선형 회귀 분석 (Linear Regression)

회귀 분석의 개념

사례 1

나는 큰 신발회사의 CEO이다. 많은 지점들을 가지고 있다.
그리고 이번에 새로운 지점을 내고 싶다. 어느 지역에 내야 될까?

내가 새로운 지점을 내고 싶어하는 지역들의 예상 수익만 파악할 수 있으면
큰 도움이 될 것인데!

내가 가지고 있는 자료(data)는 각 지점의 수익(profits)과 각 지점이 있는 지역의 인구수(populations)이다.

이것을 통하여, 새로운 지역의 인구수를 알게 될 경우, 그 지역의 예상 수익을 구할 수 있을까?

사례 2

나는 지금 Pittsburgh로 이사를 왔다

나는 가장 합리적인 가격의 아파트를 얻기 원한다.

그리고 다음의 조건들은 내가 집을 사기 위해 고려하는 것들이다.

square-ft^2(평방미터), 침실의 수, 학교 까지의 거리...

내가 원하는 크기와 침실의 수를 가지고 있는 집의 가격은 과연 얼마일까?

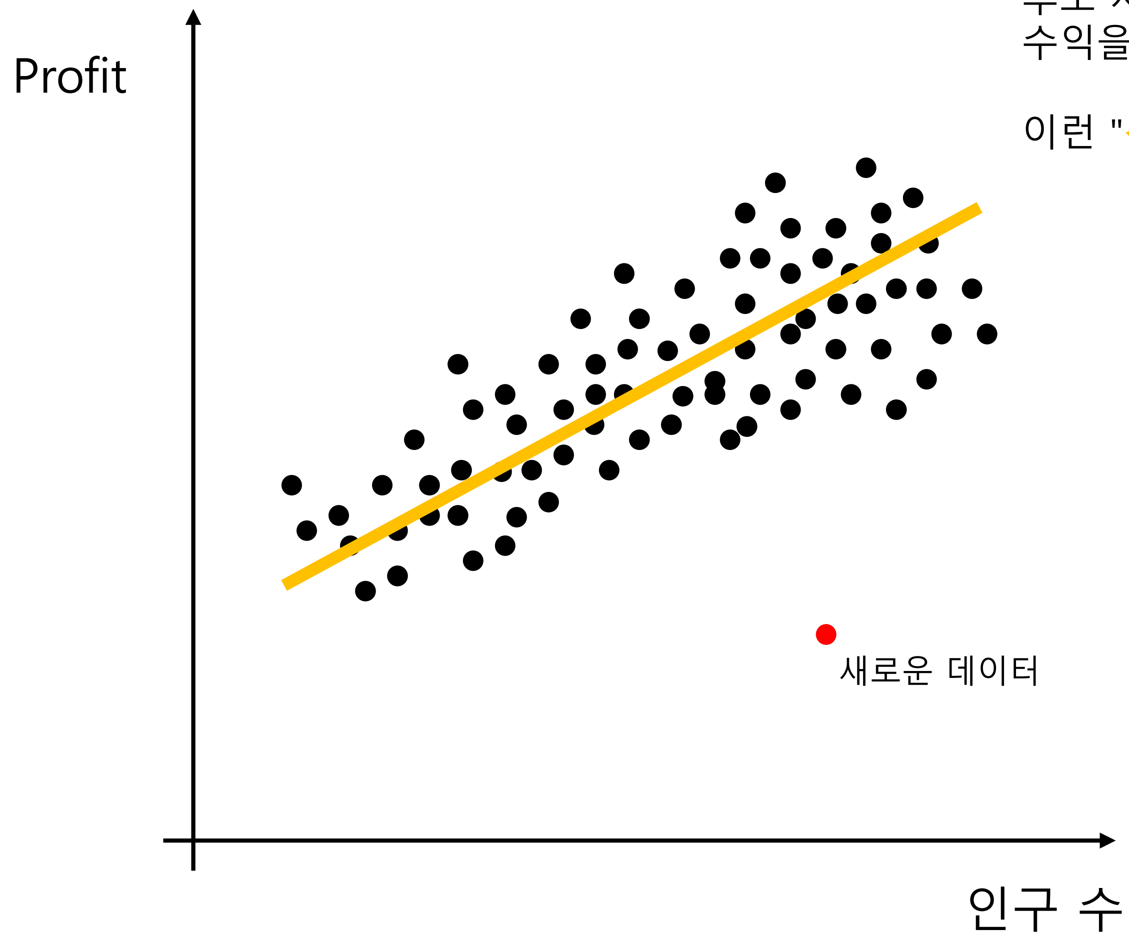
또한 방의 크기와 침실의 수가 집의 가격과 관련이 있을까?

Living area(ft^2)	# of bedroom	Rent($)
230	1	600
506	2	1000
433	2	1100
109	1	500
…	…	…
150	1.5	?

예측가능?

문제 제기

신발 판매 지점이 위치한 지역의 인구 수와 해당 판매 지점의 수익



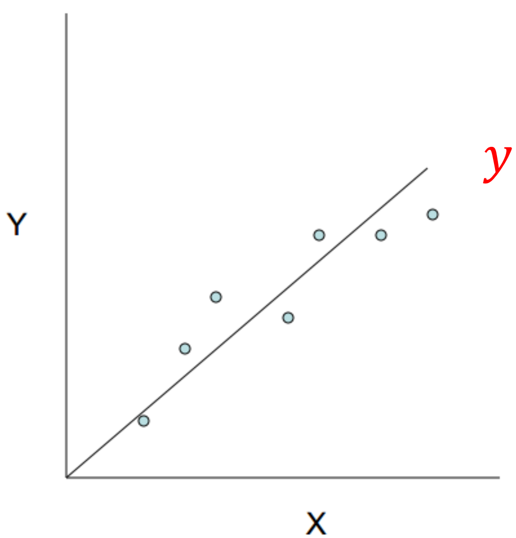
후보 지역의 인구수가 주어졌을 때, 수익을 예측?

이런 "**선**"을 찾으면 되지 않을까?

회귀 (Regression)

① Given an input x we would like to compute an output y.
(내가 원하는 집의 크기와, 방의 개수를 입력했을 때, 집 가격의 예측 값을 계산)

② For example
1) Predict height from age (height = y, age = x)
2) Predict Google`s price from Yahoo`s price (Google's price = y, Yahoo's price = x)



$$y = \theta_0 + \theta_1 x$$

즉, 기존의 data들에서 learning, training
직선($y = \theta_0 + \theta_1 x$)을 찾아내면,
새로운 값 x_{new}가 주어졌을 때,
해당하는 y의 값을 예측할 수
prediction
있겠구나!

회귀 (Regression)

Input : 집의 크기(x_1), 방의 개수(x_2), 학교까지의 거리(x_3),.....

$(x_1, x_2, \ldots, x_n)$: 특성 벡터 feature vector

$$\boldsymbol{y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n}$$

Output : 집 값(y)

training set을 통하여 학습(learning)

단순 선형 회귀 모형
(Simple Linear Regression)

최소 제곱 법 (LSM)

i번째 관찰점 (y_i, x_i)가 주어졌을 때 단순 회귀 모형은 다음과 같다.

$$y_i = \theta_0 + \theta_1 x_i + \varepsilon_i$$

종속 변수

설명 변수, 독립 변수

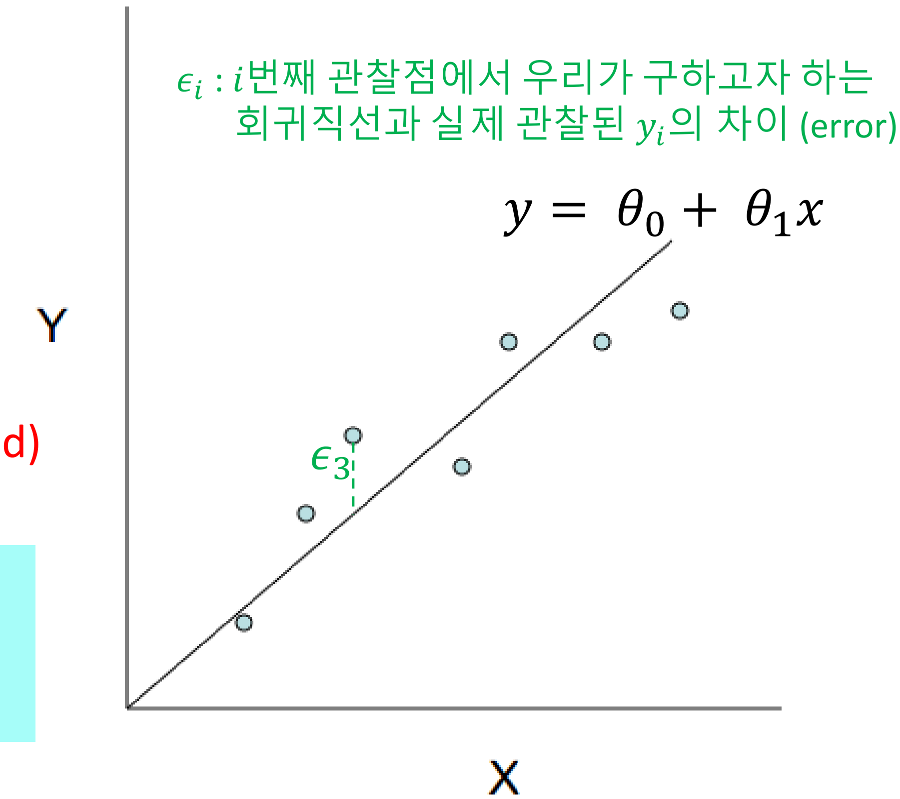
우리는 **오류의 합을 가장 작게 만드는 직선**을 찾고 싶다. 즉 그렇게 만드는 **θ_0와 θ_1을 추정**하고 싶다 !

How!! 최소 제곱 법! (Least Squares Method)

$$min \sum_i \{y_i - (\theta_0 + \theta_1 x_i)\}^2 = min \sum_i \epsilon_i^2$$

실제 관측 값

회귀 직선의 값(예측 값)



최소 제곱법 (LSM)

$$\min \sum_i \{y_i - (\theta_0 + \theta_1 x_i)\}^2 = \min \sum_i \epsilon_i^2$$

실제 관측 값   회귀 직선의 값(예측 값)

위의 식을 최대한 만족 시키는 θ_0, θ_1을 추정하는 방법은 무엇일까?

(이러한 θ_1, θ_2를 $\hat{\theta}_1, \hat{\theta}_2$ 라고 하자.)

- Normal Equation

- Steepest Gradient Descent

What is normal equation?

극대 값, 극소 값을 구할 때, 주어진 식을 미분한 후에, 미분한 식을 0으로
만드는 값을 찾는다.

$$\min \sum_i \{y_i - (\theta_0 + \theta_1 x_i)\}^2$$

먼저, θ_0에 대하여 미분하자. $$\frac{\partial}{\partial \theta_0} \sum_i \{y_i - (\theta_0 + \theta_1 x_i)\}^2 = -\sum_i \{y_i - (\theta_0 + \theta_1 x_i)\} = 0$$

다음으로, θ_1에 대하여 미분하자. $$\frac{\partial}{\partial \theta_1} \sum_i \{y_i - (\theta_0 + \theta_1 x_i)\}^2 = -\sum_i \{y_i - (\theta_0 + \theta_1 x_i)\} x_i = 0$$

위 의 두 식을 0으로 만족시키는 θ_0, θ_1를 찾으면 된다. 이처럼 2개의 미지수에 대하여,
2개의 방정식(system)이 있을 때, 우리는 이 system을 normal equation(정규방정식)이라 부른다.

The normal equation form

$\mathbb{x}_i = (1, x_i)^T,\ \ \Theta = (\theta_0, \theta_1)^T,\ \ \mathbb{y} = (y_1, y_2, \dots, y_n)^T,\ \ X = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ \dots & \dots \\ 1 & x_n \end{pmatrix},\ \ \mathbb{e} = (\epsilon_1, \dots, \epsilon_n)$ 라고 하자.

n개의 관측 값 $(x_i,\ y_i)$은 아래와 같은 회귀 모형을 가진다고 가정하자.

$$y_1 = \theta_0 + \theta_1 x_1 + \epsilon_1$$

$$y_2 = \theta_0 + \theta_1 x_2 + \epsilon_2$$

....

$$y_{n-1} = \theta_0 + \theta_1 x_{n-1} + \epsilon_{n-1}$$

$$y_n = \theta_0 + \theta_1 x_n + \epsilon_n$$

$$\Rightarrow \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ \dots \\ y_n \end{pmatrix} = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & x_3 \\ \dots & \dots \\ 1 & x_n \end{pmatrix} \begin{pmatrix} \theta_0 \\ \theta_1 \end{pmatrix} + \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \dots \\ \epsilon_n \end{pmatrix} \Rightarrow \mathbb{y} = X\Theta + \mathbb{e}$$

최소 제곱법 (LSM) Normal Equation

$$\mathbb{y} = X\Theta + \mathbb{e} \quad \Rightarrow \quad \mathbb{e} = \mathbb{y} - X\Theta$$

$$\text{Minimize} \sum_{j=1}^{n} \epsilon_j^2$$

$$\sum_{j=1}^{n} \epsilon_j^2 = \mathbb{e}^T\mathbb{e} = (\mathbb{y} - X\Theta)^T(\mathbb{y} - X\Theta)$$

1 by 1 행렬이므로
전치행렬의 값이 같다!

$$= \mathbb{y}^T\mathbb{y} - \Theta^T X^T \mathbb{y} - \mathbb{y}^T X\Theta + \Theta^T X^T X\Theta$$

$$= \mathbb{y}^T\mathbb{y} - 2\Theta^T X^T \mathbb{y} + \Theta^T X^T X\Theta$$

$$\frac{\partial(\mathbb{e}^T\mathbb{e})}{\partial\Theta} = \mathbf{0} \quad \Rightarrow \quad \frac{\partial(\mathbb{e}^T\mathbb{e})}{\partial\Theta} = -2X^T\mathbb{y} + 2X^T X\Theta = \mathbf{0}$$

$$\Rightarrow \quad X^T X\boldsymbol{\Theta} = X^T\mathbb{y} \quad \Rightarrow \quad \hat{\boldsymbol{\Theta}} = (X^T X)^{-1} X^T \mathbb{y}$$

정규방정식

최소 제곱법 (LSM)

machine learning에서는 매개 변수(parameter, 선형회귀에서는 θ_0, θ_1)가 수십~수백 차원의 벡터인 경우가 대부분이다. 또한 목적 함수(선형회귀에서는 $\Sigma\epsilon_i^2$)가 모든 구간에서 미분 가능하다는 보장이 항상 있는 것도 아니다.

따라서 <u>한 번의 수식 전개로 해를 구할 수 없는 상황이 적지 않게 있다</u>.

이런 경우에는 초기 해에서 시작하여 해를 반복적으로 개선해 나가는 수치적 방법을 사용한다. (미분이 사용 됨)

최소 제곱법 (LSM)

Gradient Descent

초기해 α_0 설정
$t = 0$

α_t가 만족스럽나?

Yes

$\hat{\alpha} = \alpha_t$

No

$\alpha_{t+1} = U(\alpha_t)$
$t = t + 1$

Gradient Descent

현재 위치에서 경사가 가장 급하게 하강하는 방향을 찾고,
그 방향으로 약간 이동하여 새로운 위치를 잡는다.
이러한 과정을 반복함으로써 가장 낮은 지점(즉 최저 점)을 찾아 간다.

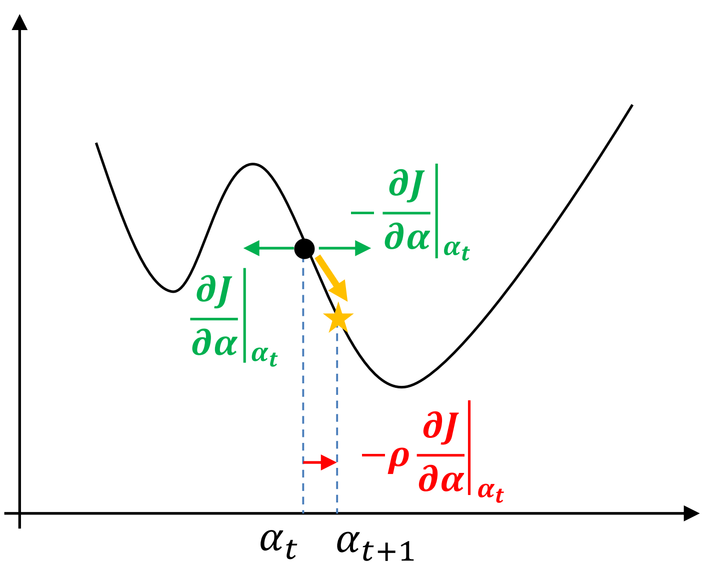
Gradient Ascent

현재 위치에서 경사가 가장 급하게 상승하는 방향을 찾고,
그 방향으로 약간 이동하여 새로운 위치를 잡는다.
이러한 과정을 반복함으로써 가장 높은 지점(즉 최대 점)을 찾아 간다.

J = 목적함수

$\left.\frac{\partial J}{\partial \alpha}\right|_{\alpha_t}$: α_t에서의 도함수 $\frac{\partial J}{\partial \alpha}$의 값

$$\boldsymbol{\alpha_{t+1} = \alpha_t - \rho \left.\frac{\partial J}{\partial \alpha}\right|_{\alpha_t}}$$



α_t에서의 미분값은 음수이다.

그래서 $\left.\frac{\partial J}{\partial \alpha}\right|_{\alpha_t}$ 를 더하게 되면 왼쪽으로 이동하게 된다.
그러면 목적함수의 값이 증가하는 방향으로 이동하게 된다.

<u>따라서 $\left.\frac{\partial J}{\partial \alpha}\right|_{\alpha_t}$ 를 빼준다.</u>
<u>그리고 적당한 ρ를 곱해주어서 조금만 이동하게 한다.</u>

J = 목적함수

$\left.\frac{\partial J}{\partial \alpha}\right|_{\alpha_t}$: α_t에서의 도함수 $\frac{\partial J}{\partial \alpha}$의 값

Gradient Descent

$$\alpha_{t+1} = \alpha_t - \rho \left.\frac{\partial J}{\partial \alpha}\right|_{\alpha_t}$$

Gradient Ascent

$$\alpha_{t+1} = \alpha_t + \rho \left.\frac{\partial J}{\partial \alpha}\right|_{\alpha_t}$$

Gradient Descent, Gradient Ascent는 전형적인 Greedy algorithm이다.
<u>과거 또는 미래를 고려하지 않고 현재 상황에서 가장 유리한 다음 위치를 찾아 Local optimal point로 끝날 가능성</u>을 가진 알고리즘이다.

$$\mathbb{x}_i = (1, x_i)^T,\quad \Theta = (\theta_0, \theta_1)^T,\quad \mathbb{y} = (y_1, y_2, \ldots, y_n)^T,\quad X = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ \ldots & \ldots \\ 1 & x_n \end{pmatrix},\quad \mathbb{e} = (\epsilon_1, \ldots, \epsilon_n)$$ 라고 하자.

미분할 때 이용.

Gradient descent를 중지하는 기준이 되는 함수

- The Cost Function

$$J(\Theta) = \frac{1}{2}\sum_{i=1}^{n}(\theta_0 + \theta_1 x_i - y_i)^2 = \frac{1}{2}\sum_{i=1}^{n}(\Theta^T \mathbb{x}_i - y_i)^2$$

- Consider a gradient descent algorithm

$$\theta_0^{t+1} = \theta_0^t - \alpha \frac{\partial}{\partial \theta_0} J(\Theta)\Big|_t$$

θ_0의 t번째 값을,
$J(\Theta)$를 θ_0으로 미분한 식에다가 대입.
그 후에, 이 값을 θ_0에서 빼 줌.

$$\theta_1^{t+1} = \theta_1^t - \alpha \frac{\partial}{\partial \theta_1} J(\Theta)\Big|_t$$

최소 제곱법 (LSM)

$\mathbb{x}_i = (1, x_i)^T$, $\Theta = (\theta_0, \theta_1)^T$, $\mathbb{y} = (y_1, y_2, \dots, y_n)^T$, $X = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ \dots & \dots \\ 1 & x_n \end{pmatrix}$, $\mathbb{e} = (\epsilon_1, \dots, \epsilon_n)$ 라고 하자.

$$J(\Theta) = \frac{1}{2}\sum_{i=1}^{n}(\theta_0 + \theta_1 x_i - y_i)^2 = \frac{1}{2}\sum_{i=1}^{n}(\Theta^T \mathbb{x}_i - y_i)^2$$

$$\nabla J(\Theta) = \left[\frac{\partial}{\partial\theta_0}J(\Theta), \frac{\partial}{\partial\theta_1}J(\Theta)\right]^T = \sum_{i=1}^{n}(\Theta^T \mathbb{x}_i - y_i)\mathbb{x}_i$$

Gradient of $J(\Theta)$

$$\frac{\partial}{\partial\theta_0}J(\theta) = \sum_{i=1}^{n}(\Theta^T \mathbb{x}_i - y_i)\,1 \qquad \frac{\partial}{\partial\theta_1}J(\theta) = \sum_{i=1}^{n}(\Theta^T \mathbb{x}_i - y_i)\,x_i$$

$\mathbb{x}_i = (1, x_i)^T, \ \ \Theta = (\theta_0, \theta_1)^T, \ \ \mathbb{y} = (y_1, y_2, \dots, y_n)^T, \ \ X = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ \dots & \dots \\ 1 & x_n \end{pmatrix}, \ \ \mathbb{e} = (\epsilon_1, \dots, \epsilon_n)$ 라고 하자.

$$\theta_0^{t+1} = \theta_0^t - \alpha \sum_{i=1}^{n} (\Theta^T \mathbb{x}_i - y_i)\, 1$$

$$\theta_1^{t+1} = \theta_1^t - \alpha \sum_{i=1}^{n} (\Theta^T \mathbb{x}_i - y_i) x_i$$

단, 이 때의 Θ자리에는
t번째에 얻어진 Θ값을 대입해야 한다.

최소 제곱법 (LSM)

Normal Equations

장점 : a single-shot algorithm! Easiest to implement.

단점 : need to compute pseudo-inverse $(X^T X)^{-1}$, expensive, numerical issues (e.g., matrix is singular..), although there are ways to get around this ...

$$\hat{\mathbb{e}} = (X^T X)^{-1} X^T \mathbb{y}$$

Steepest Descent

장점 : easy to implement, conceptually clean, guaranteed convergence

단점 : often slow converging

$$\Theta^{t+1} = \Theta^t - \alpha \sum_{i=1}^{n} \{(\Theta^t)^T \mathbb{x}_i - y_i\} \mathbb{x}_i$$

다중 선형 회귀 모델
(Multivariate Linear Regression)

Multi Linear Regression

단순 선형 회귀 분석은, input 변수가 1.

다중 선형 회귀 분석은, input 변수가 2개 이상.

$$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

Google의 주식 가격

Microsoft의 주식 가격

Yahoo의 주식 가격

Multi Linear Regression

예를 들어, 아래와 같은 식을 선형으로 생각하여 풀 수 있는가?

$$\boldsymbol{y = \theta_0 + \theta_1 x_1^2 + \theta_2 x_2^4 + \epsilon}$$

물론, input 변수가 polynomial(다항식)의 형태이지만, coefficients θ_i가 선형(linear)이므로 선형 회귀 분석의 해법으로 풀 수 있다.

$$\hat{\boldsymbol{\Theta}} = (X^T X)^{-1} X^T \mathbb{y}$$

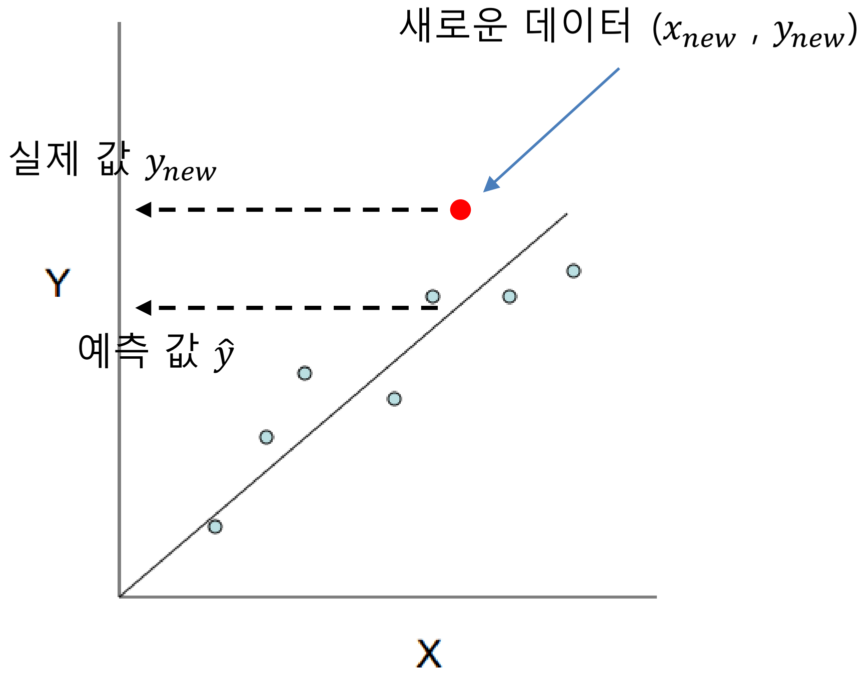
$(\theta_0, \theta_1, \dots, \theta_n)^T$

모델의 성능 측정

일반화 (generalization)

- 성능이 좋은 모델
  - 훈련 데이터로 모델의 학습을 끝낸 후, 훈련 데이터와 같은 특성을 가진 새로운 데이터가 주어졌을 때, 정확하게 예측할 수 있는 모델
- 일반화 (generalization)
  - 훈련 데이터에 의해 학습이 완료된 모델이, 새로운 데이터에 대해 그 값을 정확하게 예측할 수 있는 것

ex)
지역 주민수(X) 와 신발가게 매출(Y)
데이터로 모델을 학습하였다.
새로운 데이터 x_{new} 가 주어졌을 때,
신발가게 매출을 정확히 예측할 수 있을까?
즉 y_{new}과 $\hat{y}$가 같을까?



모델의 복잡성

고객 성별, 자녀 수, 직장 데이터, 구매 데이터

학습 (training)

고객이 맥북을 구매할 것인지를 예측하는 모델

간단한 모델

복잡한 모델

20대~40대 남성의 고객들은
맥북을 구매할 것이다.

20살~40살 남성이고,
자녀가 없고,
분당에 살고,
컴퓨터 회사에 다니는 고객들은
맥북을 구매할 것이다.

?

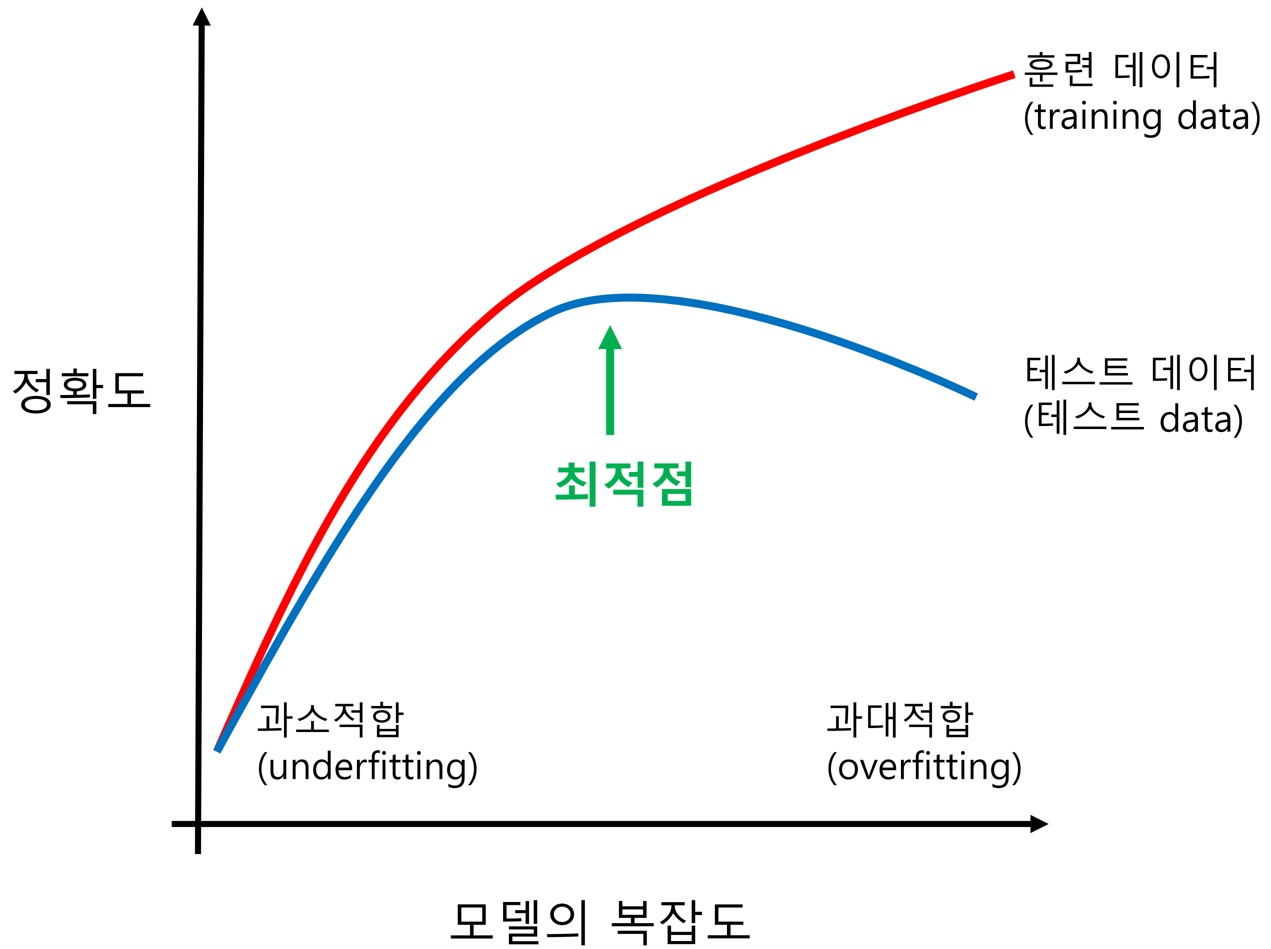
new client!
(41살, 1 아들, 컴퓨터 회사)

과적합 1

트레이닝 데이터에서 발생한 오차와 테스트 데이터에서 발생한 오차 사이에 아주 심각한 차이가 있다는 것이다.

- 과대적합 (Overfitting)
  - 주로 복잡한 모델에서 발생
  - 모델이 훈련 세트의 각 샘플에 너무 가깝게 맞춰져서 새로운 데이터에 일반화되기 어려운 경우
  - Training set에 대한 오차가 Test set에 대한 오차보다 심각하게 작은 경우 발생한다.
- 과소적합 (Underfitting)
  - 주로 간단한 모델에서 발생
  - 데이터의 면면과 다양성을 잡아내지 못하며 훈련 세트에도 잘 맞지 않은 경우
- Bias-Variance Trade off
  - Bias가 크면, 과소적합의 경향성이 크다.
  - Variance가 크면, 과대적합의 경향성이 크다.

과적합 2


훈련 데이터
(training data)
정확도
테스트 데이터
(테스트 data)
최적점
과소적합
(underfitting)
과대적합
(overfitting)
모델의 복잡도

성능 측정 Metric 1

- MSE (Mean Square Error)
  - target과 예측값의 차이의 제곱의 평균
  - $\frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{y_i})^2$

- MAE (Mean Absolute Error)
  - target과 예측값의 차이의 절대값의 평균
  - $\frac{1}{n}\sum_{i=1}^{n}|y_i - \widehat{y_i}|$

- RMSE (Root Mean Square Error)
  - MSE에 root를 사용한 값

성능 측정 Metric 2

- R^2 (결정계수)
  - 모델이 주어진 데이터에 얼마나 적합한지 알아볼 수 있는 척도
  - 종속 변수의 총 변화량 중 모델이 잡아낼 수 있는 변화량
  - 재조정된(정규화 된) MSE

$$R^2 = 1 - \frac{SSE}{SST} = 1 - \frac{\frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{y_i})^2}{\frac{1}{n}\sum_{i=1}^{n}\left(y_i - \mu_y\right)} = 1 - \frac{MSE}{Var(y)}$$

규제 (Regularization)

규제 (Regularization)

- 모델의 복잡도를 통제하고, 과적합을 피하기 위해, 가중치의 절대값을 최소화 하는 것

$$\mathbb{x}_i = (1, x_i)^T, \quad \Theta = (\theta_0, \theta_1)^T$$

- Linear Regression

$$\min \sum_i \{y_i - (\theta_0 + \theta_1 x_i)\}^2$$

L²-Regularization

- Ridge Regression

$$\min \sum_i \{y_i - (\theta_0 + \theta_1 x_i)\}^2 + \alpha \sum_i \theta_i^2$$

L¹-Regularization

- Lasso Regression

$$\min \sum_i \{y_i - (\theta_0 + \theta_1 x_i)\}^2 + \alpha \sum_i \theta_i$$

- ElasticNet

$$\min \sum_i \{y_i - (\theta_0 + \theta_1 x_i)\}^2 + l_1 \alpha \sum_i \theta_i + \frac{1}{2}(1 - l_1)\alpha \sum_i \theta_i^2$$

Ridge Regression

$$\min \sum_i \{y_i - (\theta_0 + \theta_1 x_i)\}^2 + \alpha \sum_i \theta_i^2$$

- 계수(θ)의 값을 작게 하여, 과대적합을 방지하는 것이 목적
- 패널티 항 : L_2 Regularization
- α가 0이면 Linear regression과 동일하다.
- α가 커질수록
  - 계수(θ)의 값은 작아지며, 모델의 복잡도가 낮아진다.
  - 과적합을 막아주며, Test set에서의 성능은 상대적으로 좋아진다.
- α의 최적값은 데이터셋에 따라서 달라진다.
- 데이터가 충분히 많을 경우
  - 과대적합하기 어려워, Training set에 대한 회귀 모형의 성능이 떨어진다.
  - Linear regression의 성능 또한 Ridge regression의 성능과 비슷해진다.

Lasso Regression

$$\min \sum_i \{y_i - (\theta_0 + \theta_1 x_i)\}^2 + \alpha \sum_i |\theta_i|$$

- 계수(θ)의 값을 작게 하여, 과대적합을 방지하는 것이 목적
- 패널티 항 : L_1 Regularization
- α가 커질수록
  - 가중치(θ)의 값은 작아지며, 모델의 복잡도가 낮아진다.
  - 과적합을 막아주며, Test set에서의 성능은 상대적으로 좋아진다.
- Lasso를 사용하는 경우,
  - 0이 되는 계수가 발생할 수 있다.
  - 모델에서 완전히 제외되는 특성(feature)가 발생할 수 있다.
  - 특성 선택(feature selection)이 자동으로 이루어진다.