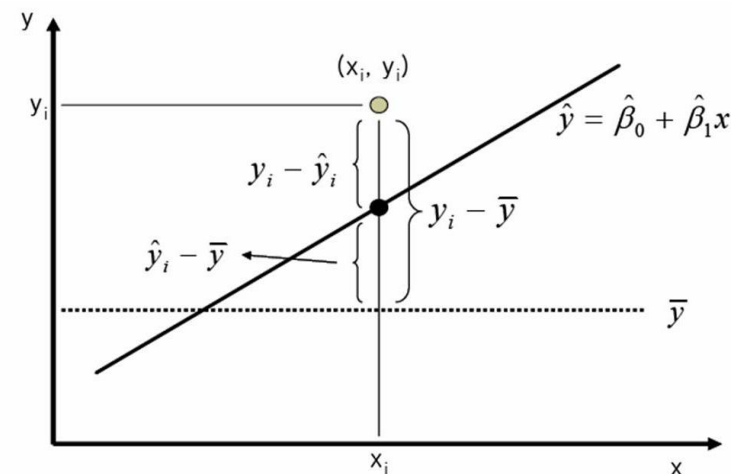
$$y_i - \bar{y} = (y_i - \hat{y}_i) + (\hat{y}_i - \bar{y})$$

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

총제곱합	잔차제곱합	회귀제곱합
SST	SSE	SSR
$n - 1$	$n - 2$	1 (자유도)

*SST (Total sum of squares)

*SSR (Regression sum of squares)



03 단순 · 다중 회귀분석

Regularization

- 정규화란 '과최적화' 막는 방법
- 베타가 커지면 커질수록 해당 모델에게 페널티를 주는 방법

Regularization.

$$\min J(\theta) = \frac{1}{2m} \left[\underbrace{\sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2}_{\text{data fit}} + \underbrace{\lambda \sum_{j=1}^n \theta_j^2}_{\text{regularization parameter}} \right]$$

Regularization.

Small values for parameters $\theta_0, \theta_1, \dots, \theta_n$

- "Simpler" hypothesis
- Less prone to overfitting

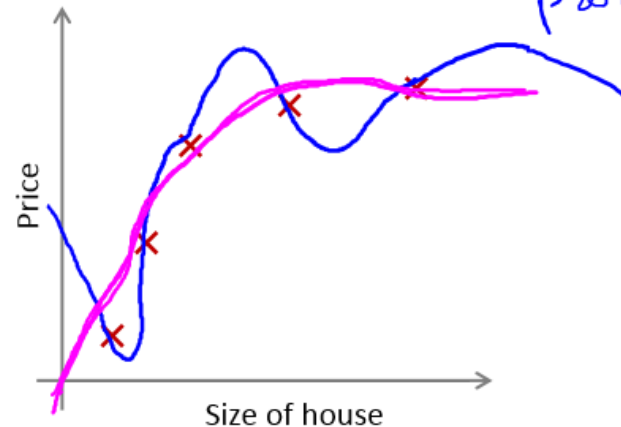
$$\theta_3, \theta_4 \approx 0$$

Housing:

- Features: x_1, x_2, \dots, x_{100}
- Parameters: $\theta_0, \theta_1, \theta_2, \dots, \theta_{100}$

$$J(\theta) = \frac{1}{2m} \left[\sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^n \theta_j^2 \right]$$

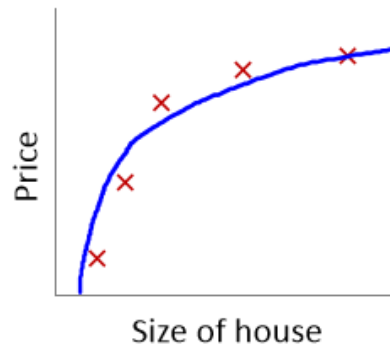
~~$\theta_1, \theta_2, \theta_3, \dots, \theta_{100}$~~



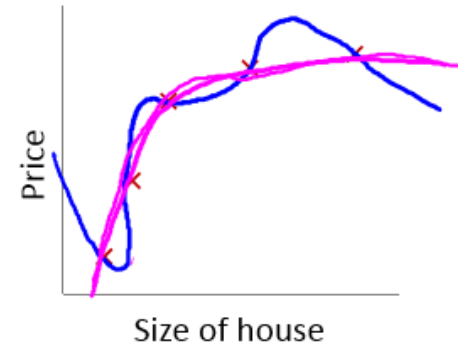
03 단순 · 다중 회귀분석

Regularization

Intuition



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



$$\theta_0 + \theta_1 x + \theta_2 x^2 + \cancel{\theta_3 x^3} + \cancel{\theta_4 x^4}$$

Suppose we penalize and make θ_3, θ_4 really small.

$$\rightarrow \min_{\theta} \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2 + 1000 \theta_3^2 + 1000 \theta_4^2$$

$\theta_3 \approx 0$ $\theta_4 \approx 0$

03 단순 · 다중 회귀분석

Regularization

Ridge 회귀모형

Ridge 회귀 모형에서는 가중치들의 제곱합(squared sum of weights)을 최소화하는 것을 추가적인 제약 조건으로 한다.

$$\text{cost} = \sum e_i^2 + \lambda \sum w_i^2$$

Lasso 회귀모형

Lasso(Least Absolute Shrinkage and Selection Operator) 회귀 모형은 가중치의 절대값의 합을 최소화하는 것을 추가적인 제약 조건으로 한다.

$$\text{cost} = \sum e_i^2 + \lambda \sum |w_i|$$

Elastic Net 회귀모형

Elastic Net 회귀 모형은 가중치의 절대값의 합과 제곱합을 동시에 제약 조건으로 가지는 모형이다.

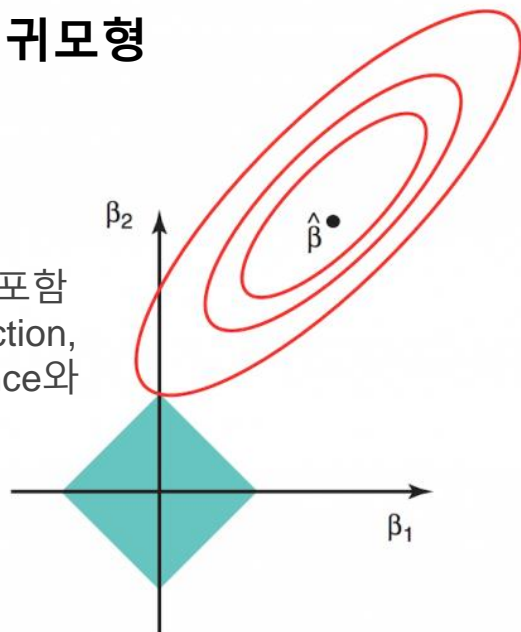
$$\text{cost} = \sum e_i^2 + \lambda_1 \sum |w_i| + \lambda_2 \sum w_i^2$$

03 단순 · 다중 회귀분석

Regularization(FYI)

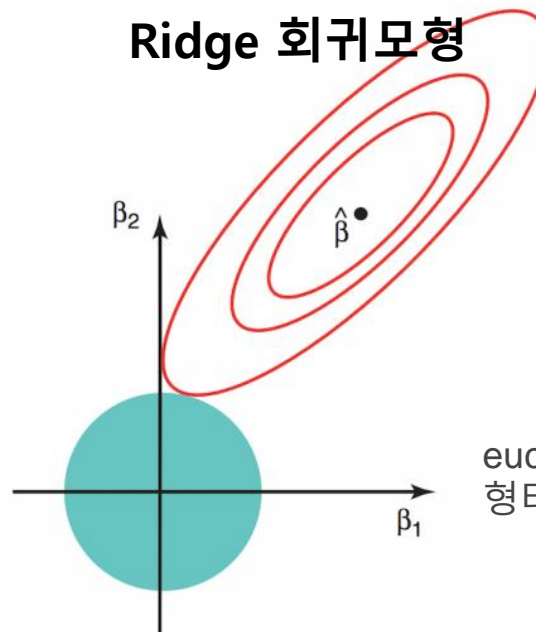
Lasso 회귀모형

파란 영역은 λ 가 포함
된 constraint function,
manhattan distance와
형태의 함수



Ridge 회귀모형

euclidean distance
형태의 함수를 제공



Lasso의 경우 최적값은 **모서리 부분에서 나타날 확률이 Ridge에 비해 높아** 몇몇 유의미하지 않은 변수들에 대해 계수를 0에 가깝게 추정해 주어 **변수 선택 효과**를 가져오게 된다. 반면 Ridge의 경우 어느정도 상관성을 가지는 변수들에 대해서 pulling이 되는 효과를 보여줘 변수 선택보다는 **상관성이 있는 변수들에 대해서 적절한 가중치 배분**을 하게 된다. 따라서 Ridge의 경우 PCA와 상당한 관련성이 있다.

04 단순 선형 회귀분석

Importing library

```
from __future__ import division
from probability import mean, correlation, standard_deviation, de_mean
```

probability.py
참조

```
def correlation(x, y):
    stdev_x = standard_deviation(x)
    stdev_y = standard_deviation(y)
    if stdev_x > 0 and stdev_y > 0:
        return covariance(x, y) / stdev_x / stdev_y
    else:
        return 0 # if no variation, correlation is zero
```

```
def standard_deviation(x):
    return math.sqrt(variance(x))
```

```
def de_mean(x):
    """translate x by subtracting its mean (so the result has mean 0)"""
    x_bar = mean(x)
    return [x_i - x_bar for x_i in x]
```

04 단순 선형 회귀분석

Predict함수와 최소자승법

```
5
6 def predict(alpha, beta, x_i):           추정치
7     return beta * x_i + alpha
8
9
10 def error(alpha, beta, x_i, y_i):        실제값 - 추정치
11     return y_i - predict(alpha, beta, x_i)
12
13
14 def sum_of_squared_errors(alpha, beta, x, y):    SSE
15     return sum(error(alpha, beta, x_i, y_i) ** 2 (실제값 - 추정치 의 제곱합)
16               for x_i, y_i in zip(x, y))
17
18
19 def least_squares_fit(x, y):
20     """given training values for x and y,
21     find the least-squares values of alpha and beta"""
22     beta = correlation(x, y) * standard_deviation(y) / standard_deviation(x)
23     alpha = mean(y) - beta * mean(x)
24     return alpha, beta
25
```

04 단순 선형 회귀분석

결정계수(r^2)

```
def total_sum_of_squares(y):
    """the total squared variation of y_i's from their mean"""
    return sum(v ** 2 for v in de_mean(y))
```

SST

(실제값 - 평균의 제곱합)

```
def r_squared(alpha, beta, x, y):
    """the fraction of variation in y captured by the model, which equals
    1 - the fraction of variation in y not captured by the model"""

    return 1.0 - (sum_of_squared_errors(alpha, beta, x, y) /
                  total_sum_of_squares(y))
```

결정계수(r^2)

$= 1 - (SSE/SST)$

04 단순 선형 회귀분석

최소자승법으로 적합시키기 & 결정계수 구하기

num_friends_good 과 daily_minutes_good 은 data 폴더 참조

```
alpha, beta = least_squares_fit(num_friends_good, daily_minutes_good)
print("alpha", alpha)
print("beta", beta)

print("r-squared", r_squared(alpha, beta, num_friends_good, daily_minutes_good))
```

simple_linear_regression

/Library/Frameworks/Python.framework/Versions/

alpha 22.94755241346903

beta 0.903865945605865

r-squared 0.3291078377836305

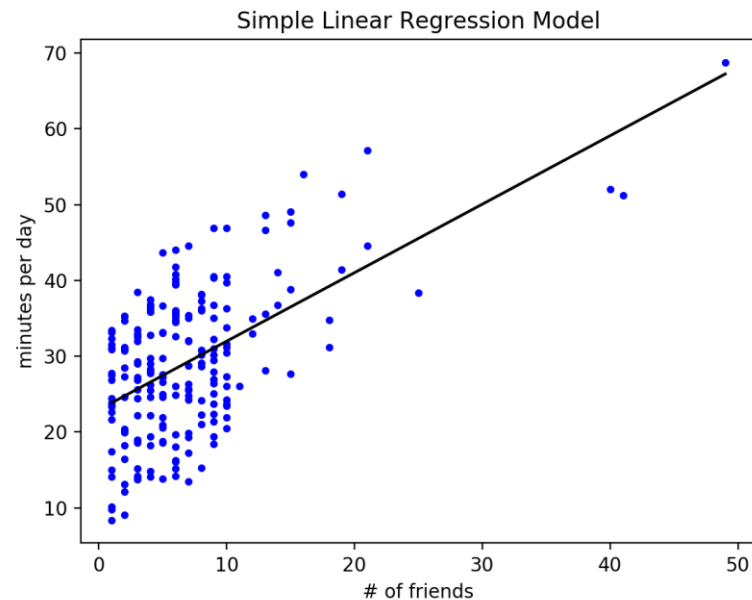
Process finished with exit code 0

04 단순 선형 회귀분석

result

from matplotlib import pyplot as plt

```
def yfit(alpha, beta, x):  
    return [predict(alpha, beta, i) for i in x]  
  
plt.scatter(num_friends_good, daily_minutes_good, marker='.', color='blue')  
plt.plot(num_friends_good, yfit(alpha, beta, num_friends_good), color='k')  
plt.title("Simple Linear Regression Model")  
plt.xlabel("# of friends")  
plt.ylabel("minutes per day")  
plt.show()
```



04 단순 선형 회귀분석

경사하강법(SGD)

```
def squared_error(x_i, y_i, theta):  
    alpha, beta = theta  
    return error(alpha, beta, x_i, y_i) ** 2  
  
def squared_error_gradient(x_i, y_i, theta):  
    alpha, beta = theta  
    return [-2 * error(alpha, beta, x_i, y_i), # alpha partial derivative  
            -2 * error(alpha, beta, x_i, y_i) * x_i] # beta partial derivative
```

```
random.seed(0)  
theta = [random.random(), random.random()]  
alpha, beta = minimize_stochastic(squared_error,  
                                   squared_error_gradient,  
                                   num_friends_good,  
                                   daily_minutes_good,  
                                   theta,  
                                   0.0001)  
  
print("alpha", alpha)  
print("beta", beta)
```

04 단순 선형 회귀분석

경사하강법(SGD)

from gradient_descent import minimize_stochastic_2

```
def minimize_stochastic_2(target_fn, gradient_fn, x, y, theta_0, alpha_0=0.01):
    data = zip(x, y)
    theta = theta_0           # initial guess
    alpha = alpha_0           # initial step size
    min_theta, min_value = None, float("inf") # the minimum so far
    iterations_with_no_improvement = 0

    # if we ever go 100 iterations with no improvement, stop
    while iterations_with_no_improvement < 100:
        value = sum(target_fn(x_i, y_i, theta) for x_i, y_i in data)

        if value < min_value:
            # if we've found a new minimum, remember it
            # and go back to the original step size
            min_theta, min_value = theta, value
            iterations_with_no_improvement = 0
            alpha = alpha_0
        else:
            # otherwise we're not improving, so try shrinking the step size
            iterations_with_no_improvement += 1
            alpha *= 0.9

        # and take a gradient step for each of the data points
        for x_i, y_i in in_random_order(data):
            gradient_i = gradient_fn(x_i, y_i, theta)
            theta = vector_subtract(theta, scalar_multiply(alpha, gradient_i))

    return min_theta
```

04 단순 선형 회귀분석

경사하강법(SGD)

from gradient_descent import minimize_stochastic

```
def minimize_stochastic(target_fn, gradient_fn, x, y, theta_0, alpha_0=0.01):
```

```
#data = zip(x, y)
```

```
theta = theta_0 # initial guess
```

```
alpha = alpha_0 # initial step size
```

```
min_theta = None
```

```
min_value = float("inf") # the minimum so far
```

```
iterations_with_no_improvement = 0
```

```
#value = 0
```

```
# if we ever go 100 iterations with no improvement, stop
```

```
while iterations_with_no_improvement < 35:
```

```
    k = 0
```

```
    value = 0
```

```
    while k < len(x):
```

```
        value += target_fn(x[k], y[k], theta)
```

```
        k += 1
```

```
    print(value, min_value)
```

```
    if value < min_value:
```

```
        # if we've found a new minimum, remember it
```

```
        # and go back to the original step size
```

```
        min_theta, min_value = theta, value
```

```
        iterations_with_no_improvement = 0
```

```
        alpha = alpha_0
```

```
        print(min_theta)
```

```
    else:
```

```
        # otherwise we're not improving, so try shrinking the step size
```

```
        iterations_with_no_improvement += 1
```

```
        print(iterations_with_no_improvement)
```

```
        alpha *= 0.95
```

```
    i = 0
```

```
    while i < len(x):
```

```
        gradient_i = gradient_fn(x[i], y[i], theta)
```

```
        theta = vector_subtract(theta, scalar_multiply(alpha, gradient_i))
```

```
        i += 1
```

```
    return min_theta
```

Zip(x,y)에 대한 for 문이
작동하지 않는 경우
수정 코드

04 다중 선형 회귀분석

Importing library

```
from __future__ import division
from functools import partial
from linear_algebra import dot
from probability import de_mean
from statistics import mean
from gradient_descent import minimize_stochastic
import random
```

```
def dot(v, w):
    """v_1 * w_1 + ... + v_n * w_n"""
    return sum(v_i * w_i for v_i, w_i in zip(v, w))
```

Data 폴더 내의 linear_algebra.py 파일 참조

04 다중 선형 회귀분석

Predict 함수

```
def predict(x_i, beta):  
    return dot(x_i, beta)  
  
def error(x_i, y_i, beta):  
    return y_i - predict(x_i, beta)  
  
def squared_error(x_i, y_i, beta):  
    return error(x_i, y_i, beta) ** 2  
  
def total_sum_of_squares(y):  
    """the total squared variation of y_i's from their mean"""  
    return sum(v ** 2 for v in de_mean(y))  
  
def squared_error_gradient(x_i, y_i, beta):  
    """the gradient corresponding to the ith squared error term"""  
    return [-2 * x_ij * error(x_i, y_i, beta)  
            for x_ij in x_i]
```

04 다중 선형 회귀분석

estimate_beta(x, y)

```
def estimate_beta(x, y):  
    beta_initial = [random.random() for x_i in x]  
    #print(beta_initial)  
    return minimize_stochastic(squared_error, squared_error_gradient, x, y, beta_initial, 0.002)  
  
def multiple_r_squared(x, y, beta):  
    sum_of_squared_errors = sum(error(x_i, y_i, beta) ** 2  
                                for x_i, y_i in zip(x, y))  
    return 1.0 - sum_of_squared_errors / total_sum_of_squares(y)
```

```
random.seed(0)  
beta = estimate_beta(x, daily_minutes_good) # [30.63, 0.972, -1.868, 0.911]  
print("beta", beta)  
print("r-squared", multiple_r_squared(x, daily_minutes_good, beta))
```

04 다중 선형 회귀분석

result

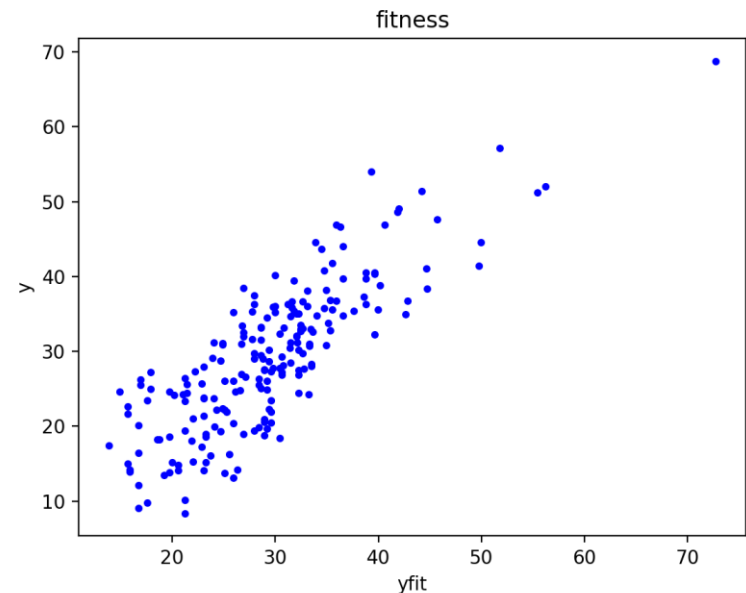
```
beta [30.522657701970747, 1.0111632007085694, -1.8416213803688282, 0.7515629864517557]
r-squared 0.6784650836048222
```

Process finished with exit code 0

```
def yfit(x, beta):
    return [predict(x_i, beta) for x_i in x]
```

from matplotlib import pyplot as plt

```
plt.scatter(yfit(x, beta), daily_minutes_good, marker='.', color='blue')
plt.xlabel("yfit")
plt.ylabel("y")
plt.title("fitness")
plt.show()
```



sklearn.linear_model.LinearRegression()

plot.ols.py

Importing library & loading data

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

# Load the diabetes dataset
diabetes = datasets.load_diabetes()

#print(diabetes.data.shape)
#print(diabetes.data)

# Use only one feature
diabetes_X = diabetes.data[:, np.newaxis, 2]
```

sklearn.linear_model.LinearRegression()

plot.ols.py

Slicing data for training/testing
& fitting the model

```
# Split the data into training/testing sets
diabetes_X_train = diabetes_X[:-20]
diabetes_X_test = diabetes_X[-20:]

# Split the targets into training/testing sets
diabetes_y_train = diabetes.target[:-20]
diabetes_y_test = diabetes.target[-20:]

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

# Train the model using the training sets
regr.fit(diabetes_X_train, diabetes_y_train)

# Make predictions using the testing set
diabetes_y_pred = regr.predict(diabetes_X_test)
```

sklearn.linear_model.LinearRegression()

plot.ols.py

result

```
# The coefficients
print('Coefficients: \n', regr.coef_)
# The mean squared error
print("Mean squared error: %.2f"
      % mean_squared_error(diabetes_y_test, diabetes_y_pred))
# Explained variance score: 1 is perfect prediction
print('Variance score: %.2f' % r2_score(diabetes_y_test, diabetes_y_pred))

# Plot outputs
plt.scatter(diabetes_X_test, diabetes_y_test, color='black')
plt.plot(diabetes_X_test, diabetes_y_pred, color='blue', linewidth=3)

plt.xticks(())
plt.yticks(())

plt.show()
```

THANK YOU !