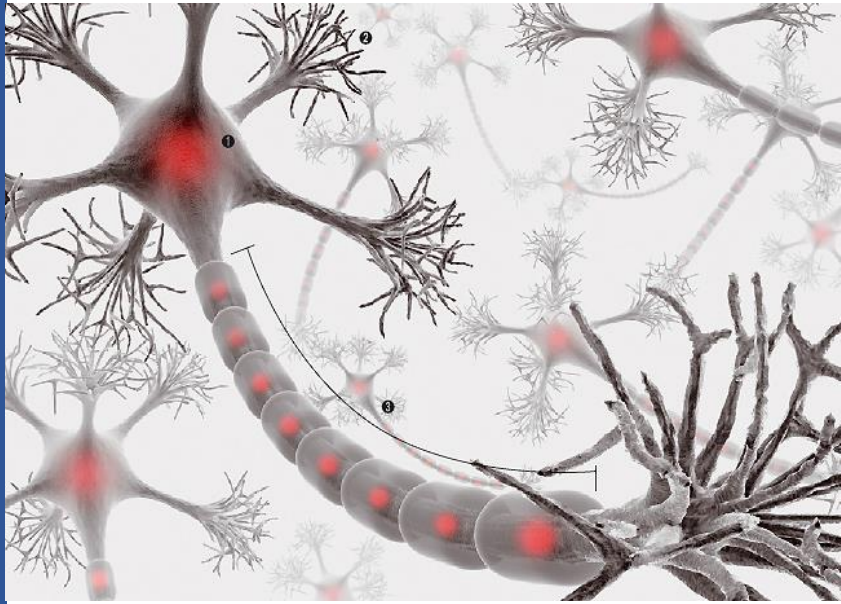
신경망 학습

학습 목표

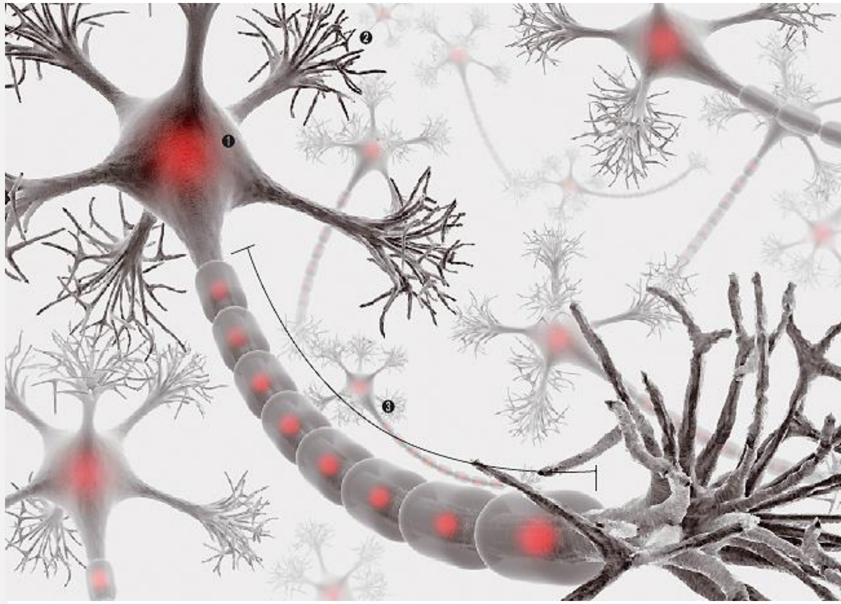
- 신경망 학습과 역전파 알고리즘을 이해한다.

주요 내용

1. 최적화 문제로서의 인공신경망 학습
2. 손실 함수
3. 경사 하강법과 역전파 알고리즘
4. 데이터셋 구성과 훈련 데이터 단위



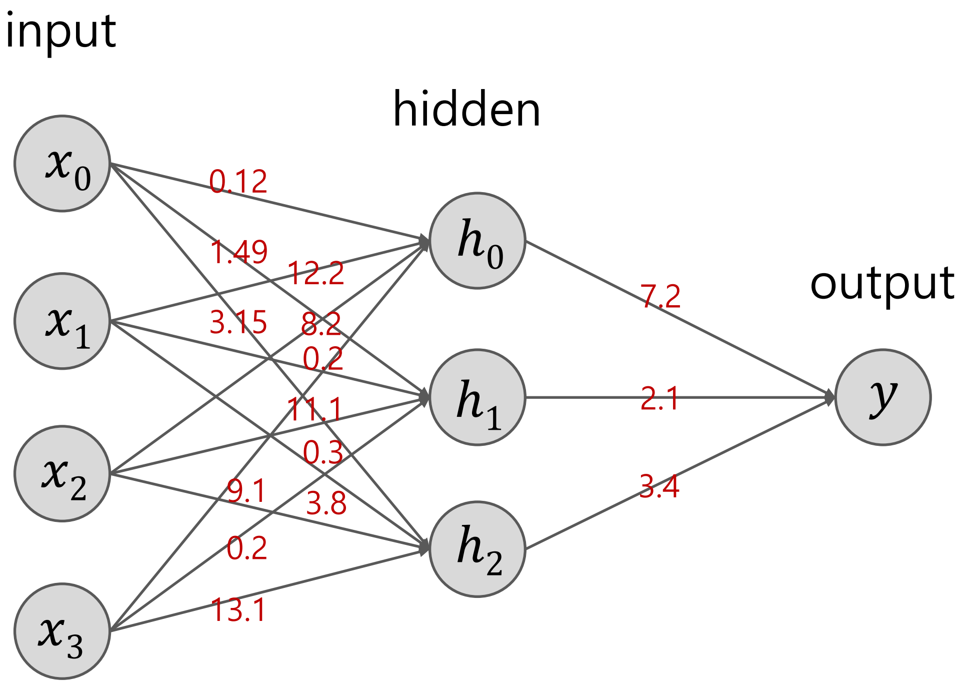
1 최적화 문제로서의 인공신경망 학습



인공 신경망의 학습

주어진 입력과 출력 데이터를 이용해서 신경망 스스로 함수를 찾아내는 것

집값 예측 신경망



방의 수 (개) **3**

면적 (평) **100**

집 종류 (1. 아파트, 2. 주택) **1**

역과의 거리 (km) **1.5**

집값

200,000,000

신경망 스스로 파라미터를 찾아 함수를 정의하는 것을 학습이라고 한다!

최적화 기반의 학습 방식

인공 신경망을 학습한다는 것은?

최적해를 향해 반복적으로 수렴하도록 하는 **최적화 문제**를 푸는 것이다!

Loss 계산

- 모델의 출력과 실제 값의 오차를 이용해서 Loss 계산
- 평균 오차 제곱 (Mean Squared Error), 크로스 엔트로피 (Cross Entropy)

최적화

- Loss가 최소화되도록 파라미터를 변경
- 역전파 알고리즘 + Gradient Descent의 변형 알고리즘

최적화란?

어떤 문제에 대해 선택가능한 해가 여러 개 존재할 때
최적해(Optimal Solution) 또는 근사해(Approximation)을 찾는 방식

예 : Supply Chain Optimization

최적화란?

특정 집합 위에서 정의된 함수, 실수, 정수에 대해 그 값이 최대나 최소가 되는 상태를 해석하는 문제

Standard Form

목적 함수 →

제약 조건 →

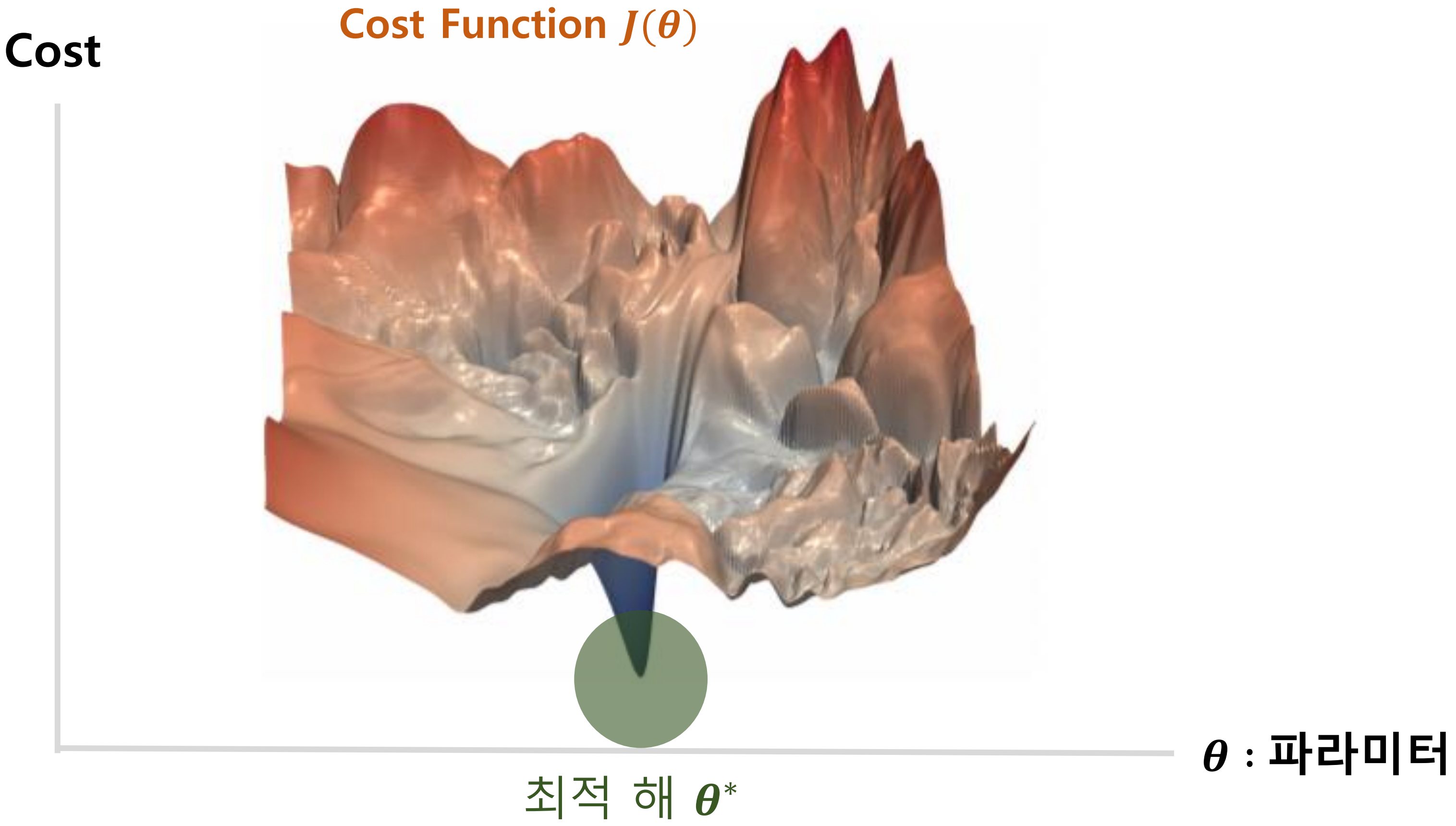
$$
\begin{array}{lll}
\min\limits_{x \in D} & f(x) & \\
\text{subject to} & g_i(x) \le 0, & i = 1, \dots, m \\
 & h_j(x) = 0, & j = 1, \dots, r
\end{array}
$$

목적 함수(Objective Function)

최소화 : 비용 함수 (Cost Function), 손실 함수 (Loss Function)

최대화 : 유틸리티 함수 (Utility Function)

최적화란?



비용 함수를 최소화 시키는 파라미터 값을 찾는 것

최적화 문제

식단 (Diet Problem)

필요한 영양을 만족하는 음식의 조합 중 가장 저렴한 조합을 찾으시오.

$$\begin{aligned} \text{식단} \longrightarrow \min_{x} \quad & c^T x && \longleftarrow \text{식단의 가격} \\ \text{subject to} \quad & Dx \geq d && \longleftarrow \text{최소 영양을 만족하는지 조건} \\ & x \geq 0 && \longleftarrow \text{음식의 양} \end{aligned}$$

x_j	식단을 구성하는 음식 j의 양
c_j	음식 j의 단위 별 가격
D_{ij}	음식 j의 단위 별 각 영양소 i의 함유량
d_i	영양소 i의 최소 섭취량

최적화 문제

회귀 (Regression)

타깃과 인공신경망이 예측한 값의 차이를 최소화하는 파라미터를 찾아라.

파라미터 →

$$\min_{\theta} \quad \frac{1}{n} \sum (t - f(x;\theta))^{2}$$

타깃 (관측 레이블)

모델의 예측

평균 제곱 오차 (Mean Squared Error)

최적화 문제

분류 (Classification)

관측 분포와 인공신경망이 예측한 분포의 차이를 최소화하는 파라미터를 찾아라.

파라미터 →

$$\min_{\theta} \quad -\frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{K} t_k \cdot \log f(x;\theta)_k$$

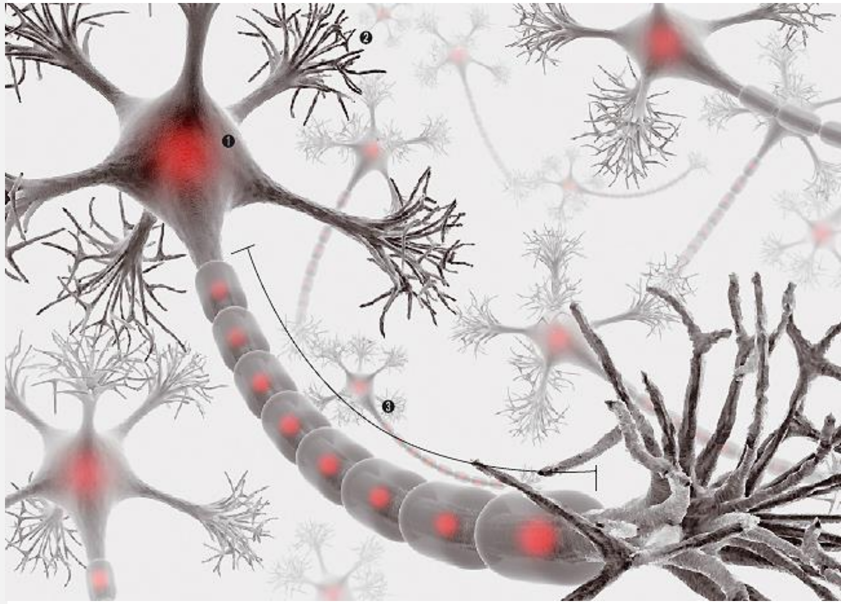
K : Class 개수

관측 분포

모델이 예측한 분포

크로스 엔트로피 (Cross Entropy)

2 손실 함수



Loss Function 정의

Loss Function을 정의하는 두가지 관점

Error Minimization

$$f^* = \underset{f}{\operatorname{argmin}}\ J(f(x;\theta),t)$$

함수 공간에서 오차를 최소화하는
함수를 찾아 보자!

=

Maximum Likelihood

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}}\, J(\theta|x)$$

관측 데이터의 확률을 최대화 하는
모델의 파라미터를 찾아보자!

오차를 최소화하는 것은 최대 우도를 찾는 것과 동일하다

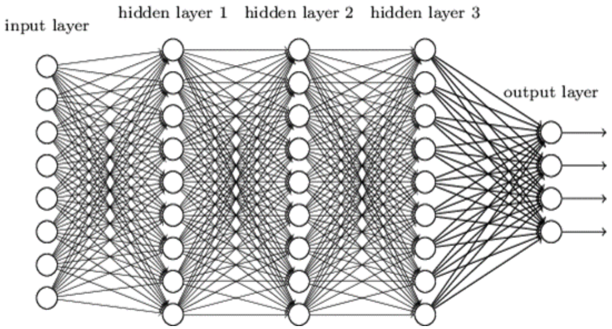
Error Minimization

$$y = f(x;\theta)$$

Input

x

Smooth and Continuous
Geometric Transformation



“함수의 출력과 타깃의 오차로 Loss를 계산”

Output Target Error

$$f(x;\theta) - \boxed{t} = \varepsilon$$

Loss Function

$J(f(x;\theta), t)$

함수 공간에서 Loss를 최소화하는 함수를 찾아 보자!

$$f^* = \underset{f}{\operatorname{argmin}}\ J(f(x;\theta), t)$$

일종의 변분법! Loss function은 functional이다.

Error Minimization

평균 제곱 오차 (MSE : Mean Squared Error)

$\mathbf{L_2}$ Norm으로 오차를 측정한 경우

$$f^* = \underset{f}{\operatorname{argmin}} \mathbb{E}_{x,t \sim \hat{p}_{\text{data}}} \parallel t - f(x;\theta) \parallel_2^2$$

함수 f^*는 각 x에 대해 t의 mean을 예측

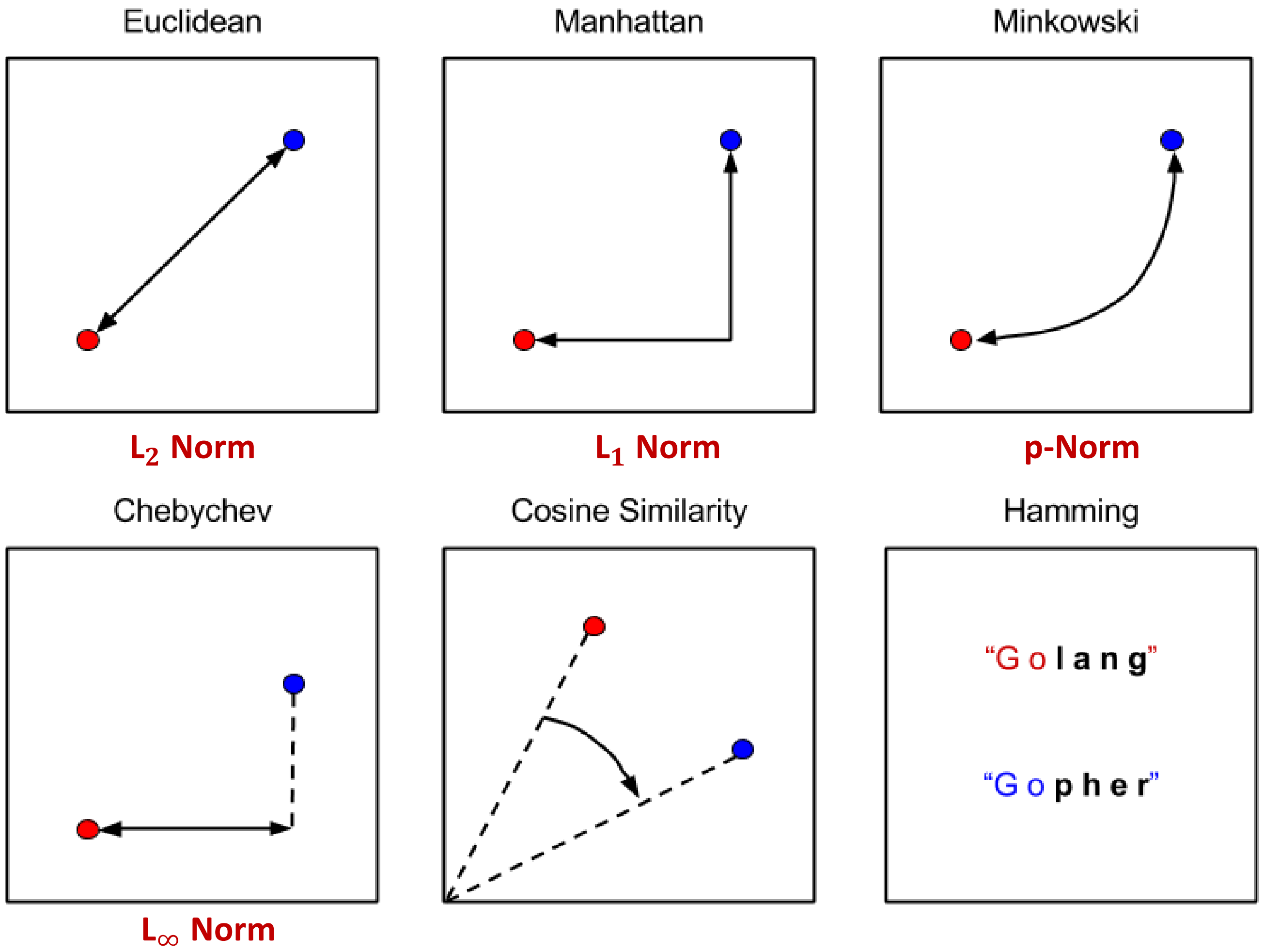
평균 절대값 오차 (MAE : Mean Absolute Error)

$\mathbf{L_1}$ Norm으로 오차를 측정한 경우

$$f^* = \underset{f}{\operatorname{argmin}} \mathbb{E}_{x,t \sim \hat{p}_{\text{data}}} \parallel t - f(x;\theta) \parallel_1^2$$

함수 f^*는 각 x에 대해 t의 median을 예측

참고 Distance (거리)



참고 Norm (크기)

p-Norm

$$\| x \|_p = \left(\sum_{i=0}^{n} |x_i|^p \right)^{1/p} \quad \text{for } P \geq 1$$

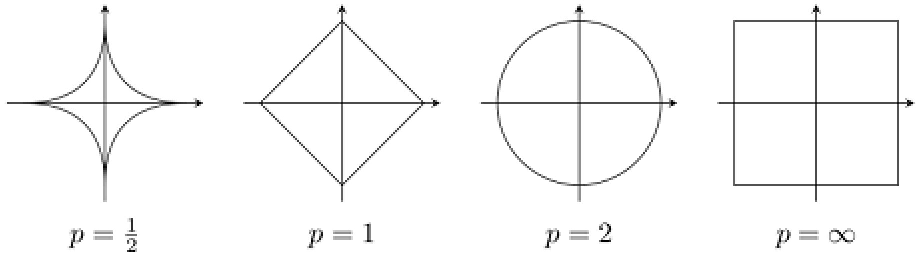
- p=1 : L_1 Norm (Manhattan)
- p=2 : L_2 Norm (Euclidean)
- p=∞ : L_∞ Norm (Chebychev)

Norm Ball

$$\{x \mid \| x - x_c \|_p \leq r \text{ for } P \geq 1$$

- P가 1이상이어야 convex set이 됨

2D norm ball

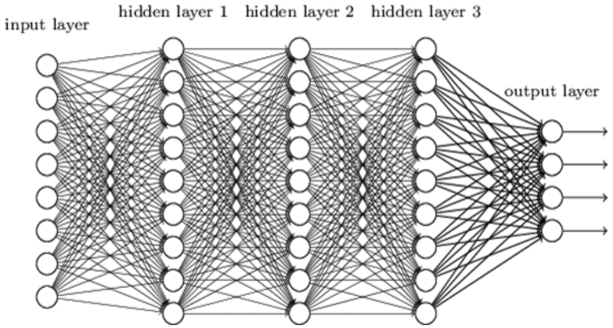


3D norm ball

Maximum Likelihood

$$p_{\text{model}}(t \mid x; \theta)$$

Smooth and Continuous
Geometric Transformation



Input

x

Output

$p_{\text{model}}(t \mid x; \theta)$

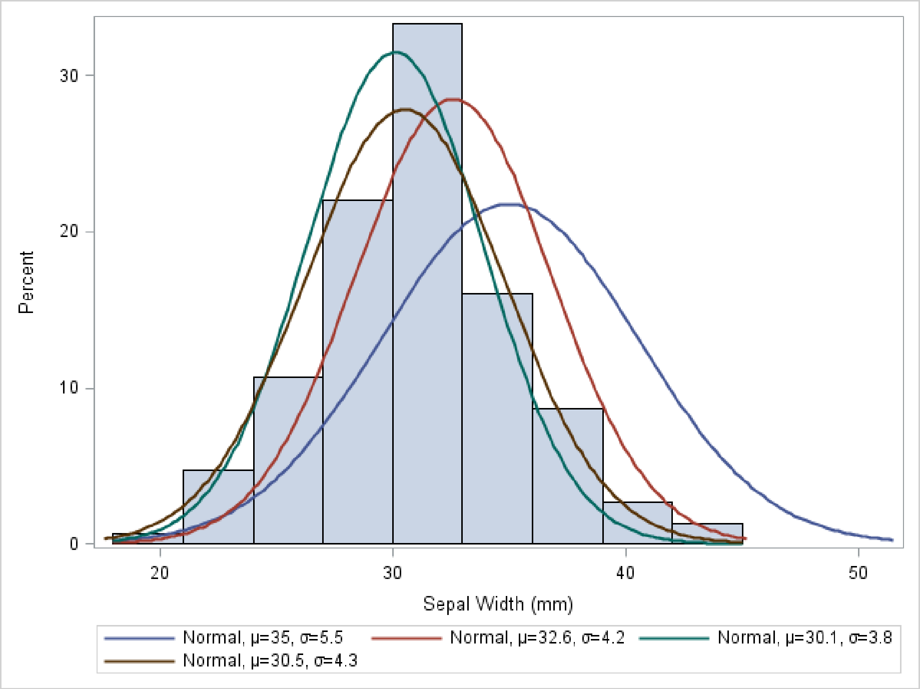
Loss Function

Log−likelihood

관측 값의 확률을 최대화 하는 모델의 파라미터를 찾아보자!

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}} \, \mathbb{E}_{x,t \sim \hat{p}_{\text{data}}} \log p_{\text{model}}(t \mid x; \theta)$$

Maximum Likelihood



Likelihood

파라미터 θ로 추정된 분포에서 관측 값의 확률 값

$$\mathcal{L}(\theta|x) = p(x|\theta)$$

MLE (Maximum Likelihood Estimate)

최대 우도(Maximum_Likelihood) 즉, 관측 값의 확률을 최대화 하는 추정 분포의 파라미터 θ를 찾는 방법

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}} \mathcal{L}(\theta|x)$$

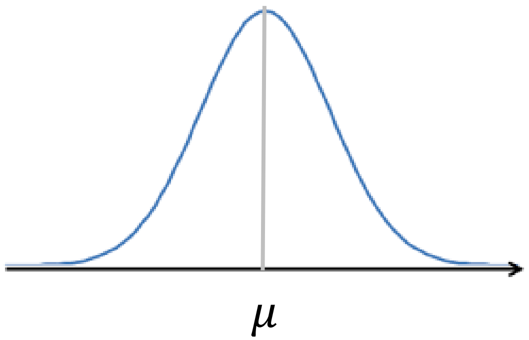
Maximum Likelihood

Gaussian Distribution

$$p_{model}(t \mid x; \theta) = \mathcal{N}(t \mid f(x;\theta), \beta^{-1})$$

μ : 함수의 출력 $f(x;\theta)$이 평균

x : 실제 값 y 가 데이터



$$N(x \mid \mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Loss Function

$$J(\theta) = \mathbb{E}_{x,t \sim \hat{p}_{\text{data}}} \log p_{\text{model}}(t \mid x; \theta)$$

log likelihood

$$= \mathbb{E}_{x,t \sim \hat{p}_{\text{data}}} \log \frac{1}{\sqrt{\beta^{-1}2\pi}} e^{-\frac{(t-f(x;\theta))^2}{2\beta^{-1}}}$$

$$= \boxed{\frac{\beta}{2} \mathbb{E}_{x,t \sim \hat{p}_{\text{data}}} \| t - f(x;\theta) \|_2^2} + \text{const}$$

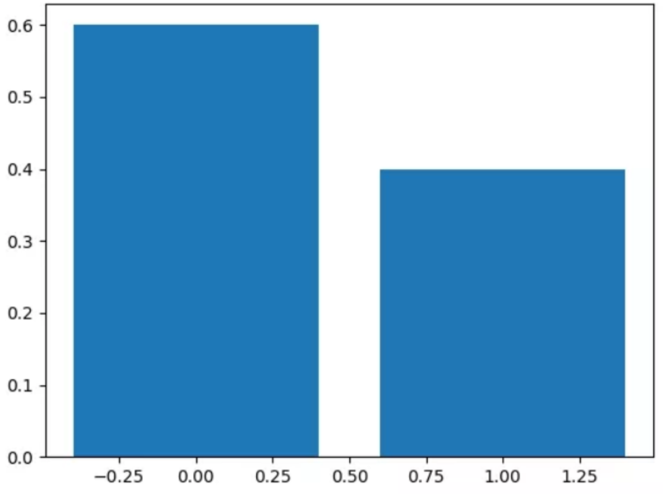
평균 제곱 오차 (MSE - Mean Squared Error)

* Laplacian Distribution인 MAE와 동일

Maximum Likelihood

Bernoulli Distribution

$$p_{model}(t \mid x;\theta) = \mathcal{B}ernoulli(f(x;\theta))$$



$$p(x\,;\mu) = \mu^{x}(1-\mu)^{1-x}$$

$$x = 0, 1$$

Loss Function

$$J(\theta) = \mathbb{E}_{x,t\sim\hat{p}_{\text{data}}} \log p_{\text{model}}(t \mid x;\theta)$$

log likelihood

$$= \mathbb{E}_{x,t\sim\hat{p}_{\text{data}}} \log f(x;\theta)^{\text{t}}(1-f(x;\theta))^{1-\text{t}}$$

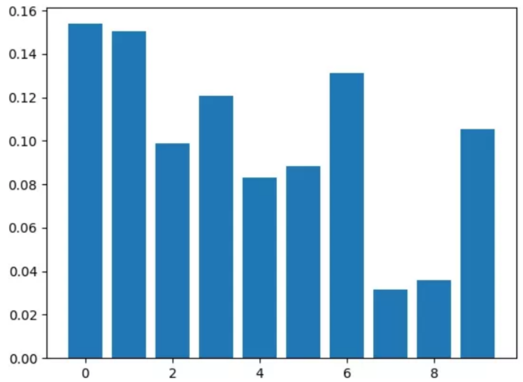
$$= \boxed{\mathbb{E}_{x,t\sim\hat{p}_{\text{data}}} \text{t}\cdot\log f(x;\theta) + (1-\text{t})\cdot\log(1-f(x;\theta))}$$

바이너리 크로스 엔트로피 (Binary Cross Entropy)

Maximum Likelihood

Categorical Distribution

$$p_{model}(t \mid x;\theta) = Categorical(f(x;\theta))$$



$$p(x \mid \mu) = \prod_{k=1}^{K} {\mu_k}^{x_k}$$

$$x = (x_1, x_2, \dots, x_{\mathrm{K}})^{\mathrm{T}} \qquad \mu = (\mu_1, \mu_2, \dots, \mu_{\mathrm{K}})^{\mathrm{T}}$$

$$x_k = \begin{cases} 1, k = i \\ 0, k \neq i \end{cases} \quad i \in \{1, 2, \dots, K\}$$

i번째 Category에 속할 경우

$$\sum_{k=1}^{K} \mu_k = 1$$

Loss Function

$$J(\theta) = \mathbb{E}_{x,t \sim \hat{p}_{\text{data}}} \log p_{\text{model}}(t \mid x;\theta)$$

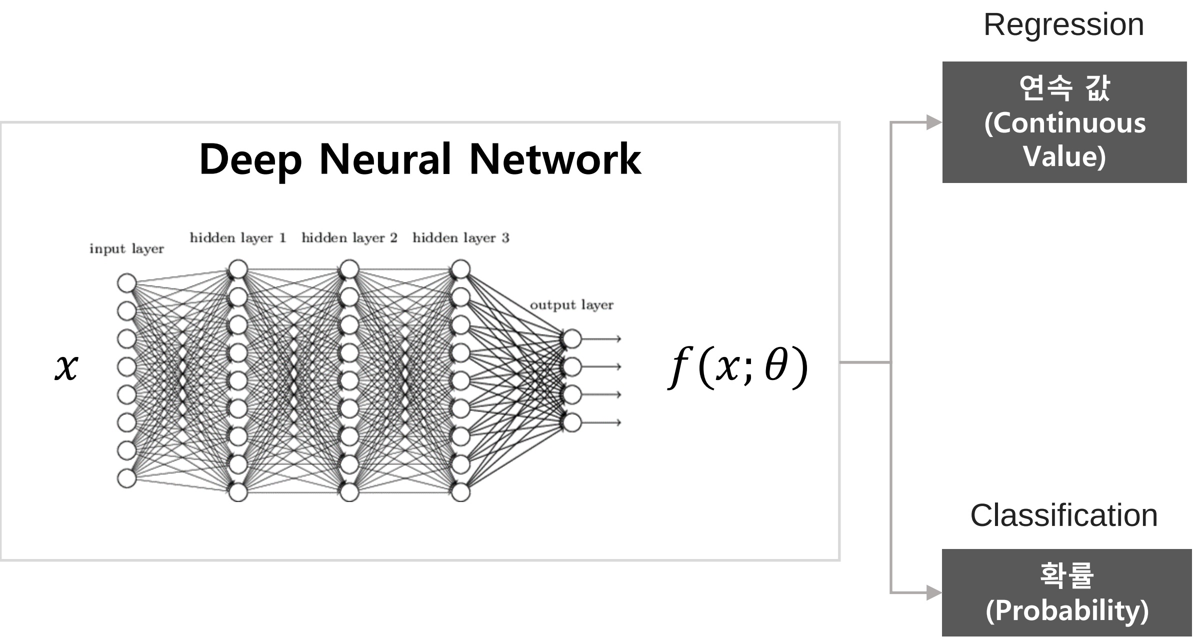
log likelihood

$$= \mathbb{E}_{x,t \sim \hat{p}_{\text{data}}} \log \prod_{k=1}^{K} {f(x;\theta)_k}^{t_k}$$

$$= \boxed{\mathbb{E}_{x,t \sim \hat{p}_{\text{data}}} \sum_{k=1}^{K} t_k \cdot \log f(x;\theta)_k}$$

크로스 엔트로피 (Cross Entropy)

Loss Function 요약



Deep Neural Network

x

$f(x;\theta)$

Regression

연속 값 (Continuous Value)

Classification

확률 (Probability)

K : Class 개수

평균 제곱 오차 (Mean Squared Error)

$$J(f(x;\theta),t) = \mathbb{E}_{x,t\sim\hat{p}_{\text{data}}} \| t - f(x;\theta) \|_2^2$$

$$\approx \frac{1}{n}\sum_{i=1}^{n}(t - f(x;\theta))^2$$

크로스 엔트로피 (Cross Entropy)

$$J(f(x;\theta),t) = \mathbb{E}_{x,t\sim\hat{p}_{\text{data}}} \sum_{k=1}^{K} t_k \cdot \log f(x;\theta)_k$$

$$\approx \frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{K} t_k \cdot \log f(x;\theta)_k$$

참고 정보량 (Self-Information)

확률 변수의 정보량은?

- 정보란 **놀라움의 정도**를 의미한다. 😮
- 발생 확률이 낮은 사건일수록 정보가 크다.

$$\frac{1}{p(x)}$$

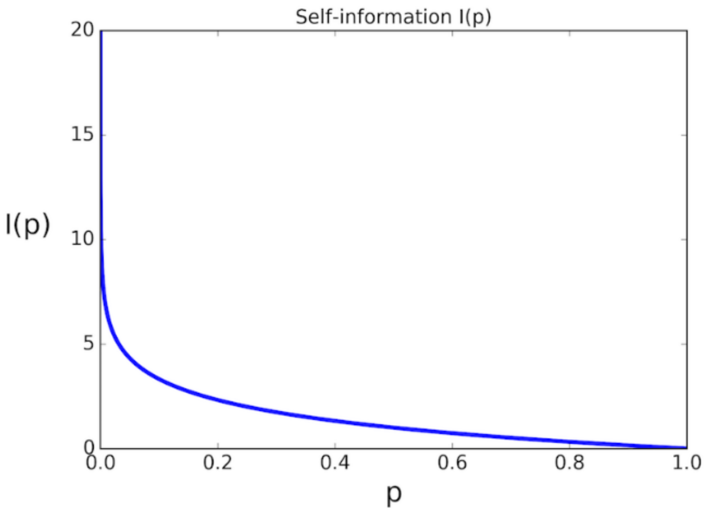
- 정보를 표현하는 Bit 수를 구해보자.

$$\log\frac{1}{p(x)} = -\log p(x)$$

정보량 (Self-Information)

Random Variable의 확률 값을 표현하기 위해 필요한 Bit 수

$$I(\mathrm{x}) = -\log p(x)$$

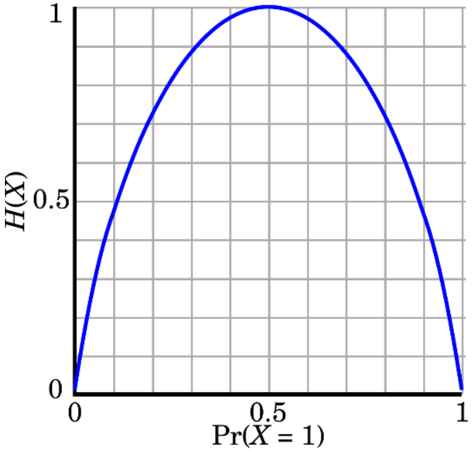


참고 엔트로피 (Entropy)

엔트로피 (Entropy)

- 확률 분포가 얼마나 불확실한지 또는 랜덤한지를 나타냄
- Random Variable의 정보량의 기댓값

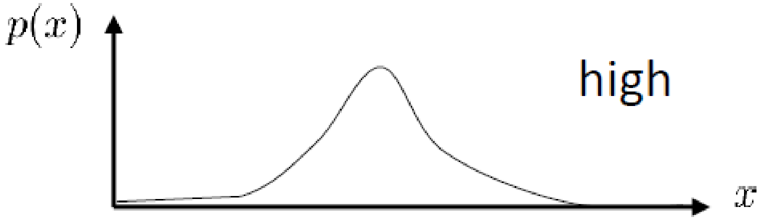
$$\mathcal{H}(p) = \mathbb{E}_{x \sim p(x)}[-\log p(x)] = -\int_{x} p(x) \log p(x) \; dx$$



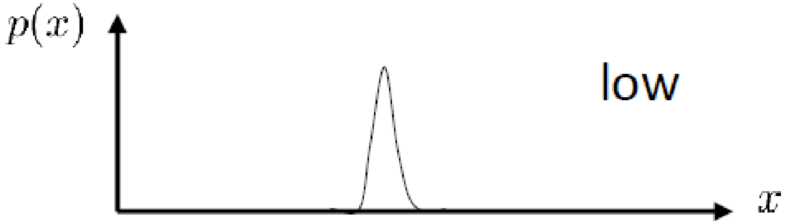
동전을 던졌을 때 앞면이 나올 확률에 대한 엔트로피

Random variable p가 얼마나 random한가?

엔트로피가 높으며 기대 확률은 낮다



엔트로피가 낮으면 기대 확률은 높다



참고 크로스 엔트로피 (Cross Entropy)

크로스 엔트로피 (Cross Entropy)

- 두 확률 분포의 차이 또는 유사하지 않은 정도(dissimilarity)를 나타냄
- 추정 확률 q 의 정보량을 확률 p 에 대해 구한 기댓값

$$\mathcal{H}(p,q) = -\,\mathbb{E}_{x\,\sim\,p(x)} \log q(x) = -\int_x p(x) \log q(x)\ dx$$

$p(x)$: Random Variable의 확률

$q(x)$: 추정 확률

참고 크로스 엔트로피 (Cross Entropy)

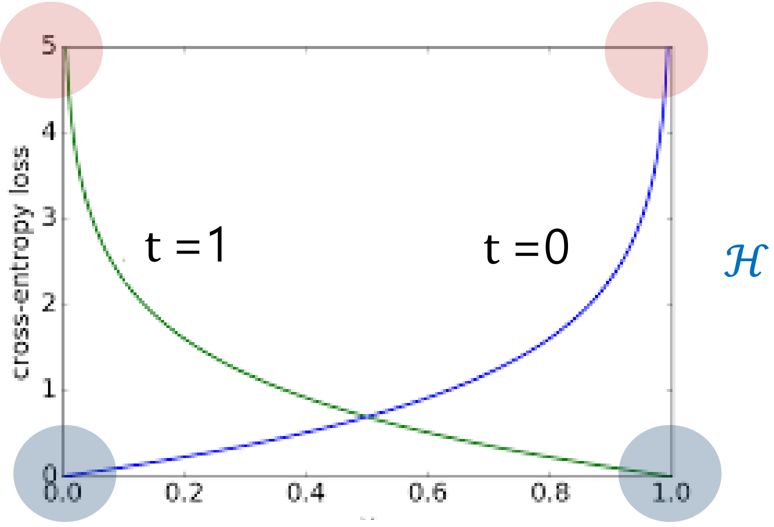
Cross Entropy Loss $\mathcal{H}(t, f(x;\theta))$

$$J(\theta) = -\mathbb{E}_{x,\mathrm{t} \sim \hat{p}_{\mathrm{data}}} \mathrm{t} \cdot \log f(x;\theta) + (1-\mathrm{t}) \cdot \log(1 - f(x;\theta))$$

t =1일 때 $f(x;\theta)$ =0이면
Cross Entropy > 0

t =0일 때 $f(x;\theta)$ =1이면
Cross Entropy > 0

$$\mathcal{H}(\mathrm{t}=1, f(x;\theta)) = -\log(f(x;\theta))$$



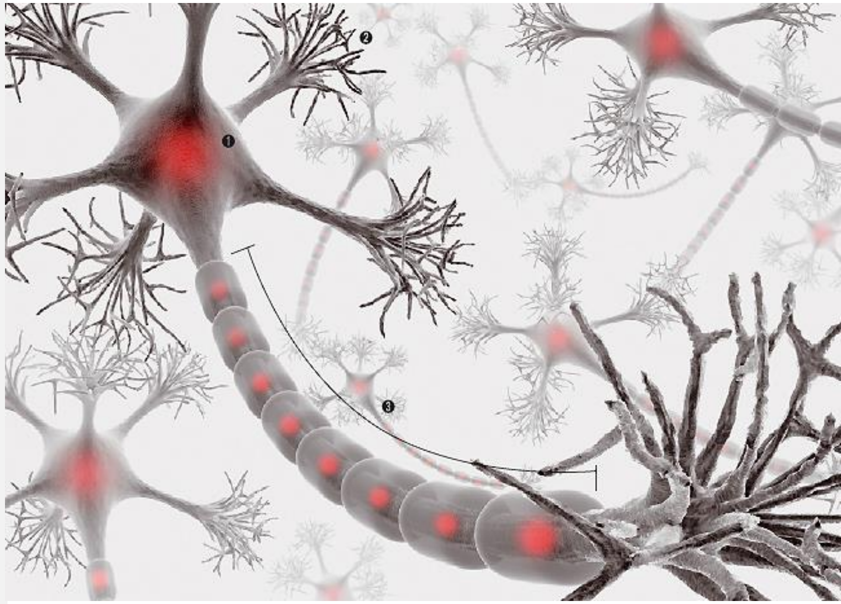
$$\mathcal{H}(\mathrm{t}=0, f(x;\theta)) = -\log(1 - f(x;\theta))$$

$f(x;\theta)$

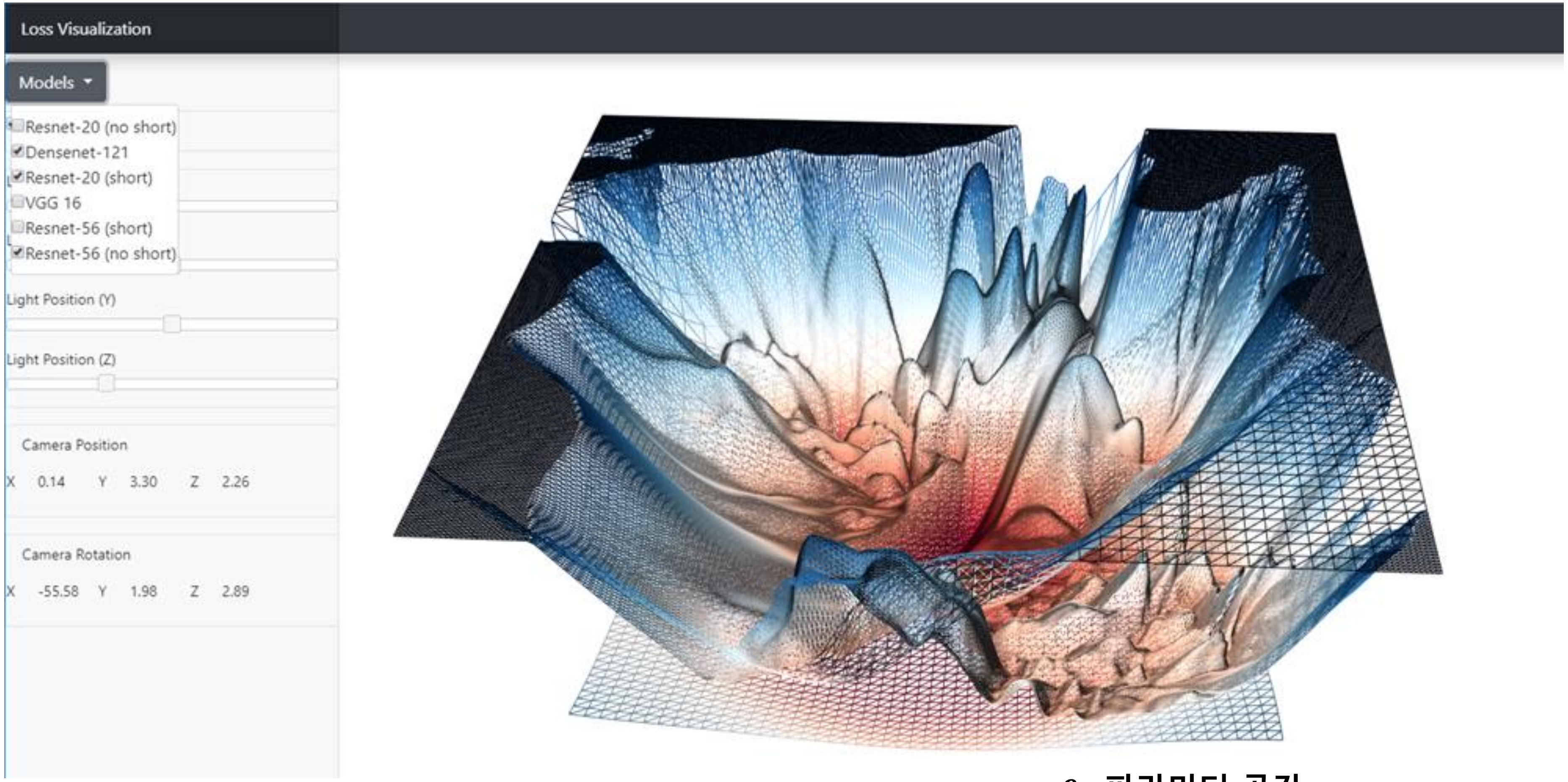
t =0일 때 $f(x;\theta)$ =0 이면
Cross Entropy = 0

t =1일 때 $f(x;\theta)$ =1이면
Cross Entropy = 0

3 경사 하강법과
역전파 알고리즘



Loss Surface



☐ Resnet-20 (no short)
☑ Densenet-121
☑ Resnet-20 (short)
☐ VGG 16
☐ Resnet-56 (short)
☑ Resnet-56 (no short)

Light Position (Y)

Light Position (Z)

Camera Position

X 0.14 Y 3.30 Z 2.26

Camera Rotation

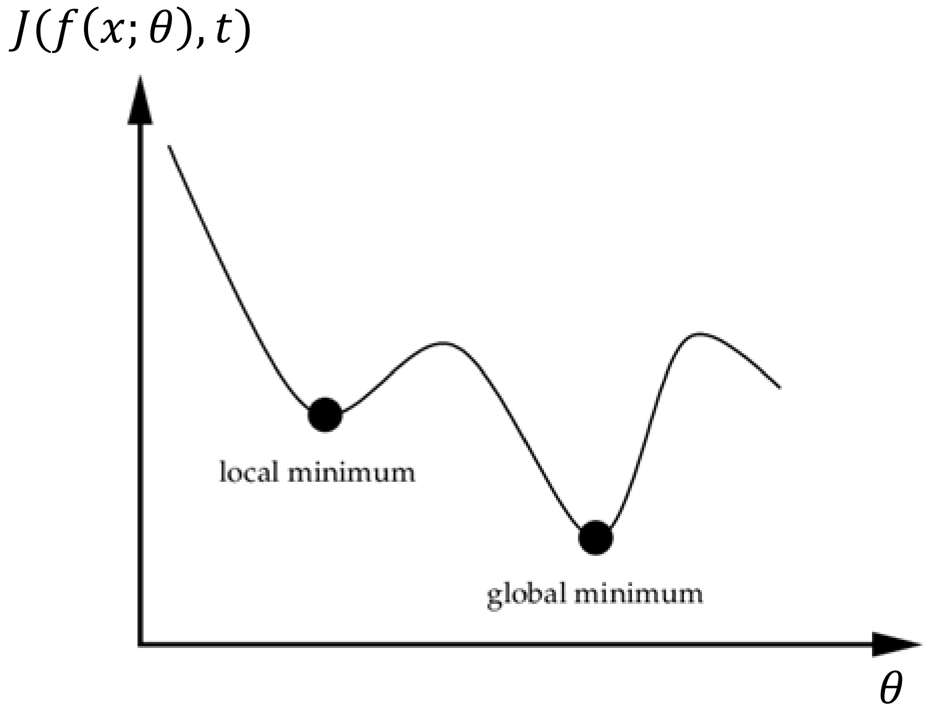
X -55.58 Y 1.98 Z 2.89

θ : 파라미터 공간

http://www.telesens.co/2019/01/16/neural-network-loss-visualization/

Loss를 최소화 하려면?

Loss Minimization



최적화 알고리즘

1차 미분

- Gradient Descent
- Variants of Gradient Descent :
  : SGD, Adagrad, Momentum, RMS prob, Adam

Deep Learning에서 주로 사용하는 방법

1.5차 미분

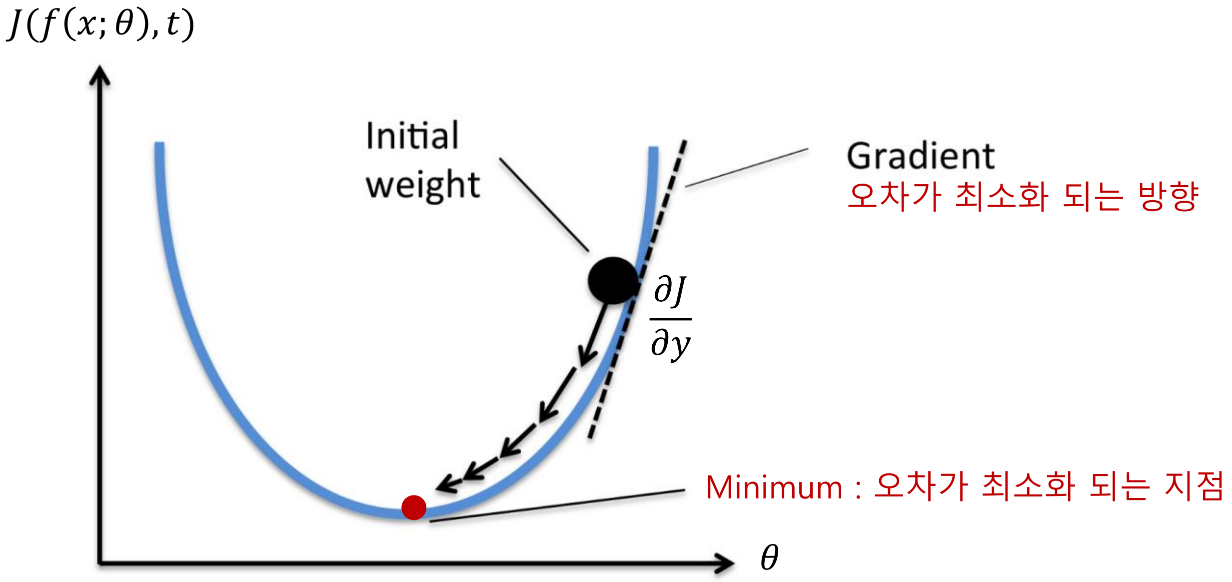
- Quasi-Newton Method
- Conjugate Gradient Descent
- Levenberg-Marquardt Method

2차 미분

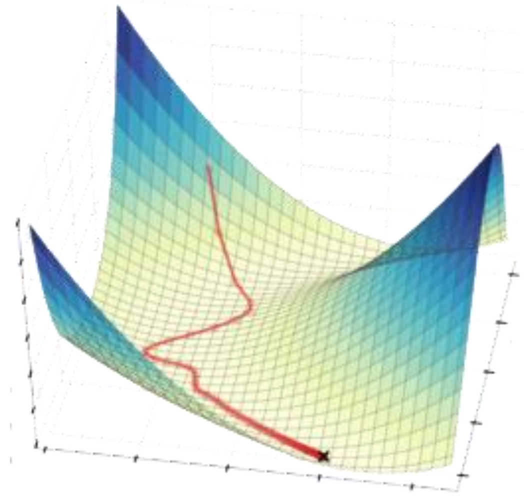
- Newton Method
- Interior Point Method

Gradient Descent

Gradient Descent



3D View



Parameter Update

$$\theta^{+} = \theta - \alpha \frac{\partial J}{\partial \theta}$$

Step Size

Gradient

참고 연속 함수의 미분

Real Valued Function

$$f : \mathbb{R}^n \rightarrow \mathbb{R}$$

Gradient

$$\nabla f = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{pmatrix}$$

Hessian

$$\nabla^2 f = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{pmatrix}$$

$$f(x) = f(x_1, x_2, \ldots, x_n)$$

Vector Function

$$f : \mathbb{R}^n \rightarrow \mathbb{R}^m$$

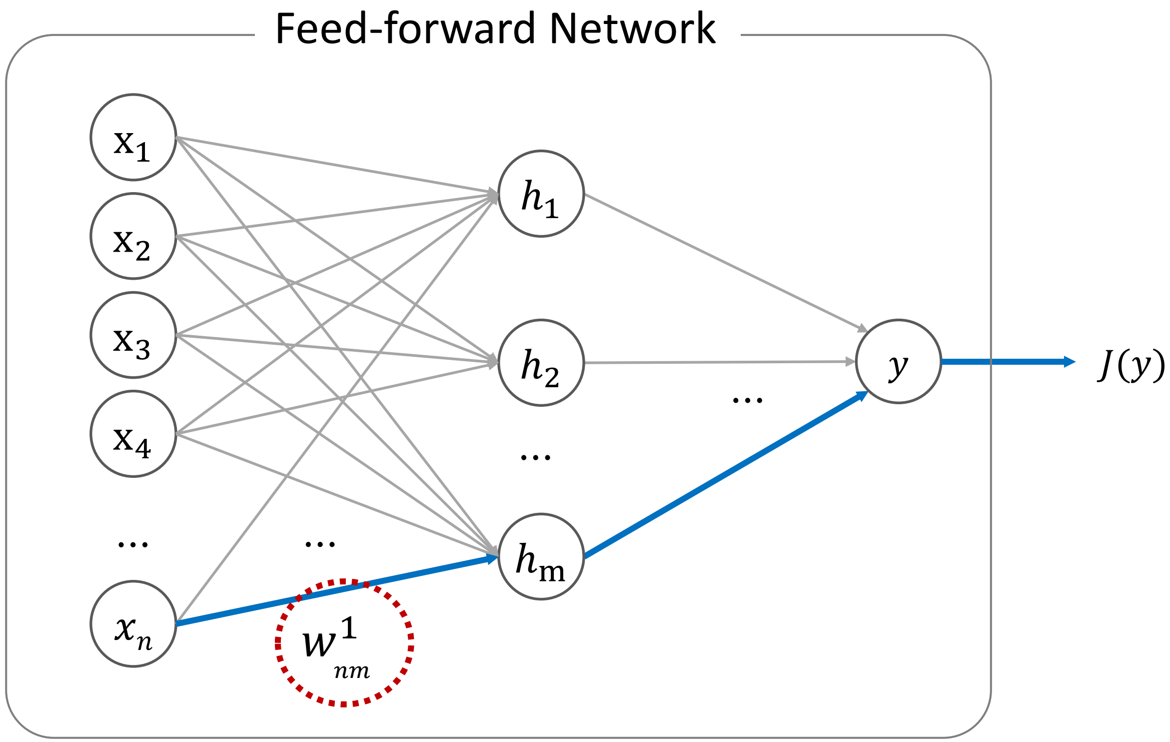
Jacobian

$$\boldsymbol{J}f = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & & \frac{\partial f_2}{\partial x_n} \\ \vdots & & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & \cdots & \frac{\partial f_m}{\partial x_n} \end{pmatrix}$$

$$f(x) = (f_1(x), f_2(x), \ldots, f_m(x))$$

$$x = (x_1, x_2, \ldots, x_n)$$

Gradient Descent



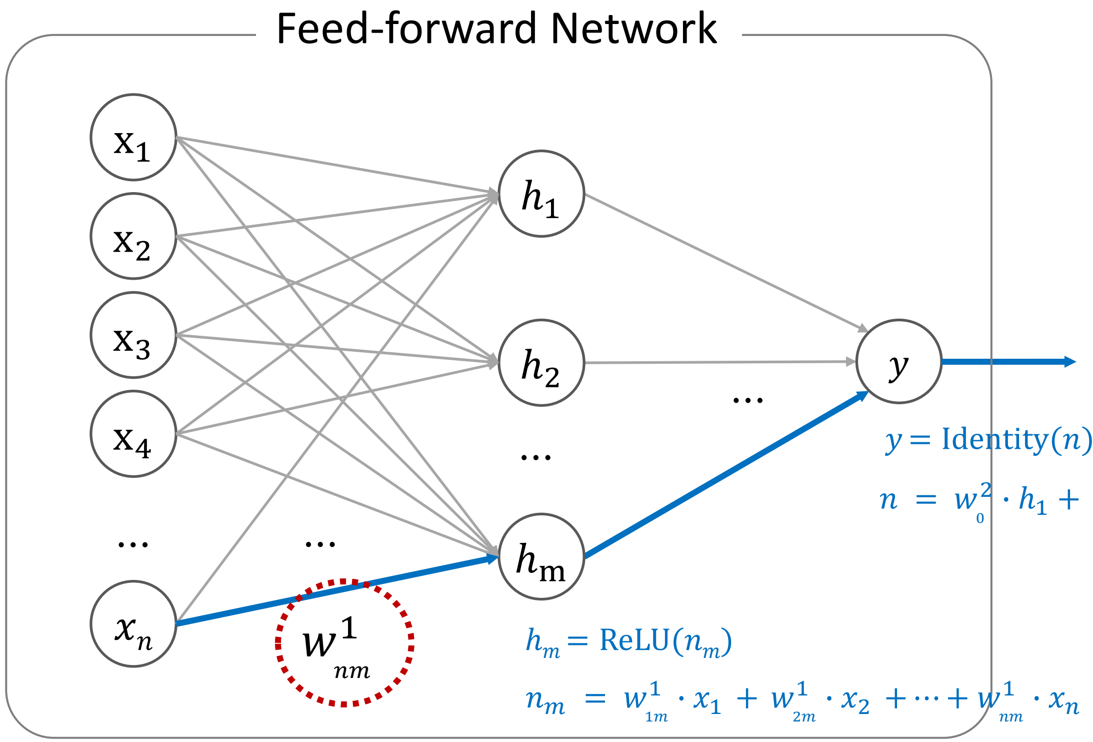
Parameter Update

$$w^{1}_{nm}{}^{+} = w^{1}_{nm} - \alpha \frac{\partial J}{\partial w^{1}_{nm}}$$

Step Size

Gradient

Gradient Descent



$h_m = \mathrm{ReLU}(n_m)$

$n_m = w^{1}_{1m} \cdot x_1 + w^{1}_{2m} \cdot x_2 + \cdots + w^{1}_{nm} \cdot x_n$

$y = \mathrm{Identity}(n)$

$n = w^{2}_{0} \cdot h_1 + w^{2}_{2} \cdot h_2 + \cdots + w^{2}_{m} \cdot h_m$

$J(y) = \frac{1}{\mathrm{N}} \sum_{i=1}^{\mathrm{N}} (y - \mathrm{t})^2$

Gradient of Parameter

$$w^{1\,+}_{nm} = w^{1}_{nm} - \alpha \frac{\partial J}{\partial w^{1}_{nm}}$$

Step Size

Gradient

“가중치는 Loss Function의 간접 파라미터이므로 직접 미분이 안됨”

참고 합성 함수의 미분

$z = t^2$

$t = 2x + y$ 과 같은 식이 있을 때 미분 $\frac{\partial z}{\partial x}$ 를 구해보자.

⇔ $z = (2x + y)^2$

$$\frac{\partial z}{\partial x} = 2(2x + y).2 = 8x + 4y$$

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial t} \cdot \frac{\partial t}{\partial x}$$

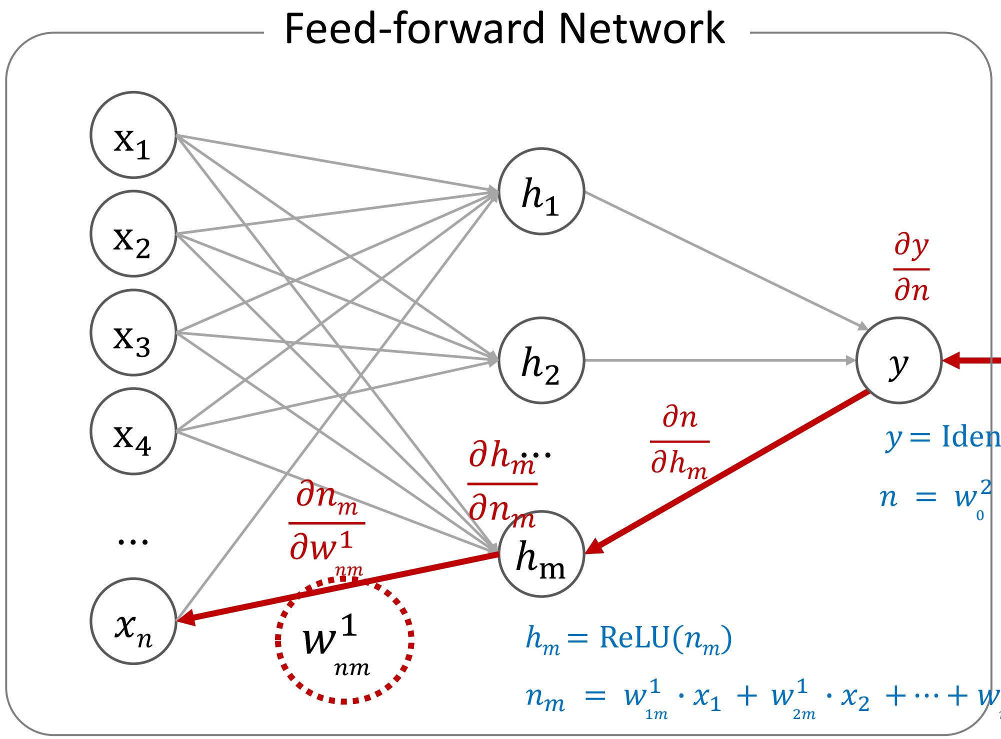
연쇄 법칙(Chain Rule)을 사용

$$\left.\begin{aligned} \frac{\partial z}{\partial t} &= 2t \\ \frac{\partial t}{\partial x} &= 2 \end{aligned}\right\}$$

각 식에 대한 미분을 구함

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial t} \cdot \frac{\partial t}{\partial x} = 2t.2 = 4(2x + y) = 8x + 4y$$

Backpropagation



$y = \text{Identity}(n)$

$n = w^2_0 \cdot h_1 + w^2_2 \cdot h_2 + \cdots + w^2_m \cdot h_m$

$h_m = \text{ReLU}(n_m)$

$n_m = w^1_{1m} \cdot x_1 + w^1_{2m} \cdot x_2 + \cdots + w^1_{nm} \cdot x_n$

$J(y) = \frac{1}{N} \sum_{i=1}^{N} (y - t)^2$

Gradient of Parameter

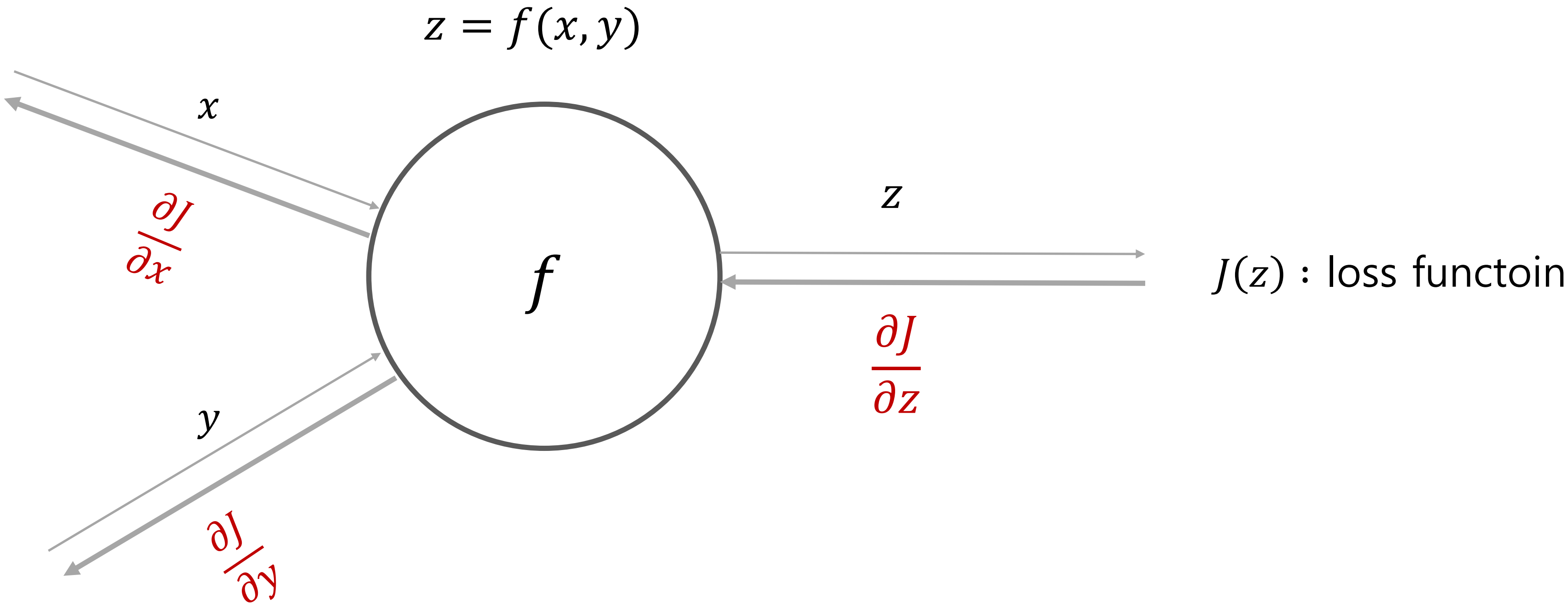
$$\frac{\partial J}{\partial w^1_{nm}} = \boxed{\frac{\partial J}{\partial y} \cdot \frac{\partial y}{\partial n} \cdot \frac{\partial n}{\partial h_m} \cdot \frac{\partial h_m}{\partial n_m} \cdot \frac{\partial n_m}{\partial w^1_{nm}}}$$

연쇄 법칙 (Chain Rule) 사용

$$= \frac{1}{N} \sum_{i=1}^{N} 2(y - t) . \text{Identity}'(n) . w^2{}_m . \text{ReLU}'(n_m) . x_n$$

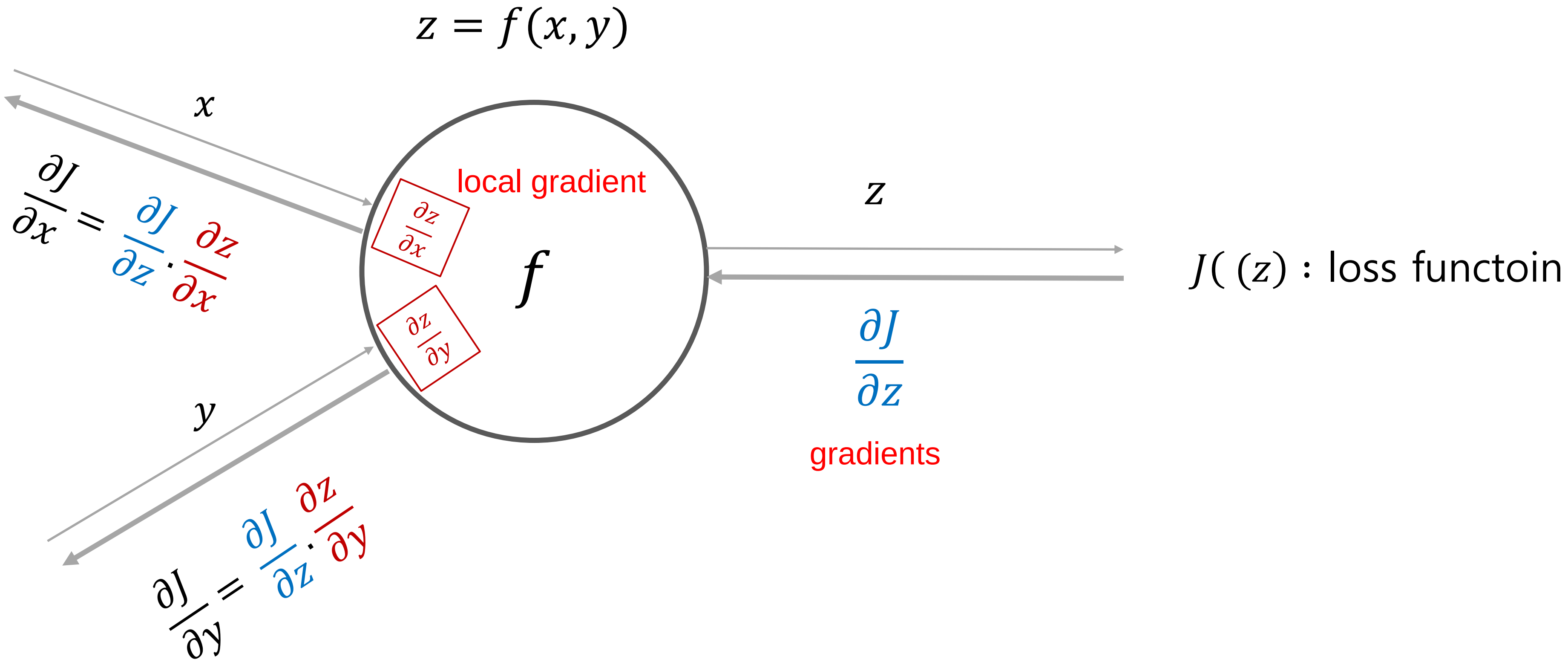
$$\begin{cases} \frac{\partial J}{\partial y} = \frac{1}{N} \sum_{i=1}^{N} 2(y - t) \\ \frac{\partial y}{\partial n} = \text{Identity}'(n) \\ \frac{\partial n}{\partial h_m} = w^2{}_m \\ \frac{\partial h_m}{\partial n_m} = \text{ReLU}'(n_m) \\ \frac{\partial n_m}{\partial w^1_{nm}} = x_n \end{cases}$$

Backpropagation



Loss $J(z)$에 대해 x, y, z의 미분을 구하라!

Backpropagation

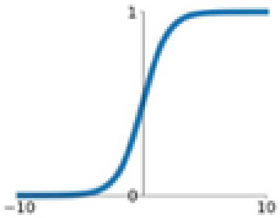


각 노드에서 Local Gradient를 구한 후 전달 받은 Gradient와 곱해서 이전 노드에 전달

주요 Activation Function의 미분

Sigmoid

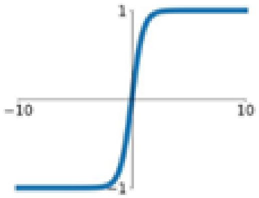
$$f(x) = \frac{1}{1+e^{-x}}$$



$$f'(x) = f(x)(1 - f(x))$$

Tanh

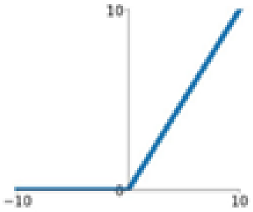
$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$



$$f'(x) = 1 - f(x)^2$$

ReLU

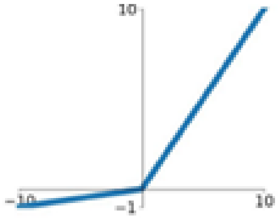
$$f(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$$



$$f'(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$$

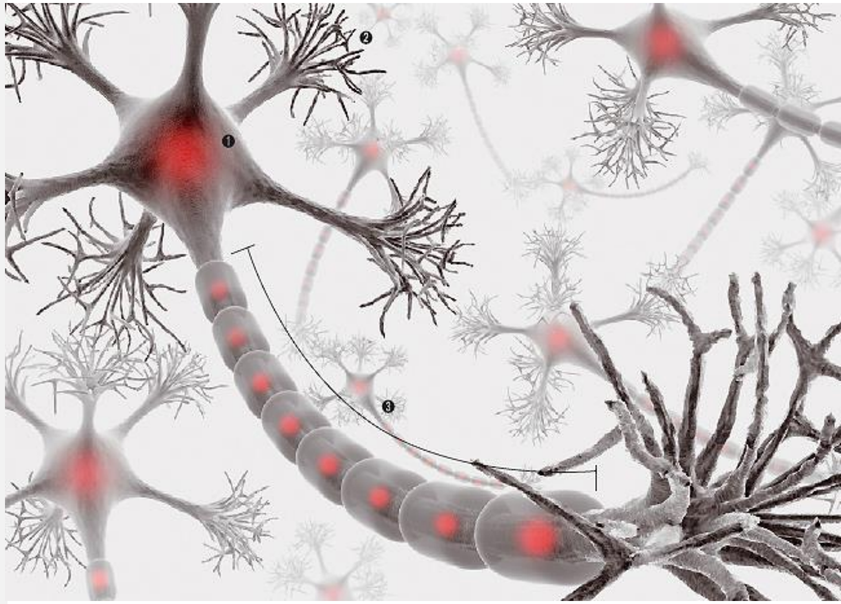
Leaky ReLU

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ 0.01x & \text{otherwise} \end{cases}$$



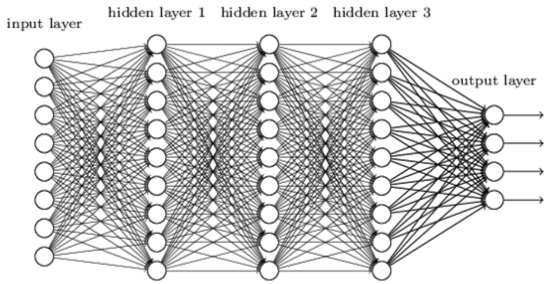
$$f'(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0.01 & \text{otherwise} \end{cases}$$

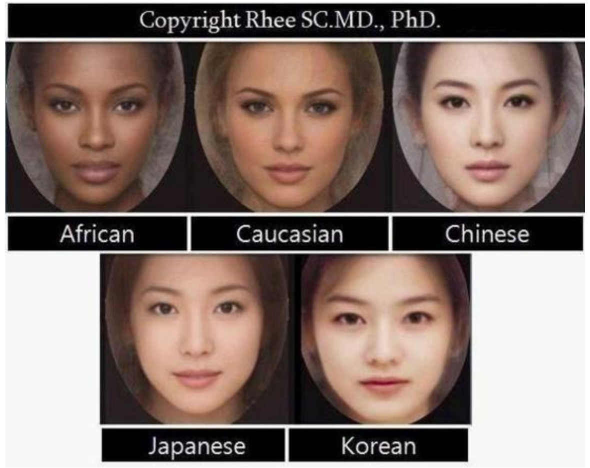
4 데이터셋 구성과 훈련 데이터 단위



관측 데이터 vs. 훈련 데이터

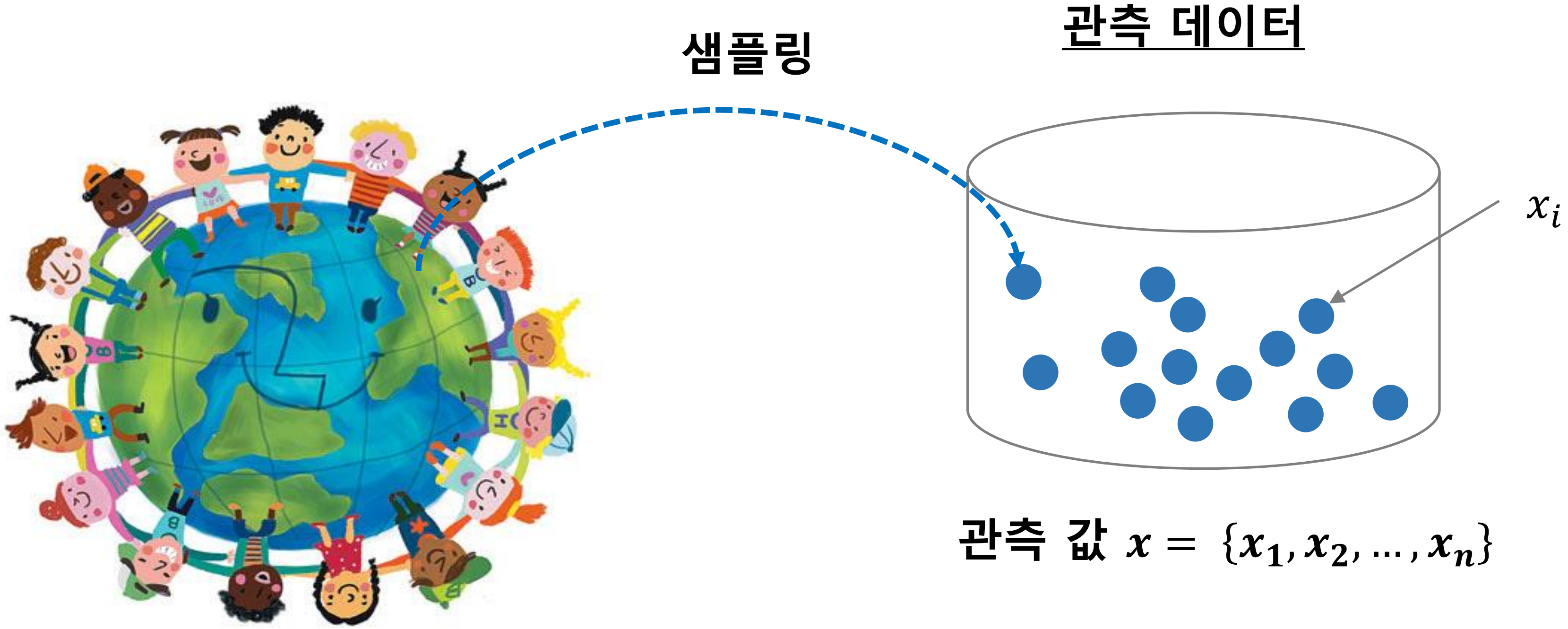
예 : 인종 분류 문제





Real World에서 데이터 전체를 구할 수 있을까?

관측 데이터 vs. 훈련 데이터

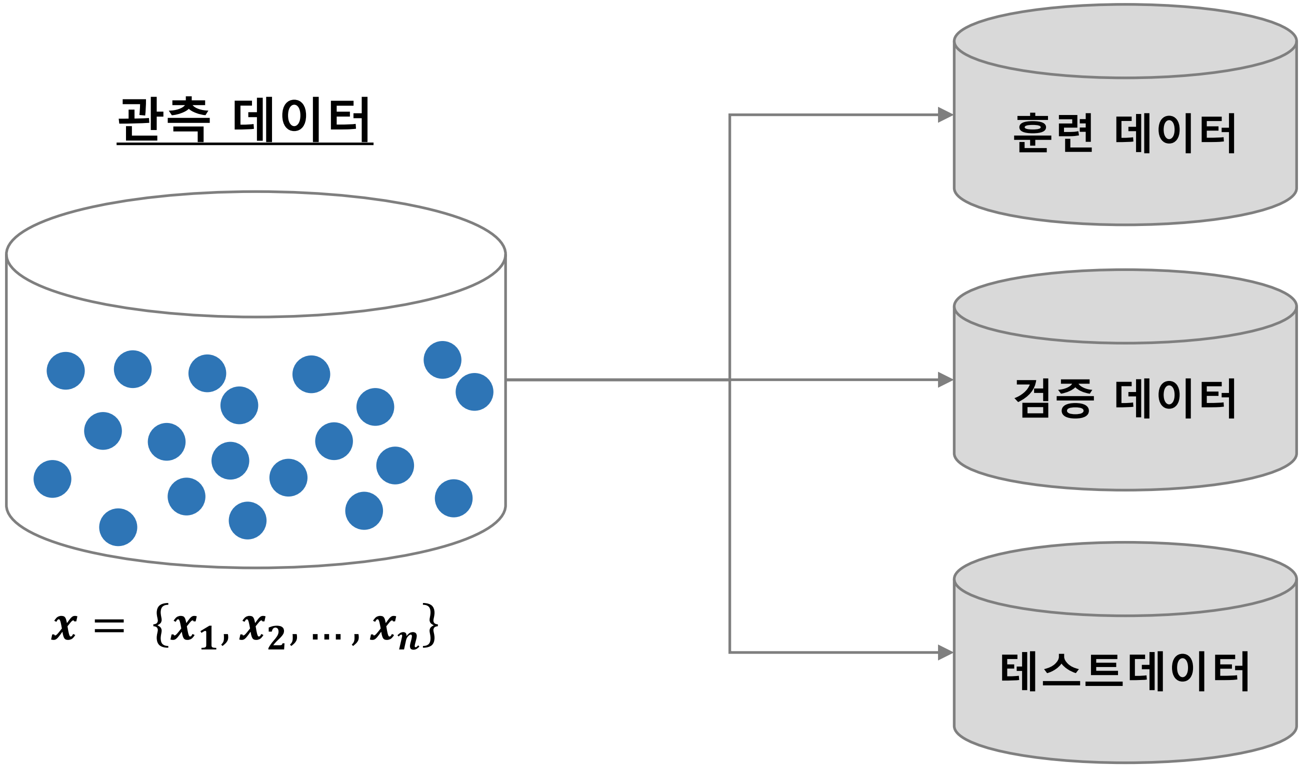


관측 값 $x = \{x_1, x_2, \ldots, x_n\}$

데이터를 샘플링을 통해 관측 데이터를 만듬

관측 데이터 vs. 훈련 데이터

관측 데이터를 세 종류의 데이터로 분리

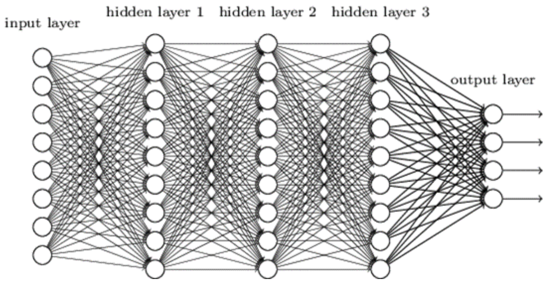


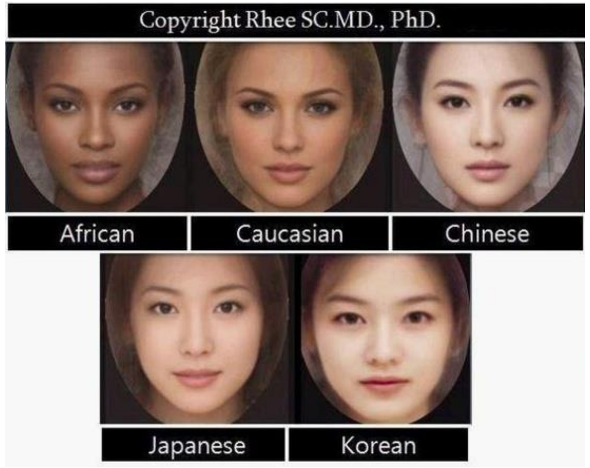
$$x = \{x_1, x_2, \dots, x_n\}$$

관측 데이터 vs. 훈련 데이터

예 : 인종 분류 문제

훈련 데이터





훈련 단위

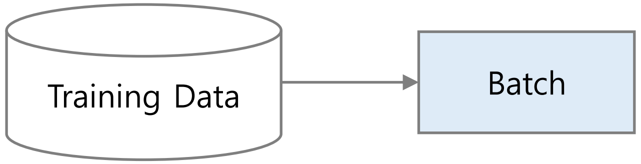
훈련 데이터를 한꺼번에 모델에 넣어서 훈련시킬 수 있을까?

- 메모리/CPU/GPU 용량이 허용되면 가능! 하지만 훈련이 속도가 매우 느림
- 훈련 데이터가 점진적으로 계속 늘어나는 경우엔 불가능 (Online Training)

훈련 단위

Batch

전체 훈련 데이터를 하나의 배치로 만들어 훈련

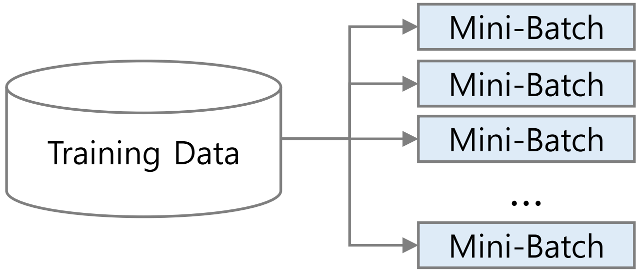

Training Data

Batch

훈련 집합이 너무 크면 불가능!

Mini-Batch

n개 샘플을 묶은 미니배치 단위로 훈련


Training Data

Mini-Batch

Mini-Batch

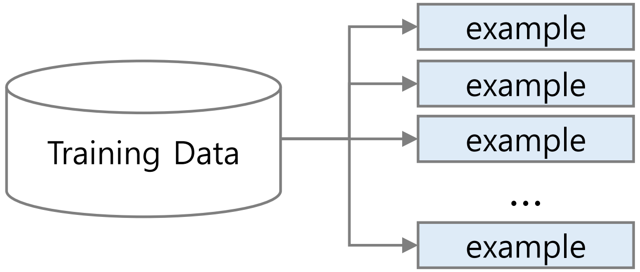
Mini-Batch

…

Mini-Batch

Stochastic

각 example 단위로 훈련


Training Data

example

example

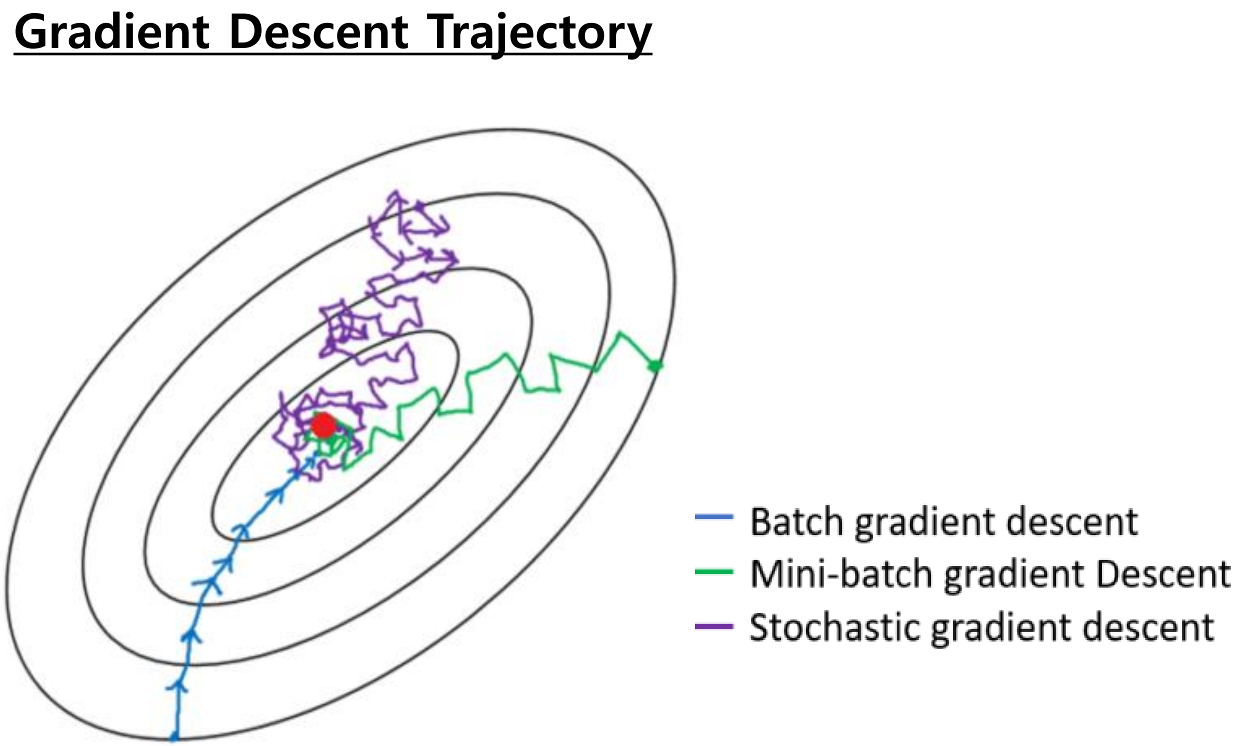
example

…

example

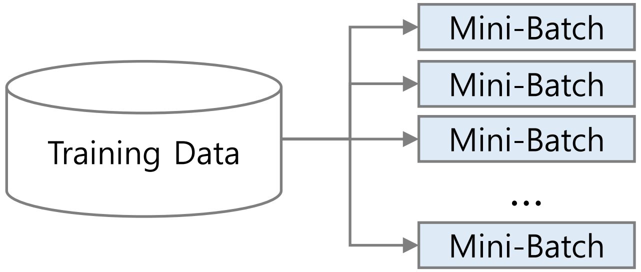
Too Noisy!

훈련 단위



[그림] https://towardsdatascience.com/gradient-descent-algorithm-and-its-variants-10f652806a3

훈련 단위



모집단의 표준 편차 : σ

표본 평균의 표준 편차 : $\dfrac{\sigma}{\sqrt{n}}$ n : 샘플 수

Mini-Bach 크기	100개	10,000개
계산 시간		100배 증가
표준 편차		10배 감소

대부분의 최적화 알고리즘은 천천히 정확하게 Gradient를 계산하는 것보다
빠르게 Gradient 근사치를 계산할 때 수렴 속도가 빠름

Thank you!