

Linear Regression

Jeonghun Yoon

Machine learning

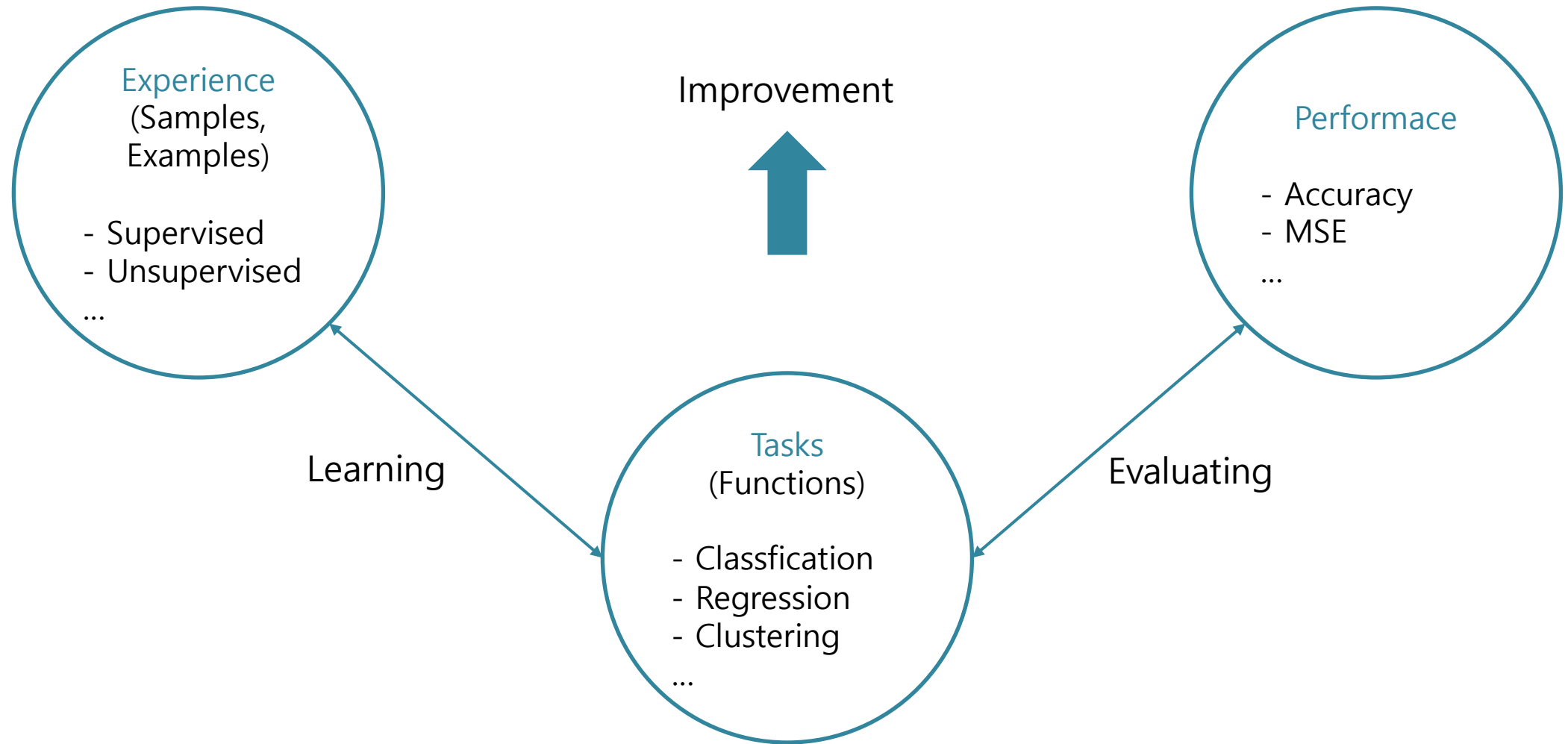
A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P , if its performance at tasks in T , as measured by P , improves with experience E .

Machine Learning by Tom M. Mitchell

컴퓨터 프로그램은 다양한 기능을 수행한다.
그 기능을 잘 수행했는지 그렇지 못했는지에 대해 성능을 평가 할 수 있다.

머신러닝은 컴퓨터가 수행하는 기능을, dataset을 통하여 더 나은 성능을 낼 수 있도록 개선하는 것이다.

Machine learning



머신러닝 학습 알고리즘의 카테고리

Broadly,

Supervised Learning algorithm

- Random sample vectors $\mathbf{x}'s$ 와 각각에 대한 t 가 관찰되었을 때, 제공된 dataset을 통해 모든 \mathbf{x} 에 대해 정답인 t 를 유추해 낼 수 있도록 일반화시키는 알고리즘(예제를 통한 학습 알고리즘)
- Roughly $P(t|\mathbf{x})$
- Classification, Regression, ..
- Input, target

Unsupervised Learning algorithm

- Random sample vectors $\mathbf{x}'s$ 가 관찰되었을 때, \mathbf{x} 의 distribution or dataset 구조의 useful properties를 찾고자 하는 알고리즘.
- Roughly $P(\mathbf{x})$
- Density estimation, Clustering, Dimensional reduction
- Input

Reinforcement Learning algorithm

....

머신러닝에서의 기본적인 용어들

Input / Input vector / Features / Feature vectors / 입력 / 특성

- 입력 벡터는 머신러닝 알고리즘의 입력으로 주어진 데이터를 의미한다. Input data가 총 m 차원이면, $\mathbf{x} = (x_1, x_2, \dots, x_m)$ 라고 표현한다. 벡터의 요소인 x_i 는 entity이다.

Output / Output vector / 출력값

- 출력 벡터는 머신러닝 알고리즘의 출력 결과 데이터를 의미한다. 출력 벡터의 각 entity는 알고리즘의 종류에 따라서 연속적인 실수값이 될 수 있고, 불연속적인 정수값이 될 수 있다. 또한 1차원이 될 수도 있고 다차원이 될 수도 있다. 출력은 회귀모형을 설명할 때는 \hat{y} 라고 표현하겠다.

Target / Label / 목표값 / 라벨

- 목표값은 입력 벡터와 함께 pair로 주어진, 즉 \mathbf{x} 에 associated 된 데이터를 의미한다. 실수값, 정수값을 가질 수 있다. 타겟은 회귀모형을 설명할 때는 y 라고 표현하겠다.

머신러닝에서의 기본적인 용어들

Training set

- 알고리즘을 학습할 때 사용되는 input dataset (supervised learning의 경우는 target도 training set에 포함)

Validation set

- 알고리즘을 학습할 때, 알고리즘의 성능을 (중간에) 측정하여 성능이 나쁜 알고리즘을 구별해내고(prune) 좋은 알고리즘을 선택할 수 있도록 하는 과정에서 사용되는 dataset

Test set

- 학습된 알고리즘을 최종 테스트하여 알고리즘의 성능을 평가할 때 사용하는 dataset

Hypothesis

- 우리가 찾고자하는 알고리즘을 의미한다. 머신러닝 알고리즘을 학습하는 것은, 결국 머신러닝 알고리즘이 존재하는 전체 셋(hypothesis set)에서 최상의 머신러닝 알고리즘(hypothesis)을 찾는 것이다.

Terms in Linear Regression

Linear Regression

Mean Squared Error

Batch / Stochastic / Mini-batch Gradient Descent

Overfitting / Underfitting

Bias Variance Trade-off

Generalization

Lasso

Ridge

ElasticNet

Example 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 구
할 수 있다.

Example 2)

나는 판교로 이사 갈 예정이다.

나는 가장 합리적인 가격의 아파트를 얻기 원한다.


그리고 다음의 조건들은 내가 집을 사기 위해 고려하는 것들이다.

평수, 침실의여기에 수식을 입력하십시오. 수, 학교 까지의 거리...

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

평수	방의 개수	집 값
34	3	8억
27	2	7억
43	4	9.5억
32	4	11억
...		

Example 3)



House Prices: Advanced Regression Techniques

Predict sales prices and practice feature engineering, RFs, and gradient boosting
4,259 teams · Ongoing

[Overview](#) [Data](#) [Kernels](#) [Discussion](#) [Leaderboard](#) [Rules](#) [Team](#) [My Submissions](#) [Submit Predictions](#)

Overview

Description

Evaluation


Tutorials

Frequently Asked Questions

Start here if...

You have some experience with R or Python and machine learning basics. This is a perfect competition for data science students who have completed an online course in machine learning and are looking to expand their skill set before trying a featured competition.

Competition Description

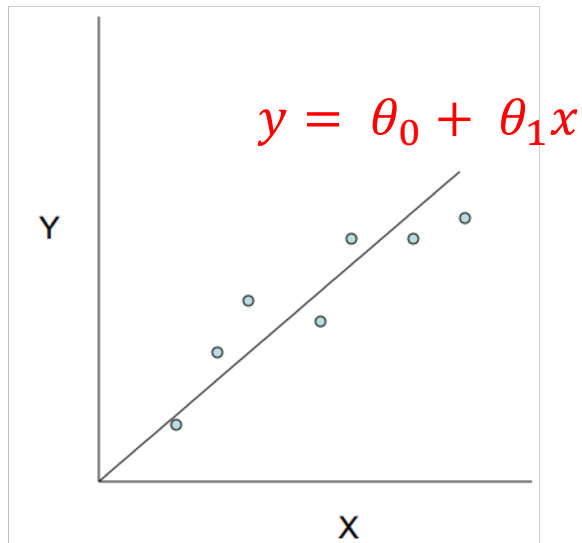


Ask a home buyer to describe their dream house, and they probably won't begin with the height of the basement ceiling or the proximity to an east-west railroad. But this playground competition's dataset proves that much more influences price negotiations than the number of bedrooms or a white-picket fence.

With 79 explanatory variables describing (almost) every aspect of residential homes in Ames, Iowa, this competition challenges you to predict the final price of each home.

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)



function / hypothesis

즉, 기존의 data들에서
함수/가설 ($y = \theta_0 + \theta_1 x$)을 찾아내면,
새로운 값 x_{new} 가 주어졌을 때,
해당하는 y 의 값을 예측할 수
있겠구나!

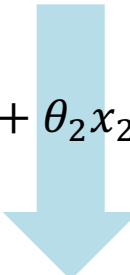
learning, training

prediction

Regression

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

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


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



Output : 집 값(y) 즉 연속적인 실수 값

training set을 통하여 학습(learning) : 학습한다는 의미는 $\theta_0, \theta_1, \dots, \theta_n$ 의 값을 찾는것이다.

Simple Linear Regression

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

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

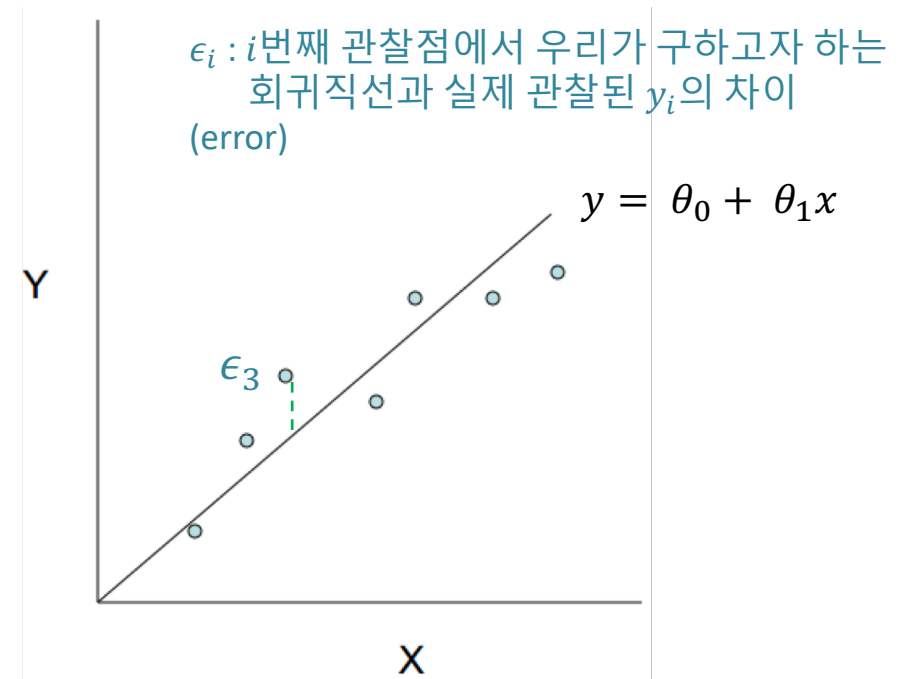
종속 변수 \nearrow \nwarrow 설명 변수, 독립 변수

우리는 오류의 합을 가장 작게
만드는 직선을 찾고 싶다. 즉 그렇게
만드는 θ_0 와 θ_1 을 추정하고 싶다.
어떻게 추정하면 좋을까?

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 \nearrow \nwarrow 회귀 직선의 값(이상적인 값)



Why least square method?

최소 제곱 합(Least square method) 을 이용하여 선형 회귀 모델을 구하는 **정당성**은 무엇일까?

정당성을 이야기하기 전에 앞서, 최소 제곱 합을 이용하여 선형 회귀 모델을 구하는 것부터 살펴보자.

Simple Linear Regression

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

실제 관측 값
(target)

회귀 직선의 값(이상적인 값)

위의 식을 최대한 만족 시키는 θ_0, θ_1 을 추정하는 방법은 무엇일까?
(추정하고자 하는 θ_1, θ_2 를 $\hat{\theta}_1, \hat{\theta}_2$ 라고 하자.)

- Normal Equation
- Steepest Gradient Descent

Normal Equation

2개의 미지수에 대하여, 2개의 방정식(system)이 있을 때, 우리는 이 system을 normal equation(정규방정식)이라 부른다.

- Cost function

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

- 먼저, θ_0 에 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_1 에 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_1 를 찾으면 된다.

Normal Equation

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

n 개의 관측 값 (x_i, y_i) 은 아래와 같은 회귀 모델을 가진다고 가정하자.

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

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

.....

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

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

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

Normal Equation

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

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

$$\begin{aligned} \sum_{j=1}^n \epsilon_j^2 &= \mathbf{e}^T \mathbf{e} = (\mathbf{y} - X\Theta)^T (\mathbf{y} - X\Theta) \\ &= \mathbf{y}^T \mathbf{y} - \Theta^T X^T \mathbf{y} - \mathbf{y}^T X \Theta + \Theta^T X^T X \Theta \\ &= \mathbf{y}^T \mathbf{y} - 2\Theta^T X^T \mathbf{y} + \Theta^T X^T X \Theta \end{aligned}$$

1 by 1 행렬이므로
전치행렬의 값이 같다!

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

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

정규방정식

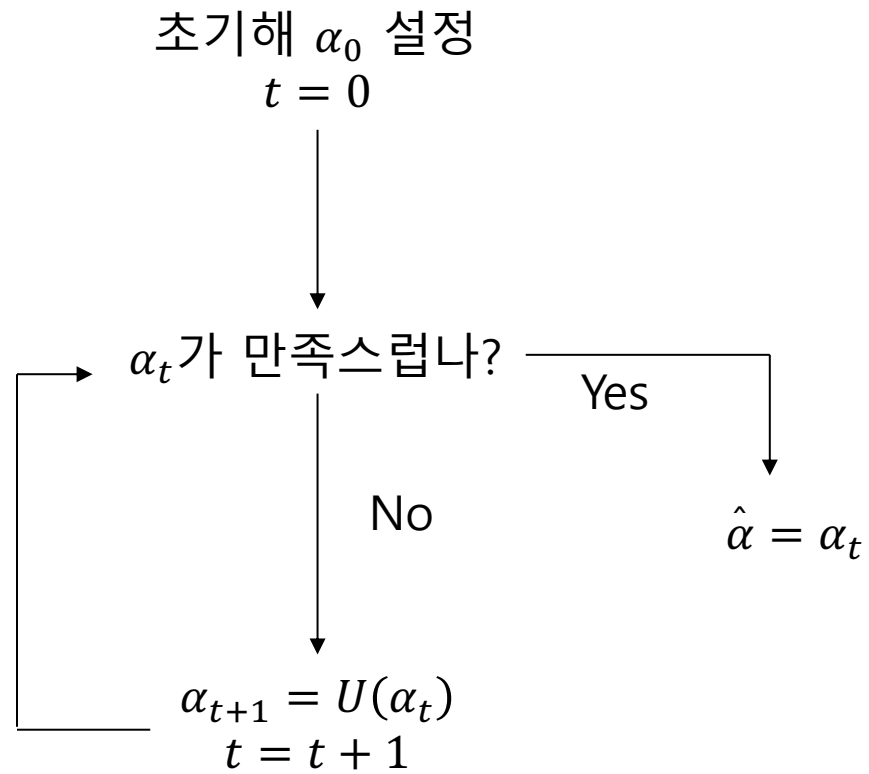
Gradient Descent (Ascent)

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

따라서 한 번의 수식 전개로 해를 구할 수 없는 상황이 적지 않게 있다.

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

Gradient Descent (Ascent)



Gradient Descent (Ascent)

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.

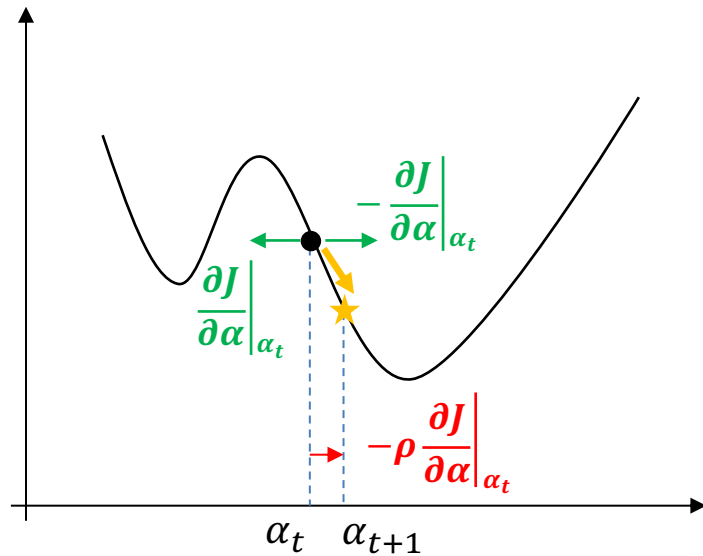
이러한 과정을 반복함으로써 가장 높은 지점(즉 최대 점)을 찾아 간다.

Gradient Descent (Ascent)

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

J = 목적함수

$\frac{\partial J}{\partial \alpha} \Big|_{\alpha_t}$: α_t 에서의 도함수 $\frac{\partial J}{\partial \alpha}$ 의 값



α_t 에서의 미분값은 음수이다.

그래서 $\frac{\partial J}{\partial \alpha} \Big|_{\alpha_t}$ 를 더하게 되면

왼쪽으로 이동하게 된다.

그러면 목적함수의 값이 증가하는
방향으로 이동하게 된다.

따라서 $\frac{\partial J}{\partial \alpha} \Big|_{\alpha_t}$ 를 빼준다.

그리고 적당한 ρ 를 곱해주어서 조금만
이동하게 한다.

Gradient Descent (Ascent)

Gradient Descent

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

J = 목적함수

$\frac{\partial J}{\partial \alpha} \Big|_{\alpha_t}$: α_t 에서의 도함수 $\frac{\partial J}{\partial \alpha}$ 의 값

Gradient Ascent

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

Gradient Descent, Gradient Ascent는 전형적인 Greedy algorithm이다.

과거 또는 미래를 고려하지 않고 현재 상황에서 가장 유리한 다음 위치를 찾아

Local optimal point로 끝날 가능성을 가진 알고리즘이다.

Gradient Descent in Linear Regression

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

Cost function

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

미분할 때 이용.

Gradient descent를 중지하는
기준이 되는 함수

Gradient descent

- $\theta_0^{t+1} = \theta_0^t - \alpha \frac{\partial}{\partial \theta_0} J(\Theta)_{at\ t}$

θ_0 의 t 번째 값을,

$J(\Theta)$ 를 θ_0 으로 미분한 식에다가 대입.

그 후에, 이 값을 θ_0 에서 빼 줌.


- $\theta_1^{t+1} = \theta_1^t - \alpha \frac{\partial}{\partial \theta_1} J(\Theta)_{at\ t}$

Gradient Descent in Linear Regression

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

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

Gradient of $J(\Theta)$


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

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

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

Gradient Descent in Linear Regression

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

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

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

단, 이 때의 Θ 자리에는
 t 번째에 얻어진 Θ 값을 대입해야 한다.

Which one?

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 ...

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

Steepest Descent

장점 : easy to implement, conceptually clean, guaranteed convergence

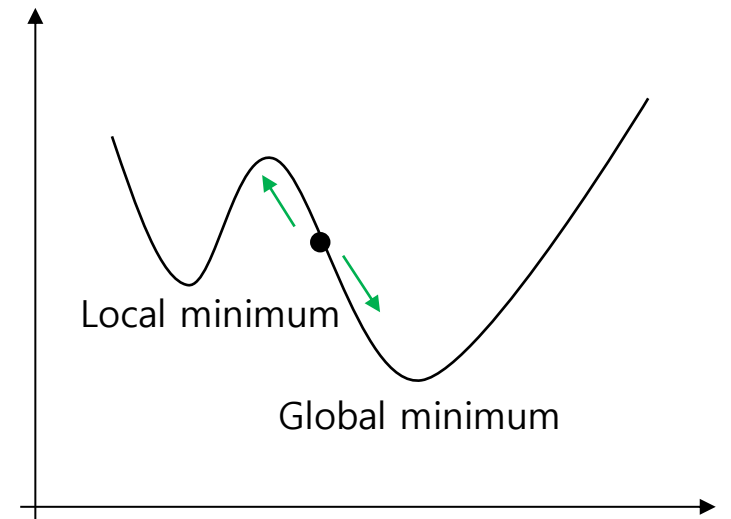
단점 : often slow converging

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

Batch Gradient Descent

$$\theta_j^{t+1} = \theta_j^t - \alpha \sum_{i=1}^n (\Theta^T \mathbb{x}_i - y_i) x_i^{(j)} \quad \text{for every } j$$

- 각 스텝에서 전체 training set X 에 대해 계산한 후에 Θ 를 1번 업데이트한다.
- Training set의 크기가 크다면 매우 느리다.
- Global minimum에 수렴한다.



Stochastic Gradient Descent

$$\text{for } i = 1 \text{ to } n : \{ \theta_j^{t+1} = \theta_j^t - \alpha(\Theta^T \mathbf{x}_i - y_i)x_i^{(j)} \text{ for every } j \}$$

- 각 스텝에서 한 개의 sample을 무작위로 선택하고, 하나의 샘플에 대한 gradient 를 계산한다.
 - 각 스텝에서 매우 적은 데이터만 처리하기 때문에 알고리즘의 속도가 빠르다.
 - 매 반복에서 하나의 샘플만 메모리에 있으면 되므로 매우 큰 훈련 세트도 훈련시킬 수 있다.
- 비용 함수가 최솟값에 다다를 때까지 부드럽게 감소하지 않고, 위아래로 불안정하게 요동치면서 평균적으로 감소한다.

(Randomness)

- 알고리즘이 멈출 때 좋은 θ 가 구해지겠지만 optimal value는 아닐 수 있다.
- Randomness로 인하여 local minimum을 탈출할 수는 있지만, global minimum에 수렴하지는 않는다. (근접하기는 함)
- 이를 해결하는 방법 중 하나는 learning rate를 점진적으로 감소시키는 것이다.
 - 학습률이 너무 빨리 줄어들면 지역 최솟값에 갇히거나, 최솟값까지 가는 도중에 멈출 수 있다.
 - 학습률이 너무 천천히 줄어들면 오랫동안 최솟값 주변을 맴돌 수 있다.

Mini-batch Gradient Descent

- 각 스텝에서 mini batch라 부르는 임의의 작은 샘플 세트(m)에 대해 gradient를 계산한다.
- Mini-batch의 크기를 어느 정도 크게 하면 SGD보다 덜 불규칙하게 움직인다.
 - SGD보다 global minimum에 더 가깝게 근접하게 된다.
 - 반면 SGD보다 local minimum에서 빠져나오기 더 힘들 수 있다.

선형회귀 모형의 Probabilistic interpretation

(Question)

Least-squares function은 선형회귀모형의 합리적인 cost function인가?

(Answer)

$y_i = \theta^T \mathbf{x}_i + \epsilon_i$ 이라 하자. ϵ_i 는 데이터의 noise 또는 error라고 하자.

ϵ_i 는 가우시안 분포를 따르고 IID(Independently and identically distributed) 를 따르는 변수라고 하자.

- $\epsilon_i \sim N(0, \sigma^2)$

ϵ_i 의 분포는 $p(\epsilon_i) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{\epsilon_i^2}{2\sigma^2}\right)$ 와 같다.

이것은 $p(y_i | \mathbf{x}_i; \theta) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y_i - \theta^T \mathbf{x}_i)^2}{2\sigma^2}\right)$ 과 같다. 즉 $y_i | \mathbf{x}_i; \theta \sim N(\theta^T \mathbf{x}_i, \sigma^2)$ 이다.

선형회귀 모형의 Probabilistic interpretation

그러면 regression에서 관심있어하는 $X = (\mathbf{x}_1, \dots, \mathbf{x}_n)$ 와 θ 가 주어졌을 때, $Y = (y_1, \dots, y_n)$ 의 분포, 즉 $p(Y|X; \theta)$ 는 무엇인가?
이 분포를 θ 의 분포로 보려고 한다면, 통계적으로 많이 사용되는 likelihood function을 사용할 수 있다.

즉, $L(\theta) = L(\theta; X, Y) = p(Y|X; \theta)$

ϵ_i 는 IID이므로, $L(\theta) = \prod_{i=1}^n p(y_i|\mathbf{x}_i; \theta) = \prod_{i=1}^n \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y_i - \theta\mathbf{x}_i)^2}{2\sigma^2}\right)$

Parameter θ 를 선택하는 좋은 방법 중 하나는 **maximum likelihood estimation**를 구하는 것이다. 즉 X, Y 를 얻을 수 있는 확률이 가장 높은 parameter를 선택하는 것이다.

Likelihood를 최대화 하는 것을 Log likelihood를 최대화 하는 것으로 바꾸어 계산한다.

선형회귀 모형의 Probabilistic interpretation

$$l(\theta) \\ = \log L(\theta) = \log \prod_{i=1}^n \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y_i - \theta^T \mathbf{x}_i)^2}{2\sigma^2}\right) = \sum_{i=1}^n \log \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y_i - \theta^T \mathbf{x}_i)^2}{2\sigma^2}\right) = n \log \frac{1}{\sqrt{2\pi}\sigma} - \frac{1}{\sigma^2} \frac{1}{2} \sum_{i=1}^n (y_i - \theta^T \mathbf{x}_i)^2$$

따라서 $l(\theta)$ 를 극대화하는 것은 $\frac{1}{2} \sum_{i=1}^n (y_i - \theta^T \mathbf{x}_i)^2$ 을 극소화하는 것과 똑같다. 그리고 이것은 least squares cost function 과 동일하다.

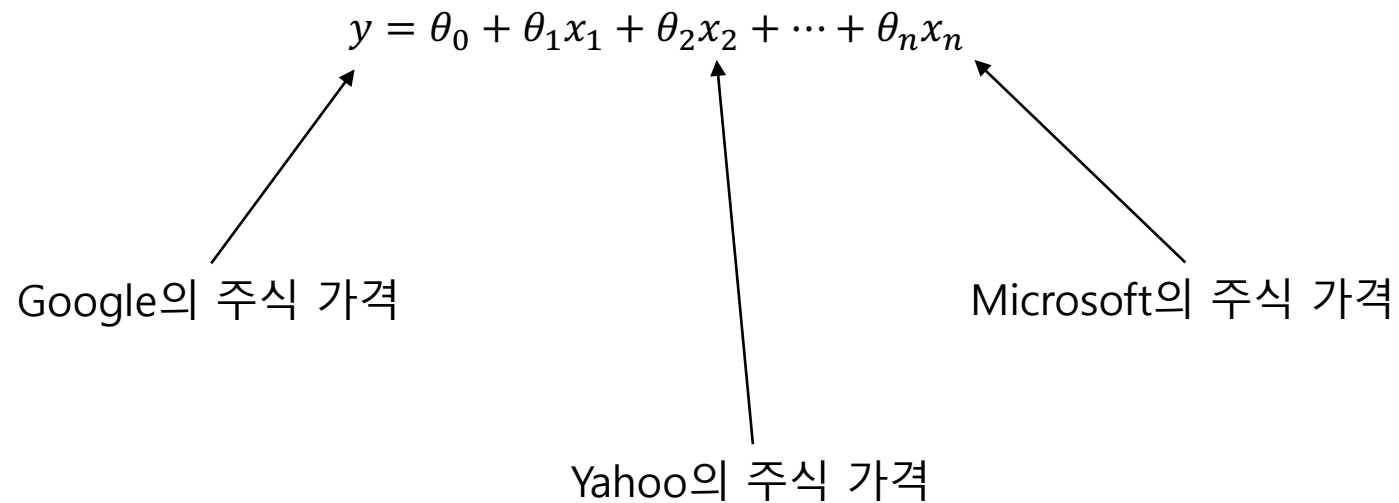
결론적으로, 데이터에 대한 probability assumption이 주어지면, **least-squares regression은 θ 의 maximum likelihood estimate를 찾는 것과 동일하다.**

Least-squares regression은 maximum likelihood estimate을 찾는 아주 자연스러운 방법이라고 정당화 될 수 있다.

Multivariate Linear Regression

단순 선형 회귀 분석은, input 변수가 1개

다중 선형 회귀 분석은, input 변수가 2개 이상




Multivariate Linear Regression

예를 들어, 아래와 같은 식을 선형으로 생각하여 풀 수 있는가?

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

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

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

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

$$y = \theta_0 + \theta_1 g_1(x_1) + \theta_2 g_2(x_2) + \dots + \theta_n g_n(x_n)$$

중 회귀 분석은?

Overfitting vs Underfitting

학습 시, 알고리즘이 얼마나 더 복잡한 것을 학습하면서 이를 얼마나 더 잘 **일반화**하고 있는지 알아야 함

모델이 학습셋 이외의 데이터에 대해서도
좋은 예측 성능을 보여주는 것

Overfitting 과 underfitting 은 무엇인가?

- Underfitting : 모델이 데이터의 특성(또는 structure) 및 패턴을 제대로 파악하지 못한 경우
- Overfitting : 모델이 학습 데이터의 특성 및 패턴, 더 나아가 데이터의 오류와 부정확성까지 학습하게 되어, 학습 데이터 이외의 데이터에 일반화(적용) 할 수 없는 경우

Overfitting 과 underfitting 을 극복하는 방법에는 무엇이 있는가?

- Underfitting :
 - 복잡한 모델을 사용.
 - 좋은 특성을 더 추가하거나 선택해야 함.
- Overfitting :
 - 검증 오차와 훈련 오차의 차이가 비슷해 질 때까지 훈련 데이터를 추가.
 - 모델을 규제해야 함.

Bias Variance Trade-off

모델의 일반화 오차(Generalization error)는 training set 외의 데이터에 대한 expected error

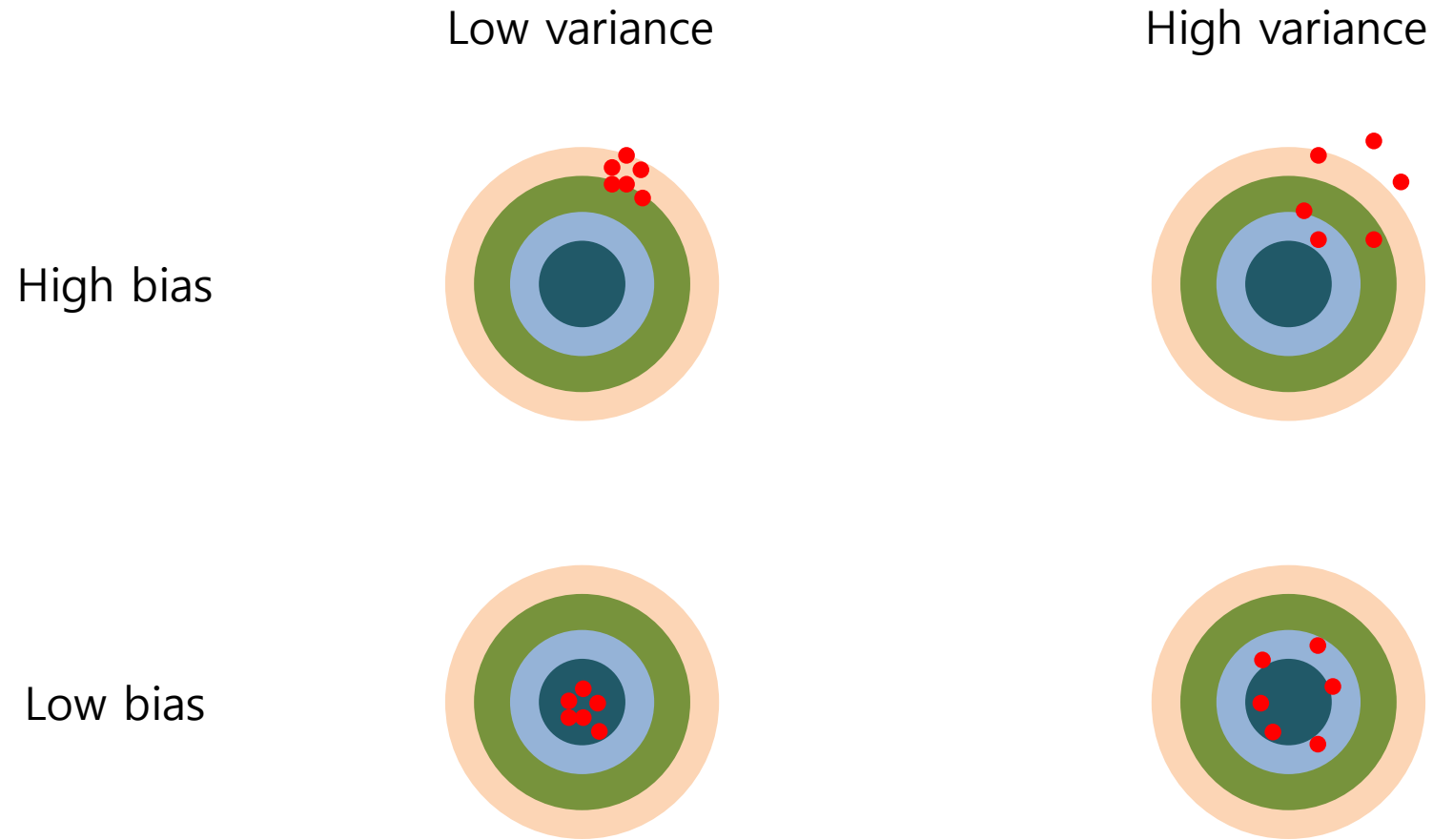
모델의 일반화 오차는 다음이 3가지 오차로 구성.

- Bias (편향) : 잘못된 가정(Hypothesis)으로 인한 오차. 모델의 예측값과 실제값의 차이로 생기는 오차. 편향이 큰 모델은 훈련 데이터에 underfitting 되기 쉬움.
- Variance (분산) : Training set이 변할 때, 모델 예측값의 변동을 나타내는 오차. 모델이 **특정한** training set에 얼마나 'over-specialized' 되었는지를 나타냄. 분산이 큰 모델은 훈련 데이터에 overfitting 되기 쉬움.
- Irreducible error (노이즈) : 데이터에 내제되어 있는 노이즈

모델의 복잡도가 커지면(complex) 통상적으로 분산이 늘어나고 편향은 줄어듦

모델의 복잡도가 줄어들면(simple) 편향이 늘어나고 분산은 줄어듦

Bias Variance Trade-off



Bias Variance Trade-off

$Y = f(X) + \epsilon$ where $\epsilon \sim N(0, \sigma_\epsilon)$

$\hat{f}(X)$: our model

D_n : training data of size n

$$Err(\hat{f})$$

$$= E_{X,Y,D_n} [(Y - \hat{f}(X))^2]$$

$$= E_{X,Y,D_n} \left[\left[(Y - E_{D_n}[\hat{f}(x)]) + (E_{D_n}[\hat{f}(x)] - \hat{f}(x)) \right]^2 \right]$$

$$= E_{X,Y,D_n} \left[\left[(Y - E_{D_n}[\hat{f}(x)])^2 + 2(Y - E_{D_n}[\hat{f}(x)])(E_{D_n}[\hat{f}(x)] - \hat{f}(x)) + (E_{D_n}[\hat{f}(x)] - \hat{f}(x))^2 \right] \right]$$

$$= E_{X,Y} [(Y - E_{D_n}[\hat{f}(x)])^2] + E_{X,Y,D_n} [(E_{D_n}[\hat{f}(x)] - \hat{f}(x))^2] + 2E_{X,Y} [(Y - E_{D_n}[\hat{f}(x)])(E_{D_n}[\hat{f}(x)] - E_{D_n}[\hat{f}(x)])]$$

0



Bias Variance Trade-off

$y = f(X) + \epsilon$ where $\epsilon \sim N(0, \sigma_\epsilon)$

$\hat{y} = \hat{f}(X)$: our model

$$Err(\hat{f}) = \text{Bias}^2 + \text{Variance} + \text{Irreducible Error}$$

$$= E_{X,Y} \left[(y - E_{D_n}[\hat{f}(x)])^2 \right] + E_{X,Y,D_n} \left[(E_{D_n}[\hat{f}(x)] - \hat{f}(x))^2 \right]$$

$$= E_{X,Y} \left[(f(X) + \epsilon - E_{D_n}[\hat{f}(x)])^2 \right] + E_{X,Y,D_n} \left[(E_{D_n}[\hat{f}(x)] - \hat{f}(x))^2 \right]$$

$$= E_{X,Y} \left[(f(X) - E_{D_n}[\hat{f}(x)])^2 + 2\epsilon(f(X) - E_{D_n}[\hat{f}(x)]) + \epsilon^2 \right] + E_{X,Y,D_n} \left[(E_{D_n}[\hat{f}(x)] - \hat{f}(x))^2 \right]$$

$$= E_{X,Y} \left[(f(X) - E_{D_n}[\hat{f}(x)])^2 \right] + 2E_{X,Y} \left[\epsilon(f(X) - E_{D_n}[\hat{f}(x)]) \right] + E_{X,Y}[\epsilon^2] + E_{X,Y,D_n} \left[(E_{D_n}[\hat{f}(x)] - \hat{f}(x))^2 \right]$$

0

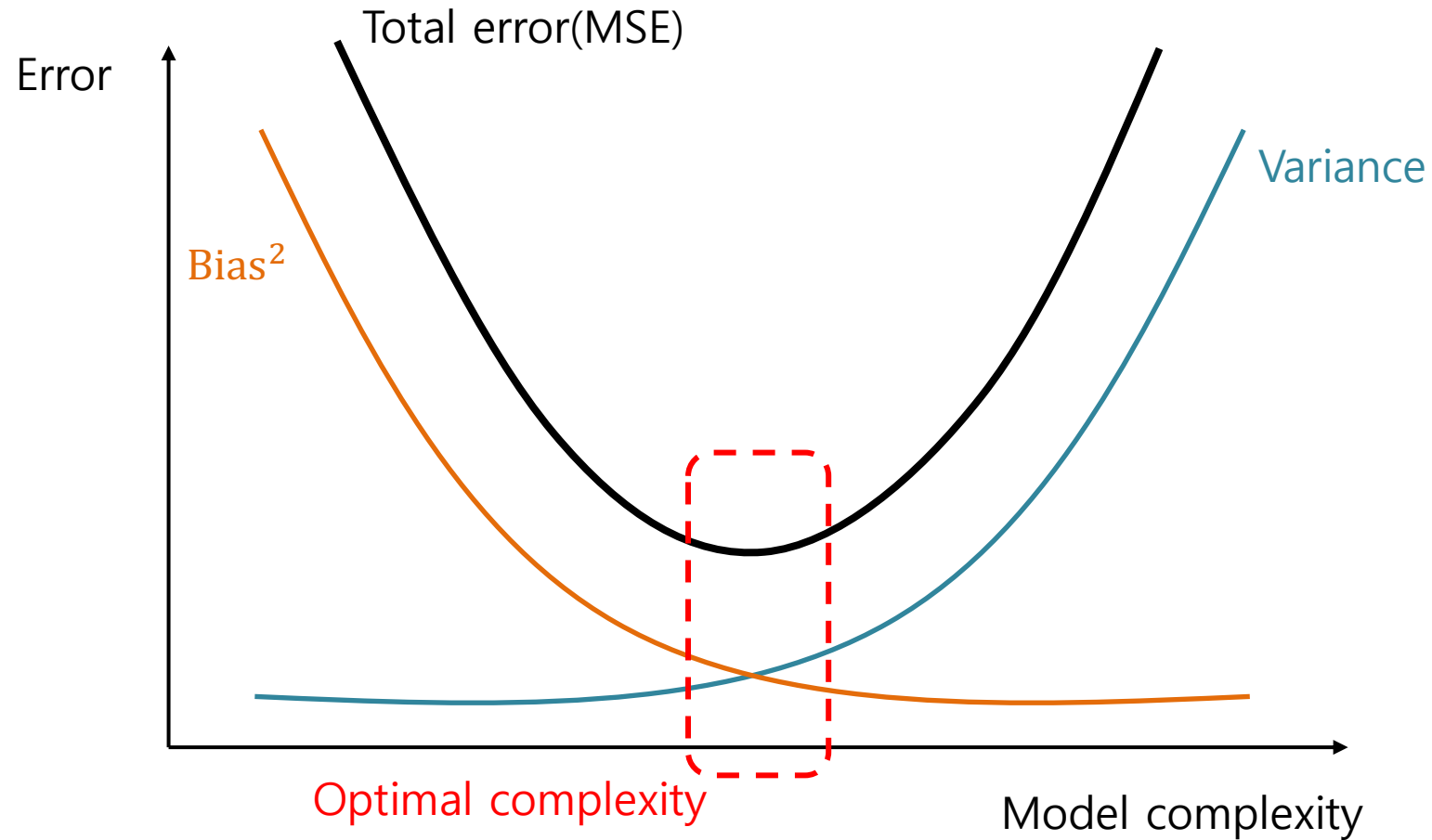
$$= E_{X,Y} \left[(f(X) - E_{D_n}[\hat{f}(x)])^2 \right] + E_{X,Y,D_n} \left[(E_{D_n}[\hat{f}(x)] - \hat{f}(x))^2 \right] + E_{X,Y}[\epsilon^2]$$

Bias²

Variance

Irreducible Error

Bias Variance Trade-off



Generalization

과대적합을 감소하기 위해서 모델을 규제(regularization, penalties)

- 선형회귀 : 모델의 가중치를 제한
 - Ridge
 - Lasso
 - Elastic net
- 다항회귀 : 다항식의 차수를 감소

Ridge Regression

$$\min(\sum_i \{y_i - (\theta_0 + \theta_1 x_i)\}^2 + \frac{\alpha}{2} \sum_{j=1} \theta_j^2)$$

α : 모델의 규제 정도를 조절하는 파라미터

- α 가 아주 크면 모든 θ 는 0에 가까워지므로 수평선(horizontal line)에 가까워짐
 - θ_i 가 완전히 제거되지는 않음 (0이 되지는 않음)
- α 가 0이면 선형회귀식

$\sum_{j=1} \theta_j^2 = (|\mathbf{w}|_2)^2$: 벡터의 l_2 norm

- \mathbf{w} 에는 θ_0 는 포함되지 않음

규제 모델 중 기본적으로 사용되는 모델

Lasso Regression

$$\min(\sum_i \{y_i - (\theta_0 + \theta_1 x_i)\}^2 + \alpha \sum_{j=1} |\theta_j|)$$

Lasso는 덜 중요하다고 판단되는 특성의 가중치를 완전히 제거

- $\theta_i = 0$ if θ_i is not significant.
- 자동으로 특성 선택을 하고 희소 모델(sparse model)을 생성

$\theta_i = 0$ 일 때는 미분 불가능

- Subgradient vector를 사용
 - 미분이 불가능한 지점 근방 gradient들의 중간값

실제로 사용되는 feature가 전체 feature에서 얼마 되지 않을 때 사용

Elastic Net

$$\min(\sum_i \{y_i - (\theta_0 + \theta_1 x_i)\}^2 + \gamma\alpha \sum_{j=1} |\theta_j| + (1 - \gamma)\frac{\alpha}{2} \sum_{j=1} \theta_j^2)$$

Lasso와 Ridge의 결합모델

- $\gamma = 0$: Ridge
- $\gamma = 1$: Lasso

Training set 수(n)보다 feature의 수(m)가 많거나, feature 몇 개가 강하게 연관되어 있을 때 사용

- $n < m$ 이면 Lasso는 최대 n 개의 특성을 선택
- 일부 feature 사이의 연관도가 높으면 이 중 feature 하나를 random하게 선택