

# 기계학습이론 기말대비 1

Loss ~ Regularization

## Hypothesis space

요소	Hypothesis space의 크기	설명
Parameter 개수	비례	
Inductive bias	반비례	
Model architecture	구조가 제한될수록 작아짐	
Prior	반비례	
Regularization	제한	$ H $ 를 줄인다기 보다 Training을 통해 좋은 solution을 찾도록 일부 영역을 제한하는 느낌
Optimization method	제한	Method마다 Implicit regularization이 달라서 Space가 일부 제한됨
Initialization	제한	탐색 가능한 영역이 달라질 수 있음

## LOSS

가장 먼저, **Linear regression**은 **Least square(MSE)**을 Loss로 사용한다.

- 이 경우 Loss가 **ERM**을 **최소화**하는 것과 동일하다.
- 이때, Loss는 **Convex**이며, **Quadratic**하다.
- Loss가 Convex & Quadratic하다는 것은 Loss=0인 지점을 찾기 쉽고, Optimization 또한 Loss=0인 지점을 찾으면 된다는 것이다.

- $W$ 에 대해 Quadratic하다.

하지만, Linear regression은 **Linearly separable**하지 않다면 **Classification** 할 수 없는 문제점이 존재한다.

- Linear regression은 그 자체로 Inductive bias가 너무 강하다.

위 문제를 해결하기 위해, 가장 먼저 사용하는 방법은 사람이 지정한 **Non-linear basis function**을 사용하는 것이다.

- Linear regression에 비해 Hypothesis space가 넓어진다.
- **Non-linear basis function**을 사용할 때의 **Loss** 역시 **Convex & Quadratic**하기 때문에 쉽게 Optimization이 가능하다.

하지만, **Non-linear basis function**을 사용하는 것 역시 Hypothesis space가 비교적 작고, 사람이 하나씩 정하는 것에 어려움이 있다.

이를 해결하기 위해 **Parameterized feature extractor**를 사용한다.

- 2NN이 기본이다.
- **Universal Approximation Theorem**에 의해 실세계의 모든 함수를 표현할 수 있다. 그러나 해당 함수를 정확히 찾을 수 있을지는 미지수이다.
- 이때부터 **Loss**가  $W$ 에 대해 **Quadratic**하지 않고, **Convex**함이 보장되지 않는다.
  - Optimization이 어려워진다.

2-Layer보다 더 깊게 쌓을수록 Feature를 계층적으로 학습하여 더 좋은 성능을 보이기에 Layer를 점점 더 깊게 쌓기 시작했다.

---

## Optimization

먼저 두 가지 용어를 정의해야 한다.

1. **Emprical risk** ( $L_S(h) = \mathbb{E}_{(x_i, y_i) \in S} [l(y_i, h(x_i))]$ ): Training dataset에 대해 계산한 loss
2. **Population risk** ( $L_P(h) = \mathbb{E}_{(x, y) \sim P} [l(y, h(x))]$ ): 실세계에 대한 Loss (일반화된)

원래의 우리의 목적은 **MAP**이다. MLE는 Emprical risk를 최소화하기 위해서 Training data만 보지만 실제 목표는 앞으로의 것, 보지 못한 것을 예측하는 것이 목표이다.

- Optimization의 최종 목표는 모르는 분포 Population에서의 risk를 낮추는 것이다.

그러나 실세계의 분포, Population은 모르는 것이 일반적이다. 따라서 우리는 간접적으로 Population에서 샘플링한 Training data를 가지고 ERM을 계산하여 간접적으로 Population risk를 최소화하고자 한다.

- ERM의 수식은 다음과 같다.

$$h_s \in \operatorname{argmin}_{h \in H} L_s(h)$$

- 여러 가능한 Hypothesis 중에 Loss를 최소화하는 Hypothesis를 찾는 것이다.
- $L_s(h)$ 가 Emprical risk이다.

그러나, Population에서 sampling한 데이터에 대한 ERM을 하는 것이 Population risk를 줄이도록 유도하는 것은 아니다.

따라서 우리는 Prior를 통해 Population에 대한 정보를 모델에게 주어 Population risk를 줄이도록 유도할 수 있다.

- 여기서 MAP가 완성된다.

Prior 부분은 잠깐 넘어가고, Optimization을 하는 방법에 대해 살펴보자.

Parameterized model이  $h_\theta$ 라고 하면 Emprical risk는 다음과 같이 작성할 수 있다.

- $L_s(\theta) = L_s(h_\theta)$
- $H = h_\theta : \theta \in \Theta$ , 모든 가능한 Parameterized hypothesis의 집합

**Parameterized model**에서 우리의 목표는  $L_s(\theta)$ 가 최소화되도록 하는  $\theta$ 를 찾는 것이다.

$$\theta_s \in \operatorname{argmin}_{\theta \in \Theta} L_s(\theta)$$

문제는 위  $L_s(\theta)$ 가 **Nonconvex**하기 때문에, **Gradient-based Optimization**을 이용해야 한다는 점이다.

$\theta$ 를 찾도록 하는 여러 **Gradient-based method**가 있다.

### 1. Gradient Flow

- 매우 작은 Step으로 Parameter를 업데이트했던 초기 방법이다.

### 2. Gradient Descent

- Gradient Flow를 근사하고자 했던 것이다.
- Loss가 Linear하지 않아도, 주어진 점에서는 Linear하다고 가정한 후 Gradient를 계산하는 방법이다.

### 3. Stochastic Gradient Descent

- GD를 근사하고자 했던 것이다.
- GD처럼 Dataset 전체를 사용하는 것이 아니라, Dataset의 일부분을 사용하여 Parameter를 업데이트 하는 것이다.
- 계산력과 상관없이 비교적 **Overfitting** 방지와 일반화 측면에서 성능이 좋다.

### 4. Momentum method

- $m \leftarrow \beta m + g(\theta)$ : 먼저 Momentum을 업데이트
- $\theta \leftarrow \theta - \eta m$ : Gradient update
- 점점 예전 Gradient에 가중치를 줄여가면서 이전 방향을 어느 정도 유지하는 효과가 있음
- **Nesterov Accelerated Gradient (NAG)**
  - $m \leftarrow \beta m + g(\theta - \beta m)$
  - 현재 Gradient 계산 시, Momentum만큼 이동한 위치에서 계산해 수렴 속도를 높인다.

## 5. Preconditioned GD/SGD

- 기본적인 Idea는 Newton's Method에서 얻을 수 있다.
  - Netwon's method의 업데이트 식에서  $f(x)$  대신  $L'(x)$ 를 사용하면  $X_{n+1} = X_n - \frac{L'(X_n)}{L''(X_n)}$ 이 된다.
  - 이때, 분모에  $L''(X_n)$  항이 곡률이 작을수록 Gradient가 커지도록 하는 **Precondition 역할**을 한다.
  - Learning rate 대신  $\frac{1}{L''(X_n)}$ 을 사용한다고 생각해도 된다.
- Gradient가 특정 방향이 너무 클 때, 최적의 방향이 아니라 해당 방향으로 쏠리게 되는 문제를 해결한다.
- 공식은 다음과 같다.
  - $M = \text{diag}(\sqrt{s} + \varepsilon), \theta \leftarrow \theta - \eta M^{-1} g(\theta)$
- **AdaGrad**
  - $s \leftarrow s + g^2$
- **RMSProp**
  - $s \leftarrow \beta s + (1 - \beta)g^2$
- **Adam (RMS + momentum)**
  - $m \leftarrow \beta_1 m + (1 - \beta_1)g$
  - $s \leftarrow \beta_2 s + (1 - \beta_2)g^2$
  - $\theta \leftarrow \theta - \eta M^{-1}m$

모든 Gradient based optimization는 공통적으로 Learning rate가 너무 작지도, 너무 크지도 않아야 한다.

- 너무 작으면 수렴 속도가 느려진다.
- 너무 크면 수렴하지 않을 수 있다.
- $\frac{2}{\lambda_{\max}(A)} > LR > 0$  의

## Convergence

Learning rate 이야기를 하면서 **Convergence** 이야기가 나왔다.

**Convergence**를 알기 위해선 **Convex set**, **Convex function**, **Epigraph** 먼저 정의해야 한다.

**Convex set:** 집합 내의 어느 두 점을 잡았을 때, 두 점을 X, Y라고 하자. 두 점을 직선으로 이은 선분 위에 모든 점이 같은 집합 내에 존재하면 Convex set이다.

### Convex function

$$f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y)$$

- X와 Y를 섞은 점의 함수 값이 X에서의 함수값과 Y에서의 함수값을 섞은 값보다 작다.

### Epigraph

$$\text{epi}(f) = \{(x, \beta) : f(x) \leq \beta\}$$

특정 함수 f의 함수값 위 영역을 가지는 집합

위 세 가지 개념으로부터 중요한 정리가 도출된다.

#### | $f$ 가 Convex function $\leftrightarrow$ Epigraph가 Convex set

- 즉, 함수가 Quadratic하면 무조건 Epograph가 Convex set이기 때문에 Convex function이 되는 것이다.
- 이는 Binary classification에서 Zero-one loss를 사용하지 않고 Binary cross entropy loss를 사용하는 이유와 관련이 있다.
  - Zero-one loss가 Convex하지 않기 때문이다.
  - 따라서 Zero-one loss를 근사하는 다른 함수를 사용하는 것이다.

이걸 이용해서 미분 가능한 함수 f에 대해 다음을 유도할 수 있다.

#### | $f$ 가 Convex function $\leftrightarrow f(x) \geq \text{Gradient}$

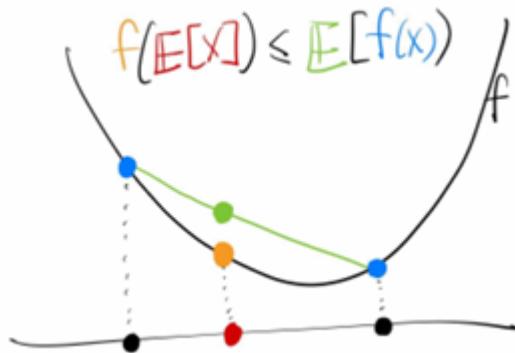
## Jexson's Inequality

$f: X \rightarrow \mathbb{R}$ 인 Convex function,  $x_i \in X, \lambda_i \in [0, 1]$

$$f\left(\sum_i \lambda_i x_i\right) \leq \sum_i \lambda_i f(x_i)$$

여기서  $\lambda_i$ 를 확률로 생각한다면? 기댓값으로 생각할 수 있다.

$$f(\mathbb{E}_X[X]) \leq \mathbb{E}_X[f(X)]$$



이를 통해 KL-divergence가 항상 0보다 크거나 같음을 증명할 수 있다.

## m- Strongly Convex

$$f(x) - (f(y) + \nabla f(y)^\top (x - y)) \geq \frac{m}{2} \|x - y\|^2$$

- $f(x) \geq \text{Linear approx} + \frac{m}{2} \|x - y\|^2$
- $f(x)$ 가 최소한  $\frac{m}{2} \|x - y\|^2$  이상 뛰어있어야 한다.
- 평평하면 안 된다!!

## $\beta$ -smoothness

$$|f(x) - (f(y) + \nabla f(y)^\top (x - y))| \leq \frac{\beta}{2} \|x - y\|^2$$

- $f(x) \leq \text{Linear approx} + \frac{\beta}{2} \|x - y\|^2$
- 너무 휘거나 너무 Sharp하면 안 된다!!

### Polyak–Łojasiewicz(PL)

$$\frac{1}{2} \|\nabla f(x)\|^2 \geq c(f(x) - f(x^*))$$

- Gradient가 너무 작아지면 안된다!!
- $f(x^*)$ 는 최적점이다.
- 최적점에 가까워질수록 Gradient가 너무 작아지는 것을 방지하기 위함이다.

$\beta$ -smoothness와 Polyak–Łojasiewicz(PL)을 이용하여 Linear convergence를 수식화하여 정리하면  $L_t - L^*$ 가 작아지기 위해선 C가 커야하고  $\beta$ 가 작아져야 한다는 결론이 나온다.

- 따라서 Linear convergence (Convex)의 경우, Loss curve가 평평한 것이 좋고, Gradient가 어느 정도 커야한다는 결론이 나온다.

그러나 SGD는 Convex하지 않은 Loss에 대해서도 충분히 좋은 solution을 찾는 경향을 보인다.

- 좋은 solution: Training loss가 작도록 하는 Parameter
- 왜 그런지는 아직도 모른다.

## Backpropagation

### | Optimization을 수행하는 방법

## Computation Graph (DAG)

| Loss와 Parameter를 연결하는 그래프로, Backpropagation에 사용된다.

Forward pass에서 이 그래프를 미리 구축해놓는다.

---

## Training error vs Test error

이제, **Emprical risk**를 줄이는 것이 어떻게 **Population risk**까지 줄일 수 있는지 알아보자.

그에 앞서, **Emprical risk**를 줄이는 것이 **Population risk**를 줄이는 데 효과가 적은 이유를 먼저 알아보자.

### Generalization error of $h_s$

- Training data를 이용한 ERM을 통해 구한  $h_s$ 와 실제 Target과의 차이
- $L_P(h_s)$

우리가 만약, Population을 알고 있다면, **Hypothesis space** 내에서의 최적의 **Hypothesis**는 아래와 같이 나타낼 수 있다.

- $h_H^* \in \operatorname{argmin}_{h \in H} L_P(h)$

그러나, 대부분에 경우에  $L_p(h)$ 를 모르기 때문에  $h_H^*$ 를 구할 수 없는 경우가 대부분이다.

### Approximation error

- $h_H^*$ 와 실제 Target간의 차이
- $L_P(h_H^*)$
- Hypothesis space가 커질수록 실제 Target을 Hypothesis space가 포함할 가능성 이 커지기 때문에  $|H|$ 에 반비례한다.

## Estimate error

- Hypothesis space 내의 최적의 hypothesis와 ERM을 통해 구한 hypothesis 간의 차이
- $h_H^*$  와  $h_s$ 의 차이
- $L_P(h_s) - L_P(h_H^*)$
- 간단하게는 Hypothesis space가 커질수록 커진다고 생각할 수 있지만, 항상 그런 것은 아니다.

Hypothesis space의 크기는 2NN이나 DNN일수록 커진다.

2-layer만 되어도 Universal Approximation Theorem (UAT)에 의해 거의 모든 함수를 표현할 수 있다.

따라서 Approximation error는 0에 가까워진다.

결국 Estimate error := Generalization error가 된다.

- $L_P(h_s) - L_P(h_H^*) = L_P(h_s)$

$L_P(h_s)$ 는 Training data로 찾은  $h_s$ 의 Population에서의 Loss이므로 Test error가 된다.

$L_P(h_s)$ 가 Test error이기 때문에 Training error인  $L_S(h_s)$ 와 비교해보자.

먼저,  $L_s(h_s)$ 를 학습했는데,  $L_p(h_s)$ 라는 본적 없는 데이터 분포에 대한 좋은 성능이 나오는 이유에 대해 알아보자.

- Training data는 Population에서 Sampling한 것이다. Central Limit Theorem에 의해 Training data가 Sampling하는 과정에서 Population을 일부 표현하기 때문이라고 생각할 수 있다.

Population에서 N번 Sampling하여  $S$ 를 얻었다고 할 때, **Train-test error gap**  $|L_s - L_p|$ 는  $\frac{1}{\sqrt{n}}$ 에 비례한다.

- 따라서 **Training Data**가 많아질수록 **Gap**이 작아지는 것이다.
- $|L_s - L_p| \leq \frac{1}{\sqrt{n}}$

그러나, **Hypothesis space**가 커질수록  $L_S(h_s)$ 는 계속 감소하지만,  $L_P(h_s)$ 는 감소하다가 다시 증가한다.

- Hypothesis space가 커진다면 **Training data**를 모두 지나는 함수를 정확히 표현할 수 있게 되어  $L_S(h_s)$ 는 0에 가까워진다.

Test error가 감소하다가 증가하는 이유를 **Bias & Variance** 관점에서 설명해보자.

**bias( $h_s$ )**:  $E_s[h_s] - f$

**variance( $h_s$ )**:  $Var_s[h_s]$

Test error를 다음과 같이 표현할 수 있다.

$$L_p(h_s) = E_{x \sim P}[(h_s(x) - f(x))^2]$$

- $x_i$ 가  $P$ 에 대해 i.i.d로 Sampling될 때,  $S = [x_i]_{i=1}^n$ 이 되어 **Randomness**를 부여하기 때문이다.

$E_{s \sim p^n}[(h_s - f)^2]$  (**Generalization error**)의 평균을 정리하면 **Variance + Bias**의 형태로 나온다.

- Hypothesis space가 커질수록 Variance가 커져 test error가 다시 증가하게 되는 것이다.
- 이를 **Overfitting**이라고 부른다.

하지만, 위는 **Classical ML**의 관점이고 **Modern ML**에서는 **Hypothesis space**가 클수록 Test error가 감소하다가 증가하고 다시 감소하는 형태를 보인다.

- **Double Descent**라고도 부른다.

---

# Regularization

**Classical ML**의 입장에서, Hypothesis space를 키우기만 하는 것은 **Overfitting**을 유발한다.

즉, Variance를 줄이기 위해 Hypothesis space를 줄이기 위한 Prior를 추가하는 것이 Regularization이다.

## Regularized Empirical Risk

$$L_s(\theta; \lambda) := L_s(\theta) + \lambda C(\theta)$$

- $C(\theta)$ : Hypothesis function  $h_\theta$ 의 복잡도를 측정
- MAP와 동일하다.
- MAP - Likelihood + Prior - Regularized ERM

## Regularization 방법

### L1 Regularization

- 1-Norm 사용한다.
- Sparse
- 대부분의  $\theta$ 가 0이 된다.

### L2 Regularization

- 2-Norm 사용
- Dense
- $\theta^2$ 는 큰 Weight에 Penalty

- $\theta$ 가 0으로 가는 경우는 잘 없다.

## Regularization 종류

**Soft prior:** 약한 Prior

- EX)  $P(\theta) \propto \theta(1 - \theta)$

**Hard prior:** 강한 Prior

- EX) Uniform(0.4; 0.6)
- $|H|$ 를 크게 줄인다.

## Implicit Regularization

그러나 **Regularized Empirical Risk**는 **Classical ML**의 관점이고, **Modern ML**에서는 **별도의 Regularization term 없이도 SGD가 충분히 좋은 Solution을 찾는다.**

- 좋은 Solution이란  $L_s, L_p$  모두 작아지는 것을 의미한다.
- 왜 그런진 모른다. 하지만, **SGD가 Generalization 성능이 가장 좋아서 자연 선택과 같** 이 살아남았다고 이해하면 좋다.

SGD가 왜 잘 동작하는지를 확인하자. (SGD를 우선 사용하다가 나중에 찾은 결과이다.)

- **SGD를 통해 구한 Minima가 더 Flat한 Minima이고 Flat minima가 더 좋은 Minima**임을 확인할 수 있었다.
- Sharpness한 부분에서는  $\theta$ 가 작게 변해도 Loss가 크게 변하기 때문이다.

찾은 Minima가 Flat한지 Sharp한지 측정할 수 있는 두 가지 방법이 있다.

1.  $E_\epsilon[L_s(\theta + \epsilon)]$ 을 계산해서 값의 변화가 클수록 Sharp하다.
2.  $\lambda_{max}(\nabla^2 L(\theta))$ 를 계산하면 가장 Sharp한 부분의 Sharpness 정도를 확인할 수 있다.

이 관점에서 **Sharpness-Aware Minimization (SAM)**이라는 Optimization 방법도 제안된다.

## Overparameterized Model

**Parameter**가 굉장히 많은 모델으로, Modern ML에서는 좋은 성능을 보인다.

**Linear Least Squares**를 통해 왜 좋은 성능을 보이는지 이해해보자.

N개의 Data가 있을 때, Regression을 생각해보자.

- $y_i = x_i^T \beta + \epsilon_i$
- $y = X^T \beta + \epsilon$

$\beta$ 는 **True oracle**으로 실세계의 함수, 파라미터로 생각하면 된다.  $\beta$ 를 정확히 알기 어렵기 때문에 Least square를 통해 최적의  $\hat{\beta}$ 를 찾고자 한다.

**Parameter**의 개수가 Data의 개수보다 많은 경우를 생각해보자

$$\begin{aligned}\hat{\beta} &= \arg \min \left\{ \|b\| : b \text{ minimizes } \frac{1}{2} \|y - Xb\|^2 \right\} \\ &= (X^\top X)^+ X^\top y \\ &= X^\top (X X^\top)^+ y \\ &= X^+ y\end{aligned}$$

위 경우에는 가능한 Solution이 무한 개이기 때문에, **Min-norm solution**을 사용하는 것이 일반적으로 좋다.

Gradient-based Optimization을 사용한다면, 따로 Prior를 통한 Regularization을 사용하지 않아도 Min-norm Solution이 사용되도록 유도된다.

- 초깃값이 0인 경우, Gradient-based Optimization은 Min-norm solution으로 수렴하게 된다.

- Gradient-based Optimization 자체에 Regularization이 존재하는 것처럼 생각할 수 있고, 이를 Implicit regularization이라고 부른다.

Min-norm solution은 모든 가능한 Solution 중, **Weight**의 크기가 가장 작은 Solution이기 때문에 그 자체로 Regularization이다. 이를 **SGD**가 갖는 Implicit regularization이라고 한다.

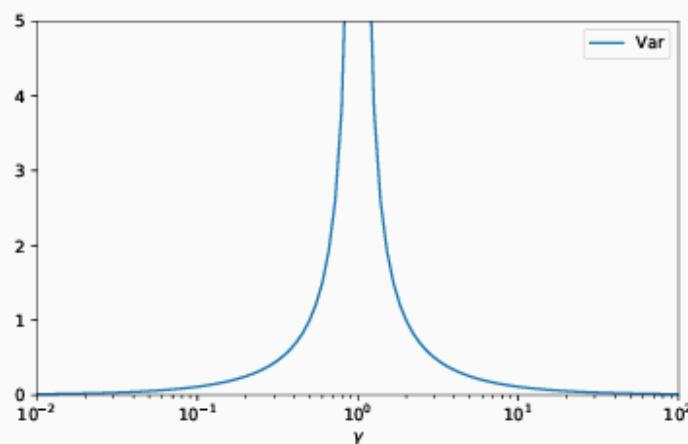
- Weight가 작기 때문에 Generalization에도 좋다.

Overparameterized model에서 Parameter 개수가 Data의 개수보다 많아도 Gradient-based Optimization가 갖는 Implicit regularization 때문에 Double Descent가 일어난다.

실제로 아래 정리에 따르면  $\gamma = \frac{p}{n}$ 이  $[0, 1]$  구간 사이라면 Variance가 계속 증가하다가  $\gamma = 1$  이후 감소하는 형태를 가짐을 확인할 수 있다.

$$\text{Var} \rightarrow \begin{cases} \frac{\gamma}{1-\gamma} & \text{if } 0 < \gamma < 1 \text{ (underparam.)}, \\ \frac{\gamma^{-1}}{1-\gamma^{-1}} = \frac{1}{\gamma-1} & \text{if } 0 < \gamma^{-1} < 1 \text{ (overparam.)}. \end{cases}$$

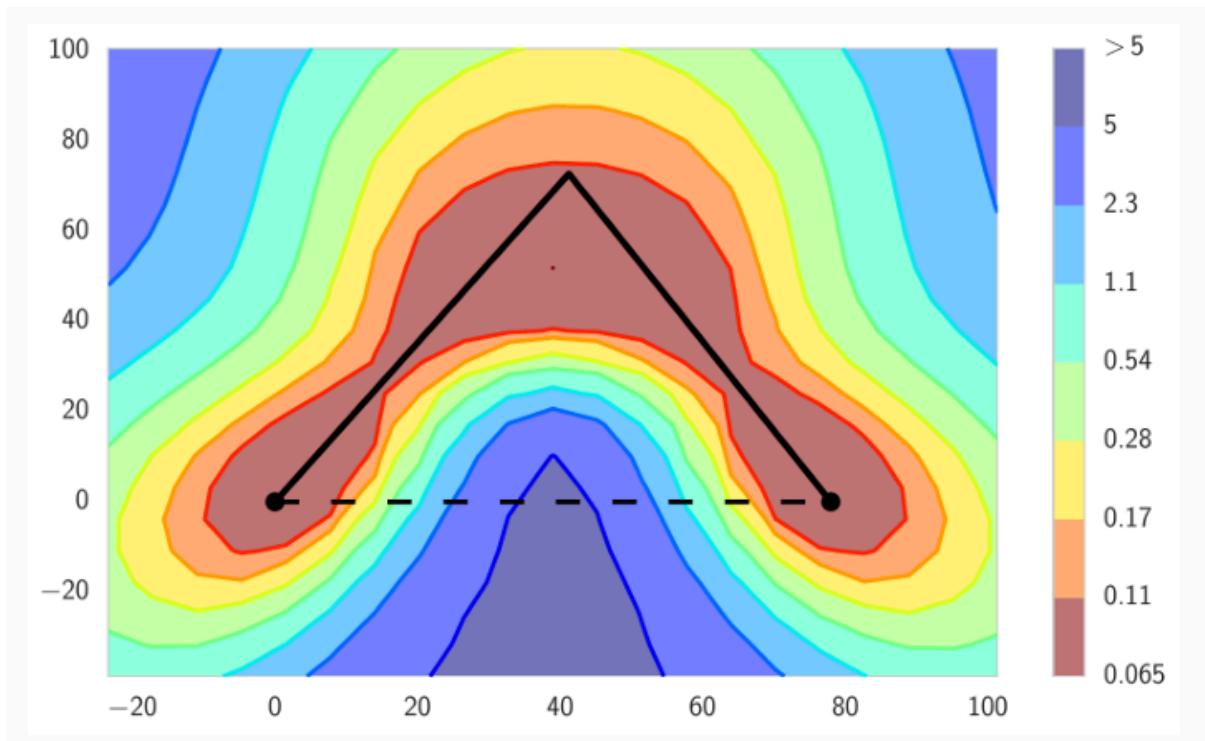
as  $p$  (# params),  $n$  (# data)  $\rightarrow \infty$  with  $p/n \rightarrow \gamma$ .



결국 Parameter, model size를 늘리다보면  $\gamma \leq 1$ 까지는 Test Loss가 감소하다가 증가하고, 더 키우다 보면 Test Loss가 다시 감소하는 것이다.

Gradient-based model이 Solution이 될 수 있는 수많은 점이 이루는 곡선에서의 Min-norm을 찾는 것을 다른 관점에서 확인할 수도 있다.

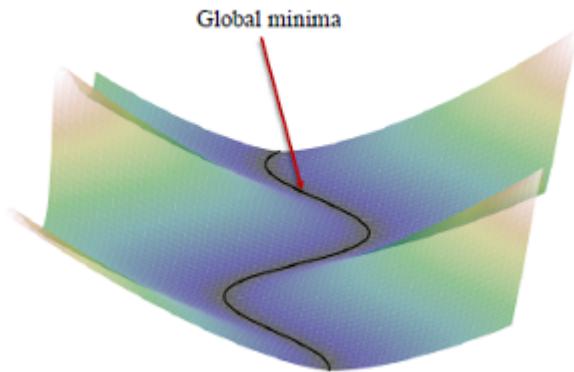
## Mode Connectivity



Overparameterized 경우에는 Loss가 작은 빨간 부분의 영역에서, 두 Minima가 독립적으로 분리된 것이 아니라, **하나의 Curve**를 이루며 연결되어 있는 것을 확인할 수 있다.

- Local minima처럼 각 Minima가 Isolated되지 않는다는 점이 중요하다.

따라서 Overparameterized 경우에 아래 그림처럼 여러 Global minima가 존재할 수 있고, 그 것들은 **하나의 곡선 (Valley)**를 이룬다.



어떤 Minima는 Test loss가 낮을 수도 있고, 다른 Minima는 클 수도 있다. 하지만 SGD는 Implicit regularization 덕분에 그 중 좋은 Minima를 일반적으로 찾을 수 있다.

- 가능한 Solution에 집합에서 Min-norm solution을 찾는 방식이 Generalization에 좋기 때문이다.