

## Section 5. 경사 하강 (Gradient Descent)

# 목차

- 섹션 0. 강의 소개
- 섹션 1. PyTorch 환경 설정
- 섹션 2. 딥러닝이란?
- 섹션 3. 손실 함수 (Loss Function)
- 섹션 4. 손실 함수에 대한 심화 이론 (Advanced Topics on Loss Function)
- 섹션 5. 경사 하강 (Gradient Descent)
- 섹션 6. 경사 하강에 대한 심화 이론 (Advanced Topics on Gradient Descent)

“Section 2, 3, 4” 에서 배웠던  
내용

# Recap

## Section 2. What is Deep Learning?

- Neural Network가 학습되는 과정 = weight값이 최적화되는 과정

**Gradient Descent** (경사 하강)을 통한 **Loss function** (손실 함수) 값을 최소화하도록

**weight** 값을 최적화하여 점진적으로 모델의 예측 정확도를 높인다.

# Recap

## Section 3. 손실 함수

Loss Function  $L$  (손실 함수) 의 정의

Neural Network 모델이 예측한 값  $\hat{Y}$  과 원래 정답  $Y$  간의 차이 (오차)의 지표

손실 함수의 값이 최소화하도록 모델의 **weight**을 최적화하면 모델의 정확도가 높아진다!

## 학습 목표

# Objective

## 학습 목표

- Gradient Descent (경사하강방법)의 기본 개념 이해
- Gradient (경사)의 의미 이해
- Learning rate (학습률)의 효과와 역할
- Mini-batch Gradient Descent

## 5-1. Gradient Descent의 기본 개념



# Gradient Descent

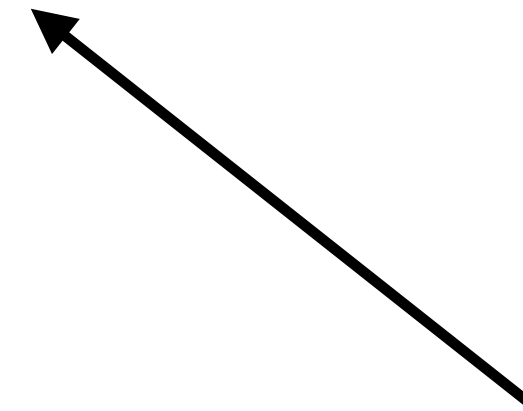
경사하강방법

경사하강방법

# Gradient Descent

경사하강방법

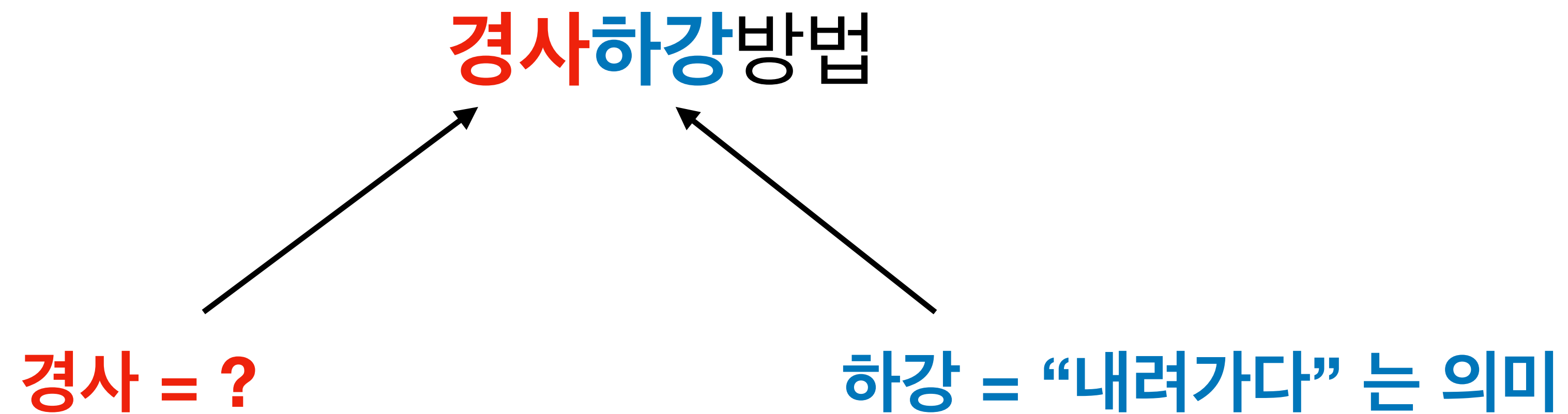
경사하강방법



하강 = “내려가다” 는 의미

# Gradient Descent

## 경사하강방법



경사하강방법에서 “경사”는 어떤 걸 의미할까?

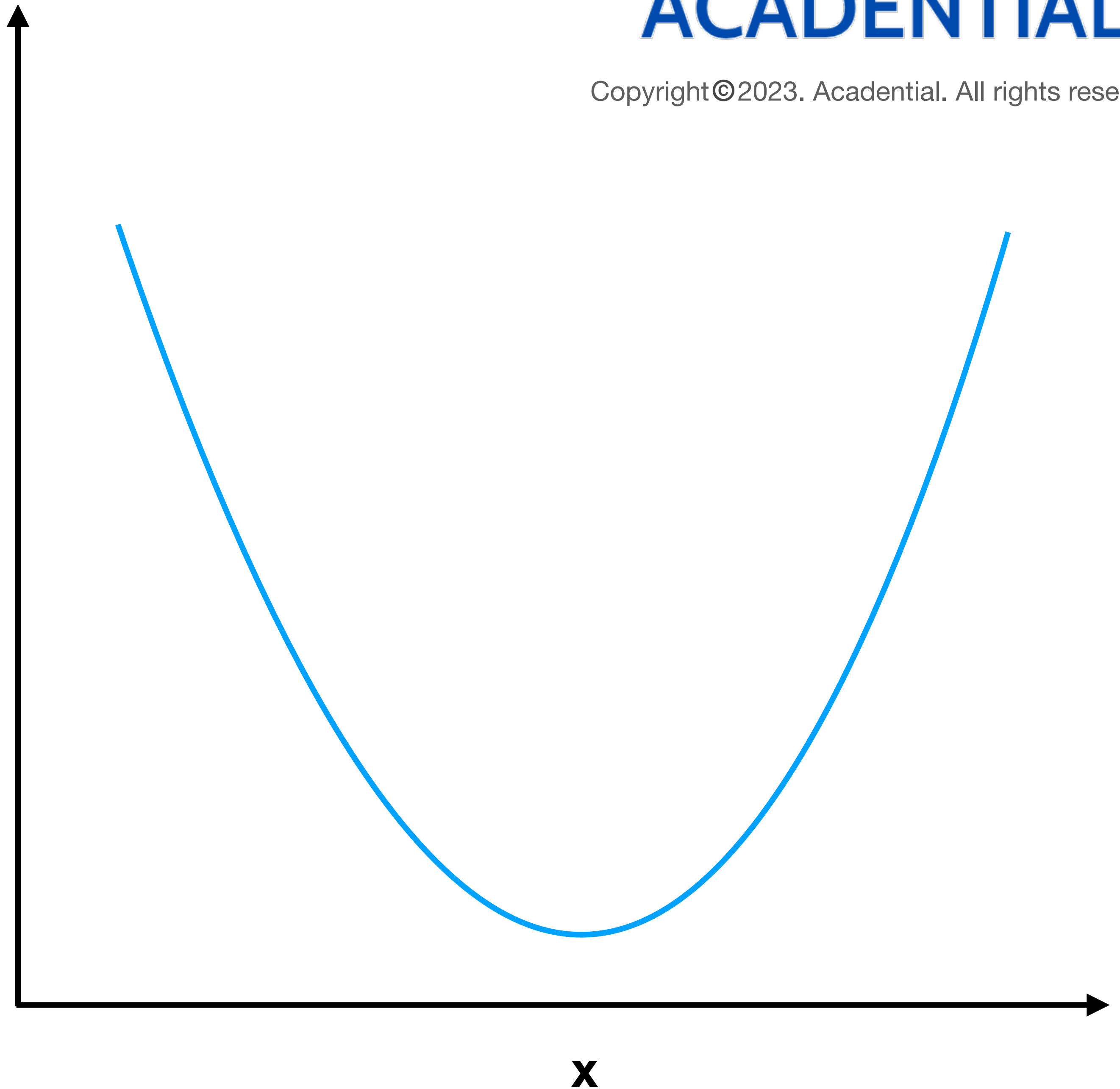
# Gradient Descent

## 경사하강방법

다음과 같은 함수  $y$ 을 가정:

$$y = y(x)$$

**y**



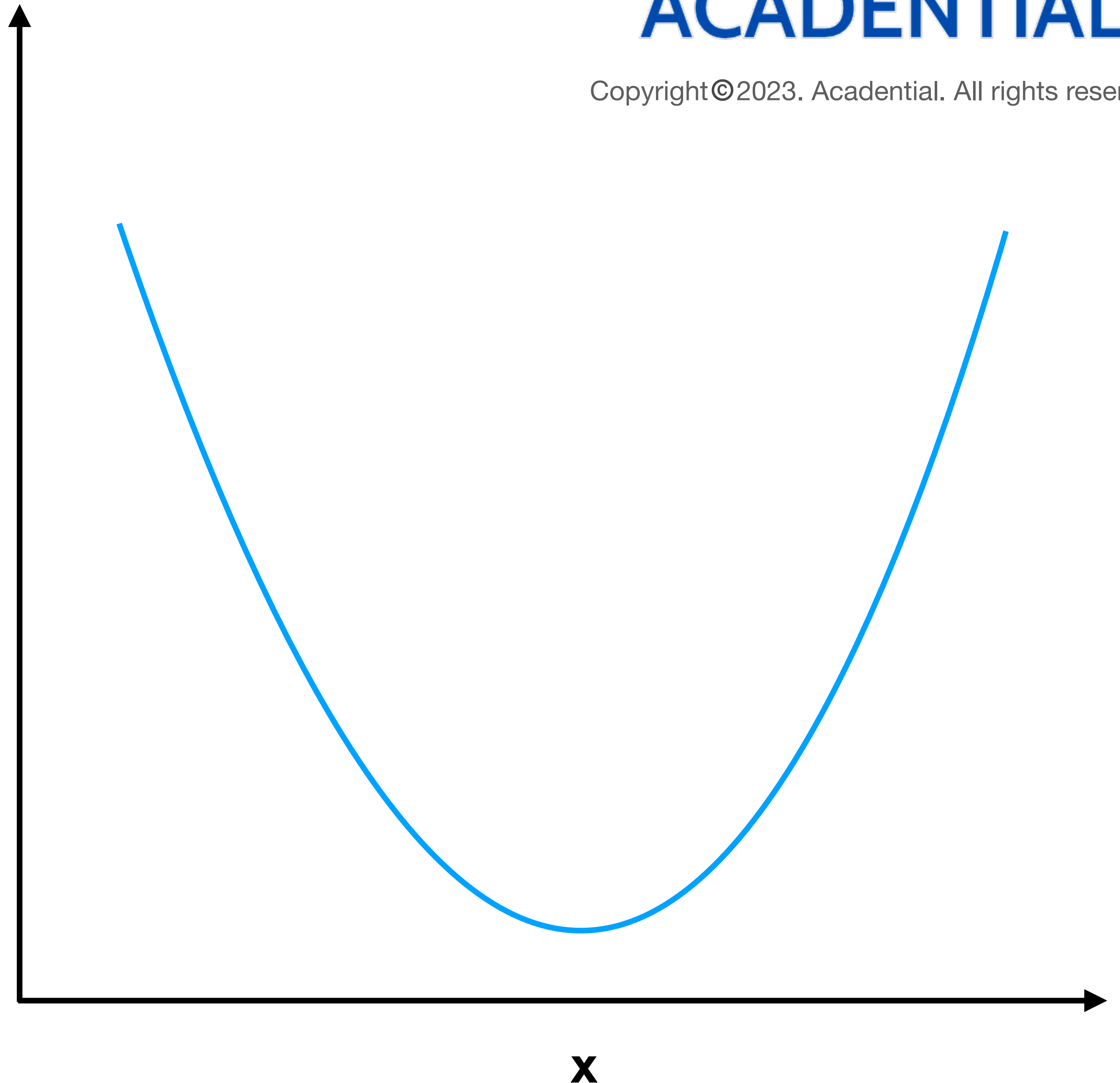
# Gradient Descent

## 경사하강방법

“경사 (Gradient)”는 기울기이다!

$$g = \frac{\Delta y}{\Delta x} = \frac{y_2 - y_1}{x_2 - x_1}$$

**y**



# Gradient Descent

## 경사하강방법

“경사 (Gradient)”는 기울기이다!

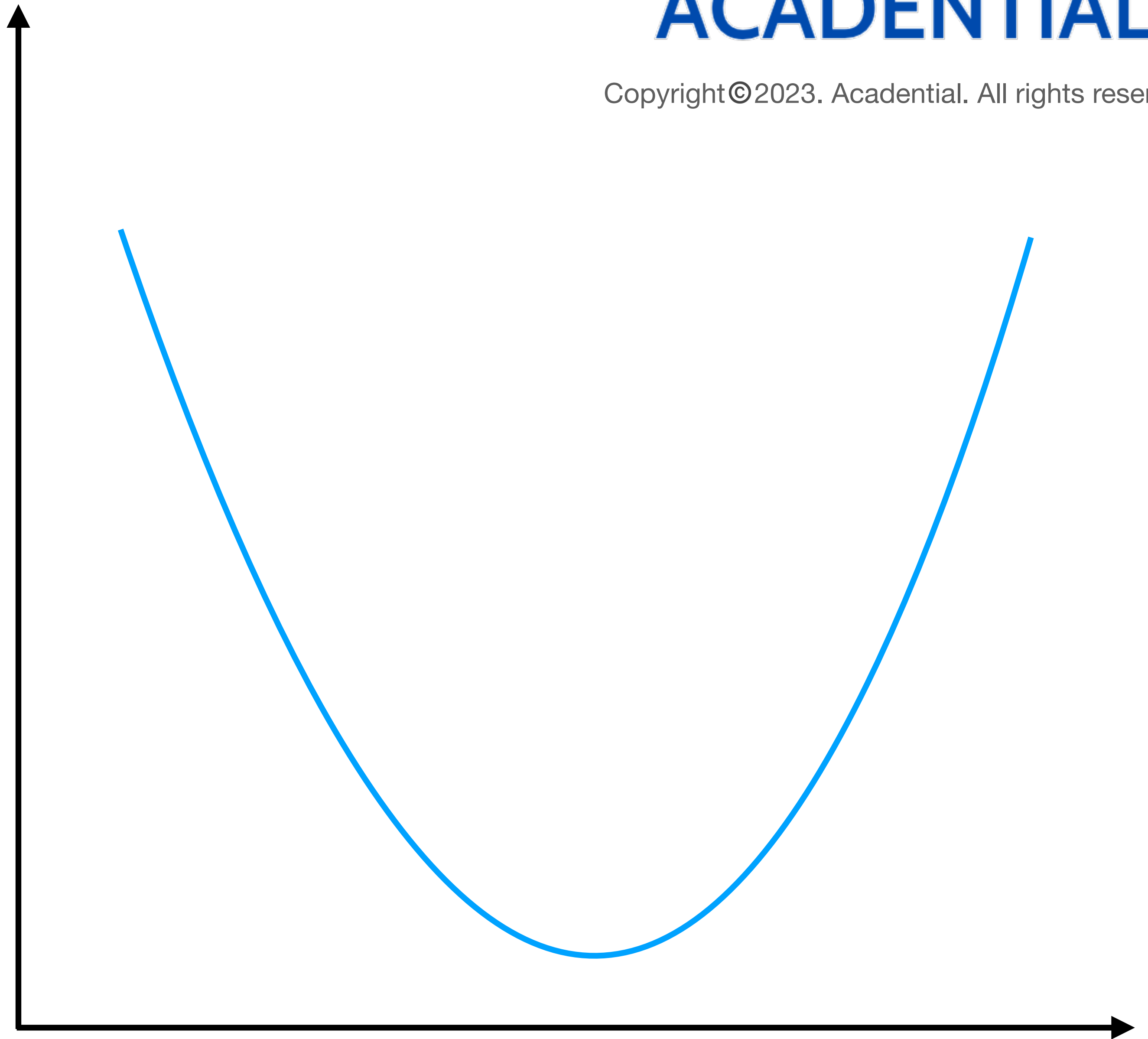
$$g = \frac{\Delta y}{\Delta x} = \frac{y_2 - y_1}{x_2 - x_1}$$

x의 변화량

y의 변화량

y

x

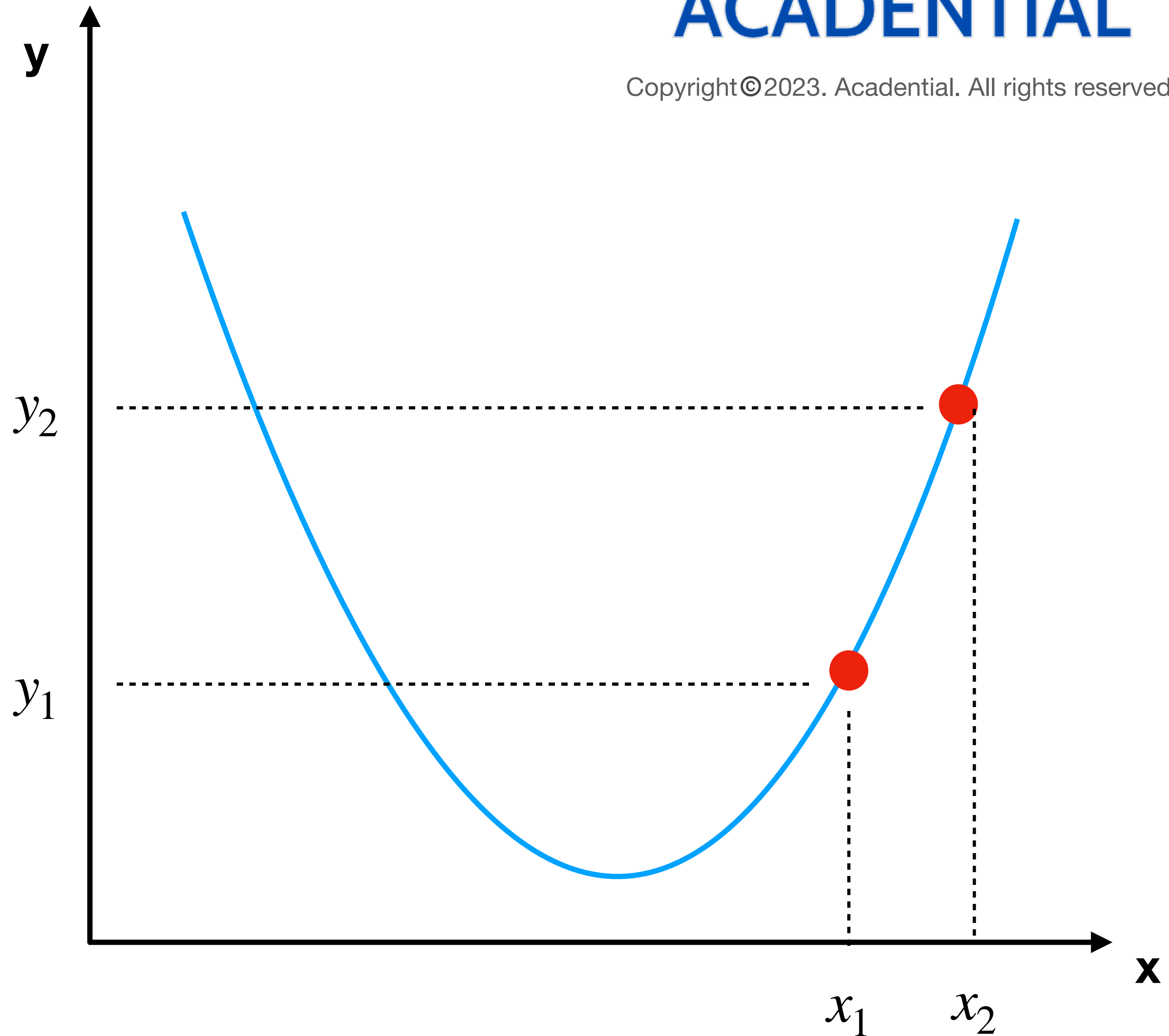


# Gradient Descent

## 경사 Gradient의 의미

경사 = 기울기  $g$

$$g = \frac{\Delta y}{\Delta x} = \frac{y_2 - y_1}{x_2 - x_1}$$

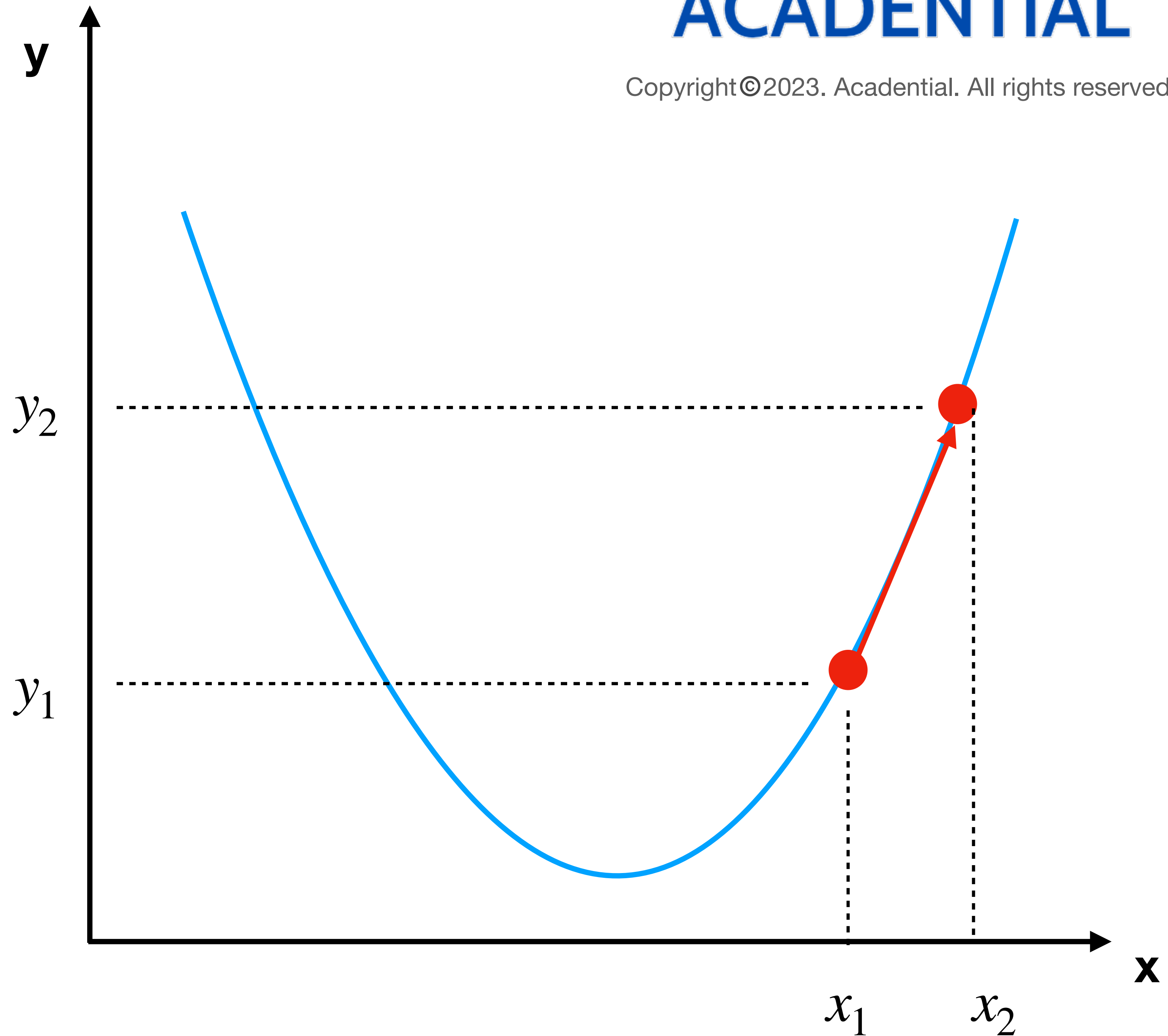


# Gradient Descent

## 경사 Gradient의 의미

경사 = 기울기  $g$

$$g = \frac{\Delta y}{\Delta x} = \frac{y_2 - y_1}{x_2 - x_1}$$



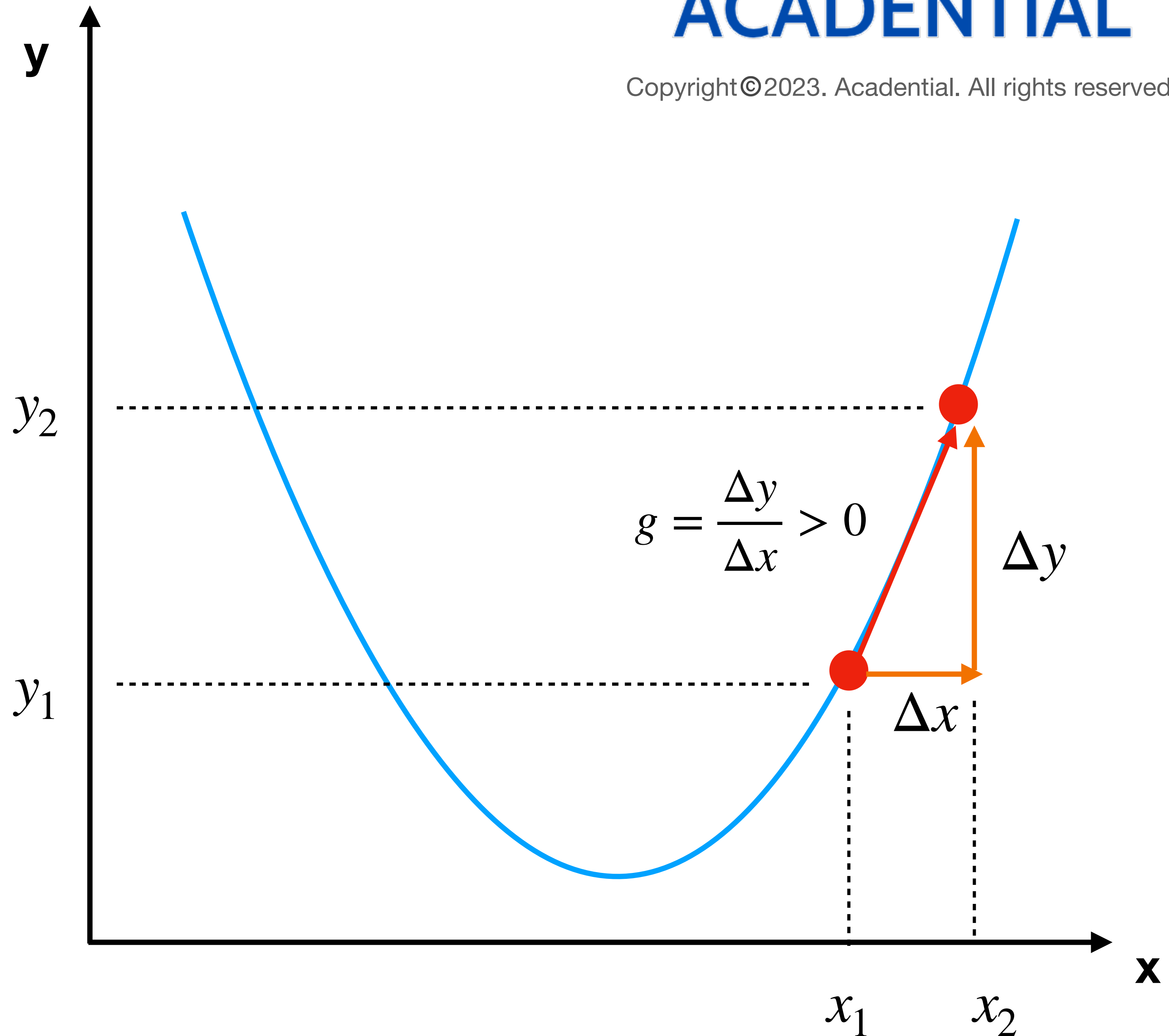


# Gradient Descent

## 경사 Gradient의 의미

경사 = 기울기  $g$

$$g = \frac{\Delta y}{\Delta x} = \frac{y_2 - y_1}{x_2 - x_1}$$



# Gradient Descent

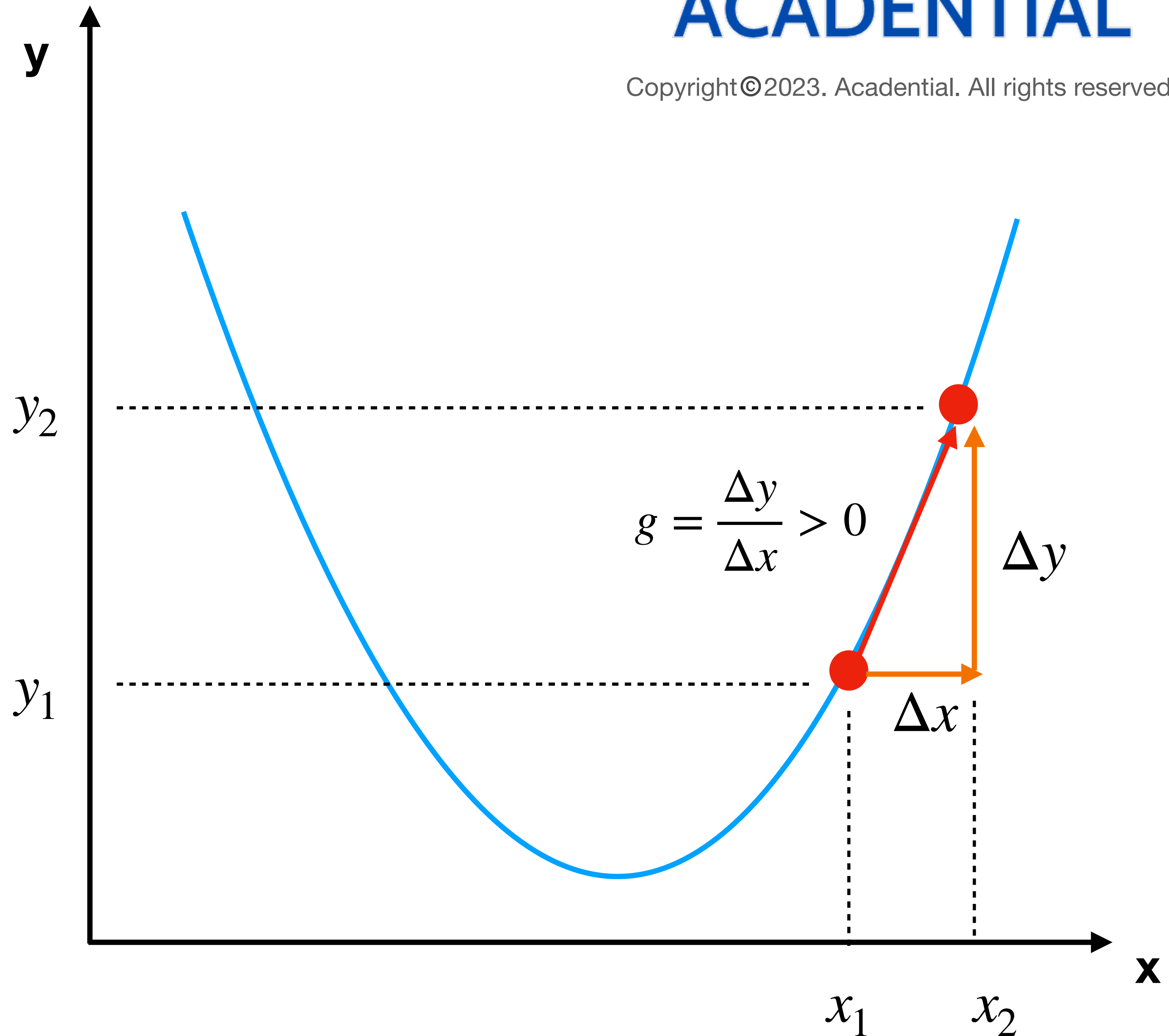
## 경사 Gradient의 의미

경사 = 기울기  $g$

$$g = \frac{\Delta y}{\Delta x} = \frac{y_2 - y_1}{x_2 - x_1}$$

$g$ 가 양수 ( $g > 0$ )이면:

- $x$ 가 증가할 때  $y$ 도 증가한다.
- $x$ 가 감소할 때  $y$ 도 감소한다.

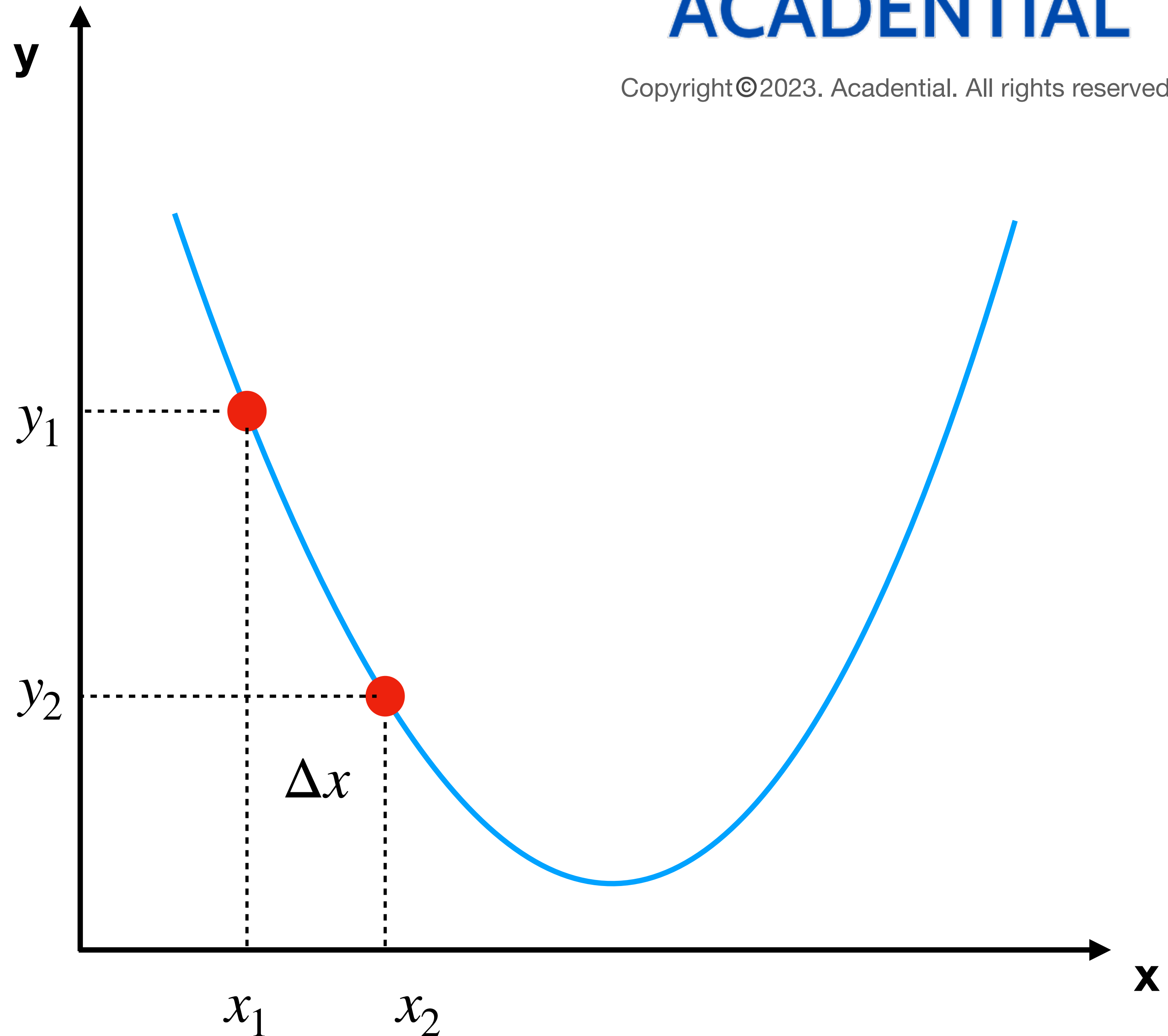


# Gradient Descent

## 경사 Gradient의 의미

경사 = 기울기  $g$

$$g = \frac{\Delta y}{\Delta x} = \frac{y_2 - y_1}{x_2 - x_1}$$



# Gradient Descent

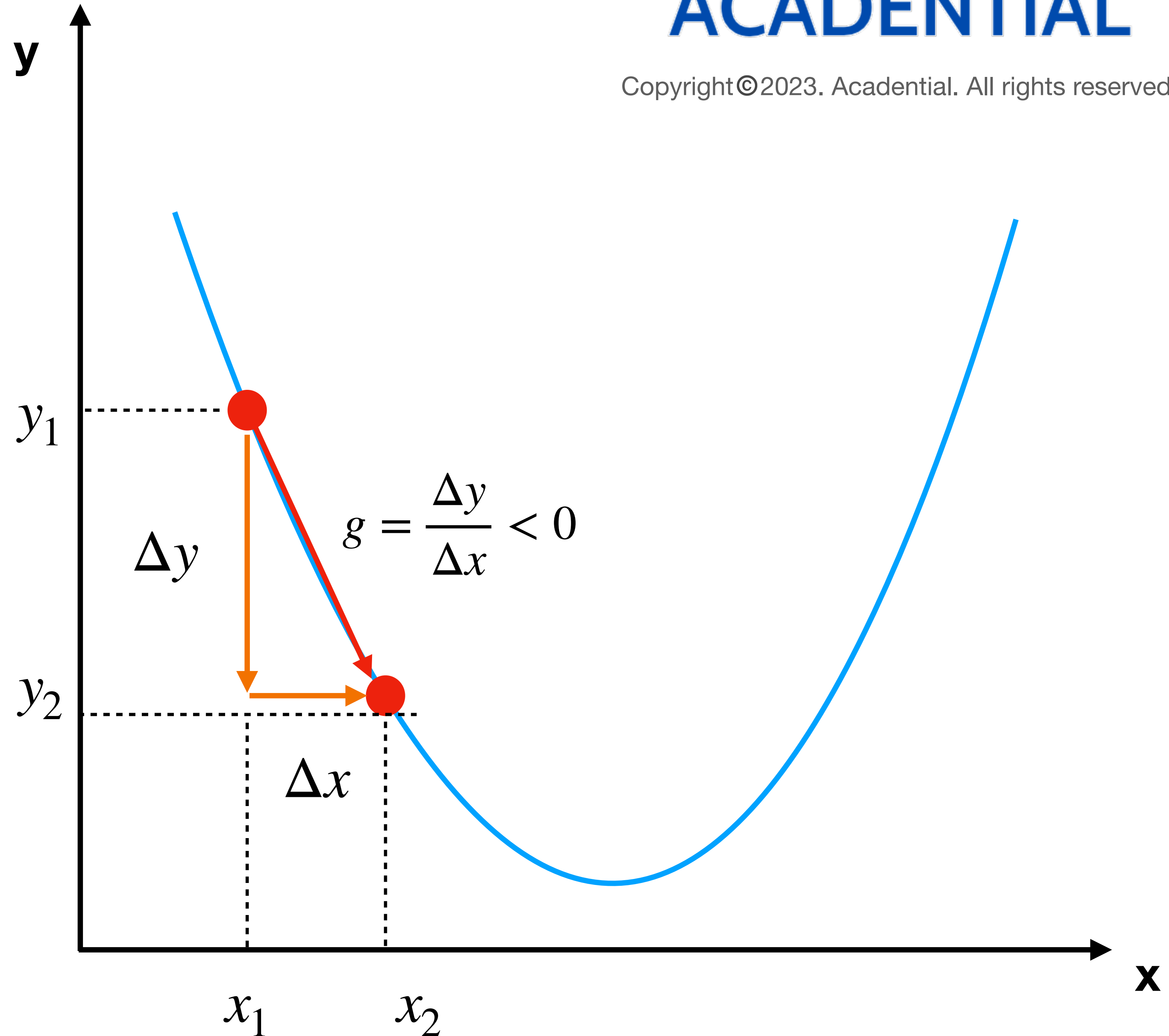
## 경사 Gradient의 의미

경사 = 기울기  $g$

$$g = \frac{\Delta y}{\Delta x} = \frac{y_2 - y_1}{x_2 - x_1}$$

$g$ 가 음수 ( $g < 0$ )이면

- $x$ 가 증가할 때  $y$ 는 감소한다.
- $x$ 가 감소할 때  $y$ 는 증가한다.



# Gradient Descent

## 경사 Gradient의 의미

$g$ 가 양수 ( $g > 0$ )이면,

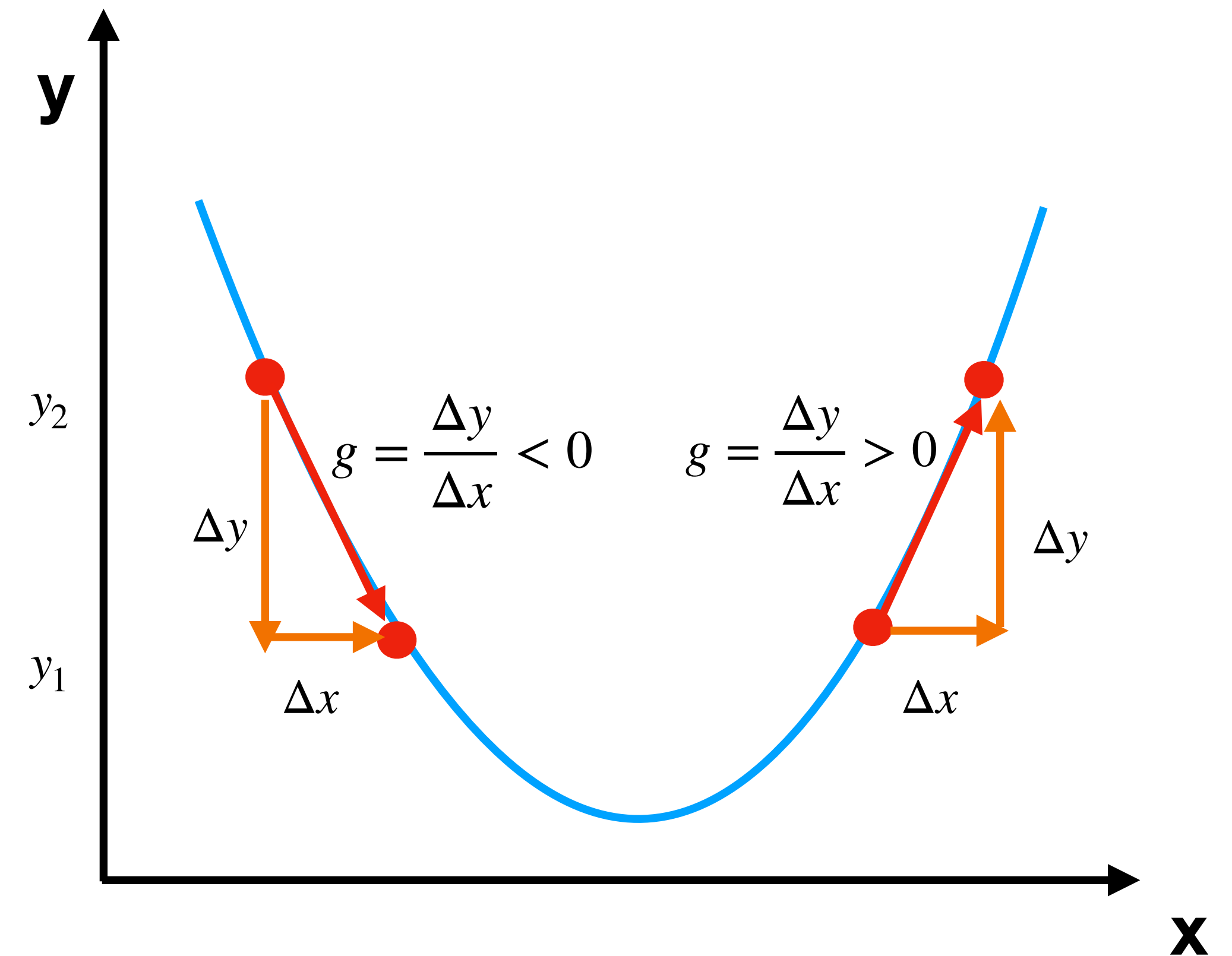
$x$ 가 증가할 때  $y$ 도 증가한다.

$x$ 가 감소할 때  $y$ 도 감소한다.

$g$ 가 음수 ( $g < 0$ )이면,

$x$ 가 증가할 때  $y$ 는 감소한다.

$x$ 가 감소할 때  $y$ 는 증가한다.



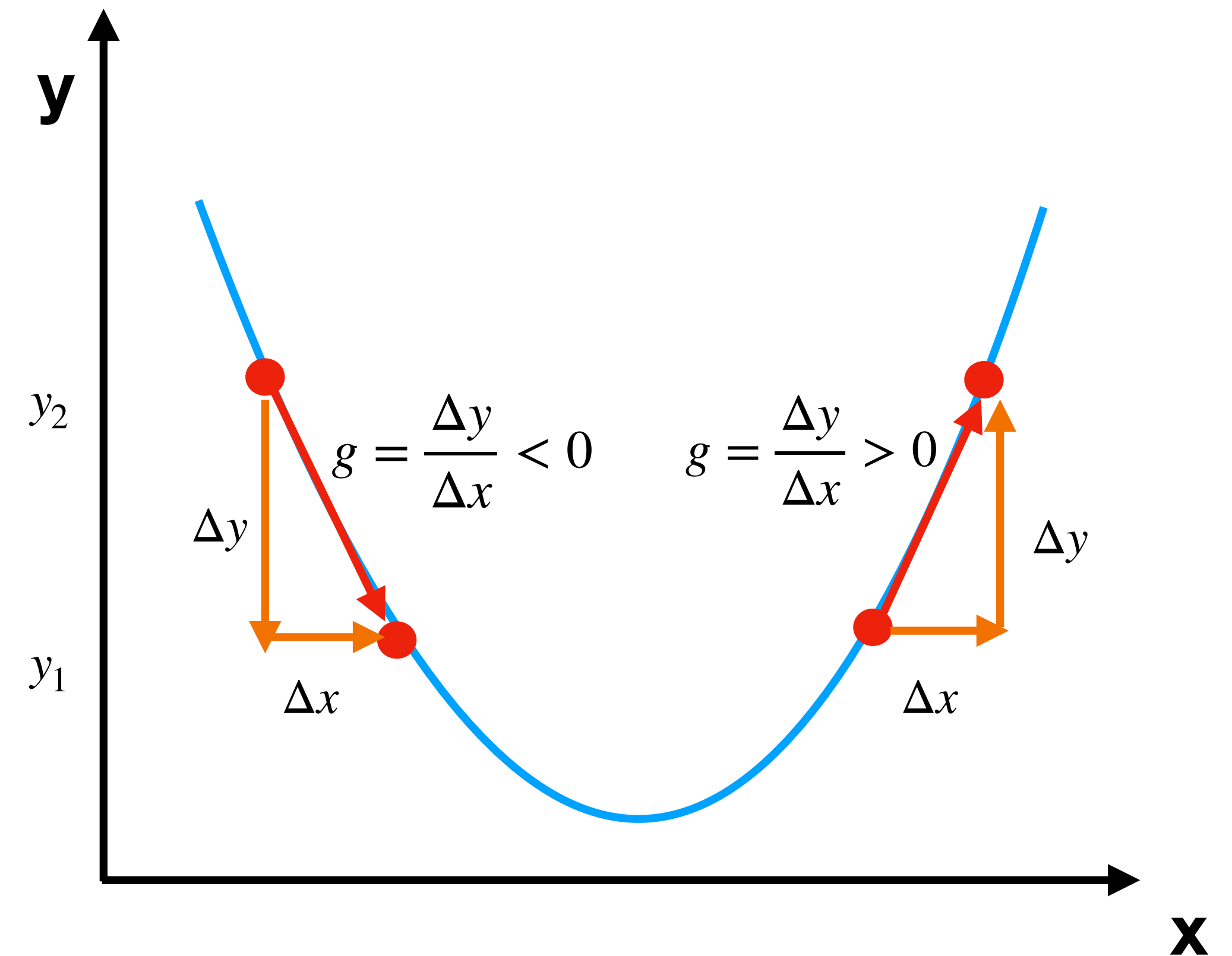
# Gradient Descent

## 경사 Gradient의 의미

앞서 살펴본 내용은

$$y \text{에 대한 미분인 } \lim_{\Delta x \rightarrow 0} g = \frac{dy}{dx}$$

이여도 유효하다!



# Gradient Descent

## Gradient와 Loss의 최소화

### 경사하강방법의 목적:

- Neural Network의 **Weight**을 어떻게 바꿔야 **Loss**을 줄일 수 있는가?
- 다음 함수 가정:

$$\text{Loss} = L(w)$$

- $w$ : Neural Network의 Weight (변수)

# Gradient Descent

## Gradient와 Loss의 최소화

$g$ 가 양수 ( $\frac{dL}{dw} > 0$ )이면,

$w$ 가 감소할 때  $L$ 도 감소한다.

$g$ 가 음수 ( $\frac{dL}{dw} < 0$ )이면,

$w$ 가 증가할 때  $L$ 는 감소한다.

$$\rightarrow \Delta w \cdot \frac{dL}{dw} < 0$$





# Gradient Descent

## Gradient와 Loss의 최소화

$$\Delta w \cdot \frac{dL}{dw} < 0$$



$$\Delta w = w_{i+1} - w_i$$

$$w_{i+1} = w_i + ?$$

# Gradient Descent

## Gradient와 Loss의 최소화

$$\Delta w \cdot \frac{dL}{dw} < 0$$

$$\rightarrow \Delta w \propto -\frac{dL}{dw}$$

$$\rightarrow \Delta w = -\lambda \frac{dL}{dw}$$

$$\Delta w = w_{i+1} - w_i$$

# Gradient Descent

## Gradient와 Loss의 최소화

$$\Delta w \cdot \frac{dL}{dw} < 0$$

$$\rightarrow \Delta w \propto -\frac{dL}{dw}$$

$$\rightarrow \Delta w = -\lambda \frac{dL}{dw}$$

$$\Delta w = w_{i+1} - w_i$$



$$\rightarrow w_{i+1} = w_i + \Delta w$$

# Gradient Descent

## Gradient와 Loss의 최소화

$$\Delta w \cdot \frac{dL}{dw} < 0$$

$$\rightarrow \Delta w \propto -\frac{dL}{dw}$$

$$\rightarrow \Delta w = -\lambda \frac{dL}{dw}$$

$$\Delta w = w_{i+1} - w_i$$

$$\rightarrow w_{i+1} = w_i + \Delta w$$

$$\rightarrow w_{i+1} = w_i - \lambda \frac{dL}{dw}$$



# Gradient Descent

## Gradient와 Loss의 최소화

$$w_{i+1} = w_i - \lambda \frac{dL}{dw}$$

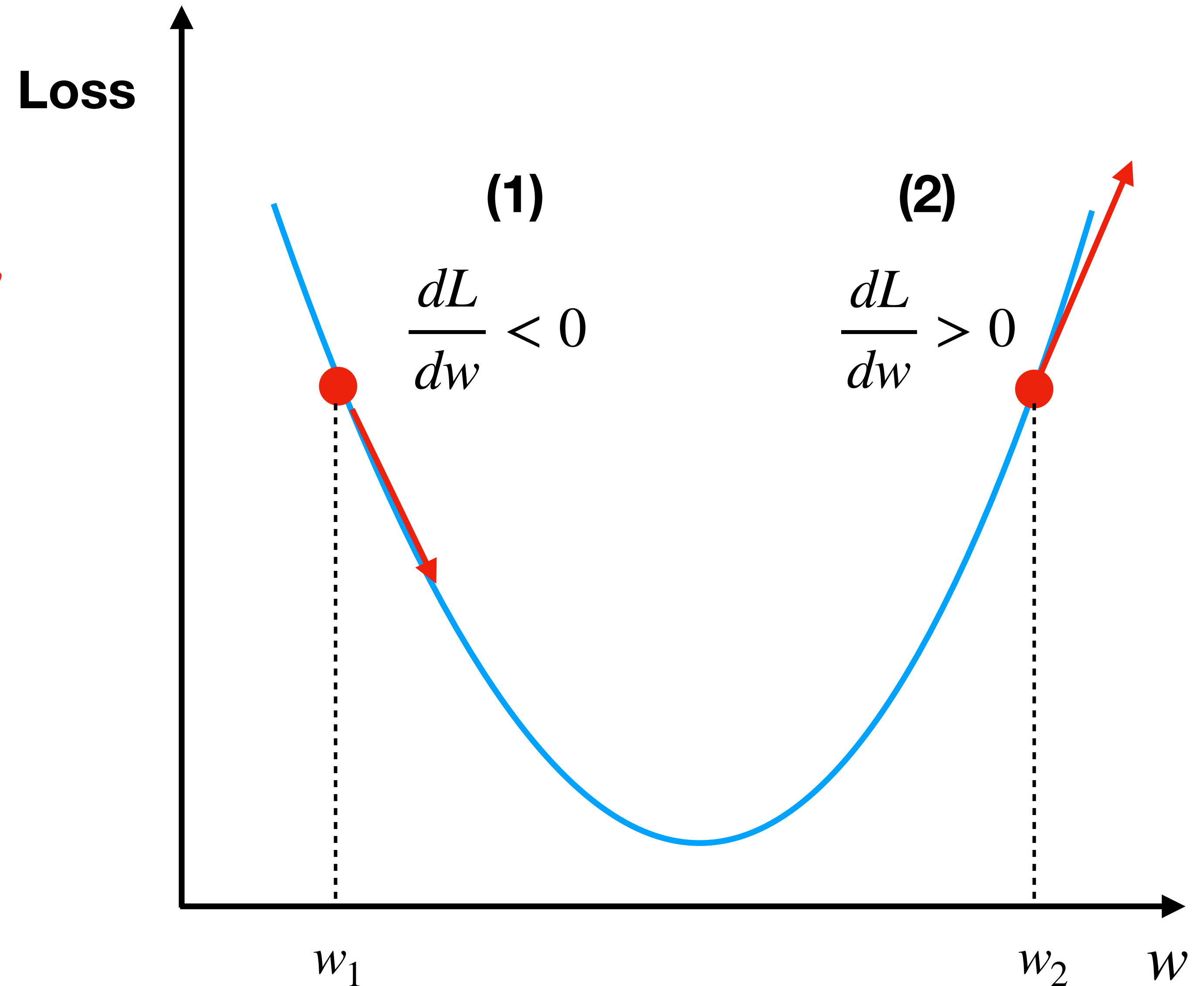
→ 경사  $\frac{dL}{dw}$ 의 반대 방향으로  $w$ 을 update하여 Loss를 줄이는 것.

→ 경사 하강 (Gradient Descent)

# Gradient Descent

경사 하강 예시

$$w_{i+1} = w_i - \lambda \frac{dL}{dw} = \Delta w$$



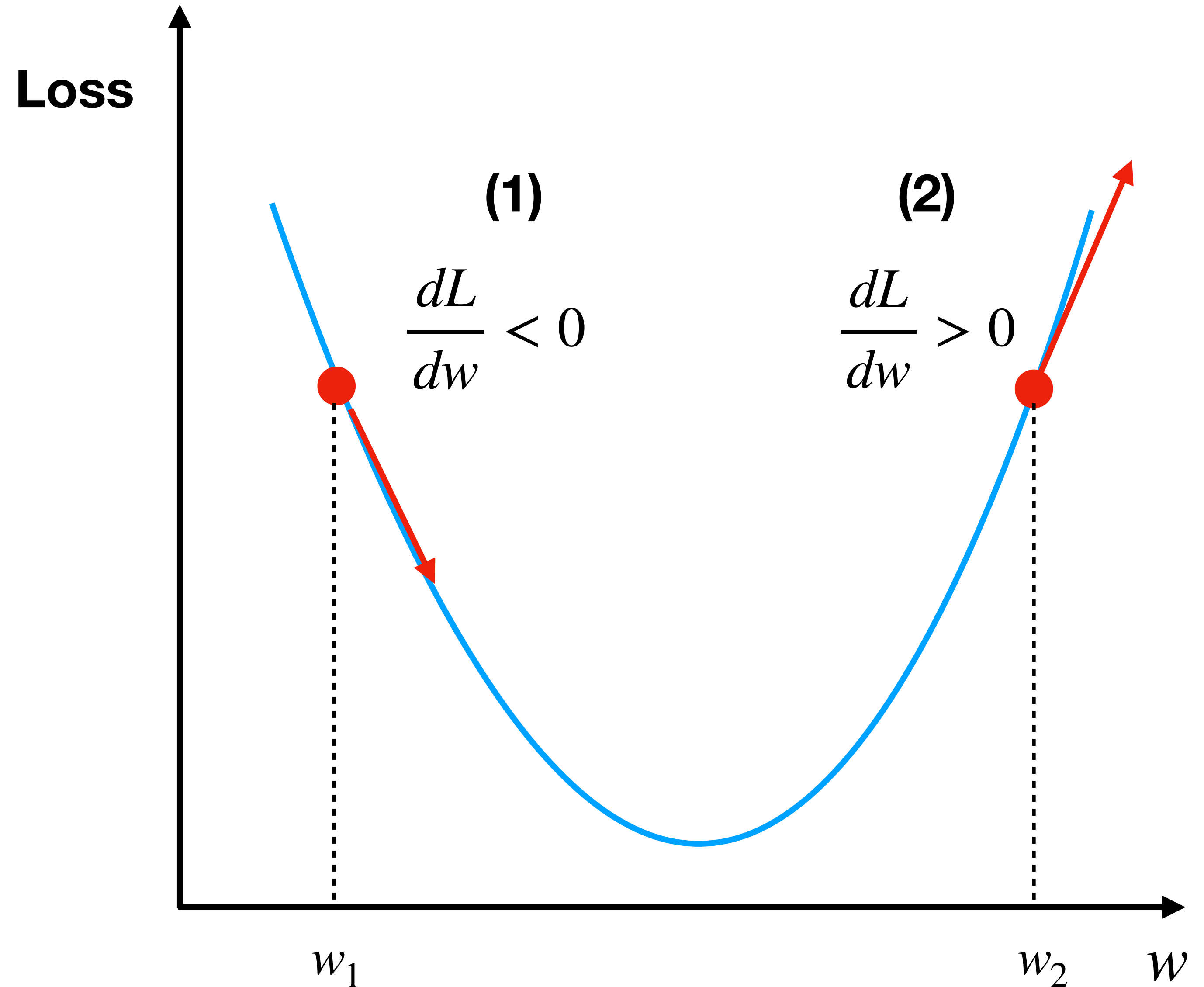
# Gradient Descent

## 경사 Gradient의 의미

$$w_{i+1} = w_i - \lambda \frac{dL}{dw} = \Delta w$$

(1)의 경우

$$\frac{dL}{dw} < 0 \rightarrow \Delta w > 0$$



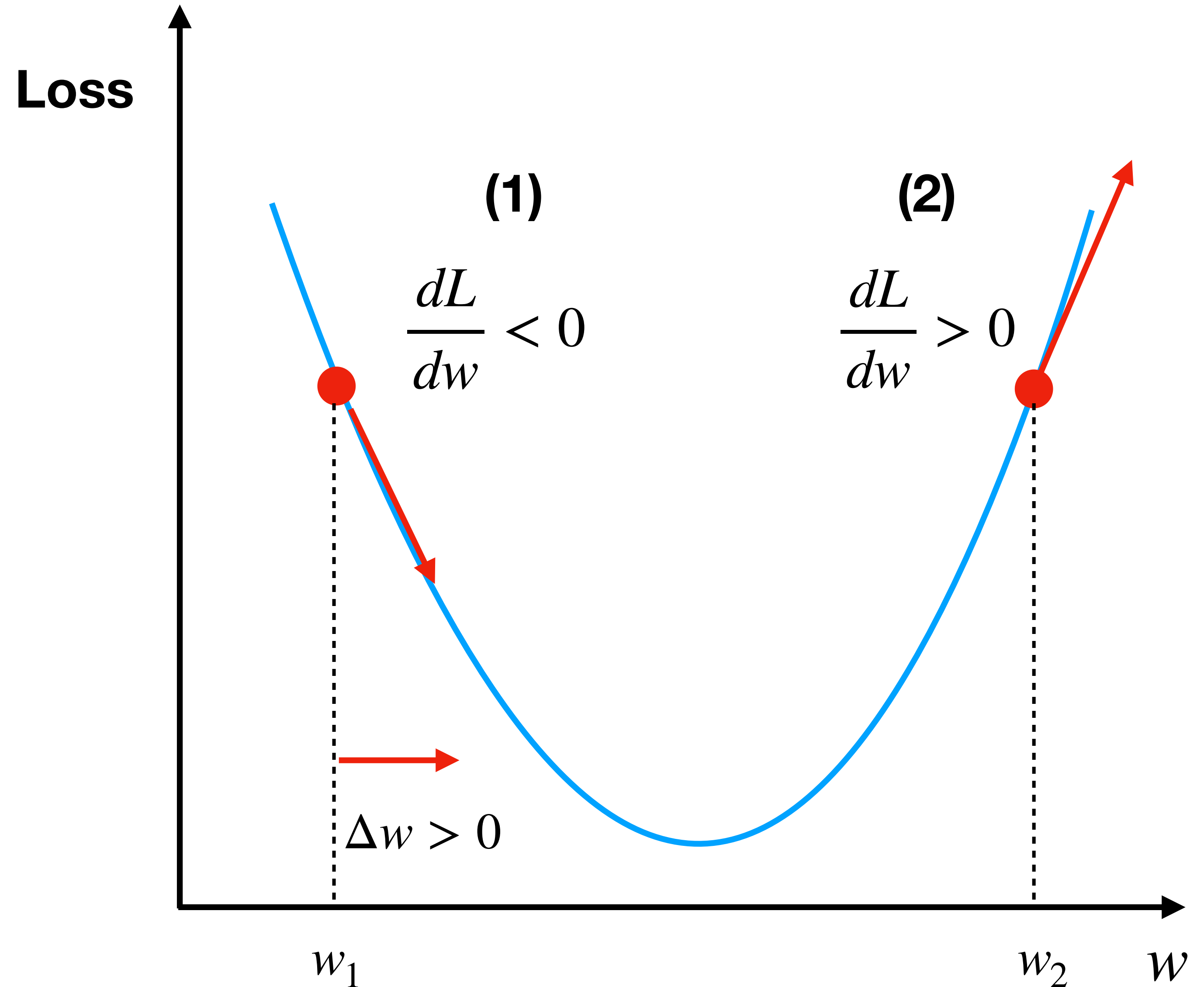
# Gradient Descent

경사 하강 예시

$$w_{i+1} = w_i - \lambda \frac{dL}{dw} = \Delta w$$

(1)의 경우

$$\frac{dL}{dw} < 0 \rightarrow \Delta w > 0$$





# Gradient Descent

경사 하강 예시

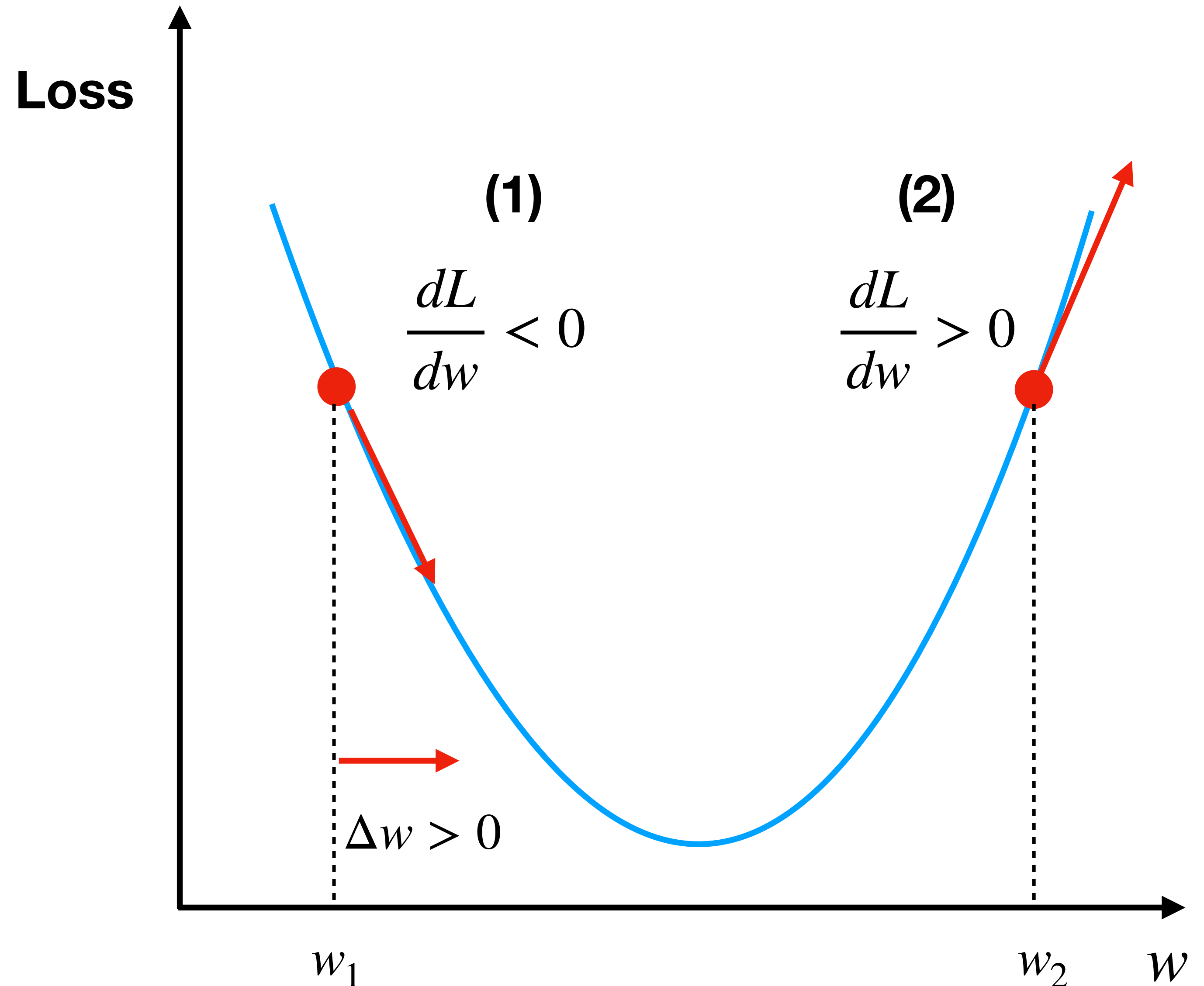
$$w_{i+1} = w_i - \lambda \frac{dL}{dw} = \Delta w$$

(1)의 경우

$$\frac{dL}{dw} < 0 \rightarrow \Delta w > 0$$

(2)의 경우

$$\frac{dL}{dw} > 0 \rightarrow \Delta w < 0$$



# Gradient Descent

경사 하강 예시

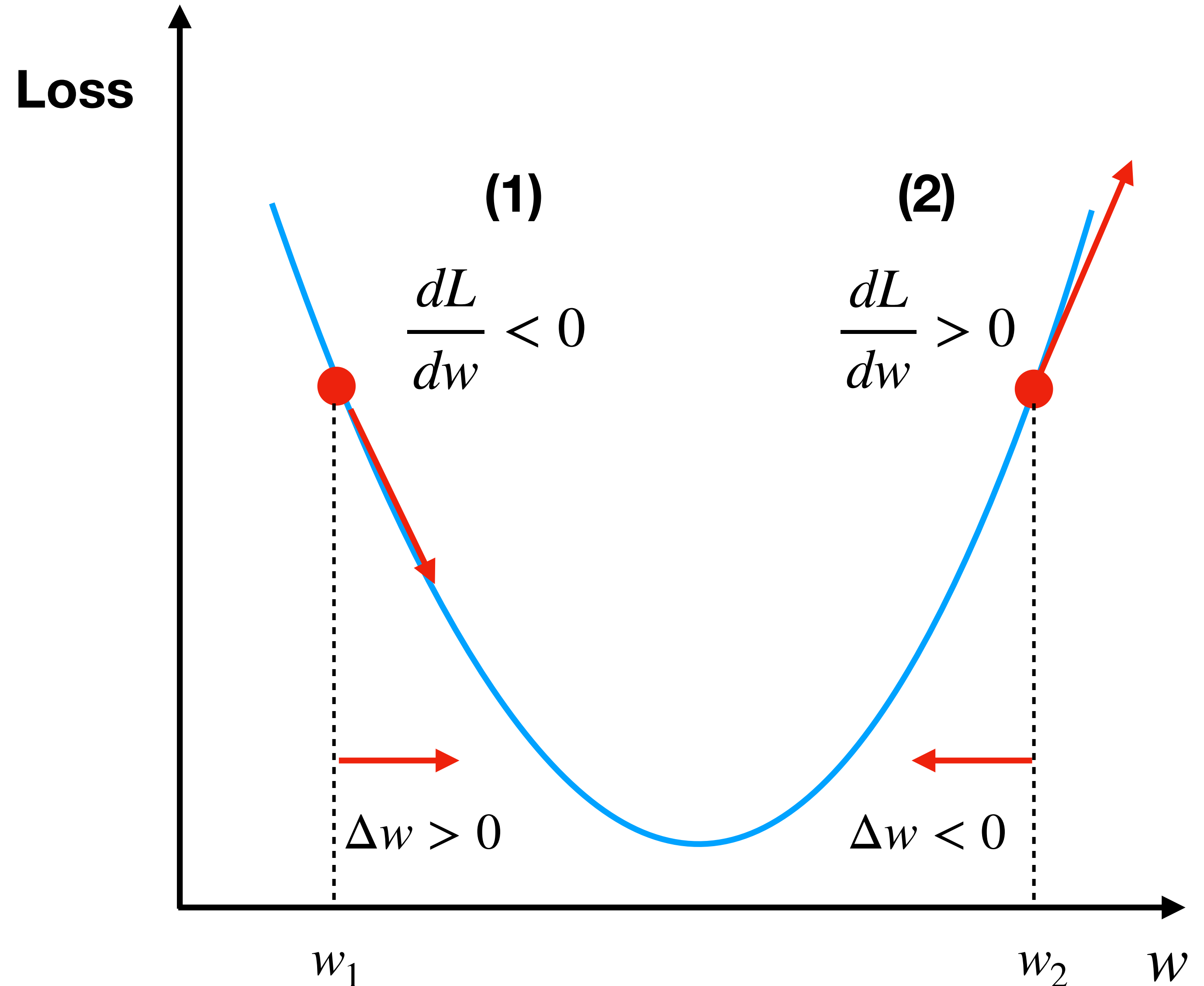
$$w_{i+1} = w_i - \lambda \frac{dL}{dw} = \Delta w$$

(1)의 경우

$$\frac{dL}{dw} < 0 \rightarrow \Delta w > 0$$

(2)의 경우

$$\frac{dL}{dw} > 0 \rightarrow \Delta w < 0$$



# Gradient Descent

## 경사 하강 요약

- Gradient Descent (경사 하강)

Loss 가 줄어드는 방향  $-\lambda \frac{dL}{dw}$  으로 Weight을 갱신하는 것.

- Loss가 줄어드는 방향

$\frac{dL}{dw}$ 의 반대 방향

- Gradient Descent의 Update rule:

$$w_{i+1} = w_i - \lambda \frac{dL}{dw}$$

# Gradient Descent

## 경사 하강 요약

- Gradient Descent (경사 하강)

Loss 가 줄어드는 방향  $-\lambda \frac{dL}{dw}$  으로 Weight을 갱신하는 것.

- Loss가 줄어드는 방향



$\frac{dL}{dw}$ 의 반대 방향

???

- Gradient Descent의 Update rule:

$$w_{i+1} = w_i - \lambda \frac{dL}{dw}$$

# Gradient Descent

## 경사하강법

$$w \rightarrow w - \lambda \cdot \frac{dL}{dw}$$

여기서  $\lambda$ 은 **Learning Rate (학습률)** 이다.



## 5-2. 학습률 (Learning Rate)의 역할과 효과

# Gradient Descent

## Learning Rate의 효과 및 역할

$$w \rightarrow w - \lambda \cdot \frac{dL}{dw}$$

learning rate  $\lambda$ 의 역할은 무엇인가?

# Learning Rate의 역할과 효과

$$w_{i+1} = w_i - \lambda \cdot \frac{dL}{dw} = \Delta w$$

- Learning Rate (학습률) = 학습을 얼마나 빠르게 진행할지를 조절해주는 값
- Weight의 변화량은 Learning Rate에 비례함. ( $\Delta w \propto \lambda$ )



# Learning Rate의 역할과 효과

$$\Delta w = -\lambda \cdot \frac{dL}{dw}$$

learning rate  $\lambda$ 의 역할은 무엇인가?

너무 작은 Learning Rate  $\lambda$

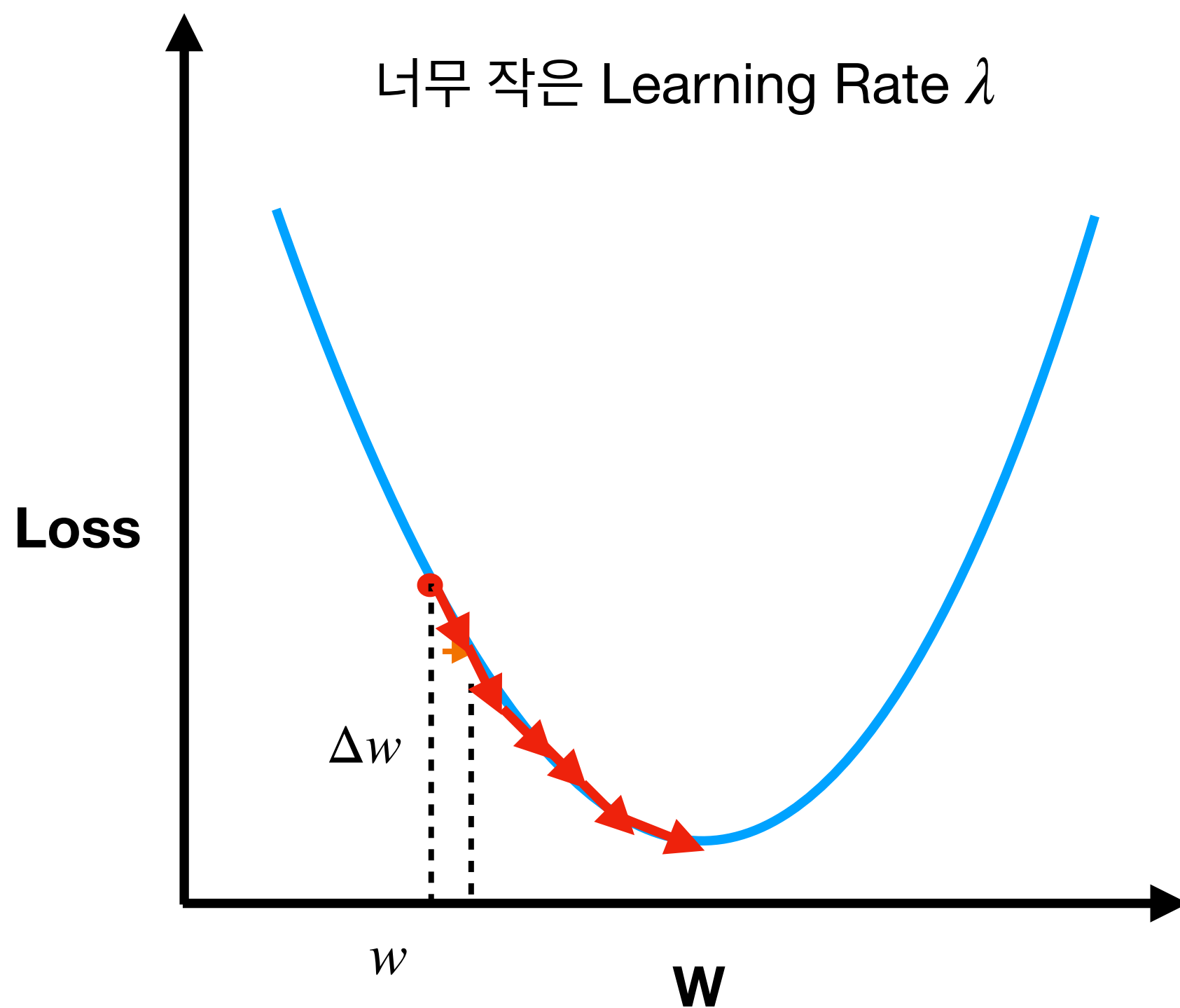
너무 큰 Learning Rate  $\lambda$

적당한 크기의 Learning Rate  $\lambda$

# Learning Rate의 역할과 효과

$$\Delta w = -\lambda \cdot \frac{dL}{dw}$$

learning rate  $\lambda$ 의 역할은 무엇인가?



너무 큰 Learning Rate  $\lambda$

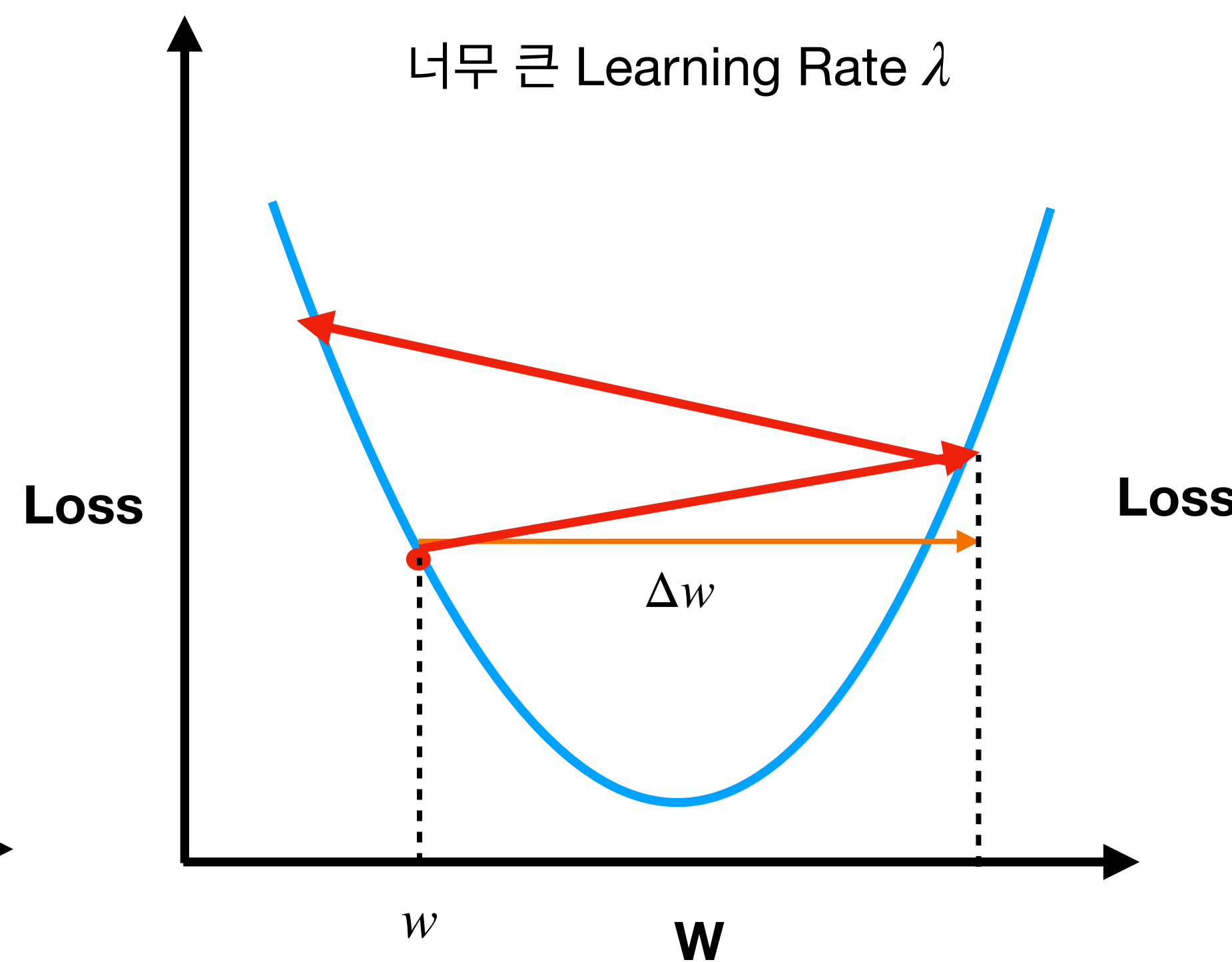
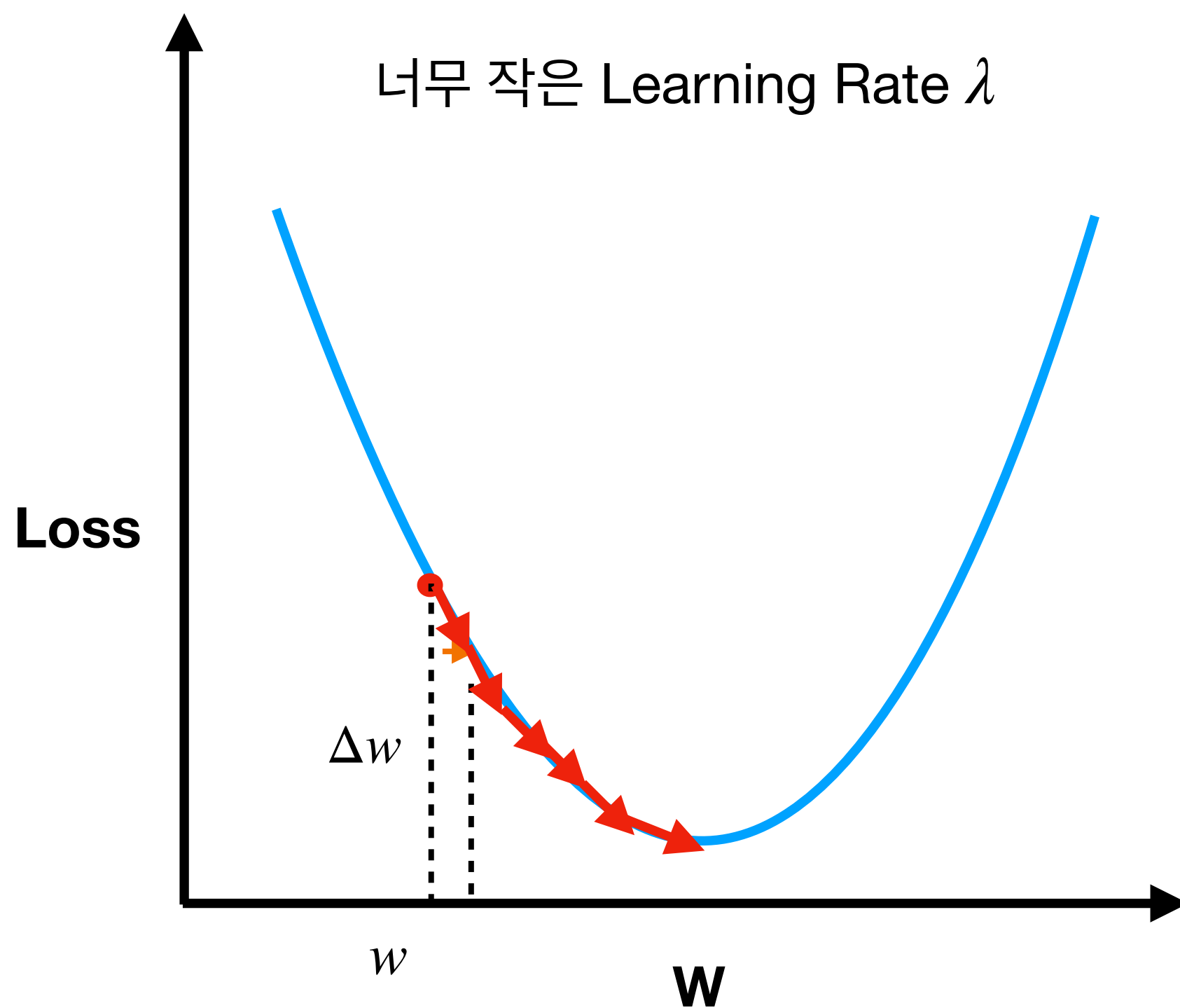
적당한 크기의 Learning Rate  $\lambda$

# Learning Rate의 역할과 효과

$$\Delta w = -\lambda \cdot \frac{dL}{dw}$$



learning rate  $\lambda$ 의 역할은 무엇인가?

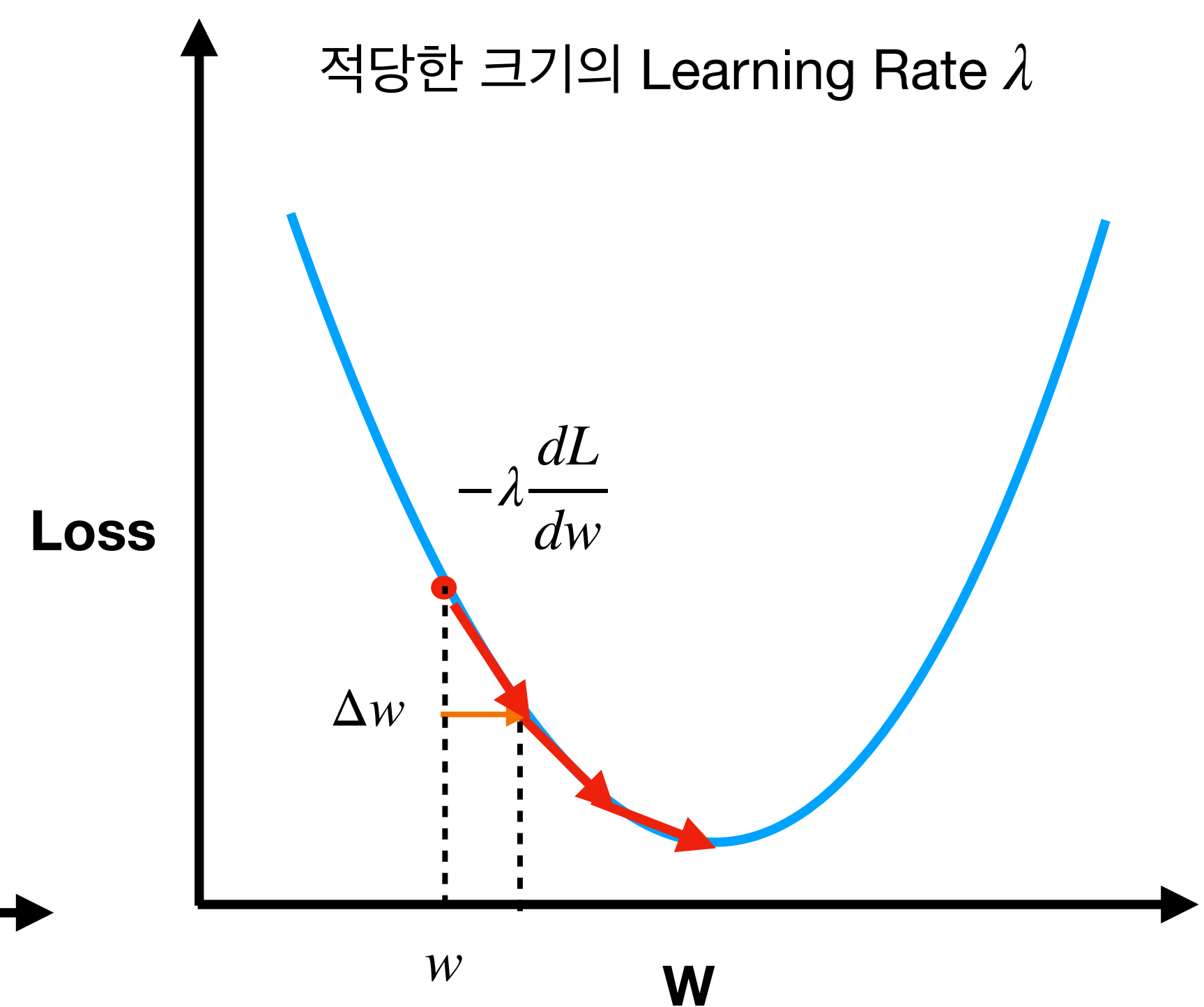
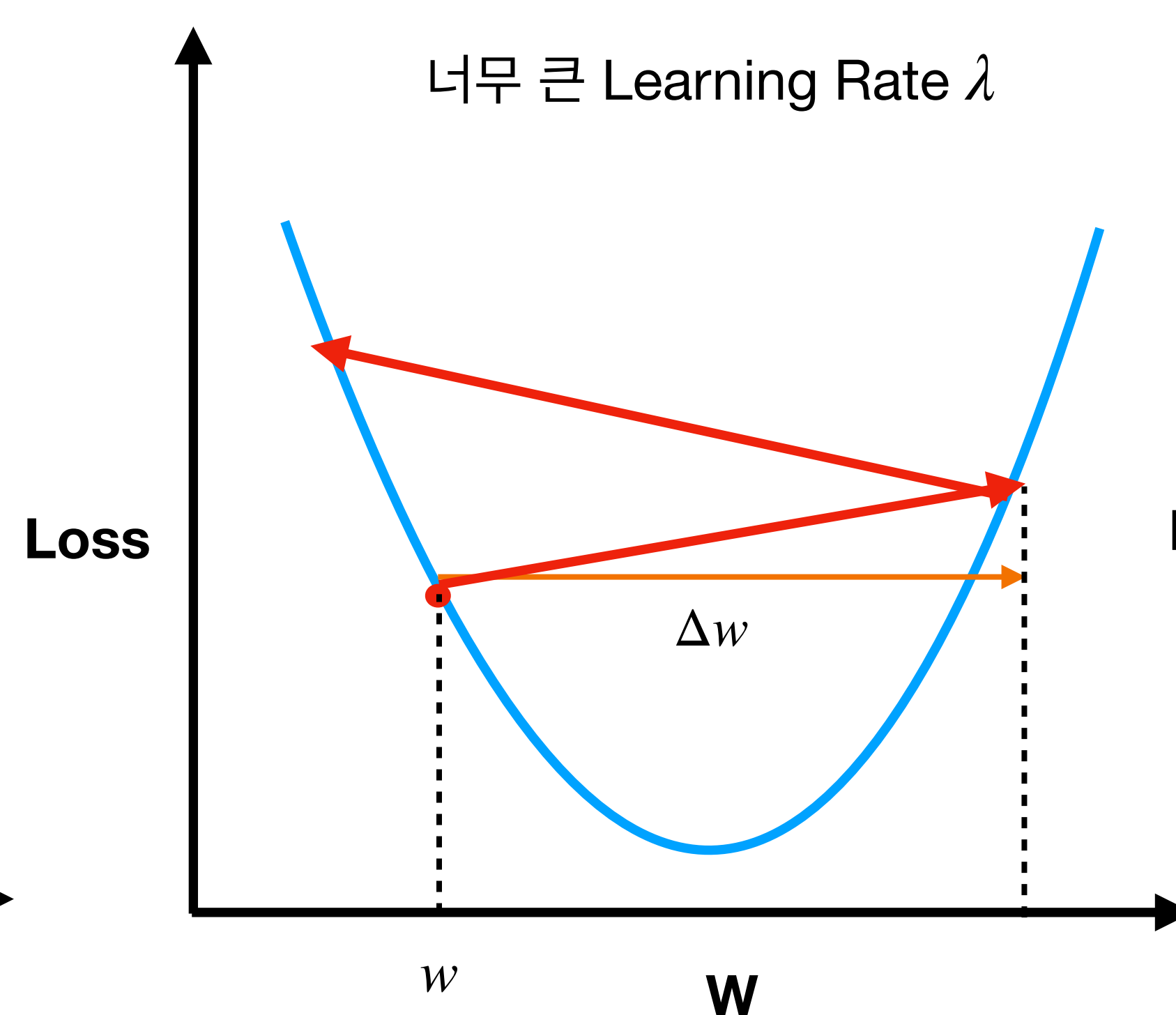
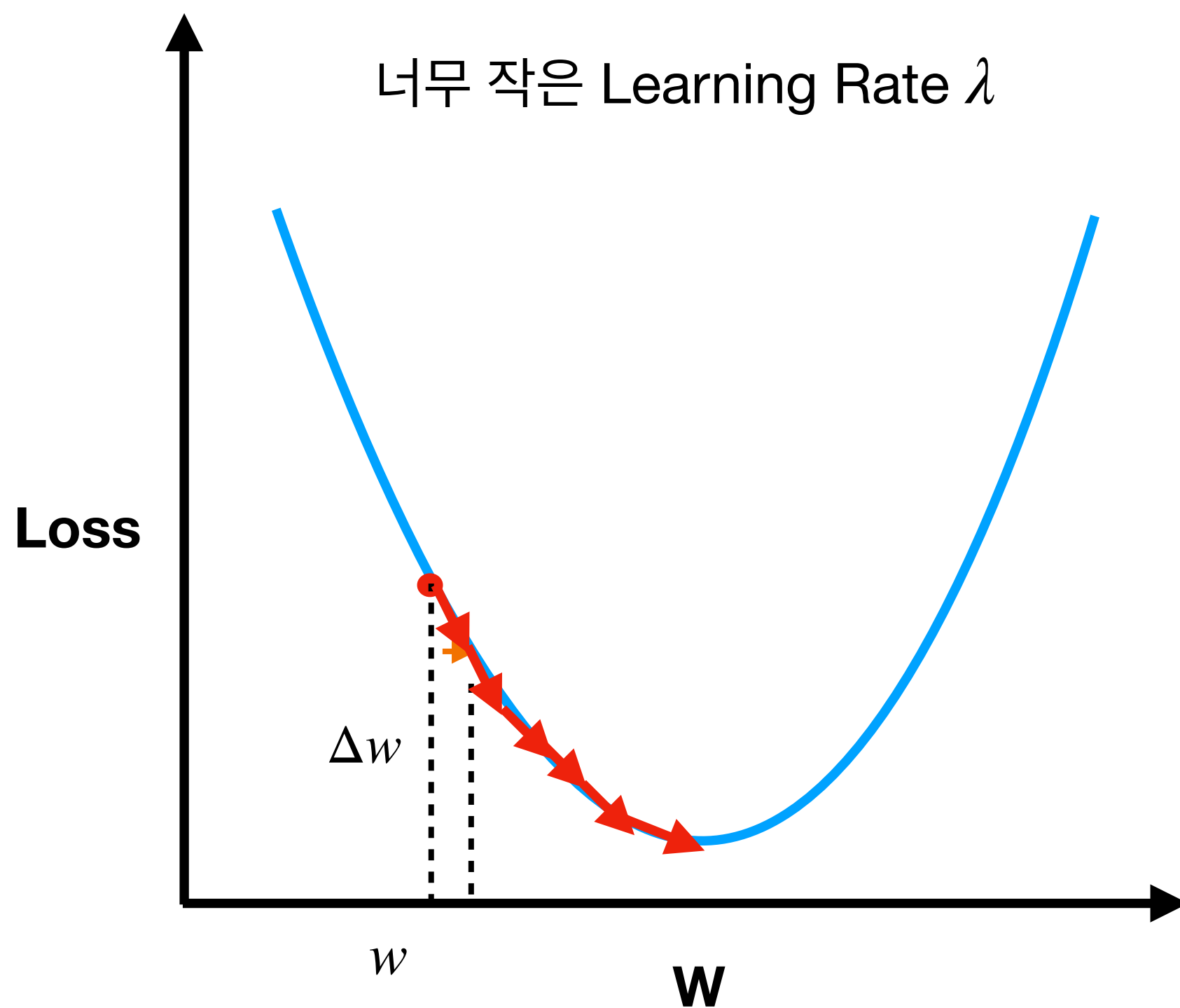


적당한 크기의 Learning Rate  $\lambda$

# Learning Rate의 역할과 효과

$$\Delta w = -\lambda \cdot \frac{dL}{dw}$$

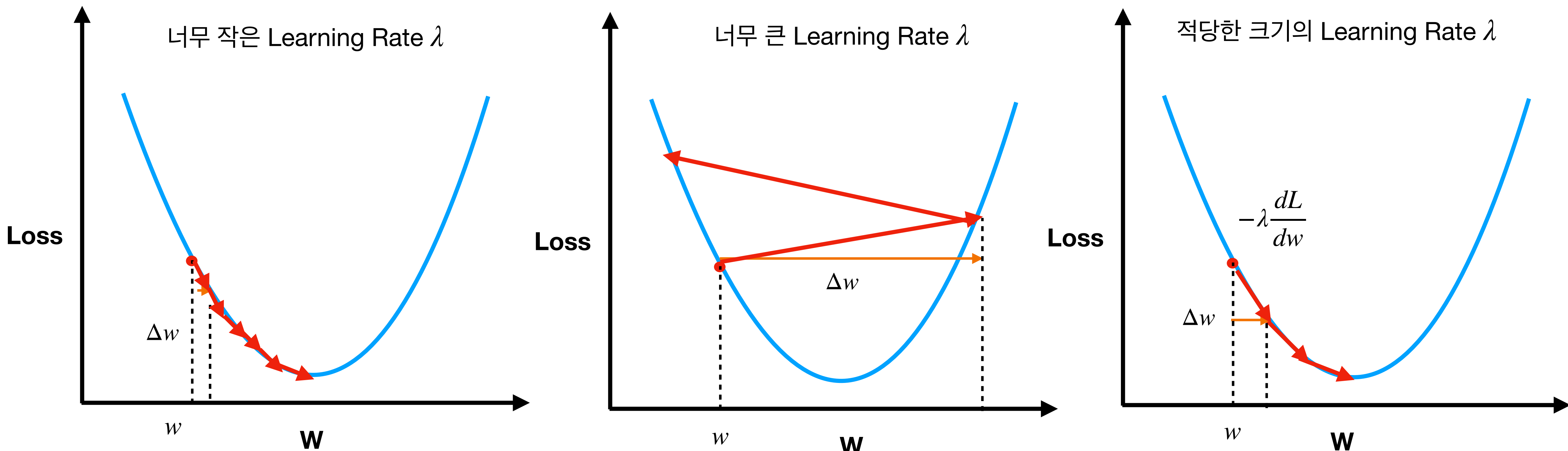
learning rate  $\lambda$ 의 역할은 무엇인가?



# Learning Rate의 역할과 효과

learning rate  $\lambda$ 의 역할은 무엇인가?

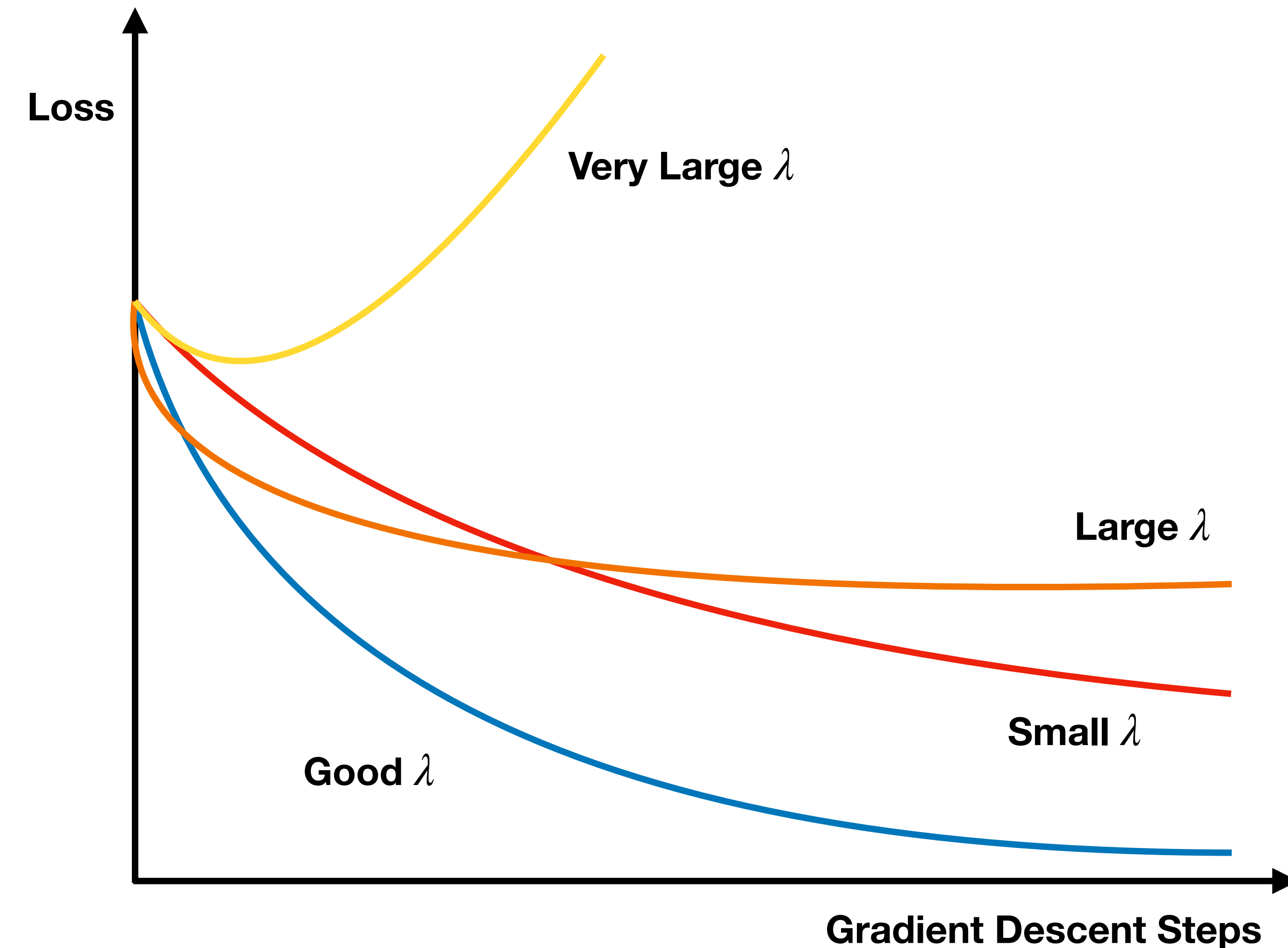
- Learning Rate가 너무 작으면 최저점 (Optimum)에 수렴 (converge) 하는데까지 많은 Gradient Descent step 필요
- Learning Rate가 너무 크면 Optimum에 수렴하지 못하고 발산 (diverge)하게 된다.



# Learning Rate의 역할과 효과

- Learning Rate가 너무 작으면 최저점 (Optimum)에 수렴 (converge) 하는데까지 많은 **Gradient Descent step** 필요 → 학습 시간이 오래 걸림.
- Learning Rate가 너무 크면 Optimum에 수렴하지 못하고 발산 (diverge) → 학습 성능이 오히려 저하됨.

# Learning Rate의 역할과 효과



- **Very large  $\lambda$** : Loss가 Gradient descent을 거듭할수록 오히려 증가함. (즉, Loss가 발산함.)
- **Large  $\lambda$** : Loss가 Gradient descent을 거듭할수록 초반에 빠르게 감소하지만 점차 정체됨.
- **Small  $\lambda$** : Loss가 천천히 감소된다.
- **Good  $\lambda$** : Small  $\lambda$ 보다 빠르게 Loss가 감소된다. Loss가 수렴했을때 Large  $\lambda$ 보다 더 낮은 Loss 지점에서 수렴하게 된다.



# Learning Rate의 역할과 효과

- Neural Network의 최종 학습 성능은 Learning Rate에 의해서도 영향을 받음.
- Learning Rate의 범위: 0.1~1e-05
- Neural Network의 구성과 종류에 따라서 적합한 Learning Rate 값은 다름.
  - 초반에 몇 번의 시행 학습 (preliminary runs)들로 적합한 범위의 값을 찾기.
  - 혹은 Hyperparameter Optimization (Grid search, Bayes Optimization 등등)로 적합한 Learning Rate 찾기.



## 5-3. Mini-batch Gradient Descent (미니 배치 경사 하강)

# Objective

## 학습 목표

- 다음 개념들에 대해서 이해하기:
  - Stochastic Gradient Descent
  - Full-batch Gradient Descent
  - Mini-batch Gradient Descent

# Mini-batch Gradient Descent

## Gradient Descent에 대한 Recap

- Neural Net 모델을 학습 데이터에 대해서 Gradient Descent 할시:

$$w_{i+1} = w_i - \lambda \cdot \frac{dL}{dw} \text{ (1-D case)}$$

$$\mathbf{w}_{i+1} = \mathbf{w}_i - \lambda \cdot \nabla_{\mathbf{w}} L \text{ (Multi-D case)}$$



Notation:

- 전체 학습 데이터:

$$X = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_i, y_i), \dots, (\mathbf{x}_N, y_N)\}$$

# Mini-batch Gradient Descent

## Gradient Descent에 대한 Recap

- Neural Net 모델을 학습 데이터에 대해서 Gradient Descent 할시:

$$w \rightarrow w - \lambda \cdot \frac{dL}{dw} \text{ (1-D case)}$$

$$\mathbf{w} \rightarrow \mathbf{w} - \lambda \cdot \nabla_{\mathbf{w}} L \text{ (Multi-D case)}$$

- $\nabla L_{\mathbf{w}}$ 을 정확히 근사하려면 모든 데이터  $X$ 에 대한 Jacobian  $\nabla L_{\mathbf{w}}|_{\mathbf{x}_i}$ 을 구해서 평균을 구해야 한다.

$$\nabla_{\mathbf{w}} L \approx \frac{1}{N} \sum_{i=1}^N \nabla_{\mathbf{w}} L|_{\mathbf{x}_i}$$



Full-batch Gradient Descent

# Mini-batch Gradient Descent

## Gradient Descent에 대한 Recap

- $\nabla L_w$ 을 정확히 근사하려면 모든 데이터에 대한 Jacobian  $\nabla L_w|_{\mathbf{x}_i}$ 을 구해서 평균을 구해야 한다.

$$\nabla_w L \approx \frac{1}{N} \sum_{i=1}^N \nabla_w L|_{\mathbf{x}_i}$$

### Full-batch Gradient Descent



- 하지만 Full-batch Gradient를 계산하는 것은

Computational Cost와 Memory Cost가 크다!

# Mini-batch Gradient Descent

## Gradient Descent에 대한 Recap

- $\nabla L_w$ 을 정확히 근사하려면 모든 데이터에 대한 Jacobian  $\nabla L_w|_{\mathbf{x}_i}$ 을 구해서 평균을 구해야 한다.

$$\nabla_w L \approx \frac{1}{N} \sum_{i=1}^N \nabla_w L|_{\mathbf{x}_i}$$

### Full-batch Gradient Descent

- 하지만 (N이 매우 커서) Full-batch Gradient를 계산하는 것은

Computational Cost와 Memory Cost가 크다!

어떻게 해야할까?

# Mini-batch Gradient Descent

## Gradient Descent에 대한 Recap

어떻게 해야할까?

- 한 가지 방법:

$$\nabla_w L \approx \nabla_w L|_{\mathbf{x}_i}$$

- 하나의 data-sample  $\mathbf{x}_i$  에 대한 Gradient로 근사하는 것:

Stochastic Gradient Descent (SGD)

“확률적”이라는 의미

# Mini-batch Gradient Descent

## Gradient Descent에 대한 Recap

어떻게 해야할까?

- 한 가지 방법:

$$\nabla_w L \approx \nabla_w L|_{\mathbf{x}_i}$$

- 하나의 data-sample  $\mathbf{x}_i$  에 대한 Gradient로 근사하는 것:

Stochastic Gradient Descent (SGD)

- 하지만, 하나의 data-sample로 전체 데이터  $X$ 을 대변하는 것은 부정확할 수 있다.

어떻게 할까?





# Mini-batch Gradient Descent

## Gradient Descent에 대한 Recap

어떻게 해야할까?

- 또다른 방법 ( $B \ll N$ ):



$$\nabla_w L \approx \frac{1}{B} \sum_{i=1}^B \nabla_w L|_{\mathbf{x}_i}$$

하나의 데이터 샘플로 근사하는 것보다는 비교적 더 정확하다

- Random하게 샘플링된 B개의 data-sample들에 대한 Gradient로 근사하는 것:

Mini-batch Stochastic Gradient Descent (Mini-batch SGD)

## (Mini-) Batch-size $B$ 에 따른 효과



# Mini-batch Gradient Descent

## Mini-batch 크기의 효과

- Mini-batch의 크기 **B**가 클수록:
  - 전체 데이터에 대한 Loss Gradient를 더 잘 근사하게 된다.

# Mini-batch Gradient Descent

## Mini-batch 크기의 효과

- Mini-batch의 크기 **B**가 클수록:
  - 전체 데이터에 대한 Loss Gradient를 더 잘 근사하게 된다.
- Mini-batch 크기 **B**가 작을수록:
  - Loss Gradient가 더 “noisy”해진다. (Stochasticity 증가)

# Mini-batch Gradient Descent

## Mini-batch 크기의 효과

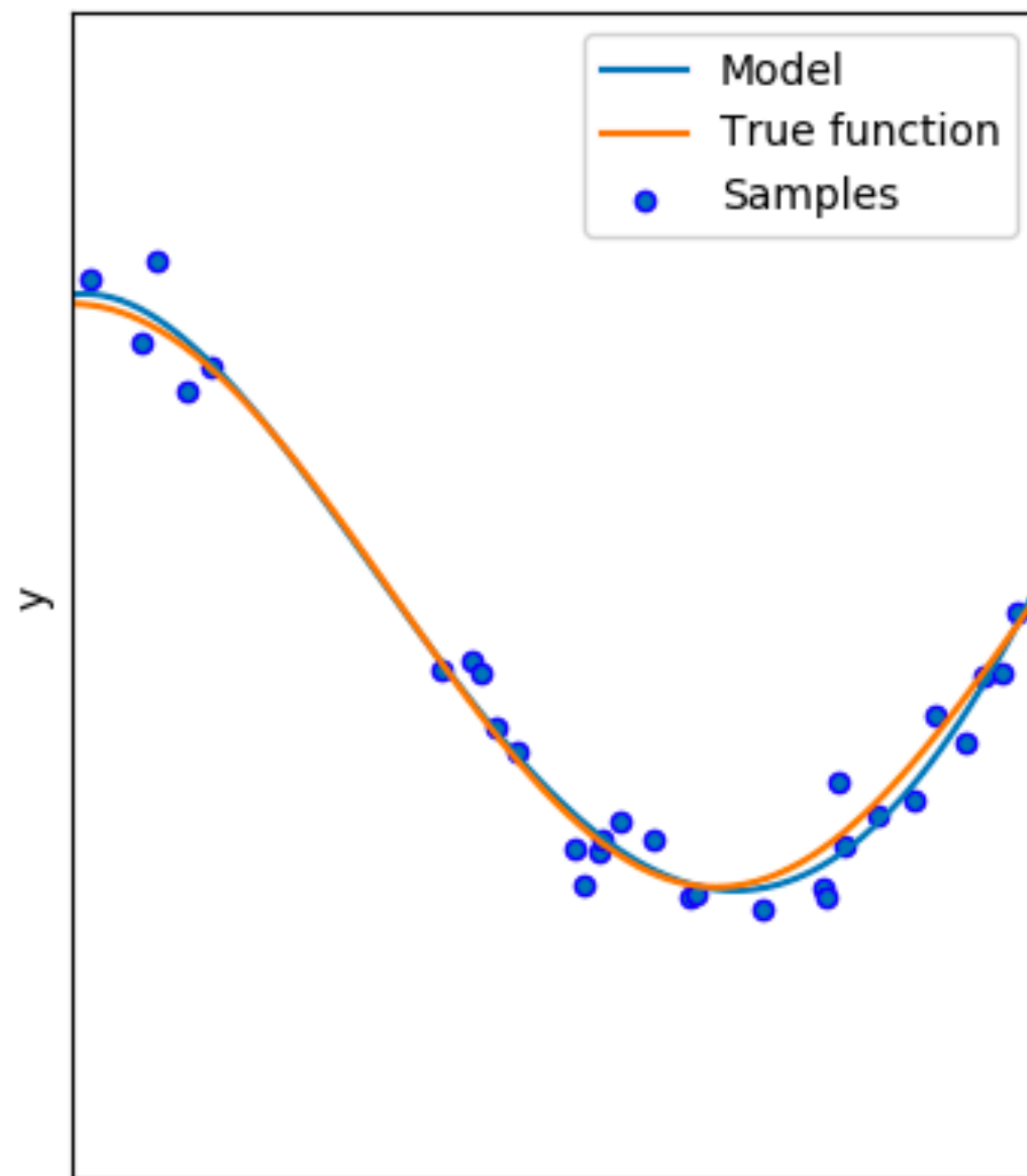
- Mini-batch의 크기 **B**가 클수록:
  - 전체 데이터에 대한 Loss Gradient를 더 잘 근사하게 된다.
- Mini-batch 크기 **B**가 작을수록:
  - Loss Gradient가 더 “noisy”해진다. (Stochasticity 증가)
  - Noise은 regularization의 효과를 주고, generalization 성능에 더 도움을 줄 수 있다.
  - Saddle Point에서 벗어나는데 도움을 줄 수 있다.

# Mini-batch Gradient Descent

## Regularization

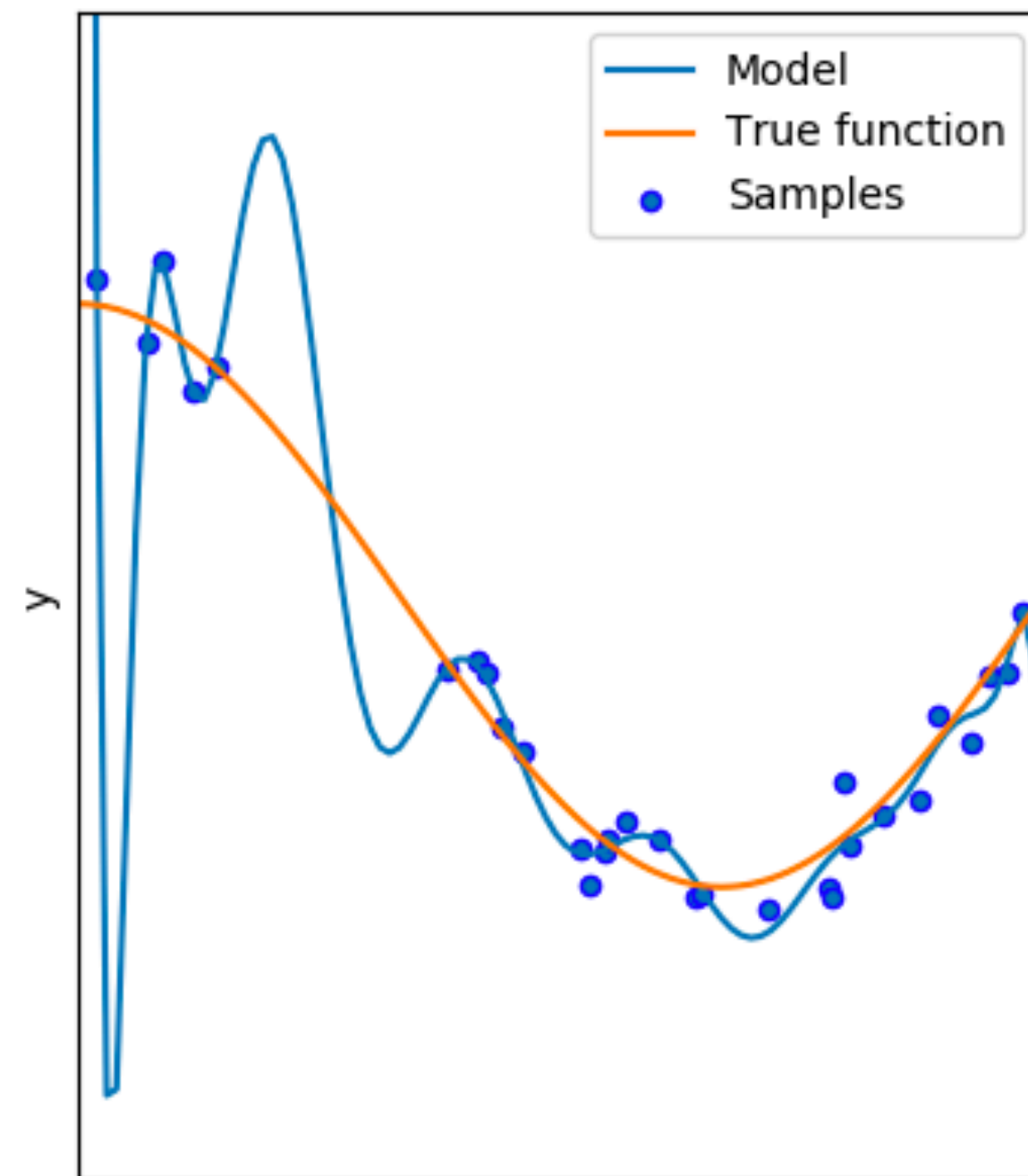
- Overfitting

Degree 4



Good Fit

Degree 15



High Variance

- Regularization 정의 = 뉴럴넷 모델이 너무 복잡해지지 않도록 **model complexity**을 통제하는 방법.
- 왜 필요한가? = 뉴럴넷 모델이 학습되는 과정에서 학습 데이터셋에 대해서 **“overfitting” (과적합)**되는 것을 막기 위해서다.
- 과적합**이란? = 뉴럴넷 모델이 학습 데이터에 있는 **noise (노이즈)**에 대해서도 학습하여 **일반화 성능 (generalizability)**가 저하되는 현상.

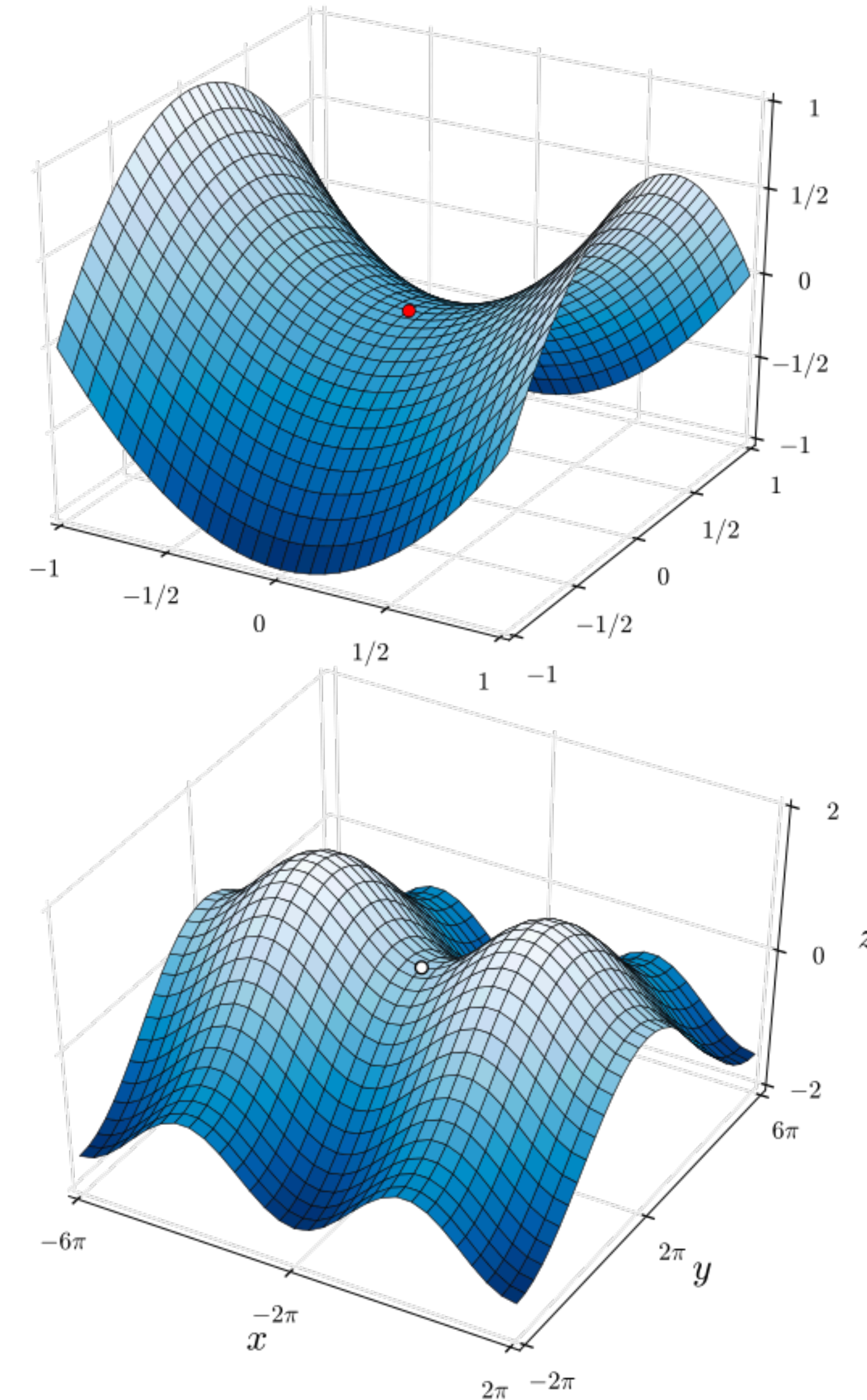


# Mini-batch Gradient Descent

## Saddle Point (안장점)

- Saddle Point (안장점) = 어떤 방향에서 보면 극대값이지만 다른 방향에서는 극소값을 가지는 지점.
- 안장점에서 Gradient는 0이므로 Gradient Descent는 멈춰버림.
- Mini-batch SGD의 경우, Gradient에 Noise가 포함되어 있어 0이 아닐 수 있음.

→ 안장점에서 벗어날 수 있다.



# Mini-batch Gradient Descent

## Mini-batch 크기의 효과

- Mini-batch의 크기 **B**가 클수록:
  - 전체 데이터에 대한 Loss Gradient를 더 잘 근사하게 된다.
- Mini-batch 크기 **B**가 작을수록:
  - Loss Gradient가 더 “noisy”해진다. (Stochasticity 증가)
  - 더 작은 GPU memory에 mini-batch를 채울 수 있다.



# Mini-batch Gradient Descent

## SGD-> Mini-batch Gradient Descent

- Mini-batch SGD은 **B**개의 데이터에 대해서 **Loss gradient**을 구하니 SGD에 비해서 **B배의 시간 (single Gradient Descent step에 소요되는 시간) 이 걸릴까?**

$$\text{Mini-batch: } \mathbf{w} \rightarrow \mathbf{w} - \lambda \cdot \frac{1}{B} \sum_i^B \nabla_{\mathbf{w}} L(y_i, \hat{y}_i)$$

$$\text{SGD: } \mathbf{w} \rightarrow \mathbf{w} - \lambda \cdot \nabla_{\mathbf{w}} L(y_i, \hat{y}_i)$$

# Mini-batch Gradient Descent

## SGD-> Mini-batch Gradient Descent

- Mini-batch Gradient Descent은 **B개의 데이터에 대해서 Loss gradient**를 구하니 SGD에 비해서 **B배**의 시간 (single Gradient Descent step에 소요되는 시간) 이 걸릴까?

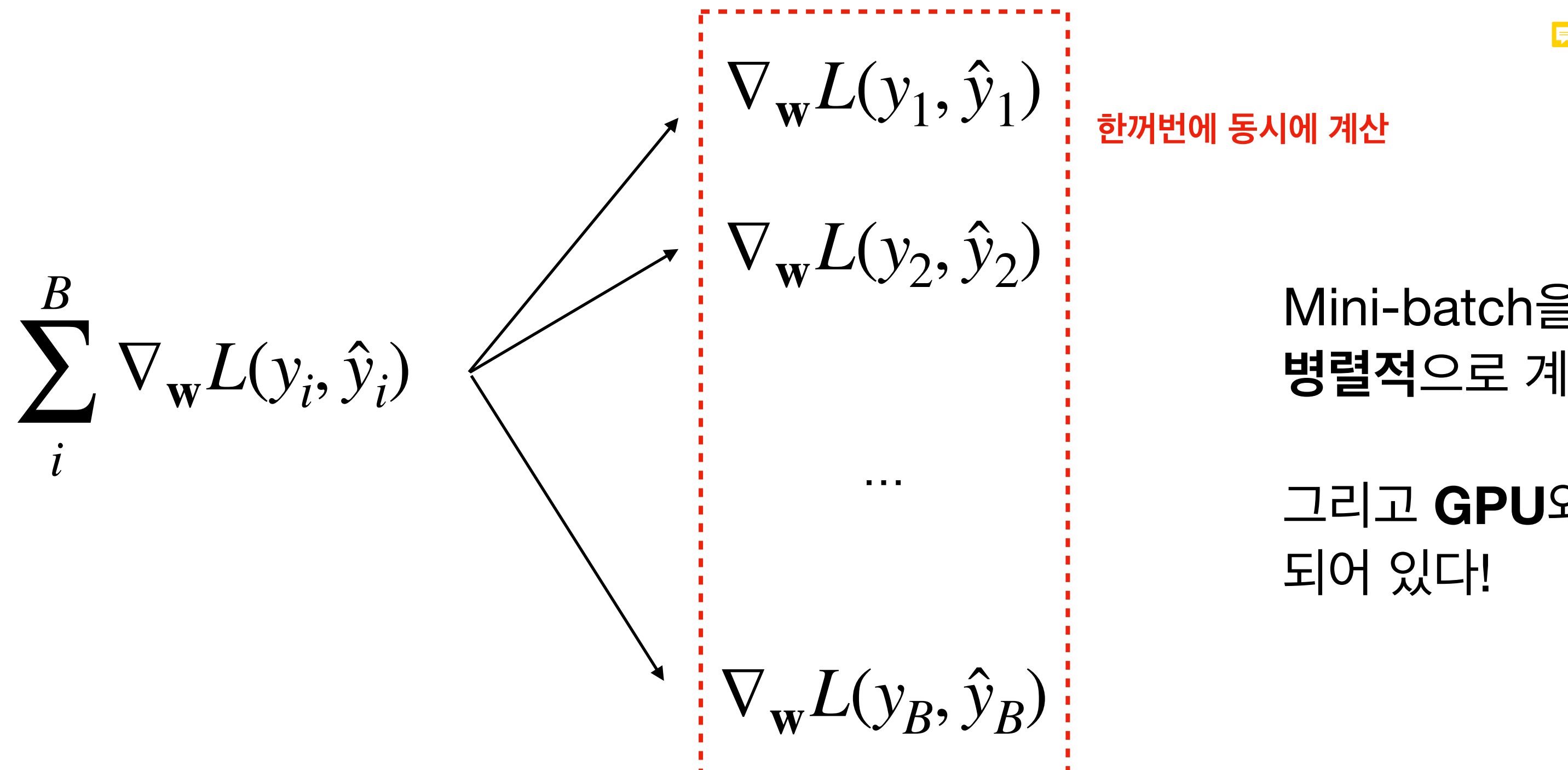
그렇지 않다!

# Mini-batch Gradient Descent

## SGD-> Mini-batch Gradient Descent

$$\text{Mini-batch: } \mathbf{w} \rightarrow \mathbf{w} - \lambda \cdot \frac{1}{B} \sum_i^B \nabla_{\mathbf{w}} L(y_i, \hat{y}_i)$$

PyTorch와 TensorFlow와 같은 딥러닝 Framework에서는



Mini-batch를 구성하는 각 data sample을 동시에,  
**병렬적으로** 계산한다!

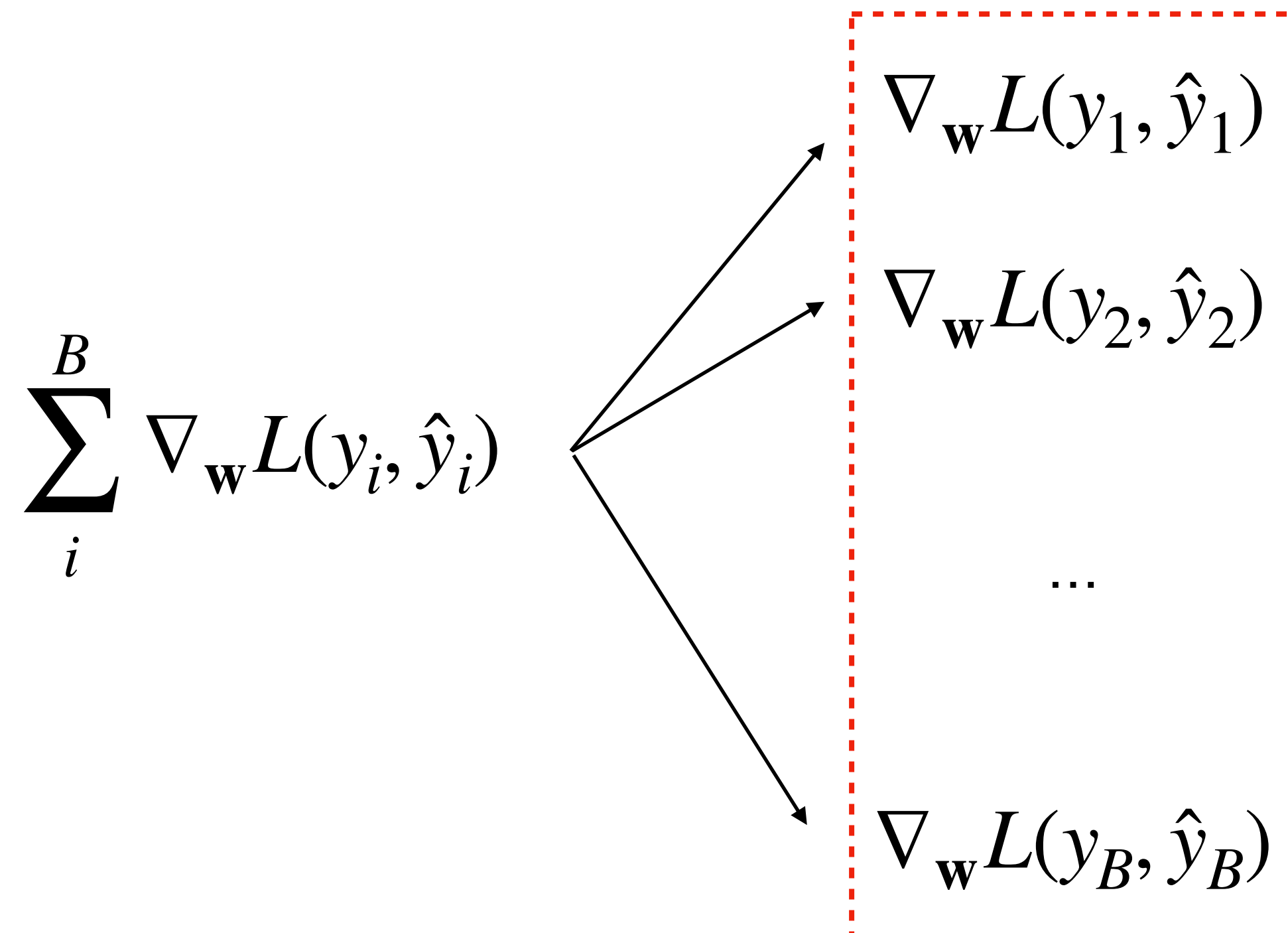
그리고 **GPU**와 **TPU**은 이러한 **병렬적 계산**에 특화  
되어 있다!

# Mini-batch Gradient Descent

## SGD-> Mini-batch Gradient Descent

$$\text{Mini-batch: } \mathbf{w} \rightarrow \mathbf{w} - \lambda \cdot \frac{1}{B} \sum_i^B \nabla_{\mathbf{w}} L(y_i, \hat{y}_i)$$

PyTorch와 Tensorflow와 같은 딥러닝 Framework에서는



따라서 Gradient Descent에 걸리는 computational 시간이 **Mini-batch B**에 비례하지는 않는다.

다만, 데이터 전처리 속도에 따라 영향을 받을 수 있다.

## 5-4. Forward Pass와 Backward Pass



# Forward Pass vs. Backward Pass

Copyright©2023. Acadential. All rights reserved.

## What is Forward Pass?

- Forward Pass란 무엇인가?
- 예를 들어서,
- $\mathbf{x}$ 을 input으로 가지고 weight parameter  $W$ 을 가지는 Neural Network  $f_{NN}(\mathbf{x}; W)$



# Forward Pass vs. Backward Pass

## What is Forward Pass?


Forward Pass란 무엇인가?

- 예를 들어서,
- $\mathbf{x}$ 을 input으로 가지고 weight parameter  $W$ 을 가지는 Neural Network  $f_{NN}(\mathbf{x}; W)$
- Gradient Descent을 하기 위해서는  $W$ 에 대한 Loss Gradient  $\nabla_W L(\hat{y}, y)$  필요.
- **Forward pass는 Gradient를 구하기에 앞서서 먼저 미분할 대상  $L(\hat{y}, y)$ 을 계산하는 과정 의미!**



# Forward Pass vs. Backward Pass

## What is Forward Pass?

- **Forward pass**
  - **Inference** : predicted output (예측값)  $\hat{y} = f_{NN}(\mathbf{x}; W)$ 을 계산. 
  - **Training**: predicted output을 구한 후 **Loss**  $L(\hat{y}, y)$ 까지 계산.
- **Forward propagation**으로도 불림.

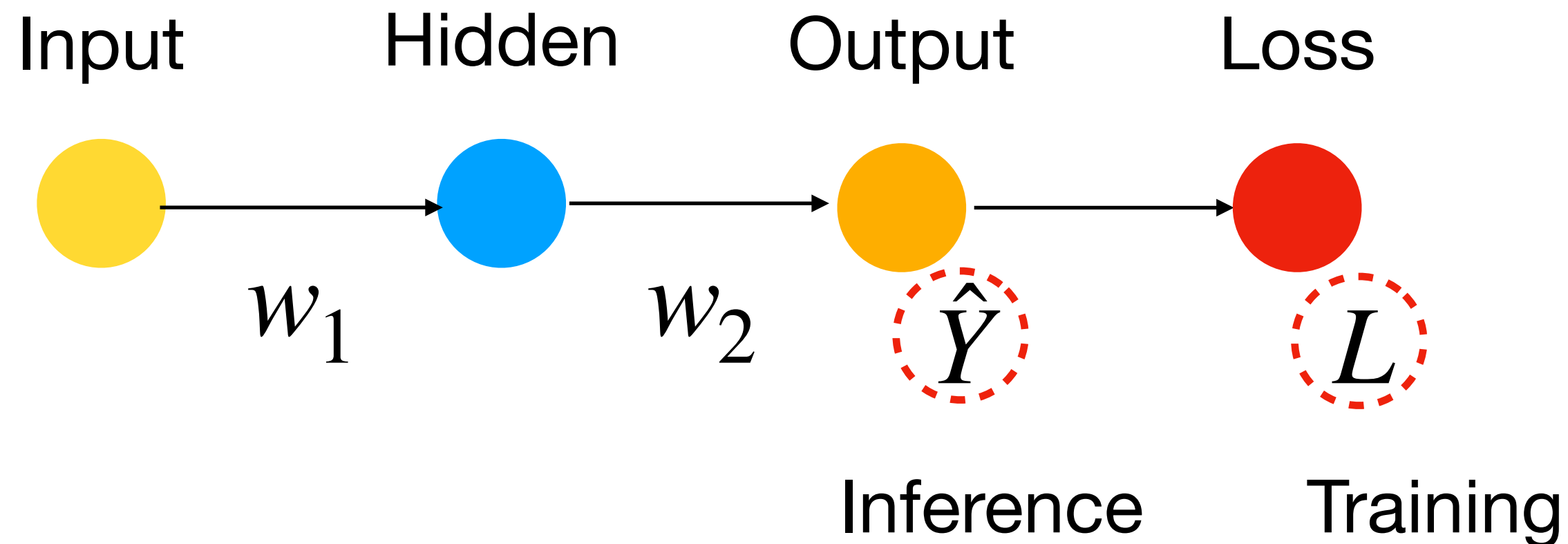


# Forward Pass vs. Backward Pass

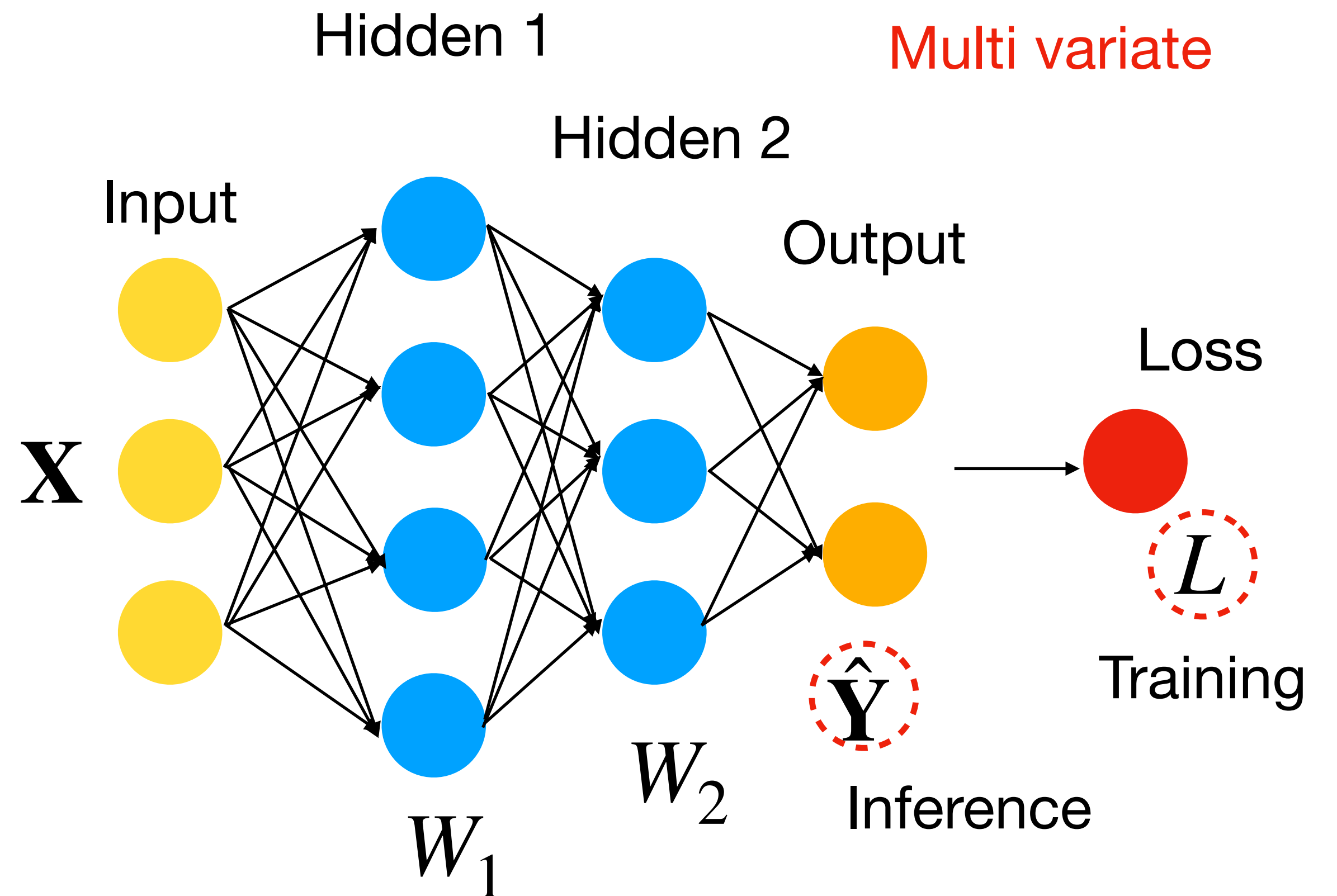
## What is Forward Pass?



Single Variate



Multi variate



# Forward Pass vs. Backward Pass

Copyright©2023. Acadential. All rights reserved.

## What is Backward Pass?

Backward Pass란 무엇인가?

- **Forward pass**에서 출력된 값을 Weight parameter에 대해서 미분하는 것!
- 예를 들어서, Loss Gradient  $\nabla_w L(\hat{y}, y)$ 을 계산하는 것.
- 참고로 **Backward propagation (Back propagation)**으로도 불림.

# Forward Pass vs. Backward Pass

Copyright©2023. Acadential. All rights reserved.

## What is Backward Pass?

**Backward pass:**

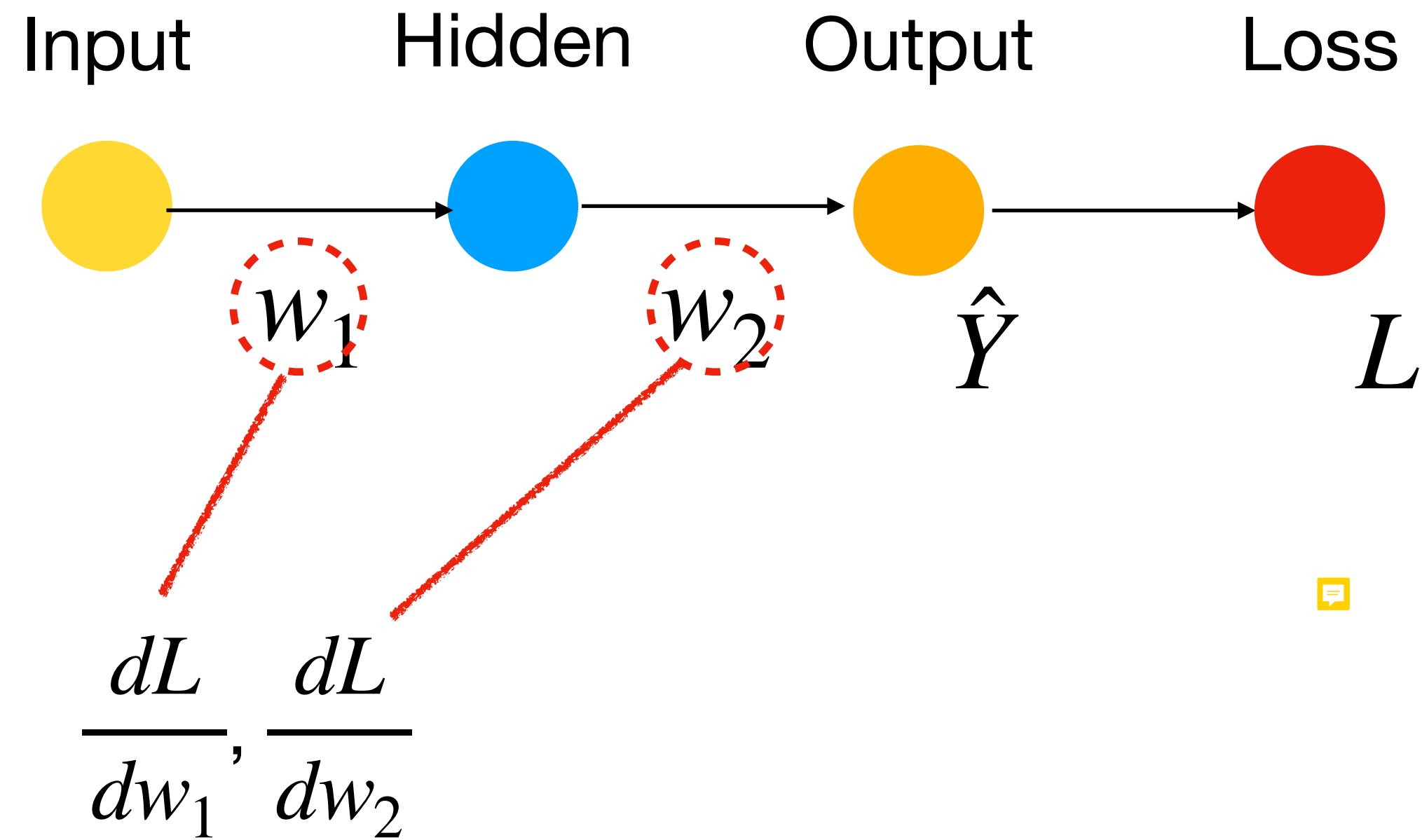
- (엄밀한 정의) Auto Differentiation에서 Reverse Differentiation을 계산하는 과정.



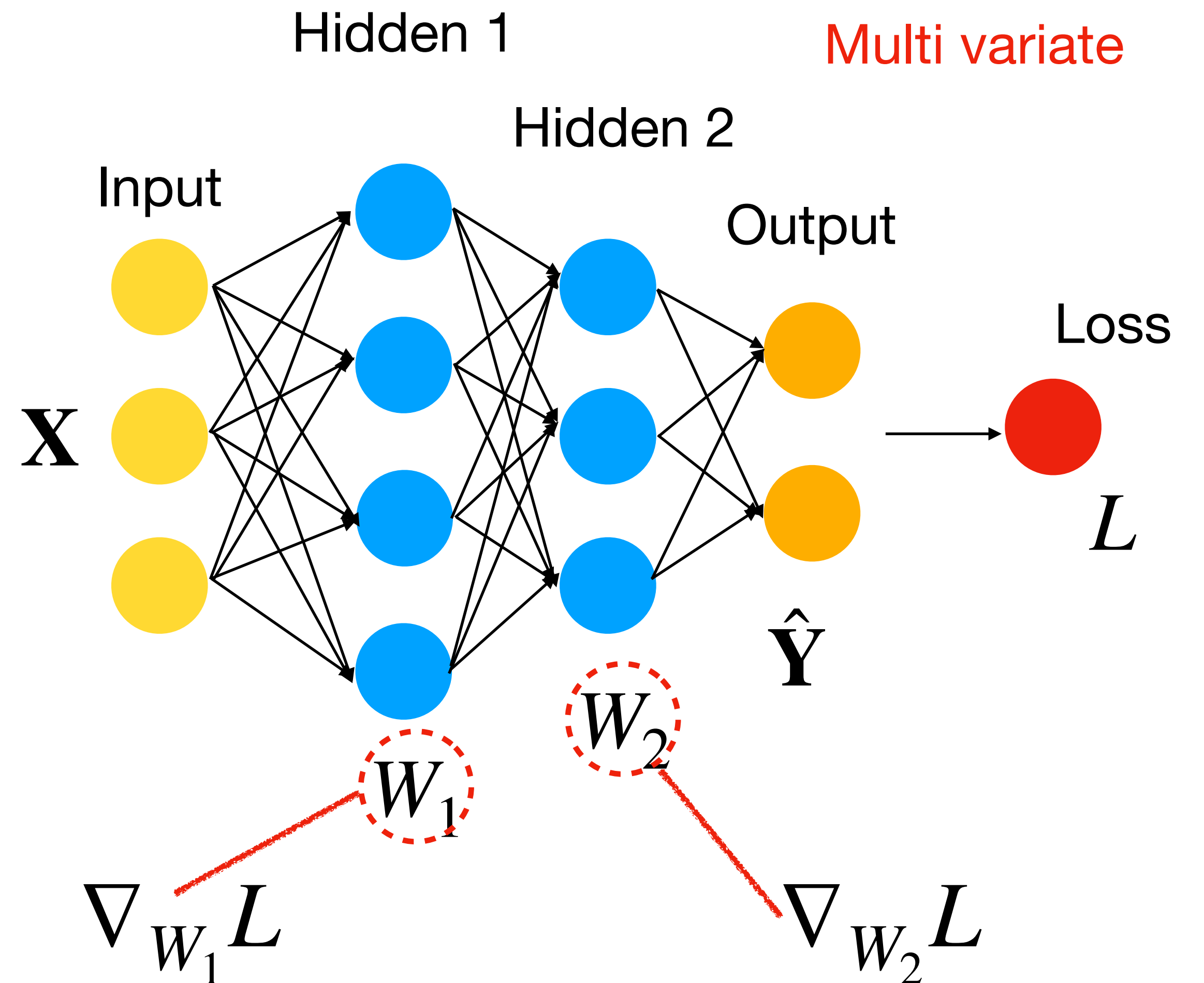
# Forward Pass vs. Backward Pass

## What is Backward Pass?

Single Variate



Multi variate



## 5-5. Section 5 요약

# Objective

## 학습 목표

- Gradient Descent (경사하강방법)의 기본 개념 이해
- Gradient (경사)의 의미 이해
- Learning rate의 효과와 역할
- Mini-batch Stochastic Gradient Descent
- Forward pass vs. Backward pass

# Section Summary

## 경사하강방법의 기본 개념

### Gradient Descent

역할: 손실 함수의 값이 **최소화**하도록 모델의 **weight**을 **최적화**하는 것

원리:

- 경사는 손실함수가 증가하는 방향을 향한다.
- 따라서 경사하강은 경사의 음의 방향으로 모델의 weight를 update해주는 것이다!

$$w_{i+1} = w_i - \lambda \cdot \frac{dL}{dw}$$

# Section Summary

## 경사 하강 예시

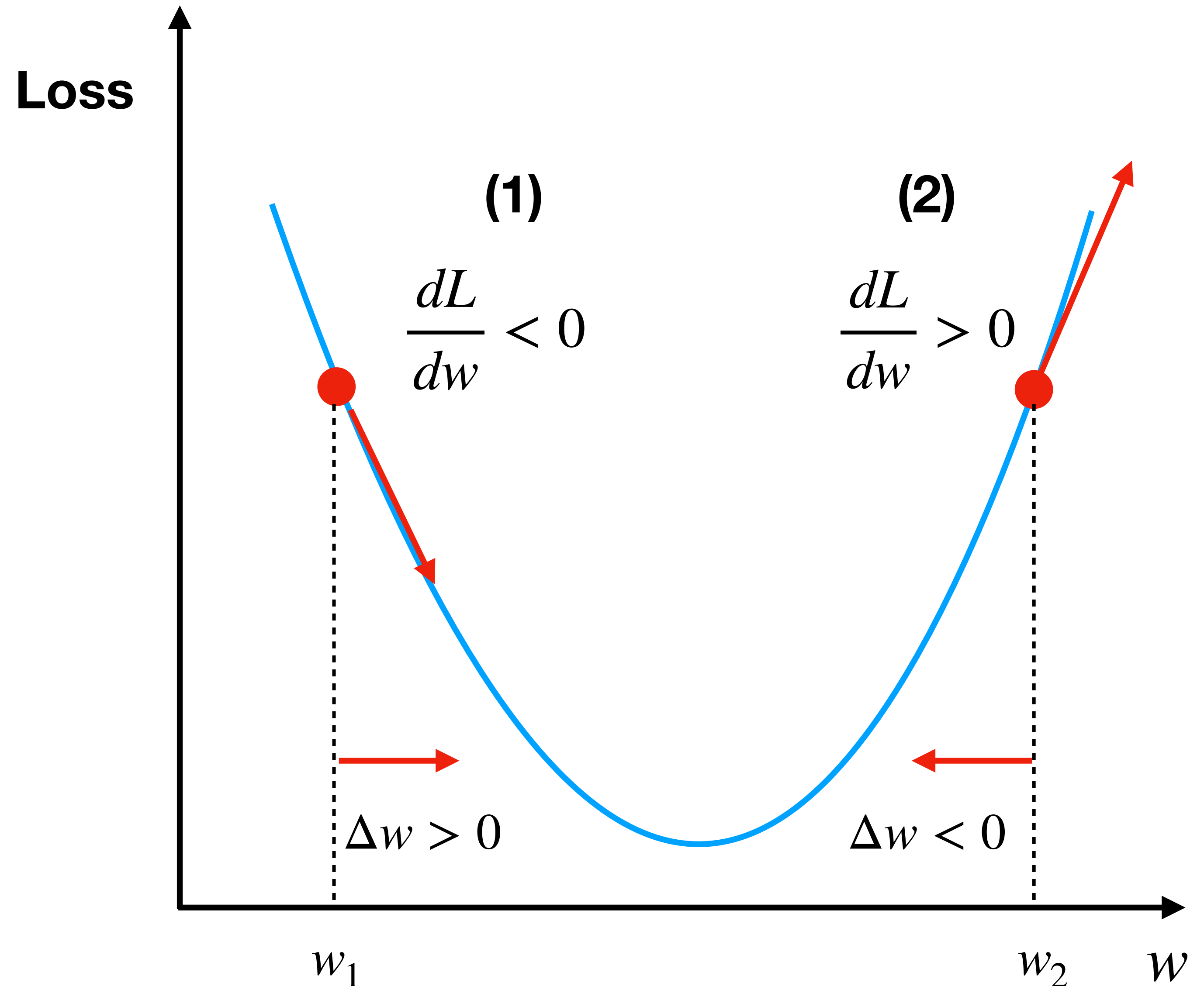
$$w_{i+1} = w_i - \lambda \frac{dL}{dw} = \Delta w$$

(1)의 경우

$$\frac{dL}{dw} < 0 \rightarrow \Delta w > 0$$

(2)의 경우

$$\frac{dL}{dw} > 0 \rightarrow \Delta w < 0$$

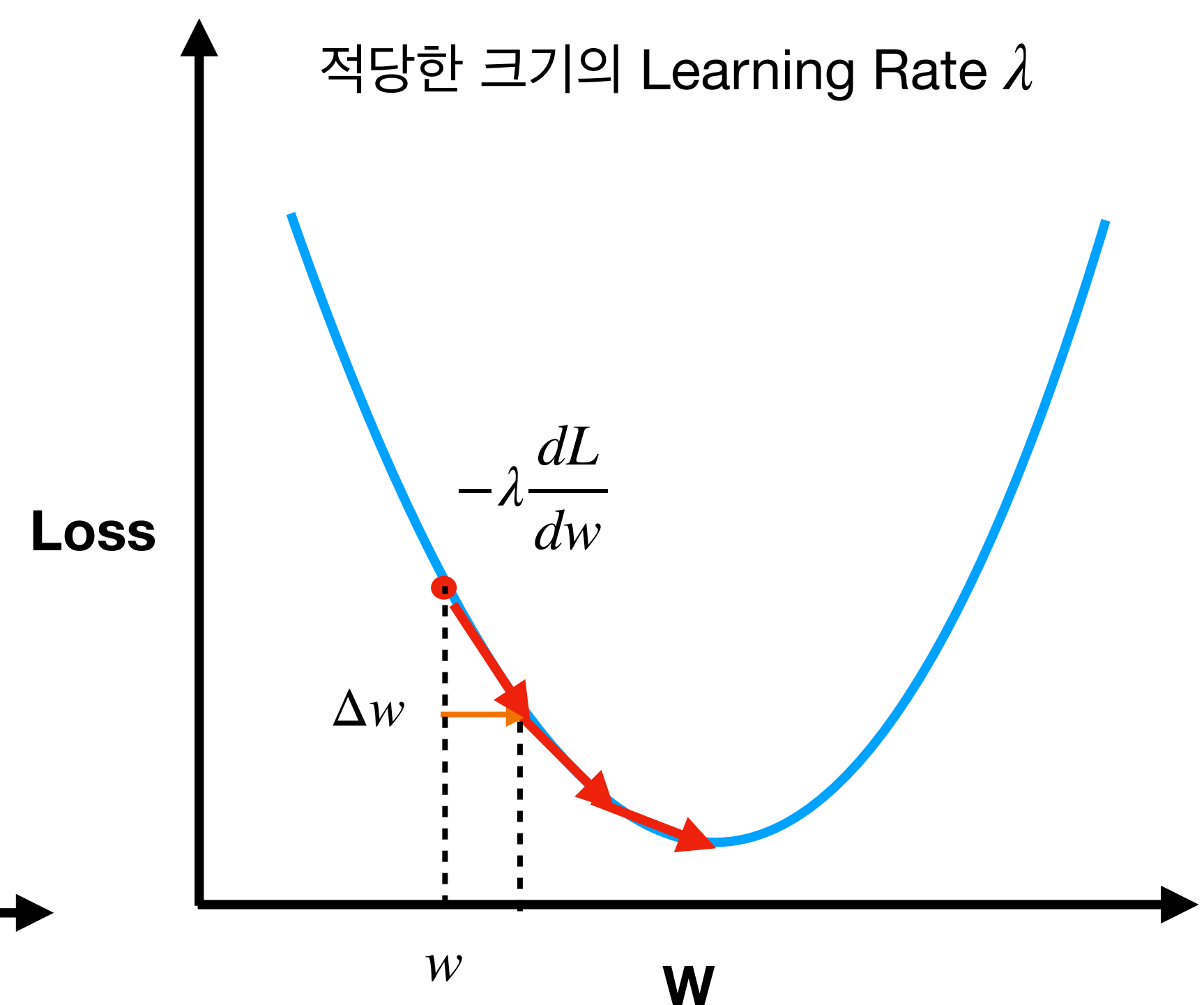
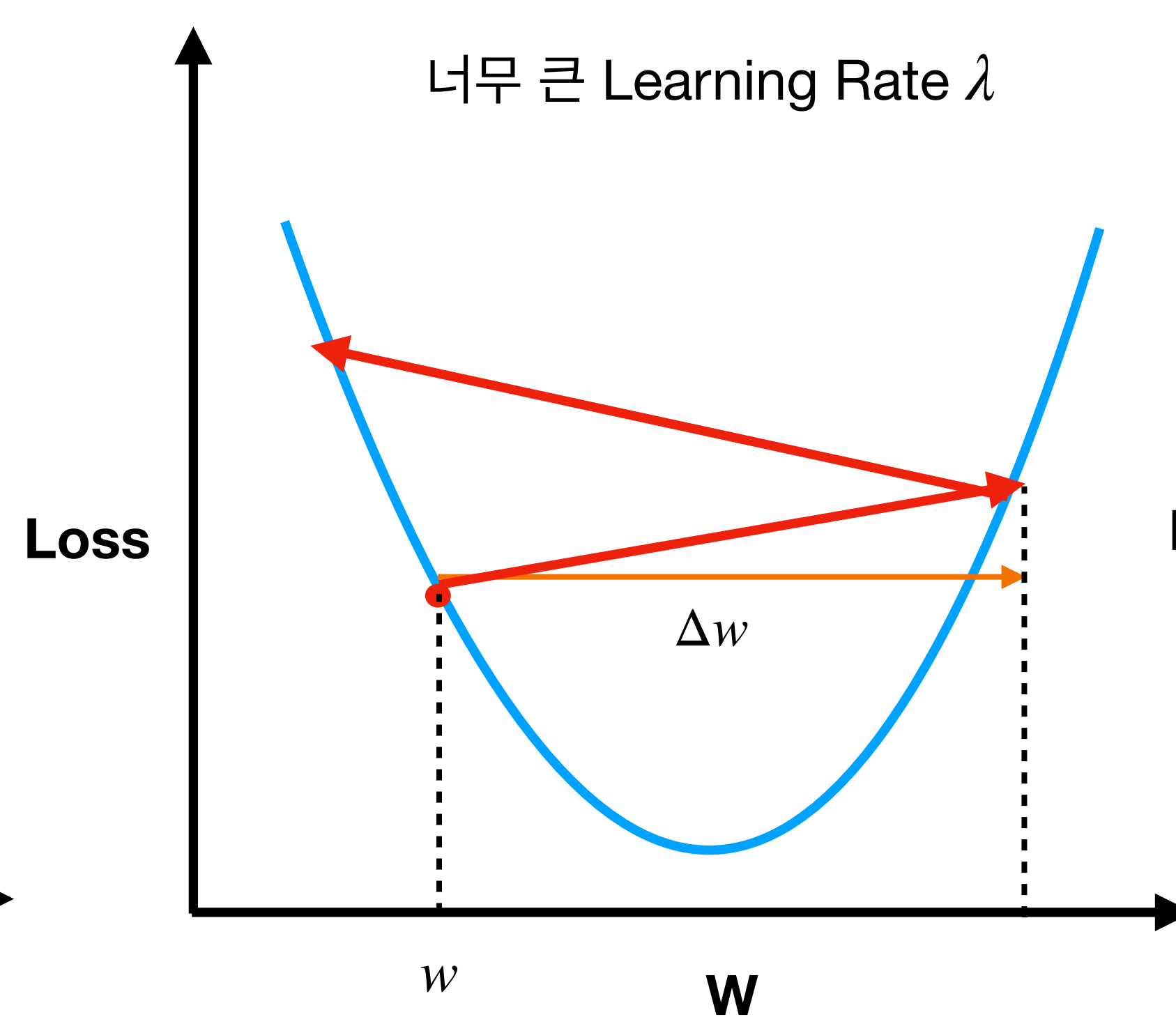
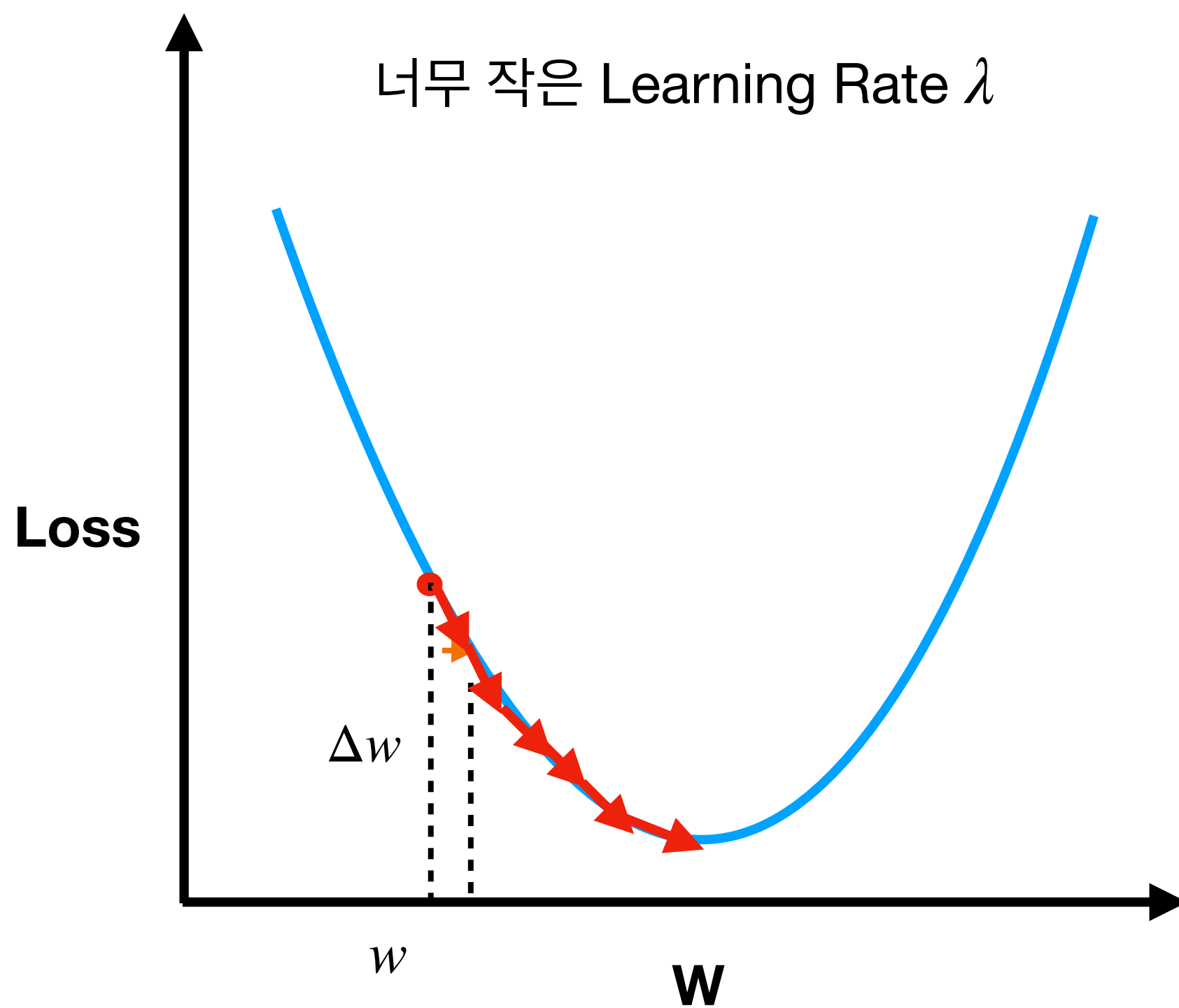




# Section Summary

## Learning Rate

$$\Delta w = -\lambda \cdot \frac{dL}{dw}$$



# Section Summary

## Gradient Descent 정리

- **Full-batch** Gradient Descent:

$$\nabla_w L \approx \frac{1}{N} \sum_{i=1}^N \nabla_w L|_{\mathbf{x}_i}$$

- **Stochastic** Gradient Descent:

$$\nabla_w L \approx \nabla_w L|_{\mathbf{x}_i}$$

- **Mini-batch Stochastic** Gradient Descent ( $B \ll N$ ):

$$\nabla_w L \approx \frac{1}{B} \sum_{i=1}^B \nabla_w L|_{\mathbf{x}_i}$$

# Section Summary

## Mini-batch 크기의 효과

- Mini-batch의 크기 **B**가 클수록:
  - 전체 데이터에 대한 Loss Gradient를 더 잘 근사하게 된다.
- Mini-batch 크기 **B**가 작을수록:
  - Loss Gradient가 더 “noisy”해진다. (Stochasticity 증가)
    - Noise은 regularization의 효과를 주고, generalization 성능에 더 도움을 줄 수 있다.
    - Saddle Point에서 벗어나는데 도움을 줄 수 있다
  - 더 작은 GPU memory에 mini-batch를 채울 수 있다.

# Section Summary

	특징	공식
<b>Full-Batch Gradient Descent</b>	주어진 전체 데이터셋에 대해서 Loss Gradient을 계산. 하지만 Computational Cost와 Memory Cost가 크다.	$\nabla_w L \approx \frac{1}{N} \sum_{i=1}^N \nabla_w L _{\mathbf{x}_i}$
<b>Stochastic Gradient Descent</b>	하나의 data sample에 대한 Loss Gradient으로 근사. 하지만 하나의 data sample로만 근사하는 것은 부정확, Noisy.	$\nabla_w L \approx \nabla_w L _{\mathbf{x}_i}$
<b>Mini-batch Stochastic Gradient Descent</b>	Random하게 샘플링된 B개의 data sample들로 Loss Gradient으로 근사	$\nabla_w L \approx \frac{1}{B} \sum_{i=1}^B \nabla_w L _{\mathbf{x}_i}$

# Forward Pass vs. Backward Pass

Copyright©2023. Acadential. All rights reserved.

## What is Forward Pass?

- **Forward pass**

**Gradient**을 구하기에 앞서서 먼저 미분할 대상을 계산하는 과정.

- **Backward pass**

Forward pass에서 출력된 값을 Weight parameter에 대해서 **미분하는 것!**

**Reverse Differentiation**을 계산하는 과정.

# Next Up!

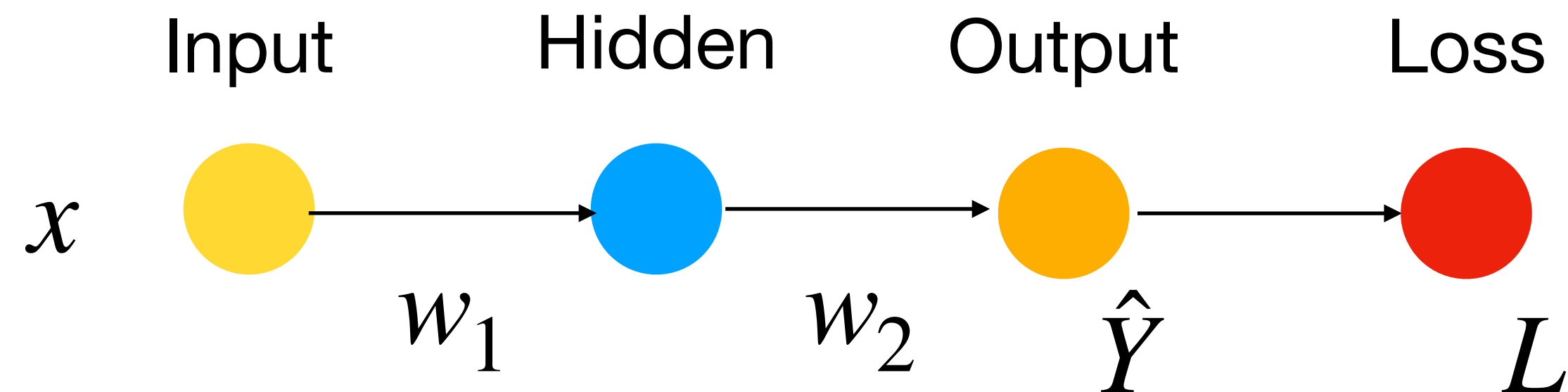
# Next Up!

## Multivariate Input

- 앞서 저희는 variable이 하나이고 weight가 scalar인 간단한 예시를 살펴보았음.

$$w_{i+1} = w_i - \lambda \cdot \frac{dL}{dw}$$

- 즉, 다음과 같은 경우라고 볼 수 있다.



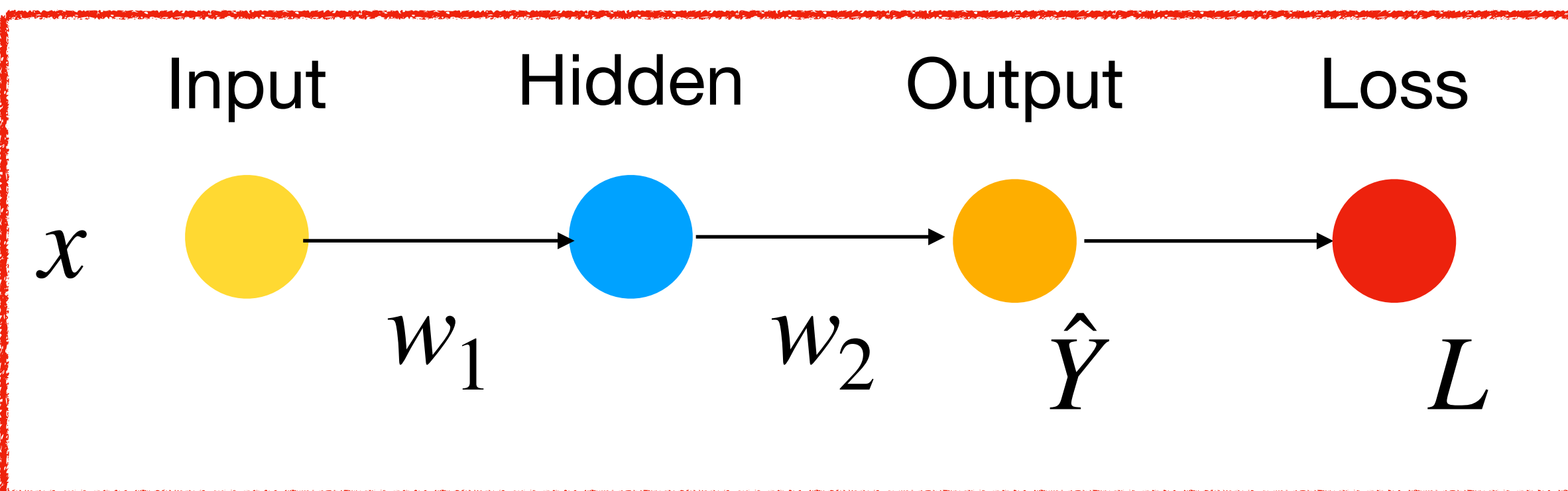
# Next Up!

## Multivariate Input

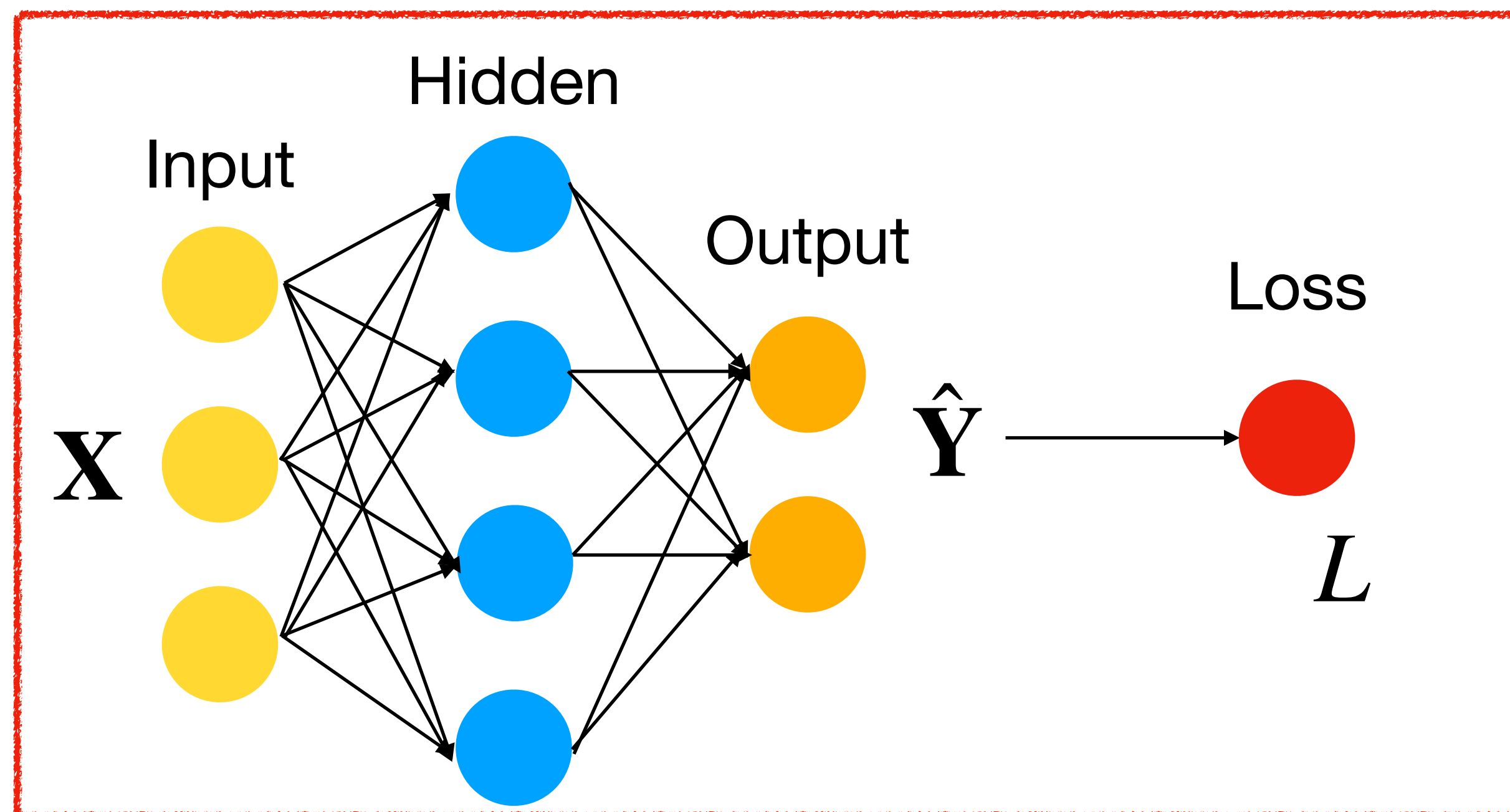


- 하지만 Input feature가 여러 개 (multi-variate)하거나 Hidden layer가 여러개의 neuron들로 구성되어 있으면 어떻게 할 것인가?

Single Variate



Multi variate



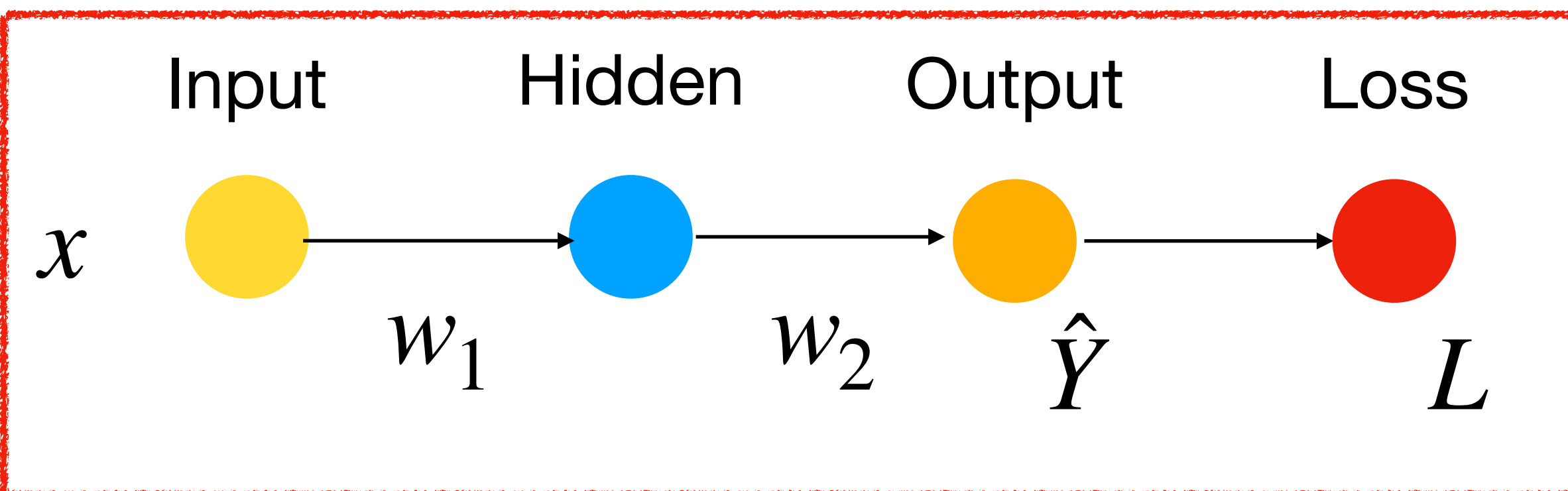


# Next Up!

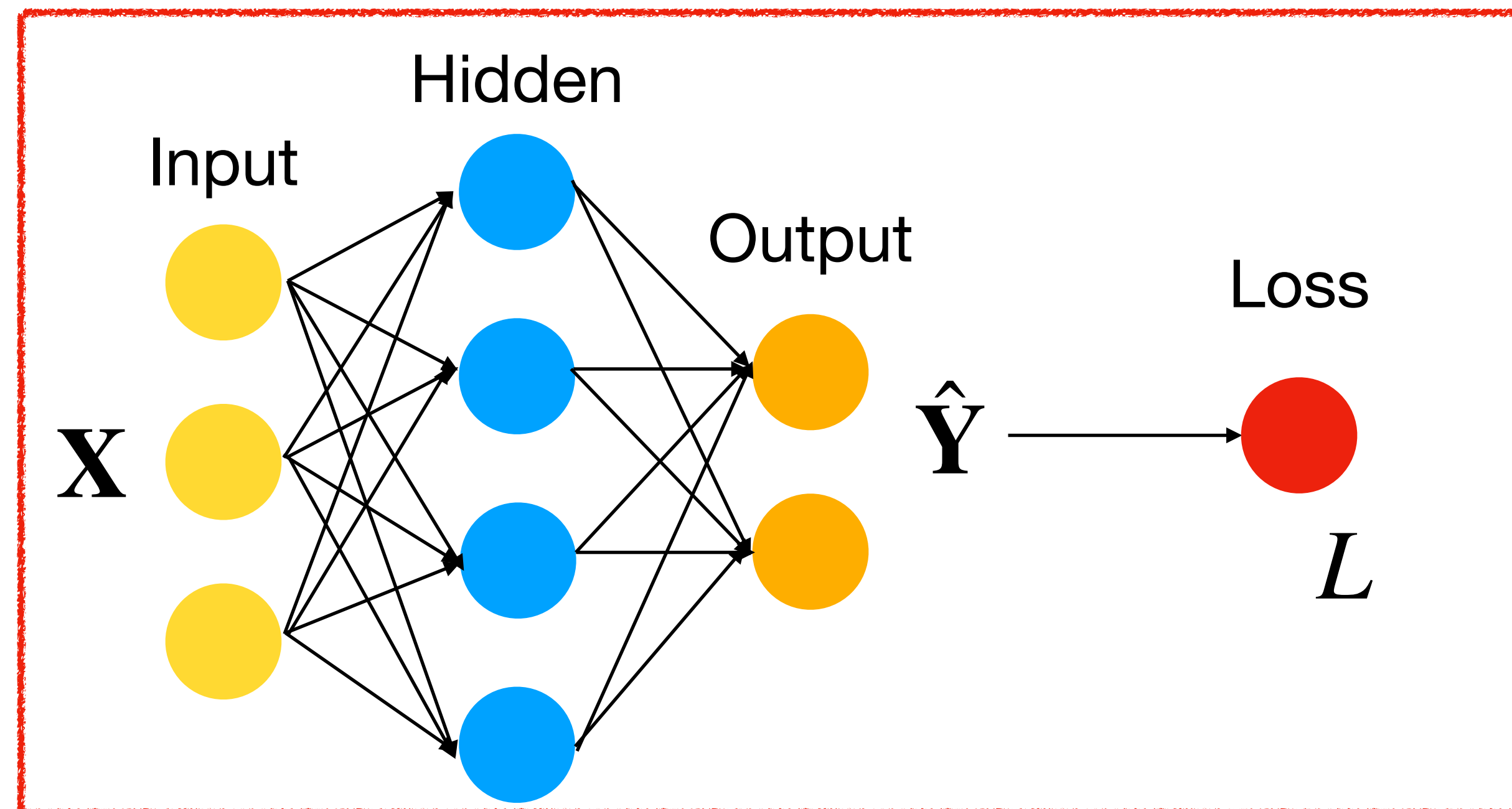
## Multivariate Input

- 다음 섹션에서는 “Multi-variate”한 경우에 대해서 더 자세히 살펴볼 것이다!

Single Variate



Multi variate



# Next Up!

## Automatic Differentiation

- 그렇다면 Gradient Descent은 PyTorch와 같은 Deep Learning framework에서 어떻게 구현되어 있을까?



# Next Up!

## Automatic Differentiation

- Neural Network의 Gradient를 효과적으로 계산하기 위해서 Deep Learning framework들은 Automatic differentiation에서 “Reverse differentiation” 개념을 사용한다.
- Reverse differentiation이 뭔지 살펴보자!

