통계기초 / 회귀분석

YBIGTA 18기 교육세션
21 Jan 2021(Thu)

학습 목차

1. 통계기초
2. 선형(Linear) 회귀
3. 경사하강법(Gradient Descent)
4. 다항(Polynomial) 회귀
5. Ridge, Lasso, Elastic Net 회귀
6. Logistic 회귀
7. 실습 A / 실습 B

연세대학교 빅데이터 학회

1. 통계기초

1-1. 통계 기초 용어 정리

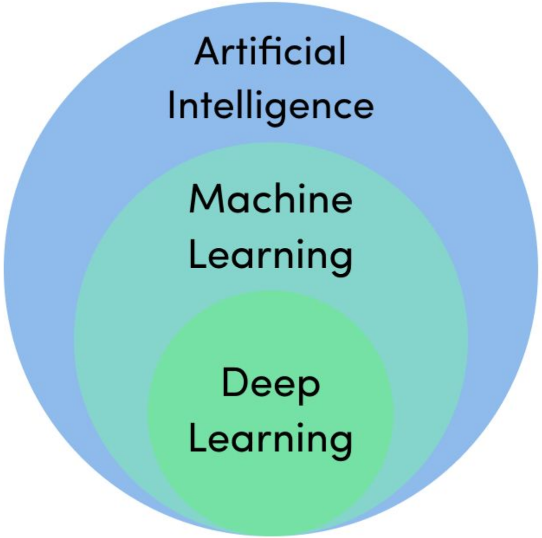
이 중에서 몇 개나 알고 있는지 체크해봅시다.

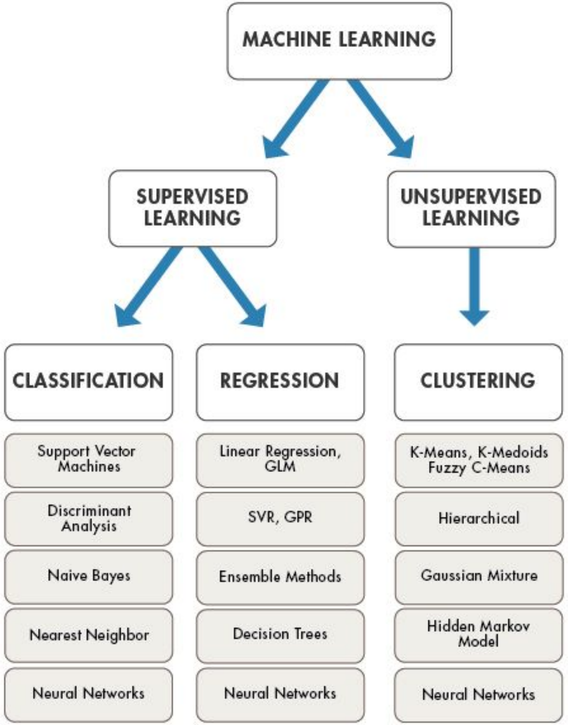
- Sample
- Population
- Mean
- Median
- Variance
- Standard deviation
- Standard error
- Covariance
- Correlation

- Distribution
- Degrees of freedom
- Regression line
- Type I / II error
- Residual
- Test statistic
- Significance level
- Confidence interval
- Hypothesis test

- Quantile
- Percentile
- Conditional probability
- Unbiased estimator
- MLE
- Discrete / Continuous
- PMF / PDF
- Central Limit Theorem
- ANOVA

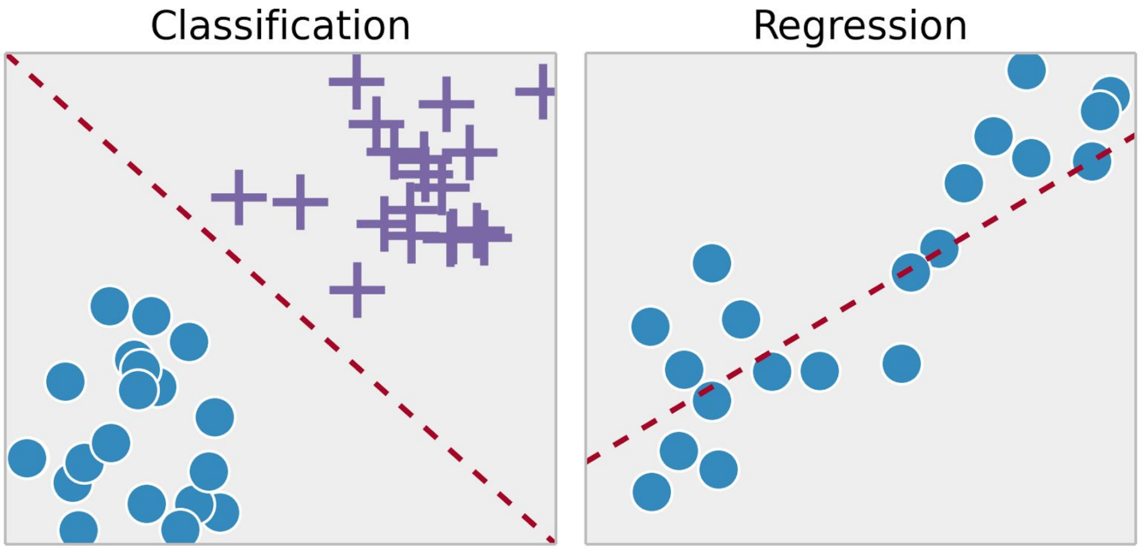
1-2. 머신러닝 소개





ybigta
연세대학교 빅데이터 학회

1-3. 분류 vs. 회귀



1-3. 분류 vs. 회귀

Classification
(분류)

↓

Categorical variable
(이산값)

Regression
(회귀)

↓

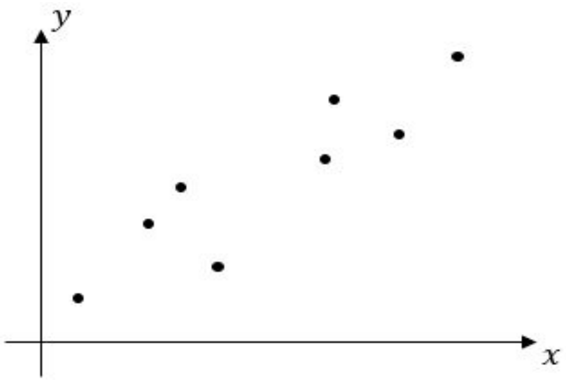
Continuous variable
(연속값)

2. 선형(Linear) 회귀

2-1. 단순선형회귀

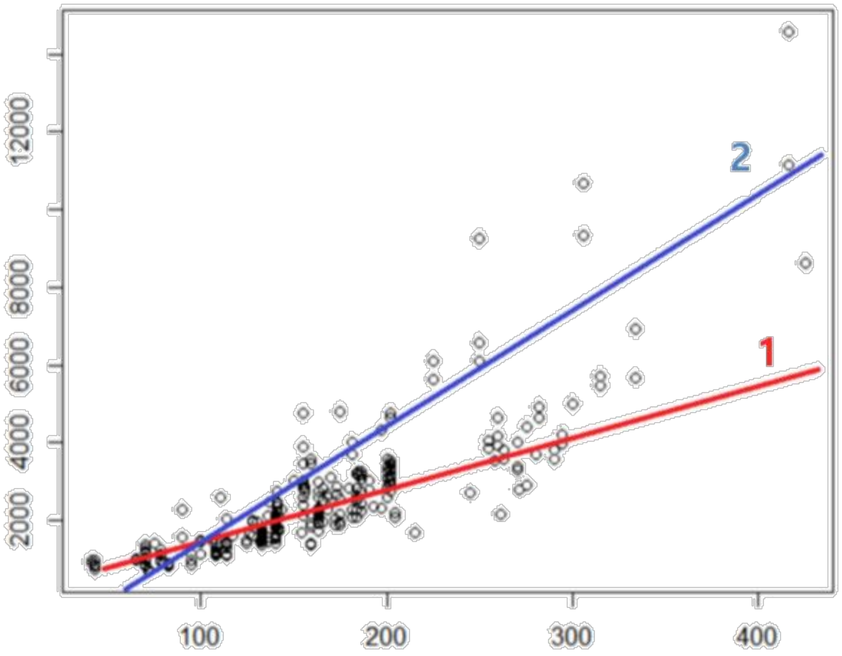
독립변수 → 인과관계 → 종속변수

변수 X **변수 Y**



- 독립변수(예측변수): 영향을 미치는 변수
- 종속변수(기준변수): 영향을 받는 변수

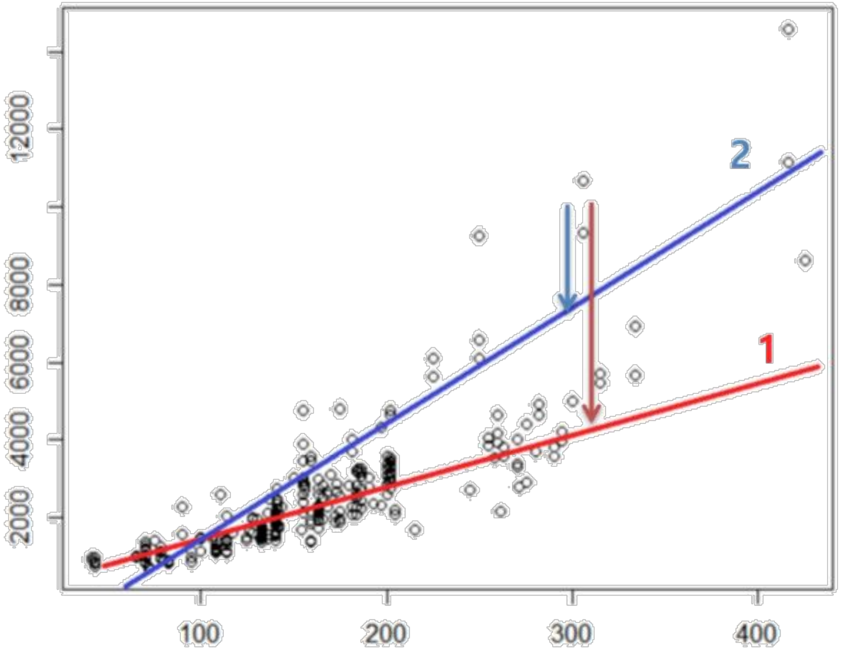
2-1. 단순선형회귀



어느 직선이 더 적절할까?

Why??

2-1. 단순선형회귀



실제값과 추정값의 차이가
작을수록 더 적절한 직선이라고
볼 수 있다.

2-1. 단순선형회귀

$$Y = \beta_0 + \beta_1 X + \epsilon.$$

(모회귀선)

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x$$

(추정회귀선)

2-1. 단순선형회귀

$$e_i = y_i - \hat{y}_i$$

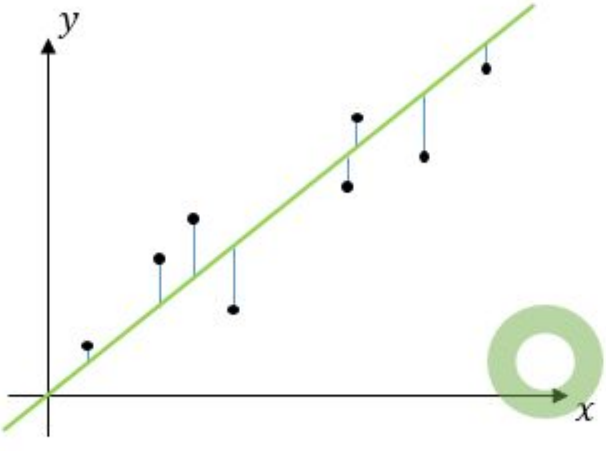
y_i : 실제 데이터의 값

$\hat{y}_i$: 회귀선으로 추정된 값

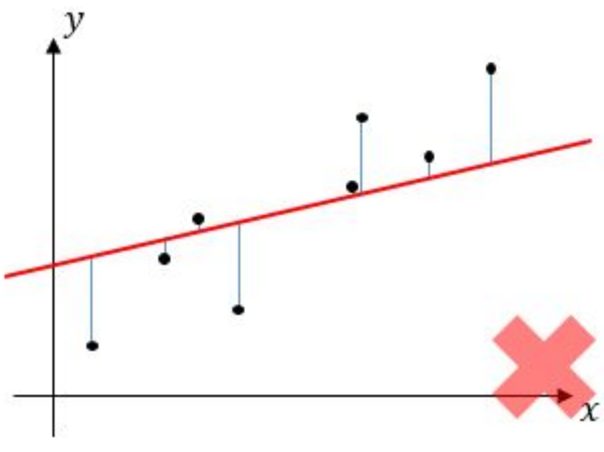
$$SSE = \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

목표: SSE를 최소화하는 회귀식을 찾자!

2-1. 단순선형회귀



SSE가 작다



SSE가 크다

SSE가 작을수록 더 좋은 회귀식이다!

2-1. 단순선형회귀

그렇다면 SSE를 최소화하는 회귀식은 어떻게 구할 수 있을까?

$$\varepsilon^2 = \sum_{i=1}^{n}(y_i - \beta_0 - \beta_1 x_i)^2 \tag{1}$$

를 가장 작게 만드는 $\beta_0 = \hat{\beta}_0$ 와 $\beta_1 = \hat{\beta}_1$ 을 찾는 것이다.

2-1. 단순선형회귀

식 (1) 에서 β_0 에 대해 편미분을 취해보면

$$\frac{\partial \varepsilon^2}{\partial \beta_0} = -2\sum_{i=1}^{n}(y_i - \beta_0 - \beta_1 x_i)$$

ε^2 이 가장 작아지려면

$$n\beta_0 = \sum_{i=1}^{n} y_i - \beta_1 \sum_{i=1}^{n} x_i$$

따라서 ε^2 은 $\beta_0 = \bar{y} - \beta_1 \bar{x}$ 일 때 가장 작아진다.

2-1. 단순선형회귀

식 (1) 에서 β_1 에 대해 편미분을 취해보면

$$\frac{\partial \varepsilon^2}{\partial \beta_1} = -2\sum_{i=1}^{n} x_i(y_i - \beta_0 - \beta_1 x_i)$$

ε^2 이 가장 작아지려면 $\beta_0 = \bar{y} - \beta_1 \bar{x}$ 이므로

$$\sum_{i=1}^{n} x_i(y_i - \bar{y} + \beta_1 \bar{x} - \beta_1 x_i) = 0$$

ybigta
연세대학교 빅데이터 학회

2-1. 단순선형회귀

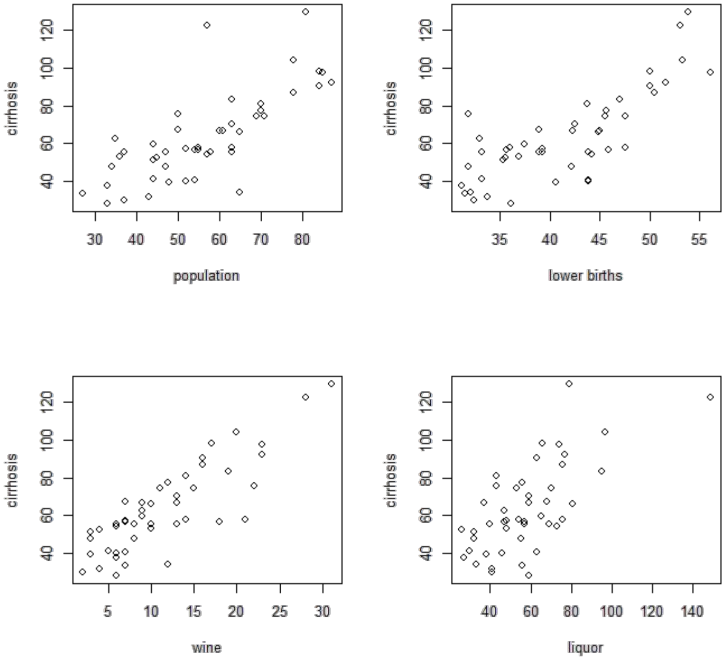
$$\beta_1 \sum_{i=1}^{n}(x_i^2 - \bar{x}x_i) = \sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \bar{y}$$

이다. 정리하면

$$\begin{aligned}
\beta_1 &= \frac{\sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \bar{y}}{\sum_{i=1}^{n}(x_i^2 - \bar{x}x_i)} \\
&= \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i^2 - \bar{x}^2)} \\
&= \frac{\mathrm{Cov}(X, Y)}{\mathrm{Var}(X)} \\
&= \mathrm{Cor}(X, Y)\frac{s_y}{s_x}
\end{aligned}$$

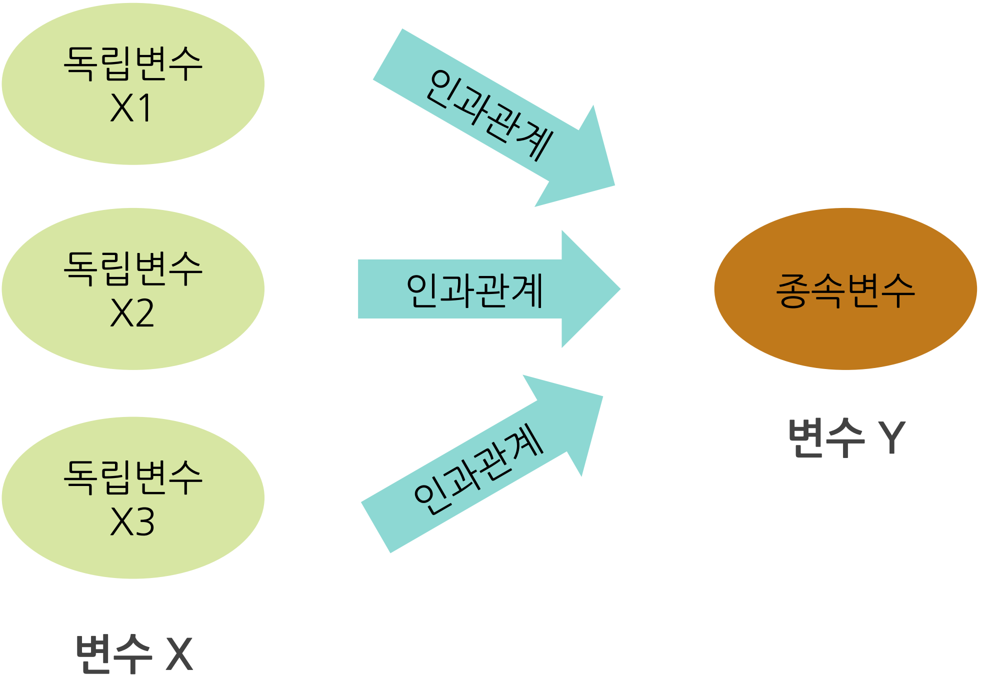
ybigta
연세대학교 빅데이터 학회

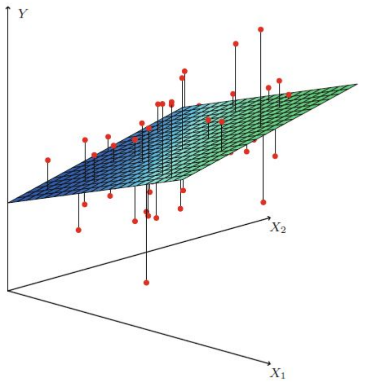
2-2. 다중선형회귀



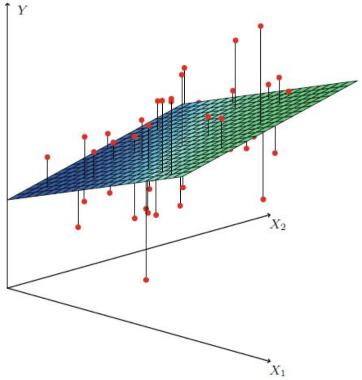
독립변수가 두 개 이상인 경우는?

2-2. 다중선형회귀





2-2. 다중선형회귀



$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \epsilon,$$ (모회귀선)

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \cdots + \hat{\beta}_p x_p.$$ (추정회귀선)

2-2. 다중선형회귀

$$y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \quad \varepsilon = \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{bmatrix} \quad X = \begin{bmatrix} 1 & x_{11} \cdots x_{1K} \\ 1 & x_{21} \cdots x_{2K} \\ \vdots & \vdots \quad \ddots \quad \vdots \\ 1 & x_{n1} \cdots x_{nK} \end{bmatrix} \quad \beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_K \end{bmatrix}$$

$$\begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_{11} & \dots & x_{1k} \\ 1 & x_{21} & \dots & x_{2k} \\ \dots & \dots & & \dots \\ 1 & x_{n1} & \dots & x_{nk} \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \dots \\ \beta_k \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \dots \\ \varepsilon_n \end{bmatrix}$$

ybigta
연세대학교 빅데이터 학회

2-2. 다중선형회귀

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

$$= \sum_{i=1}^{n} (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_{i1} - \hat{\beta}_2 x_{i2} - \cdots - \hat{\beta}_p x_{ip})^2$$

목표: SSE를 최소화하는 회귀식을 찾자!

(증명 생략)

$$b = \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_k \end{bmatrix} = (X'X)^{-1}X'Y$$

2-3. 회귀 성능 평가 지표

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}|$$

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2$$

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2}$$

$$R^2 = 1 - \frac{\sum(y_i - \hat{y})^2}{\sum(y_i - \bar{y})^2}$$

Where,

$\hat{y}$ − *predicted value of y*

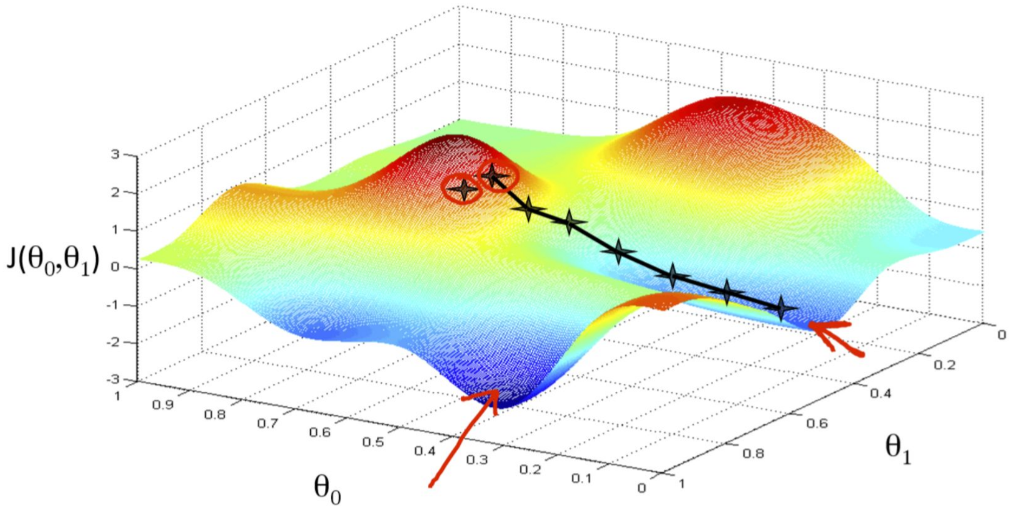
$\bar{y}$ − *mean value of y*

3. 경사하강법 (Gradient Descent)

ybigta
연세대학교 빅데이터 학회

3. 경사하강법

Gradient descent (경사하강법) is an iterative optimization algorithm for finding the **local minimum** of a function.



3. 경사하강법

Hypothesis:

$$h_\theta(x) = \theta_0 + \theta_1 x$$

Parameters:

$$\theta_0, \theta_1$$

Cost Function:

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

Goal:

$$\underset{\theta_0, \theta_1}{minimize} J(\theta_0, \theta_1)$$

3. 경사하강법

Gradient descent algorithm

$\frac{\partial}{\partial\theta_0} J(\theta_0, \theta_1)$

repeat until convergence {

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

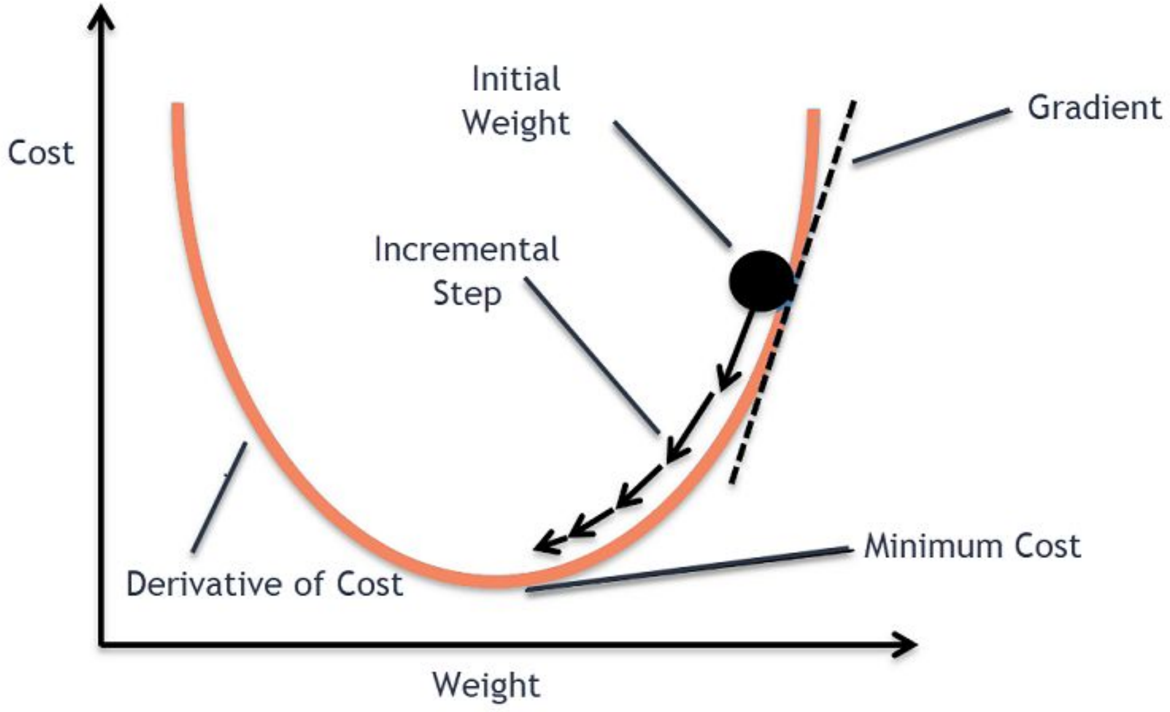
$$\theta_1 := \theta_1 - \alpha \boxed{\frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right) \cdot x^{(i)}}$$

}

update θ_0 and θ_1 simultaneously

$\frac{\partial}{\partial\theta_1} J(\theta_0, \theta_1)$

3. 경사하강법


Initial
Weight
Gradient
Cost
Incremental
Step
Minimum Cost
Derivative of Cost
Weight

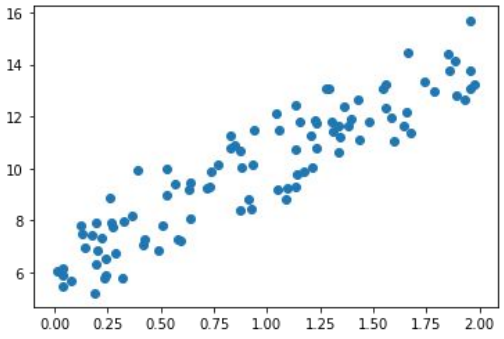
3. 경사하강법

1) 실제값을 Y = 4X + 6 시뮬레이션하는 데이터 값 생성

```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

np.random.seed(0)
# y = 4X + 6 식을 근사(w1=4, w0=6). random 값은 Noise를 위해 만듦
X = 2*np.random.rand(100, 1)
y = 6 + 4*X + np.random.randn(100,1)

# X, y 데이터 세트 scatter plot으로 시각화
plt.scatter(X, y)
```



3. 경사하강법

2) w0과 w1의 값을 최소화 할 수 있도록 업데이트 수행하는 함수 생성

```python
# w1과 w0을 업데이트 할 w1_update, w0_update를 반환
def get_weight_updates(w1, w0, X, y, learning_rate=0.01):
    N = len(y)

    # 먼저 w1_update, w0_update를 각각 w1, w0의 shape와 동일한 크기를 가진 0 값으로 초기화
    w1_update = np.zeros_like(w1)
    w0_update = np.zeros_like(w0)

    # 예측 배열 계산하고 예측과 실제 값의 차이 계산
    y_pred = np.dot(X, w1.T) + w0
    diff = y - y_pred

    # w0_update를 dot 행렬 연산으로 구하기 위해 모두 1 값을 가진 행렬 생성
    w0_factors = np.ones((N, 1))

    # w1과 w0을 업데이트할 w1_update와 w0_update 계산
    w1_update = -(2/N)*learning_rate*(np.dot(X.T, diff))
    w0_update = -(2/N)*learning_rate*(np.dot(w0_factors.T, diff))

    return w1_update, w0_update
```

3. 경사하강법

3) 반복적으로 경사 하강법을 이용해 get_weigth_updates()를 호출하여 w1과 w0를 업데이트 하는 함수 생성

```
# 입력 인자 iters로 주어진 횟수만큼 반복적으로 w1과 w0를 업데이트 적용
def gradient_descent_steps(X, y, iters=10000):
    # w0와 w1을 모두 0으로 초기화
    w0 = np.zeros((1, 1))
    w1 = np.zeros((1, 1))

    # 인자로 주어진 iters만큼 반복적으로 get_weight_updates() 호출하여 w1, w0 업데이트 수행
    for i in range(iters):
        w1_update, w0_update = get_weight_updates(w1, w0, X, y, learning_rate=0.01)
        w1 = w1 - w1_update
        w0 = w0 - w0_update

    return w1, w0
```

3. 경사하강법

4) 예측 오차 비용을 계산을 수행하는 함수 생성 및 경사 하강법 수행

```
def get_cost(y, y_pred):
    N = len(y)
    cost = np.sum(np.square(y - y_pred))/N
    return cost

w1, w0 = gradient_descent_steps(X, y, iters=1000)
print("w1:{0:.3f} w0:{1:.3f}".format(w1[0, 0], w0[0, 0]))

y_pred = w1[0,0]*X + w0
print('Gradient Descent Total Cost:{0:.4f}'.format(get_cost(y, y_pred)))
```

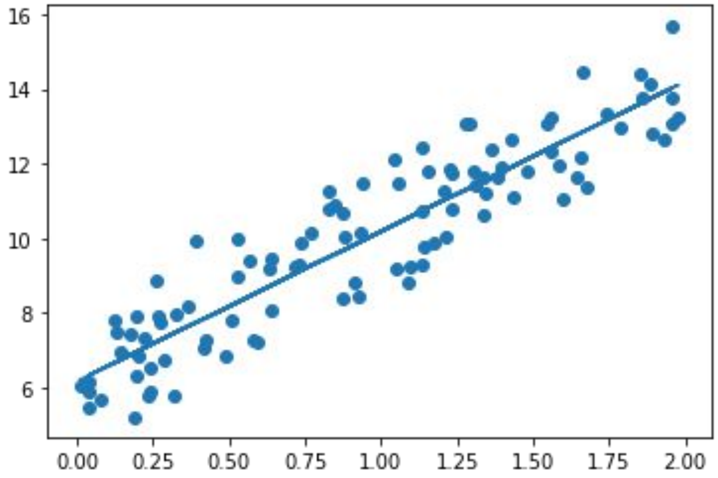
W1 : 4.022 w0 : 6.162
Gradient Descent Total Cost : 0.9935

ybigta
연세대학교 빅데이터 학회

3. 경사하강법

5) 회귀선 시각화

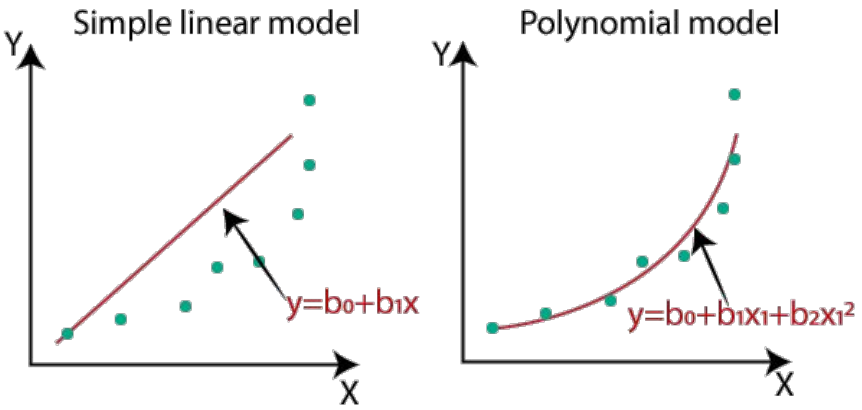
```
plt.scatter(X, y)
plt.plot(X, y_pred)
```



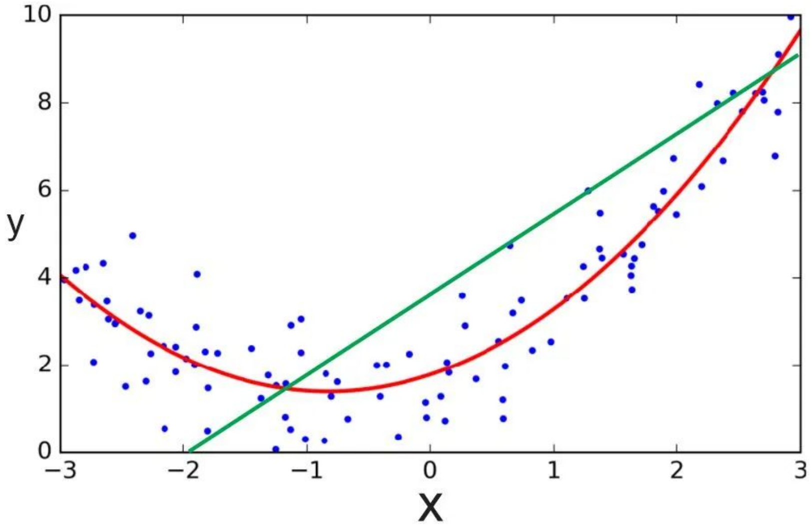
4. 다항(Polynomial) 회귀

4. 다항회귀

Simple Linear Regression	$y = b_0 + b_1 x_1$
Multiple Linear Regression	$y = b_0 + b_1 x_1 + b_2 x_2 + \ldots + b_n x_n$
Polynomial Linear Regression	$y = b_0 + b_1 x_1 + b_2 x_1^2 + \ldots + b_n x_1^n$



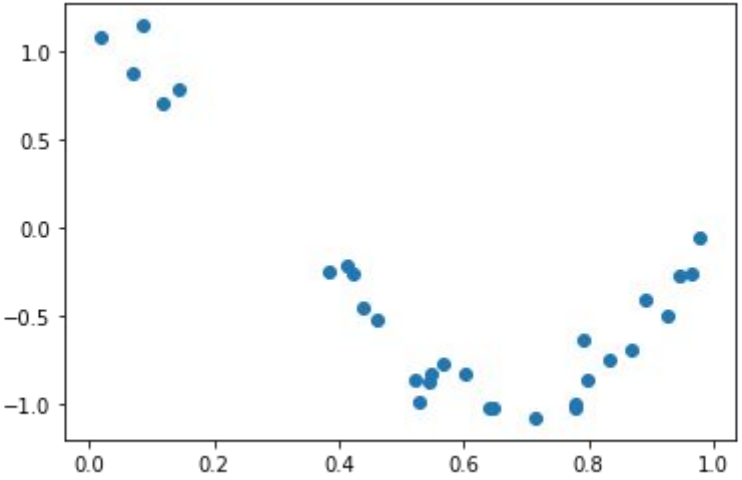
4. 다항회귀



다항(Polynomial) 회귀는
선형(Linear) 회귀일까?
비선형(Nonlinear) 회귀일까?

이 데이터 세트에서는 **다항 회귀가 더 효과적**이다.

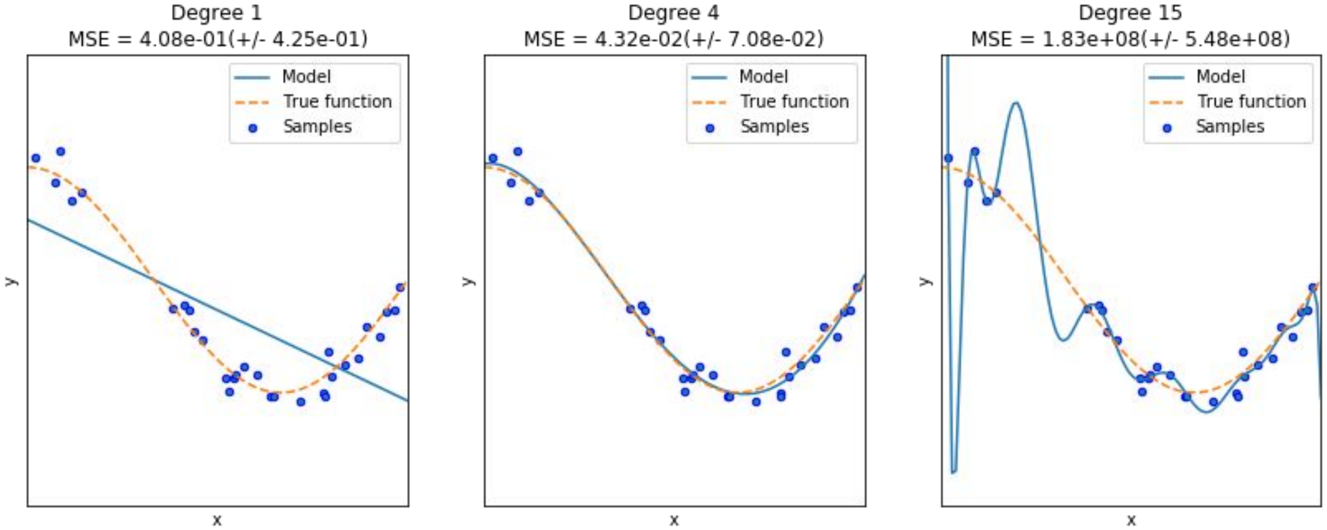
4. 다항회귀



차수(degree)는 어떻게 결정해야 할까?

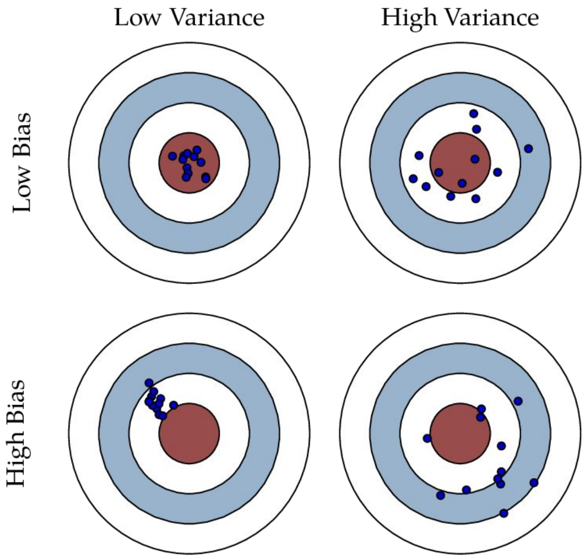
4. 다항회귀

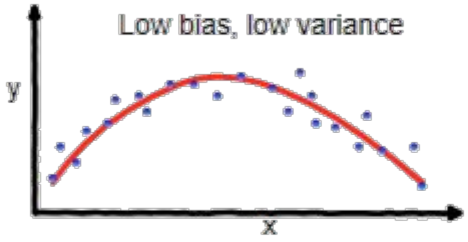
과소 적합(Underfitting)과 과대 적합(Overfitting)



4. 다항회귀

편향-분산 트레이드오프 (Bias-Variance Trade off)

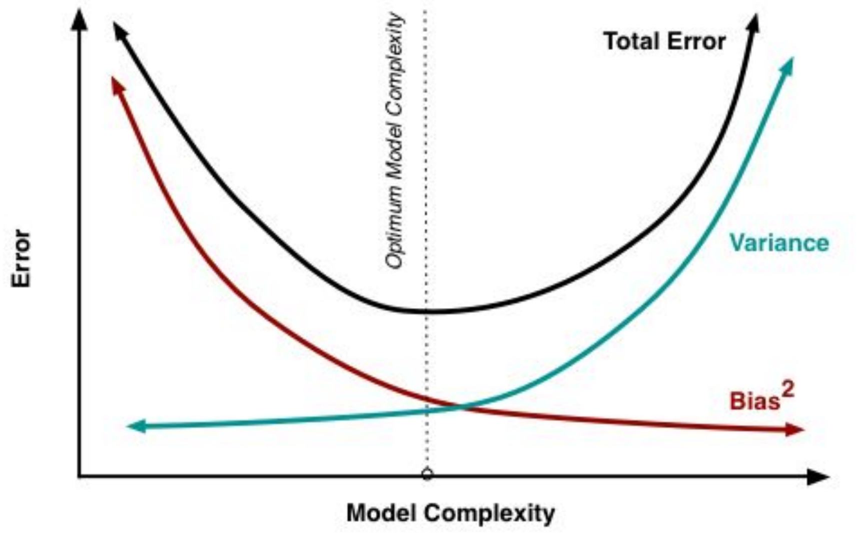




Good balance

4. 다항회귀

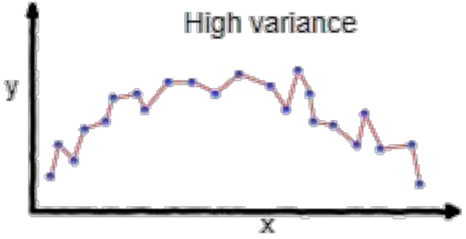
편향-분산 트레이드오프 (Bias-Variance Trade off)



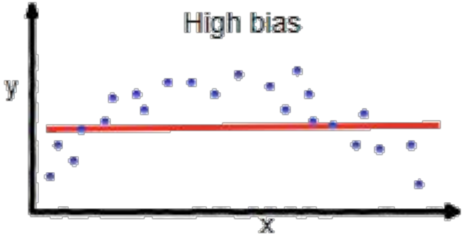
Total Error
or
Mean Square Prediction Error (MSPE)
= Variance + Bias

모델의 복잡도가 늘어날수록 (독립변수가 많아질수록)
Variance는 증가하고 Bias는 감소한다.
즉, **과대 적합(Overfitting)**이 일어난다.

4. 다항회귀



overfitting



underfitting

두 경우 모두 새로운 데이터에 대해서 적절한 예측을 하지 못한다. 따라서 Bias와 Variance 사이의 적절한 균형을 맞추는 과정이 필요하다.

정규화(Regularization)를 통해 Variance를 감소시켜 모델의 성능을 높일 수 있다. 이 과정에서 Bias가 증가할 수 있지만, 그 정도가 Variance가 감소하는 정도에 비해 미미한 수준이라면 정규화를 이용하는 것이 좋은 방법이다.

5. Ridge, Lasso, Elastic Net 회귀

5-1. 정규화(Regularization)

혼동하기 쉬운 개념!!

(1) 표준화 (Standardization) : 데이터를 표준정규분포 N(0, 1) 형태로 변환하는 과정

(2) 정규화 (Normalization) : 데이터를 동일한 정도의 scale [0, 1]로 변환하는 과정

(3) 정규화 (Regularization) : 관계형 데이터베이스 (RDBMS)에서 중복을 최소화하고 데이터의 무결성을 위하여 데이터를 구조화하는 작업

(4) 정규화 / 규제 (Regularization) : 회귀 계수에 대한 제약 조건을 추가함으로써 분산을 감소시키는 방법

5-1. 정규화(Regularization)

		Bias		Variance
최적 모델을 위한 비용 함수	=	학습 데이터 잔차 오류 최소화	+	회귀 계수 크기 제어

Balance

5-2. Ridge 회귀

$$J(w)_{Ridge} = \sum_{i=1}^{n} (y^{(i)} - y^{\wedge(i)})^2 + \boxed{\lambda \sum_{j=1}^{m} w_j{}^2}$$

$$\text{For some } c > 0,\ \sum_{j=0}^{p} w_j^2 < c$$

- L2 규제: 제곱에 대해서 패널티를 부여하는 방법
- 규제(Regularization) + 다중공선성(Multicollinearity) 해결
- 중요하지 않은 변수도 가중치(회귀 계수)가 작아질 뿐 0이 되지는 않음
- 람다가 큰 경우 → 가중치(회귀 계수)를 작게 만들어 비용 함수 최소화 / 람다가 작은 경우 → 가중치(회귀 계수)가 커져도 비용 함수 최소화

5-3. Lasso 회귀

$$J(w)_{LASSO} = \sum_{i=1}^{n}(y^{(i)} - y^{\wedge(i)})^2 + \boxed{\lambda \sum_{j=1}^{m}|w_j|}$$

$$\text{For some } t > 0,\ \sum_{j=0}^{p}|w_j| < t$$

- L1 규제: 절대값에 대해서 패널티를 부여하는 방법
- 규제(Regularization) + 변수 선택(Model Selection)
- 중요하지 않은 변수의 가중치(회귀 계수)는 0이 될 수 있음

5-4. Elastic Net 회귀

$$J(w)_{ElasticNet} = \sum_{i=1}^{n}(y^{(i)} - y^{\wedge(i)})^2 + \boxed{\lambda_1 \sum_{j=1}^{m} w_j{}^2 + \lambda_2 \sum_{j=1}^{m} |w_j|}$$

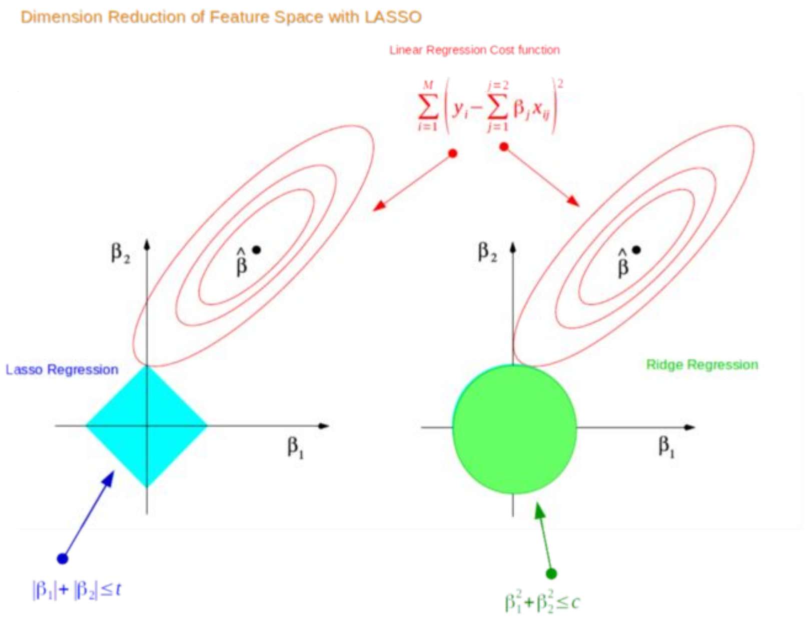
- Ridge 회귀와 Lasso 회귀를 조합한 모델
- 가중치 제곱의 합과 가중치 절대값의 합이 동시에 제약 조건이 됨

5-5. Ridge, Lasso, Elastic Net 비교

구분	Ridge 회귀	Lasso 회귀	Elastic Net 회귀
규제	L2 norm	L1 norm	L1 + L2 norm
변수 선택	불가능	가능	가능
Solution	closed form 있음	closed form 없음	closed form 없음
장점	변수 간 상관관계가 높아도 성능이 좋음	변수 간 상관관계가 높으면 성능이 떨어짐	변수 간 상관관계를 반영
특징	크기가 큰 변수를 우선적으로 줄임	중요하지 않은 변수를 우선적으로 줄임	상관관계가 큰 변수를 동시에 선택 및 배제

5-5. Ridge, Lasso, Elastic Net 비교

그런데 왜 Lasso 회귀만 변수 선택의 기능이 존재할까?



ybigta
연세대학교 빅데이터 학회

6. Logistic 회귀

6. Logistic 회귀

Recap

Classification (분류)	Regression (회귀)
↓	↓
Categorical variable (이산값)	Continuous variable (연속값)

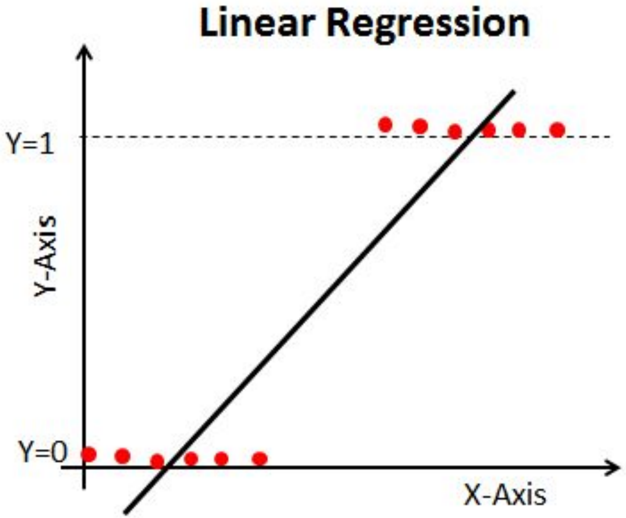
6. Logistic 회귀

종속변수가 범주형(categorical) 자료

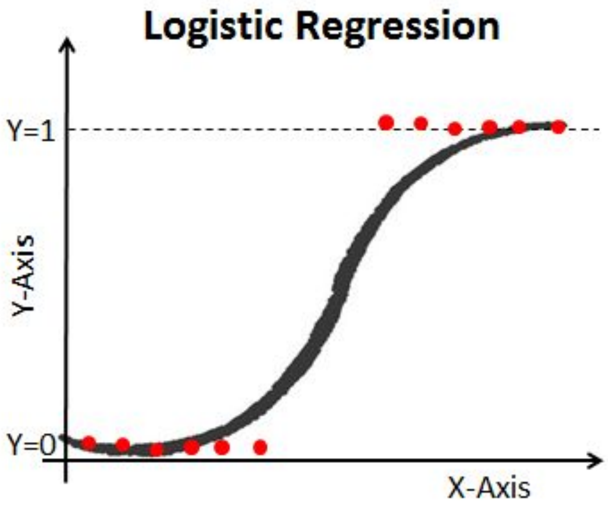
	X1	X2	…	Xp	Y (종속변수)
1			…		0
2			…		1
…	…	…	…	…	…
n-1			…		1
n			…		0

6. Logistic 회귀

Logistic 회귀란? 선형회귀 방식을 분류에 적용한 알고리즘



Continuous

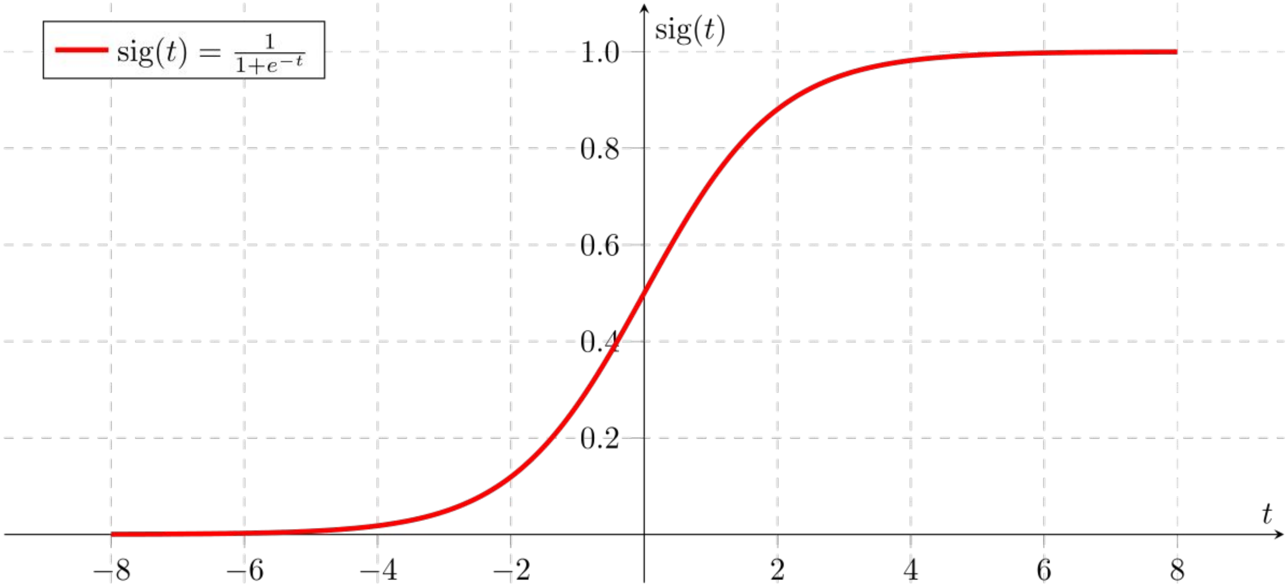


Categorical

ybigta
연세대학교 빅데이터 학회

6. Logistic 회귀

Logistic Function: Sigmoid Function



6. Logistic 회귀

$$P(Y=1 \mid X=x_1,\dots,x_p) = \beta_0 + x_1\beta_1 + \dots + x_p\beta_p$$

좌변은 [0, 1] 우변은 [−∞, ∞]

$$P(Y=1 \mid X=x_1,\dots,x_p) = \frac{e^{(\beta_0 + x_1\beta_1 + \dots + x_p\beta_p)}}{1+e^{(\beta_0 + x_1\beta_1 + \dots + x_p\beta_p)}}$$

우변에 로지스틱 함수 적용

$$\ln\left(\frac{P(Y=1 \mid X=x_1,\dots,x_p)}{1-P(Y=1 \mid X=x_1,\dots,x_p)}\right) = \beta_0 + x_1\beta_1 + \dots + x_p\beta_p$$

좌변의 관찰값을 통해
선형회귀 가능

$$\hat{p}(X) = \frac{e^{\hat{\beta}_0+\hat{\beta}_1 X}}{1+e^{\hat{\beta}_0+\hat{\beta}_1 X}}$$

7. 실습

연세대학교 빅데이터 학회

7-1. 실습 A

보스턴 주택 가격 예측

7-2. 실습 B

심장병 발병 예측

감사합니다!!

남은 교육세션도 화이팅하시고
알찬 방학이 되셨으면 좋겠습니다~~

ybigta
연세대학교 빅데이터 학회