4. Model Selection & Validation

모델은 단순히 training dataset에 잘맞는 것으로 부족, 새로운 데이터에 generalization이 잘되는 모델을 선택해야 한다 → Model을 선택하는법, 검증(Validation)하는 법.

4.1 The statistical learning setup

Problem Def)

- 우리는 어떤 입력 $x \in \mathcal{X}$ 와 출력 $y \in \mathcal{Y}$ 의 관계를 모델링해보고 싶어
- Data는 확률 분포 D에 따라 생성된 $(X,Y) \sim D$ 샘플들.
- But 실제로 우리는 D를 모르고, Sample dataset $S = \{(x_i, y_i)\}_{i=1}^{n}$ 만 가지고 있다.

- **Loss function:** 예측이 얼마나 틀렸는지 확인하기 위해 $loss : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$ 사용. ex) $loss(z,y) = (z-y)^2$ ...
- Risk

1) **Population Risk** $R(h) = E[loss(h(x), y)]$ ← 우리가 찾자 줄이고 싶은것. But D를 모르기때문에 계산 X

$h^* = \text{argmin } R(h)$

2) **Empirical Risk** $\hat{R}(h) = \frac{1}{n}\sum_{i=1}^{n} loss(h(x), y)$ ← 우리가 실제로 계산할 수 있는 것. D 기준의 loss avg.(손실 평균)

기본 Idea: 모델 학습할때 $\hat{R}(h)$를 최소화하는 방향으로 학습하되, $R(h)$가 작아야함 ($\hat{R}(h)$으로 overfitting 하면 안좋음)

4.2 Training, Validation, Test errors

ML 에서 데이터를 3부분으로 나눠서 모델 성능 평가

1) Training Set - 모델학습
2) Validation Set - 모델 선택 / Hyperparameter tuning 사용
3) Test Set - 최종 성능 평가 (unseen data)

- Risk 비교관점에서 본 Errors

(Model h에 대해)

1) **Training Error** $\hat{R}(h, D_{train}) = \frac{1}{n_{train}} \cdot \sum_{i=1}^{n_{train}} loss(h(x_i), y_i)$ — Empirical Risk on training set. 모델이 training data에 얼마나 잘 맞는지 보여줌.

2) **Validation Error** $\hat{R}(h, D_{val}) = \frac{1}{n_{val}} \cdot \sum_{i=1}^{n_{val}} loss(h(x_i), y_i)$ — 여러 모델 中 어떤 모델이 generation이 더 좋은지 비교할때 사용

3) **Test Error** $\hat{R}(h, D_{test}) = \frac{1}{n_{test}} \cdot \sum_{i=1}^{n_{test}} loss(h(x_i), y_i)$ — 학습이 끝난후 모델의 일반화 성능을 최종 평가하는 용도.

★ 왜 이렇게 나눌까?

- 모델이 training set 에만 너무 잘 맞으면(overfitting) test set 성능 떨어질 수 있음.
- Validation set이 이를 방지하고, 모델 선택 기준 제공
- test set은 한번도 보지않은 data set, 즉 실제 상황처럼 test.

- Steps 1) 여러 모델($h_1, h_2, \dots$)을 traing set 에 학습
  2) Validation error $\hat{R}_{val}(h)$ 가 가장 낮은 모델 h^* 선택
  3) 선택한 모델 h^* 에 대해 test error $\hat{R}_{test}(h)$ 평가.

True Risk $R(h) = E[loss(h(x), y)]$

Empirical Risk $\hat{R}(h) = \frac{1}{n}\sum loss(h(x), y)$

우리는 최종적으로 $R(h)$를 원하지만 바로 계산 불가. → test set 위에서 $\hat{R}(h)$를 사용.

→ 그럼 질문: Empirical Risk $\hat{R}(h)$가 True Risk $R(h)$를 얼마나 잘 근사할까?

Hoeffding's Inequality

for sample (x_i, y_i):

$$P\left[|\hat{R}(h,D) - R(h)| \le O\left(\sqrt{\frac{\log(1/\delta)}{n}}\right)\right] \ge 1-\delta$$

- $O\left(\sqrt{\frac{\log(1/\delta)}{n}}\right)$: 근사오차 (test set 크기 n)
- $1-\delta$: 허용가능한 실패확률 (e.g. $\delta = 0.05 \simeq 95\%$ 신뢰도)

해석) 좌항: test error, true risk 차이

우항: 그 차이가 $O\left(\sqrt{\frac{\log(1/\delta)}{n}}\right)$ 일 확률.

의미: test set이 충분히 크면 test error는 true risk에 가까워진다.

결론: Hoeffding's Inequality는 test error $\hat{R}(h)$가 true risk $R(h)$를 얼마나 잘 근사하는지 확률적으로 보장.

4.3 K-fold Cross Validation

- Goal: 여러 모델 중 generation 성능이 제일 좋은 모델 h 선택하자 (true risk $R(h)$가 가장 작은 모델 선택)
  - → But, $R(h)$ 직접 계산 불가 → validation set에서 측정한 $\hat{R}(h)$로 평가해보자.

- K-fold-Cross-Validation

1) Dataset D를 크기가 같은 K개의 부분집합으로 나눈다. $D_1, D_2, \cdots, D_K$

2) 각 $i \in \{1, \cdots, K\}$에 대해
   - Train $D_{train} = D \setminus D_i$
   - Validation $D_{val} = D_i$

3) 이때 모델 $h \in H$을 train set으로 학습하고, validation set으로 측정 $\hat{R}_i(h) = \frac{1}{n_{val}} \cdot \sum_{i=1}^{n_{val}} loss(h(x_i), y_i)$

4) 모든 fold에 대해 평균 validation error 계산. $\hat{R}_{cv}(h) = \frac{1}{K}\sum_{i=1}^{K} \hat{R}_i(h)$

→ 적용 목적. (언제 사용?)

모델 선택 시 단일 validation set가 아닌 여러 split을 사용 → 보다 안정적인 선택 가능. 특히 Dataset 작을때.

validation error를 줄여줌. ※) test set은 모델 성능 평가에만 사용. 모델선택 (Model Selection)에서는 사용 X

5. Convex optimization → Basic Intro. • Convexity → Aspect) Optimization algorithm.

5.1 Optimization Problems

- Basic Structure

$\min_{x \in \mathbb{R}^d} f(x)$

- $f : \mathbb{R}^d \to \mathbb{R}$ Objective function
- 목적: $f(x)$를 최소화하는 x를 찾자.

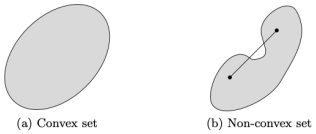
- 최적해 x^*는 $f(x^*) \le f(x)$ $\forall x \in \mathbb{R}^d$ 를 만족해야 한다
  (= global minimum)

5.2 Convexity

- Definition 1: Convex Set

A convex set $\mathcal{C}$ is any set s.t. for $x, y \in \mathcal{C}$ and $\forall \theta \in (0,1)$:

$\theta x + (1-\theta) y \in \mathcal{C}$


(a) Convex set (b) Non-convex set

≒ 두 점 사이를 잇는 선분이 모두 집합 안에 있다면 convex set.

- Definition 2: Convex Function (Strictly convex)

A function $f : \mathbb{R}^d \to \mathbb{R}$ is convex

$\Leftrightarrow$ for all $x, y \in \mathbb{R}^d$, $\theta \in (0,1)$ 에 대해

$\theta f(x) + (1-\theta) f(y) \ge f(\theta x + (1-\theta) y)$

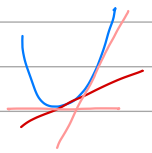
$\theta f(x) + (1-\theta) f(y) > f(\theta x + (1-\theta) y)$

그래프가 직선 연결보다 아래 있으면 convex

- Linear function $f(x) = \langle a, x \rangle + b$
- Quadratic $f(x) = \frac{1}{2} x^T Q x + b^T x + c$
- Any Norm $f(x) = \|x\|$

- Prop 1 : 함수 f가 convex 하고, diff-able 하면 다음 성립 → $f(y) \ge f(x) + \langle \nabla f(x), y - x \rangle$

→ 접선이 항상 그래프보다 아래에 있다.



- Prob 2 : convex function f에 대해, 어떤 점 $x^* \in \mathbb{R}^d$가 $\nabla f(x^*) = 0$ 이면 → x^* = global minimum 이다.

convex function의 stationary point는 global minimum이다.

→ 기울기가 0 인 지점이 global 최소값 (≈ 최적해를 가지는 x^*)

- 만약 f strictly convex 일 경우 x^*는 유일한! global minimizer

5.3 Computational aspect of optimization algorithm

우리는 $\min f(x)$ 를 풀고싶어. e.g) $f(x) = \|Ax - y\|_2^2$. 여기서 $A \in \mathbb{R}^{m \times n}$은 full column rank 행렬.

만약 A가 full rank라면 (해를 구할 수 있다면) 최적해 ⇒ $x^* = (A^T A)^{-1} A^T y$. 그런데 현실에선? 1) f가 closed form 계산이 불가한 경우에 어떡하지? 2) 역행렬 구하기 어려움 X

→ 반복해서 최대한 비슷하게 만들어보자 (근사해 Approximate Solution 구해보자)

- Oracle Access - 실제 컴퓨터가 호출 가능한 것 (할 수 있는것) : 알고리즘이 접근가능한 정보.

1) 0^{th}-order oracle : 주어진 x에 대해 $f(x)$ 만 반환
2) 1^{st}-order oracle : $f(x)$와 $\nabla f(x)$ (Gradient)를 반환
3) 2^{nd}-order oracle : $f(x)$, $\nabla f(x)$, $\nabla^2 f(x)$ (Hessian) 까지 반환

+ 몇번을 호출해야 (반복해야) 할까? = oracle complexity

호출

ε-accurate solution 을 얻기위한 횟수

$\|x^k - x^*\| \le \varepsilon$ or $f(x^k) - f(x^*) \le \varepsilon$

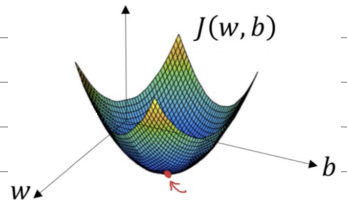
이 중 하나를 만족할때의 oracle 호출 횟수 = 복잡도.

- 우리가 어떤 정보를 가지고 최적화를 진행하나?
- 이 정보를 몇번 요청해야 충분한 정확한 해를 얻을 수 있을까?

다음 chapter 부터는 algorithm을 볼것인데 이 이전에 우리는 $f(x)$, $\nabla f(x)$ 같은 Oracle Access 정보를 사용한다 가정!

6. Gradient Descent (알고리즘 종류.)

$x^{k+1} = x^k - \alpha_k \nabla f(x^k)$ Gradient (미분)를 K번 반복하면서 α_k 만큼씩 반복해서 경사를 내려갈꺼야



- Goal: Gradient Descent 이 Quadratic function (2차 함수)에 대해 얼마나 빨리 최솟값에 수렴하는지 수학적으로 분석해보자.

6.1 Convergence for quadratic function

- Quad. function:

$f(x) = \frac{1}{2} x^T Q x - b^T x + c \quad \rightarrow \nabla f(x) = Qx - b$

- Q is symmetric, strictly positive definite.
- 이 함수 f는 convex

Gradient Descent : $x^{k+1} = x^k - \alpha \nabla f(x^k)$

- $\nabla f(x) = Qx - b$. → 최적해 식: $Qx^* = b$.

식 정리: $x^{k+1} - x^* = x^k - \alpha \nabla f(x^k) - x^*$

$= x^k - \alpha(Qx^k - b) - x^* = x^k - \alpha(Qx^k - Qx^*) - x^*$

$= x^k - \alpha Q(x^k - x^*) - x^* = (I - \alpha Q)(x^k - x^*)$

It follows: $\|x^{k+1} - x^*\|_2 \le \|I - \alpha Q\| \|x^{k+1} - x^*\|_2$

Eigen value: $\lambda_i = 1 - \alpha\lambda_i$ Symmetric (≈ invertible, singular value 有, Sym 행렬의 norm = 최대 Singular Value.)

✓ $\|I - \alpha Q\| = \max(\lambda_{max}(I - \alpha Q), \lambda_{min}(I - \alpha Q)) = \max(\alpha M - 1, 1 - \alpha m)$ M: 최대SV, m: 최소 SV

이 값을 최소화 하려면, $|\alpha M - 1| = |1 - \alpha m| \Rightarrow$ Stepsize $\alpha = \frac{2}{M+m}$

값을 $\alpha = \frac{2}{M+m}$, $\kappa = \frac{M}{m}$ 로 설정하면,

- $\lambda_{min} = m$: $1 - \frac{2}{M+m} \cdot m = \frac{M-m}{M+m}$
- $\lambda_{max} = M$: $\frac{2}{M+m} \cdot M - 1 = \frac{-(M-m)}{M+m}$

$\rightarrow \|I - \alpha Q\| = \max|1 - \alpha\lambda_i| = \frac{M-m}{M+m} = \frac{\kappa - 1}{\kappa + 1} = \frac{1 - \frac{1}{\kappa}}{1 + \frac{1}{\kappa}}$

- Prop3: GD with $\alpha = \frac{2}{M+m}$ applied to a quadratic function obeys $\kappa = \frac{M}{m}$

$\|x^k - x^*\|_2 \le \left(\frac{\kappa - 1}{\kappa + 1}\right)^k \|x^0 - x^*\|_2 \le \varepsilon$

우리가 구하는것: ε-accurate solution $\Leftrightarrow \|x^k - x^*\| \le \varepsilon$, $\rho = \frac{\kappa - 1}{\kappa + 1} \Rightarrow \log \rho < 0$.

$\rightarrow \rho^k \|x^0 - x^*\|_2 \le \varepsilon \Rightarrow \rho^k \le \frac{\varepsilon}{\|x^0 - x^*\|_2}$

$\xrightarrow{\log} k \log \rho \le \log\left(\frac{\varepsilon}{\|x^0 - x^*\|_2}\right) \Rightarrow k \ge \frac{\log(\|x^0 - x^*\|_2 / \varepsilon)}{\log(1/\rho)}$

여기서 최적 k를 N이라 해봤자.

$N = \left(\log \frac{1}{\rho}\right)^{-1} \log\left(\frac{\|x^0 - x^*\|_2}{\varepsilon}\right)$

의미)

1) 어떤 행렬 Q의 condition number (κ)는 고윳값의 비율.

* Q is well-conditioned ($\kappa \approx 1$) : 고윳값이 서로 비슷 ⇒ 모든 방향에서 함수의 곡률이 비슷 ⇒ 수렴 fast, stable
* Q is ill-conditioned ($\kappa \gg 1$) : 고윳값 차이 大 ⇒ 어떤 방향은 급경사, 평평 ⇒ 수렴 slow, zigzag 발생

2) rate of convergence $= \left(1 - \frac{1}{\kappa}\right)^k$

$\kappa \uparrow$ = 수렴率 ⇓, $\kappa \to \infty \Rightarrow$ 수렴率 매우 느림.

3) Geometrical ...

Quadratic function (2차 함수)의 등고선이 Q에 따라 타원형이 됨. Q의 condition number가 클수록 길쭉한 타원

GD는 등고선을 따라 수직하강 방식으로 움직임. 곡률 차가 크면 zigzag

$ax^2 + by^2 = 1$.