Foundation of Machine Learning 2주차

정재헌, 우지수 / 2023.01.20

Computational Data Science LAB

CONTENTS

01 | Gradient Descent

What is Gradient Descent ?

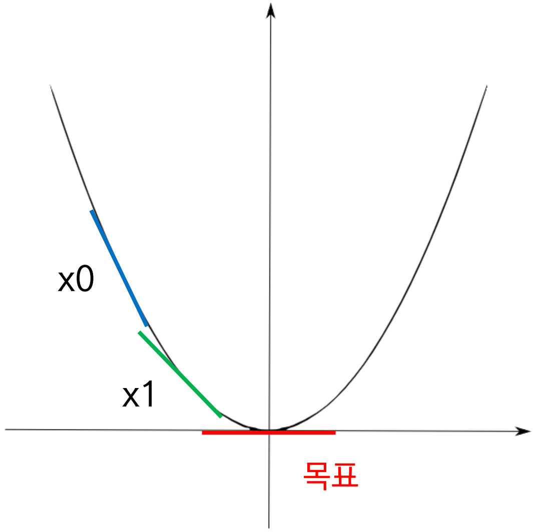
1. 함수의 최솟값을 찾는 최적화 (Optimization) 방법 중 하나

$$x^* = \arg\min_{x \in \mathbf{R}^d} f(x)$$

* 목적함수 f가 미분 가능할 때

2. 방향 도함수 $\nabla f(x)$ 를 구하고, 기울기가 낮은 쪽으로 η(파라미터) 값을 변형시켜가며 최소 기울기에 이를 때까지 반복시키는 것

$$x_1 \leftarrow x_0 - \underbrace{\eta}_{\text{step size}} \nabla f(x_0)$$



ex) 앞이 보이지 않는 안개가 낀 산을 내려올 때는 모든 방향으로 산을 더듬어가며 산의 높이가 가장 낮아지는 방향으로 한 발 씩 내딛음.

01 | Gradient Descent

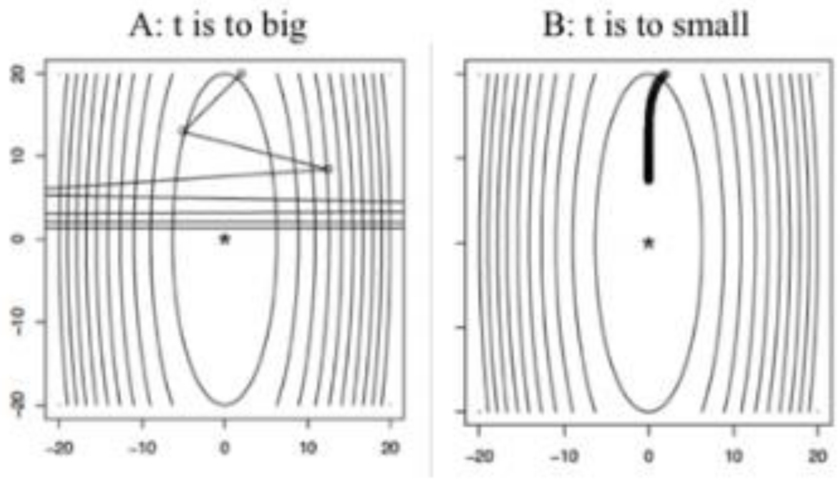
Why using Gradient Descent ?

✓ 미분계수가 0인 지점을 찾지 않고 굳이 Gradient Descent를 사용하는 이유

1. 함수가 너무 복잡해 미분계수를 구하기 어려운 경우
2. 데이터의 양이 너무 많아 효율적으로 계산해야 하는 경우

How to choose Step Size ?

$$x \leftarrow x - \underbrace{\boxed{\eta}}_{\text{step size}} \nabla f(x)$$



1. Fixed Step Size

✓ 모든 반복에서 step size를 고정하는 방법

✓ 하지만 step size에 따라 수렴할 수도 있고 발산할 수도 있음.

A : step size가 매우 큰 경우로, 8 step 이후 발산하여 최솟값에 도달 불가능함. (10, 100, 1000, ⋯)

B : step size가 매우 작은 경우로, 수렴의 속도가 매우 느림. $(10,\ 1,\ \frac{1}{10}, \frac{1}{100},\ \cdots)$

01 Gradient Descent

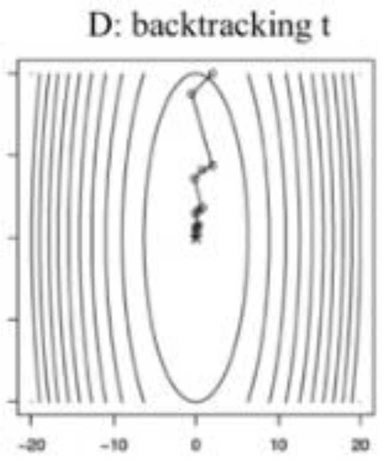
How to choose Step Size ?

$$x \leftarrow x - \underbrace{\eta}_{\text{step size}} \nabla f(x)$$

2. Backtracking Line Search

✓ 곡면의 특성에 맞춰 step size를 선택하는 방법

D : 현재 위치에서 한 step을 가보고 너무 많이 갔다고 판단되면 다시 되돌아와서 다음 step을 결정함.



Convergence Theorem for Fixed Step Size

Convergence Theorem (수렴 분석)

✓ 적절한 step size를 구하기 위함

✓ f 는 볼록하며 미분 가능할 때, 다음 식을 만족 $(L > 0)$

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|$$

Lipschitz Continuous

✓ 점 사이의 거리를 일정 수준 이상 증가시키지 않게 하는 함수

Fixed step size의 수렴 분석

✓ Fixed step size인 t가 $t \leq \frac{1}{L}$ 에 수렴하며 다음 식을 만족

$$f(x^k) - f(x^*) \leq \frac{\|x^0 - x^*\|^2}{2tk}$$

02 | Gradient Descent for Empirical Risk

✓ Empirical risk를 최소화 시켜야 함.

$$\hat{R}_n(w) = \frac{1}{n}\sum_{i=1}^{n}\left(w^T x_i - y_i\right)^2$$

✓ Empirical risk minimization은 loss를 최소화하는 w를 찾는 것이 목적 → 최적화

$$\hat{R}_n(w) = \frac{1}{n}\sum_{i=1}^{n}\ell(f_w(x_i), y_i)$$

✓ 위 식이 미분 가능한 경우, Gradient Descent를 활용해 최적화 가능

$$\nabla \hat{R}_n(w) = \frac{1}{n}\sum_{i=1}^{n}\nabla_w \ell(f_w(x_i), y_i)$$

Gradient Descent for Empirical Risk

$$\nabla \hat{R}_n(w) = \frac{1}{n}\sum_{i=1}^{n} \nabla_w \ell(f_w(x_i), y_i)$$

- n개의 학습데이터를 가지고 Gradient Descent를 한다는 것은 n개의 w를 한 번에 평균한다는 의미

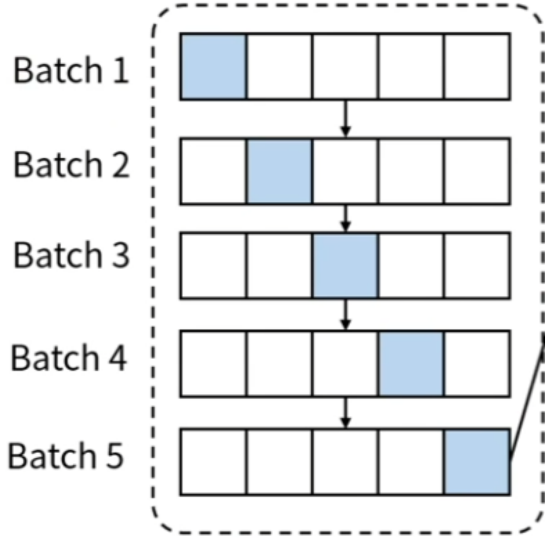
  ex) 학습데이터가 1000개이면 1000개에 대한 손실함수를 구하고, 1000개의 Gradient Descent를 구한 후 평균값 계산

- 즉, w를 한 번 업데이트 하기 위해 모든 학습 데이터를 사용
- 굉장히 비효율적인 방법

Mini-batch Gradient

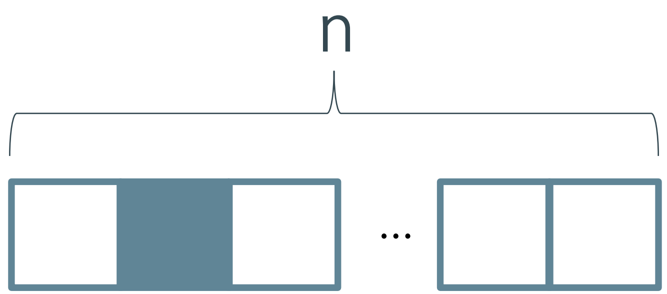
- 학습데이터 전부를 사용하는 것이 아닌 작은 batch 크기로 나누어서 batch 하나하나마다 계산 진행

w를 한 번 계산하는 단위



$$\begin{aligned}\mathbb{E}\left[\nabla \hat{R}_N(w)\right] &= \frac{1}{N}\sum_{i=1}^{N}\mathbb{E}[\nabla_w \ell(f_w(x_{m_i}), y_{m_i})] \\ &= \mathbb{E}[\nabla_w \ell(f_w(x_{m_1}), y_{m_1})] \\ &= \sum_{i=1}^{n}\mathbb{P}(m_1 = i)\nabla_w \ell(f_w(x_i), y_i) \\ &= \frac{1}{n}\sum_{i=1}^{n}\nabla_w \ell(f_w(x_i), y_i) \\ &= \nabla \hat{R}_n(w)\end{aligned}$$

Stochastic Gradient Descent (SGD)



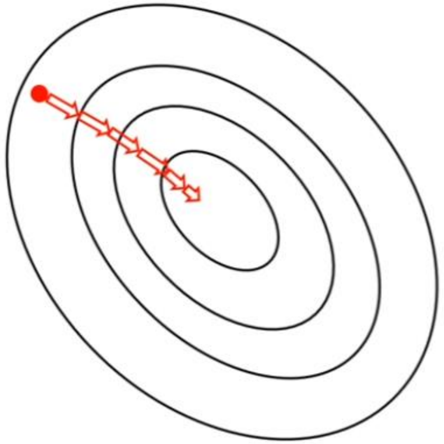
- 학습데이터 n개를 모두 사용하는 것이 아니라 확률적으로 선택된 일부의 데이터만 사용 = 샘플링
- w를 더 빨리 찾을 수 있으며 모델을 자주 업데이트 할 수 있는 장점이 존재

02 | Gradient Descent for Empirical Risk

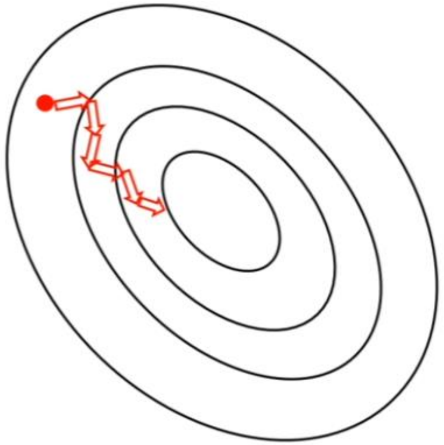
Trade – off 관계 존재

✓ n이 작은 경우 : 속도는 빠르지만 비교적 부정확한 추정

✓ n이 큰 경우 : 속도는 느리지만 비교적 정확한 추정

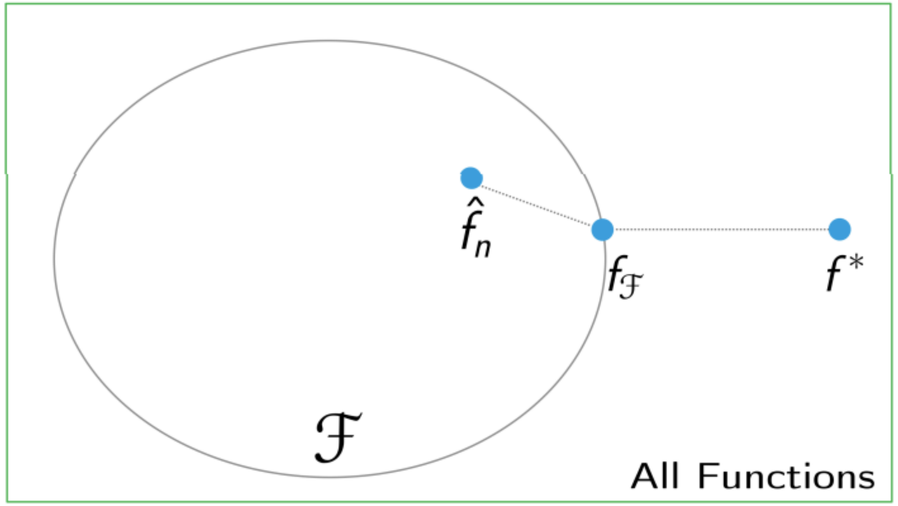


Gradient Descent



Stochastic Gradient Descent

Error Decomposition



- f^* = 모든 함수를 고려 (Bayes Decision Function) → 과적합

$$\underset{f}{\operatorname{argmin}}\, \mathbb{E}\ell(f(X), Y)$$

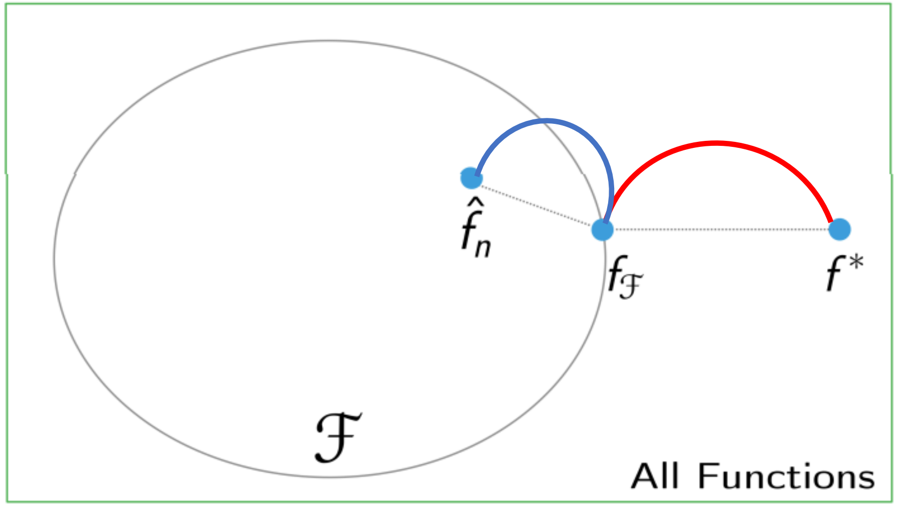
- $f_{\mathcal{F}}$ = 가설 공간 안에서 제한된 함수들만 고려 → 과적합 방지

$$\underset{f \in \mathcal{F}}{\operatorname{argmin}}\, \mathbb{E}\ell(f(X), Y)$$

- $\hat{f}_n$= 학습 데이터 고려 = loss의 기댓값 = Empirical Risk Minimizer

$$\underset{f \in \mathcal{F}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \ell(f(X_i), y_i)$$

Approximation Error & Estimation Error



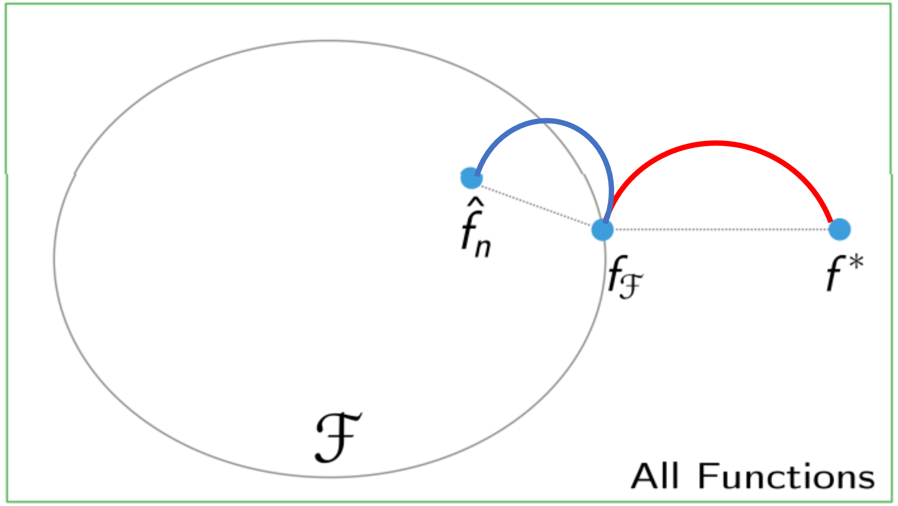
Approximation Error (근사 오차)

- ✓ $R(f_{\mathcal{F}}) - R(f^*)$

Estimation Error (추정 오차)

- ✓ $R(\hat{f}_n) - R(f_{\mathcal{F}})$

Excess Risk



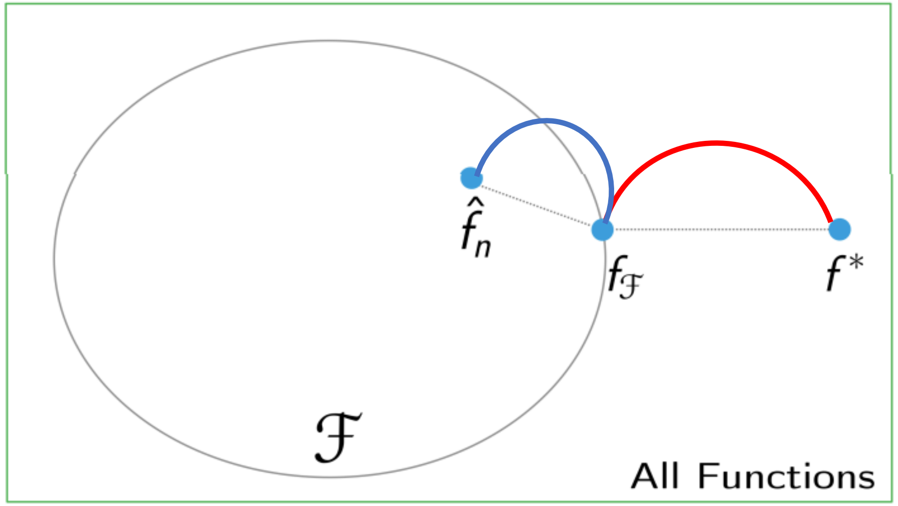
$$
\begin{aligned}
\text{Excess Risk}(\hat{f}_n) &= R(\hat{f}_n) - R(f^*) \\
&= \underbrace{R(\hat{f}_n) - R(f_{\mathcal{F}})}_{\text{estimation error}} + \underbrace{R(f_{\mathcal{F}}) - R(f^*)}_{\text{approximation error}} .
\end{aligned}
$$

Excess Risk

- ✓ Bayes Decision Function의 최적 값인 f^* 와 ERM의 f를 비교
- ✓ Estimation Error와 Approximation Error를 활용하여 Decomposition Excess Risk를 구할 수 있음.
- ✓ 어떤 $\mathcal{F}$를 사용하는가에 따라 Trade – off 관계를 가짐.

Excess Risk Decomposition

Excess Risk



$$\begin{aligned}\text{Excess Risk}(\hat{f}_n) &= R(\hat{f}_n) - R(f^*) \\ &= \underbrace{R(\hat{f}_n) - R(f_{\mathcal{F}})}_{\text{estimation error}} + \underbrace{R(f_{\mathcal{F}}) - R(f^*)}_{\text{approximation error}} .\end{aligned}$$

Function Space가 작으면 ?

- ✓ Approximation Error는 늘어남.
- ✓ Estimation Error는 줄어듦.

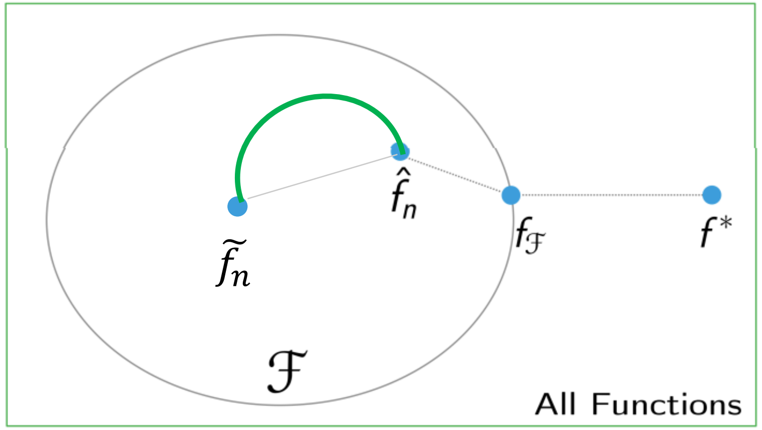
Function Space가 크면 ?

- ✓ Approximation Error는 줄어듦.
- ✓ Estimation Error는 늘어남.

ERM Overview

- ✓ $\widehat{f}_n \in \mathcal{F},\ \underset{f \in \mathcal{F}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \ell(f(X_i), y_i)$
- ✓ 위 식은 실제로 (ex. Neural Net) 활용할 수 없음.

Optimization Error



$$\hat{R}(\tilde{f}_n) - \hat{R}(\underline{\hat{f}(n)}) \geqslant 0$$

Training Data에서 가장 좋은 값

Optimization Error

- ✓ $\tilde{f}_n \in \mathcal{F}$ 을 찾고자 함.
- ✓ $\tilde{f}_n$ = 최적화 방법으로 반환되는 함수
- ✓ $\tilde{f}$ 가 얼마나 $\hat{f}$ 에 가까운지를 나타내는 식

Error Decomposition in Practice

$$\textbf{Excess Risk}(\tilde{f}_n) = R(\tilde{f}_n) - R(f^*)$$
$$= \underbrace{R(\tilde{f}_n) - R(\hat{f}_n)}_{\text{optimization error}} + \underbrace{R(\hat{f}_n) - R(f_{\mathcal{F}})}_{\text{estimation error}} + \underbrace{R(f_{\mathcal{F}}) - R(f^*)}_{\text{approximation error}}$$

Q&A

감사합니다.