

Teacher Training Presentation



MLDL

인공지능(artificial intelligence)

INTRODUCE



schedule

- 01** 퍼셉트론, 인공신경망
- 02** KNN + 회귀
결정트리, 앙상블,
- 03** 비지도 학습
- 04** DNN, CNN, RNN
- 05** 강화학습, GAN, LLM

CONTENTS



- 01 MLDL**
- 02 퍼셉트론**
- 03 인공신경망**
- 04 손실 최적화**

0
ML
DL

Machine Learning Deep Learning

ARTIFICIAL INTELLIGENCE

Any technique that enables computers to mimic human behavior



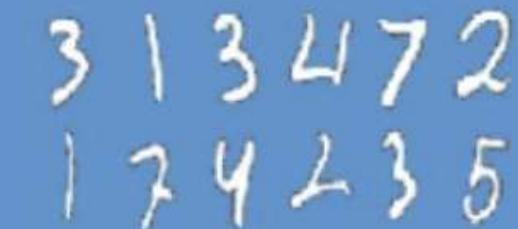
MACHINE LEARNING

Ability to learn without explicitly being programmed



DEEP LEARNING

Extract patterns from data using neural networks



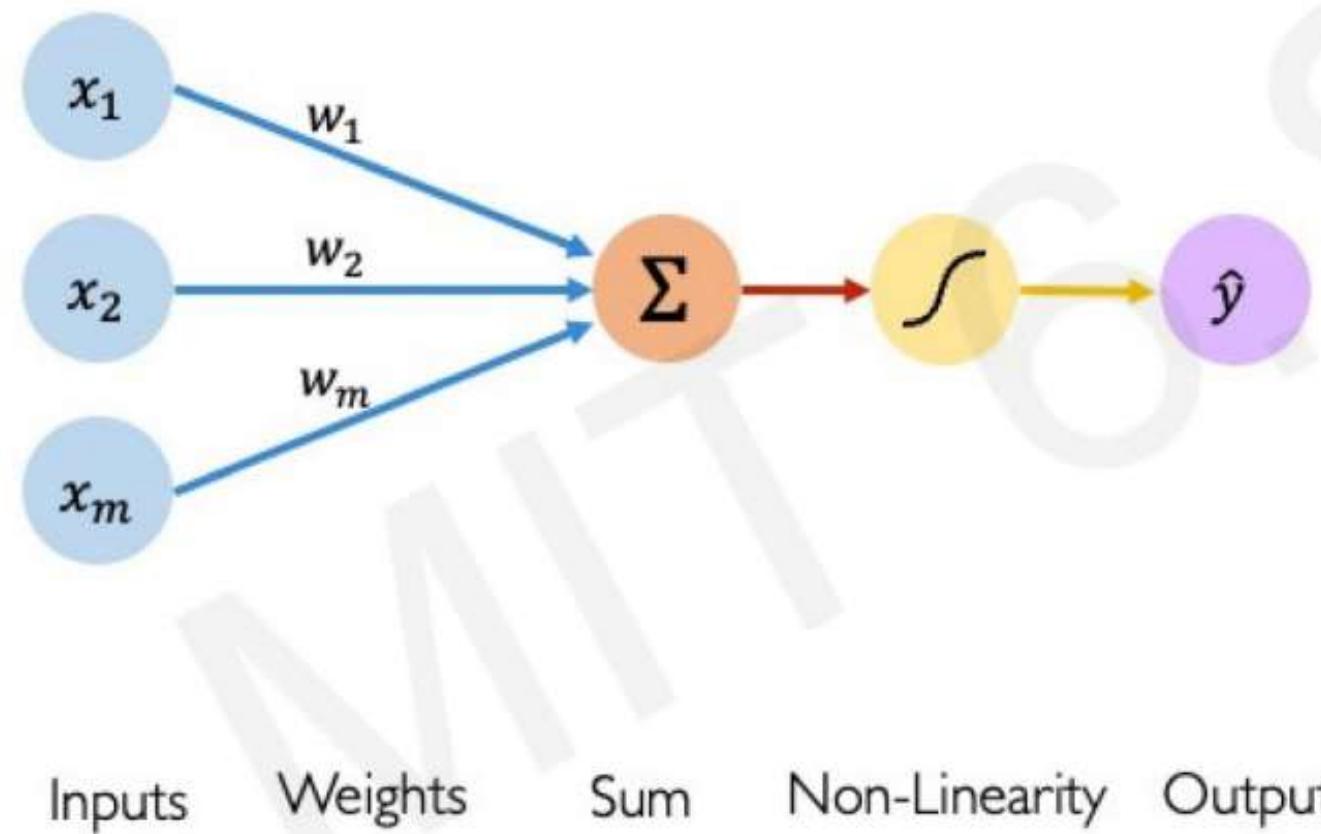
인간의 행동을 컴퓨터가 모방한
어떤 기술

프로그래밍되지 않아도
학습할 수 있는

인간의 행동을 컴퓨터가 모방한
어떤 기술

PERCEPTRON 02

퍼셉트론(초기 형태 인공신경망)



Output

Linear combination of inputs

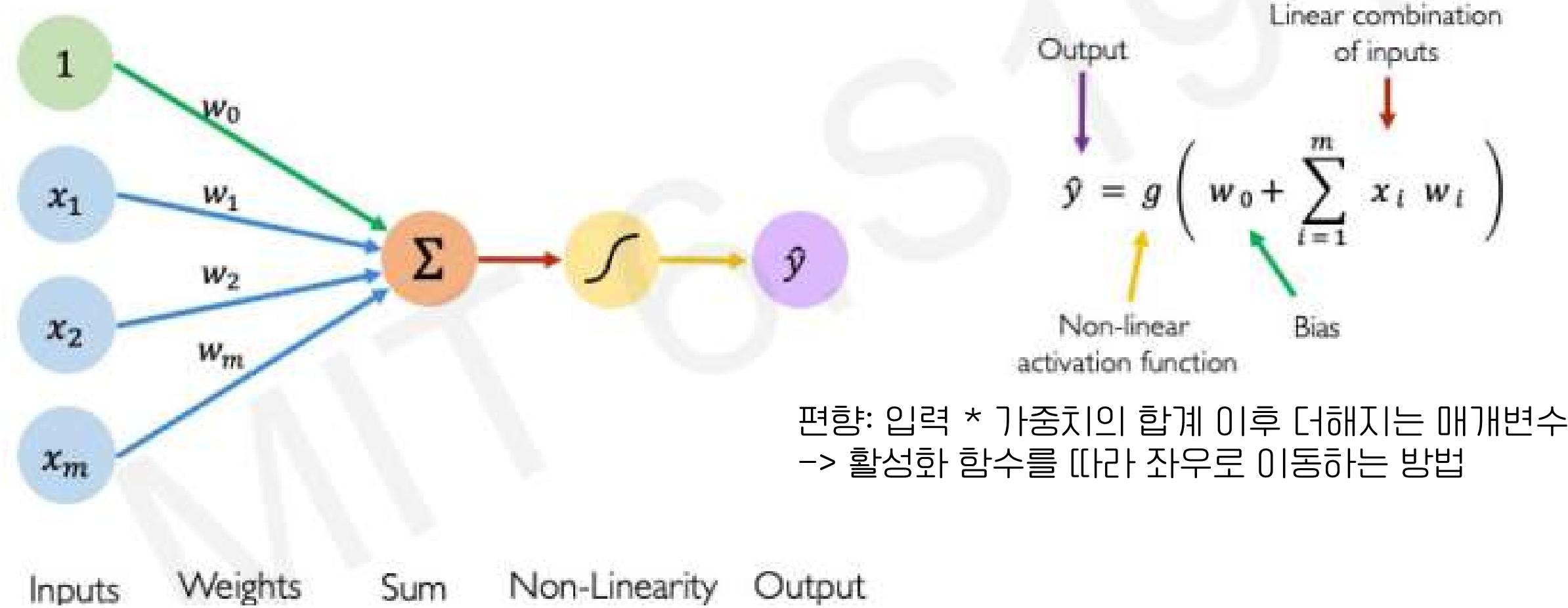
$\hat{y} = g \left(\sum_{i=1}^m x_i w_i \right)$

Non-linear activation function

가중치: 출력값에 영향을 주는 중요도를 조절하는 매개변수 \rightarrow 입력값에 곱해지는 수

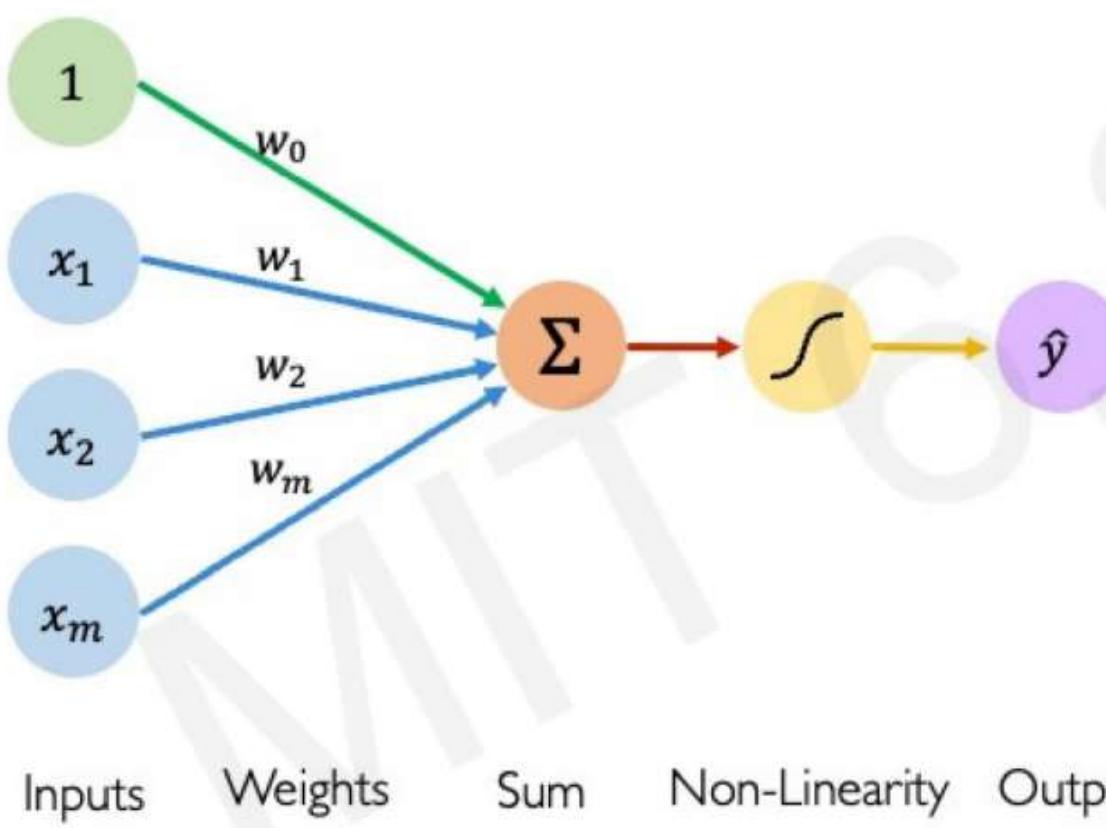
PERCEPTRON 02

퍼셉트론(초기 형태 인공신경망)



PERCEPTRON 02

퍼셉트론(초기 형태 인공신경망)



$$\hat{y} = g \left(w_0 + \sum_{l=1}^m x_l w_l \right)$$

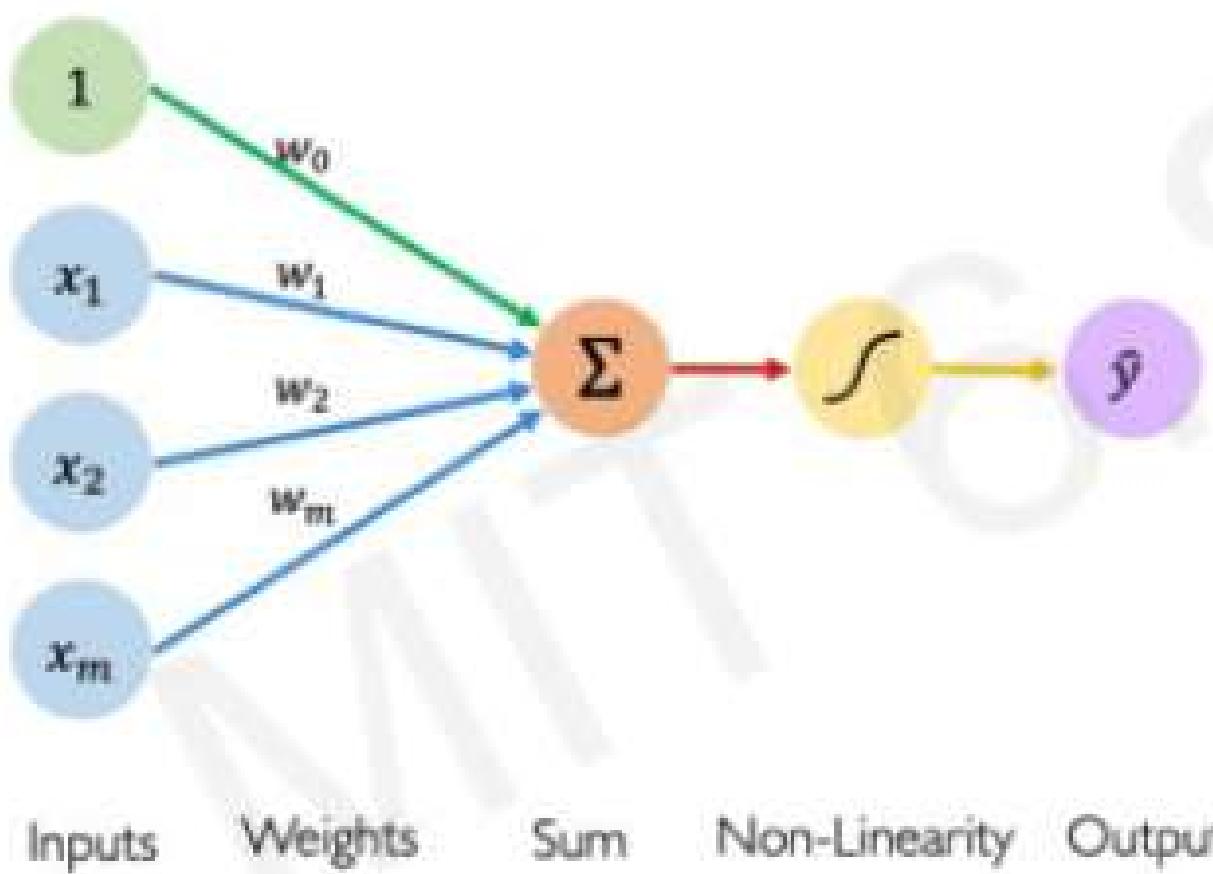
$$\hat{y} = g(w_0 + \mathbf{X}^T \mathbf{W})$$

$$\text{where: } \mathbf{X} = \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix} \text{ and } \mathbf{W} = \begin{bmatrix} w_1 \\ \vdots \\ w_m \end{bmatrix}$$

스칼라곱: 두 벡터로부터 실수 스칼라를 얻는 연산
→ 선형 연산이라는 사실을 알 수 있다.

PERCEPTRON 02

퍼셉트론(초기 형태 인공신경망)



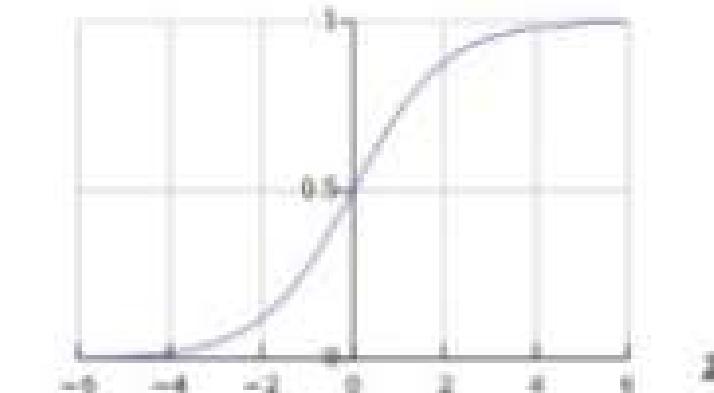
Activation Functions

활성화 함수: 비선형 함수

$$y = g(w_0 + X^T W)$$

- Example: sigmoid function

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



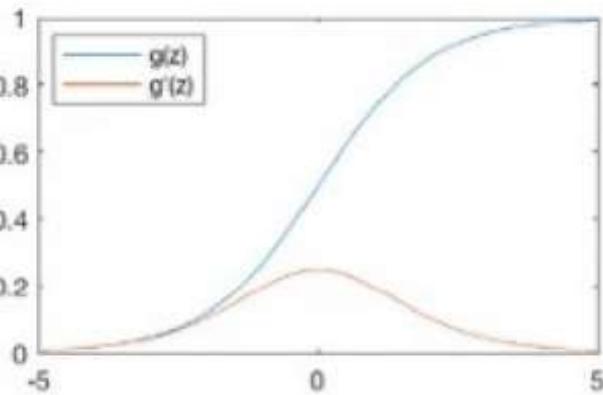
입력과 출력이 비례하지 않는 함수를 말함

PERCEPTRON 02

퍼셉트론(초) | 형태 인공신경망

시그모이드: 출력이 0과 1 사이의 값만 나옴
-> 확률에 좋음

Sigmoid Function



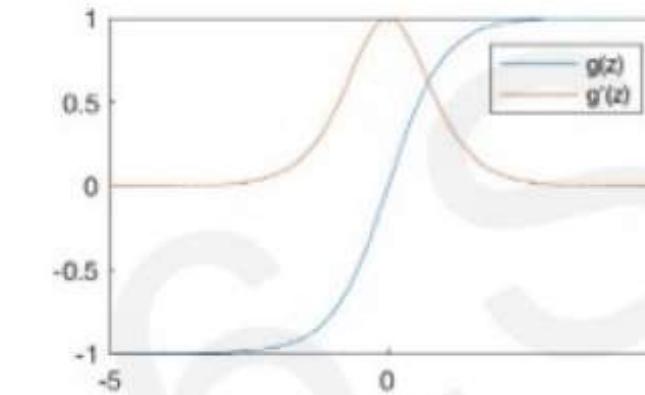
$$g(z) = \frac{1}{1 + e^{-z}}$$

$$g'(z) = g(z)(1 - g(z))$$

`tf.math.sigmoid(z)`

하이퍼볼릭 탄젠트: 출력이 $-1 \sim 1$ 사이의 값
-> 최적화를 잘함

Hyperbolic Tangent



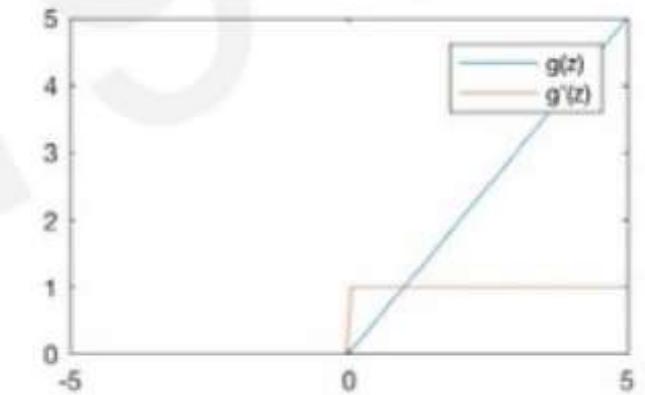
$$g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

$$g'(z) = 1 - g(z)^2$$

`tf.math.tanh(z)`

ReLU: 부분적으로 선형적이지만 x 가 0일 때 단일 비선형을 가짐
-> 계산이 빠름

Rectified Linear Unit (ReLU)



$$g(z) = \max(0, z)$$

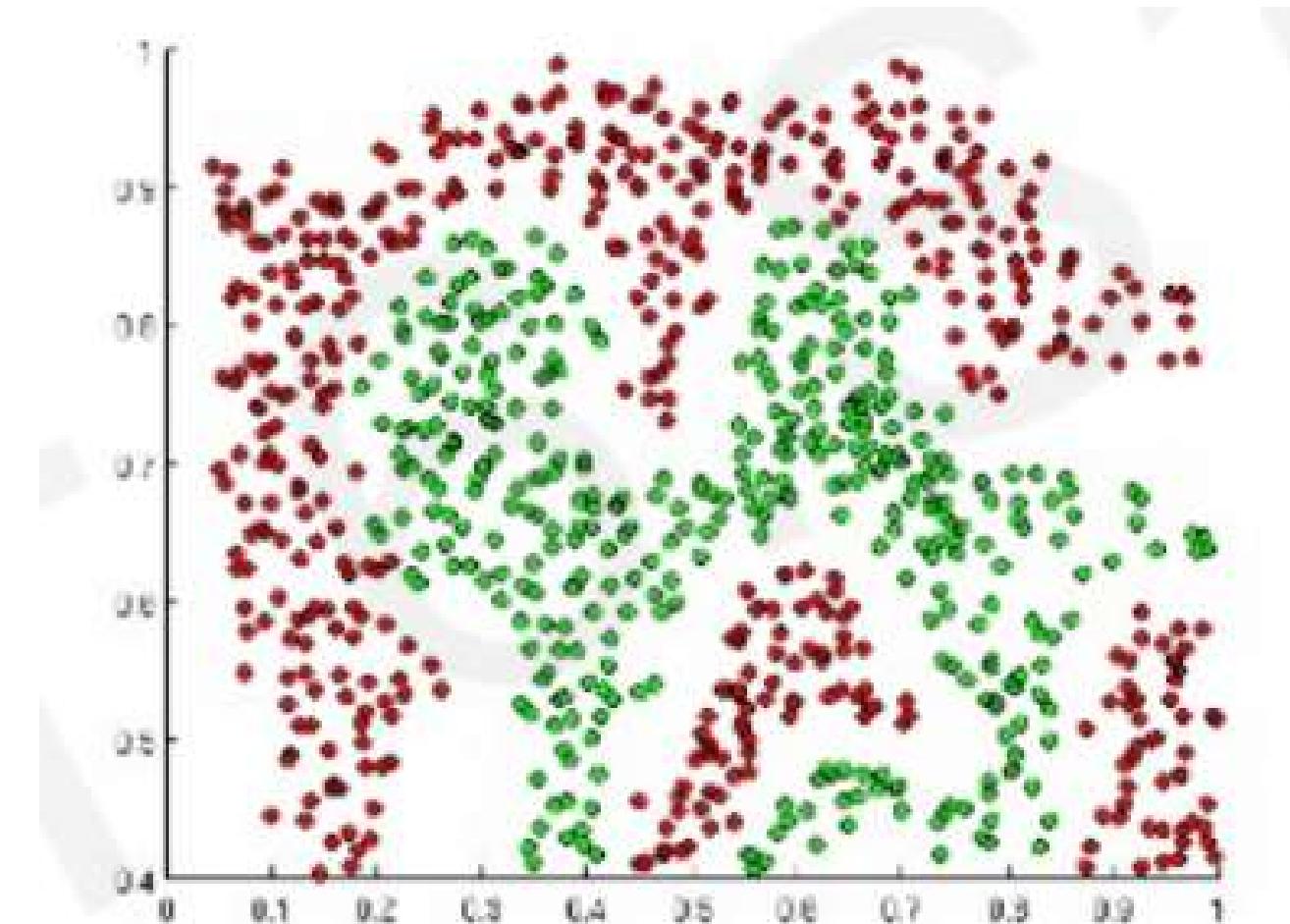
$$g'(z) = \begin{cases} 1, & z > 0 \\ 0, & \text{otherwise} \end{cases}$$

`tf.nn.relu(z)`

PERCEPTRON 02

퍼셉트론(초기 형태 인공신경망)

활성화 함수가 필요한 이유는 무엇일까?

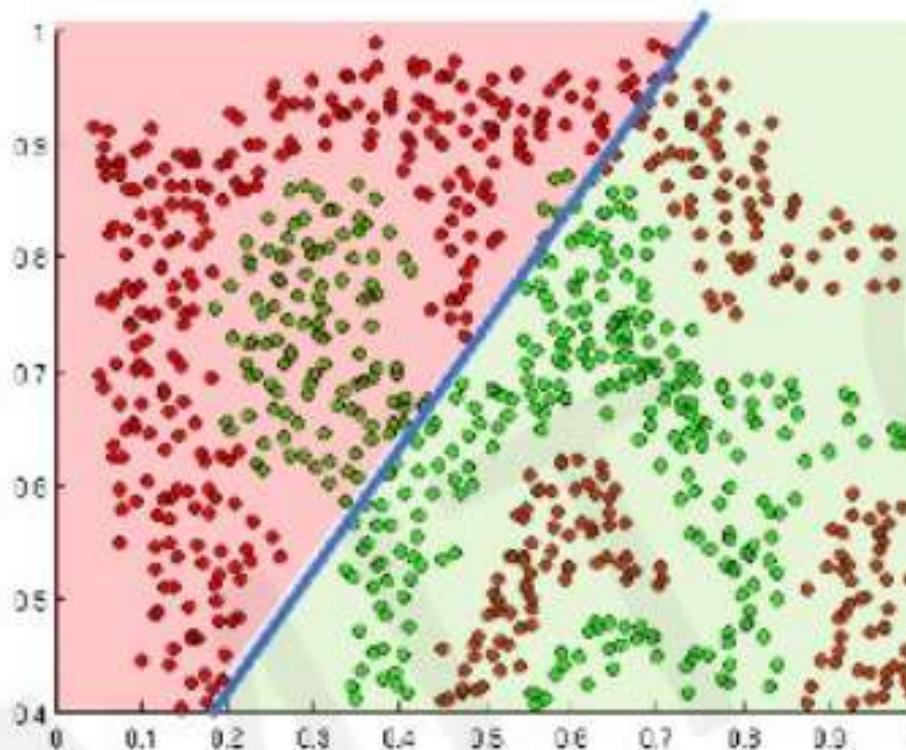


green과 red점을 구별하는 인공신경망을 구축하고 싶다면
어떻게 해야 할까?

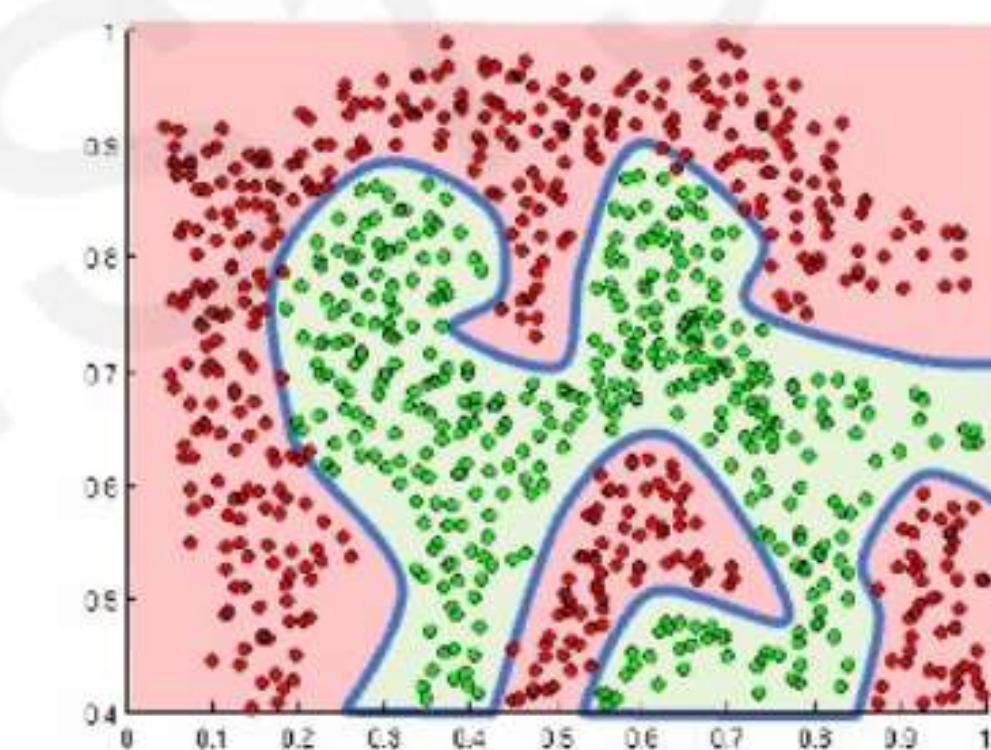
PERCEPTRON 02

퍼셉트론(초기 형태 인공신경망)

비선형성을 추가하기 위함



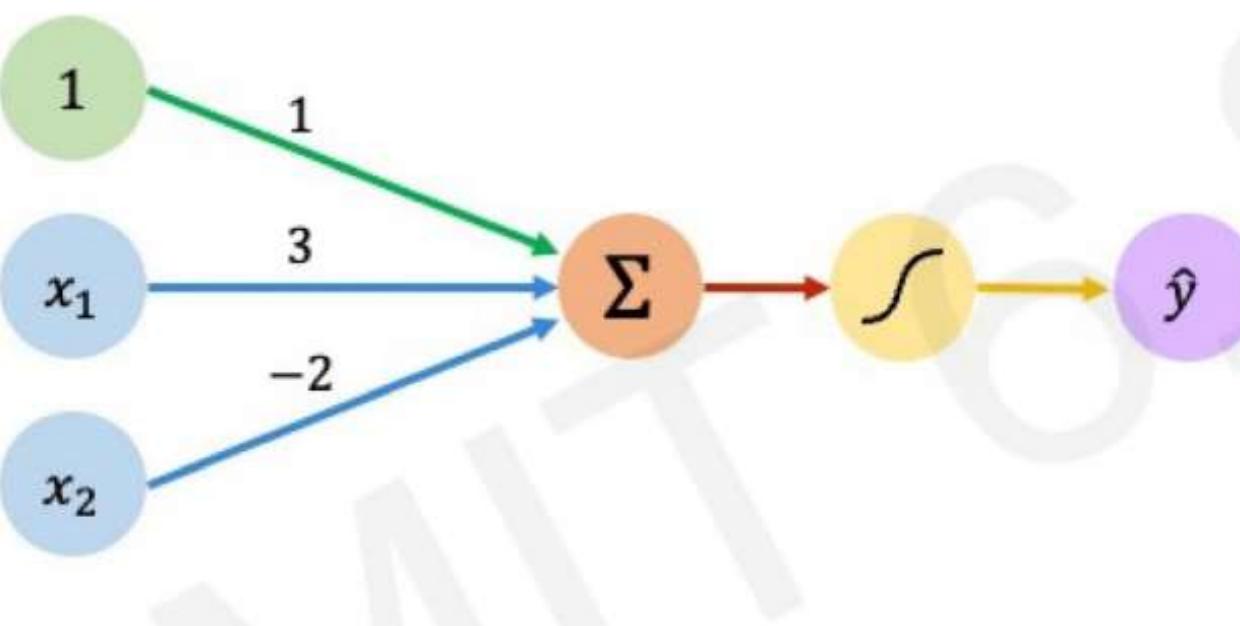
선형 활성화 함수는 신경망 크기에
관련 없이 선형 결정을 생성한다.



비선형성은 임의로 복잡한 함수에
근사할 수 있게 해준다.

PERCEPTRON 02

퍼셉트론(초기 형태 인공신경망)



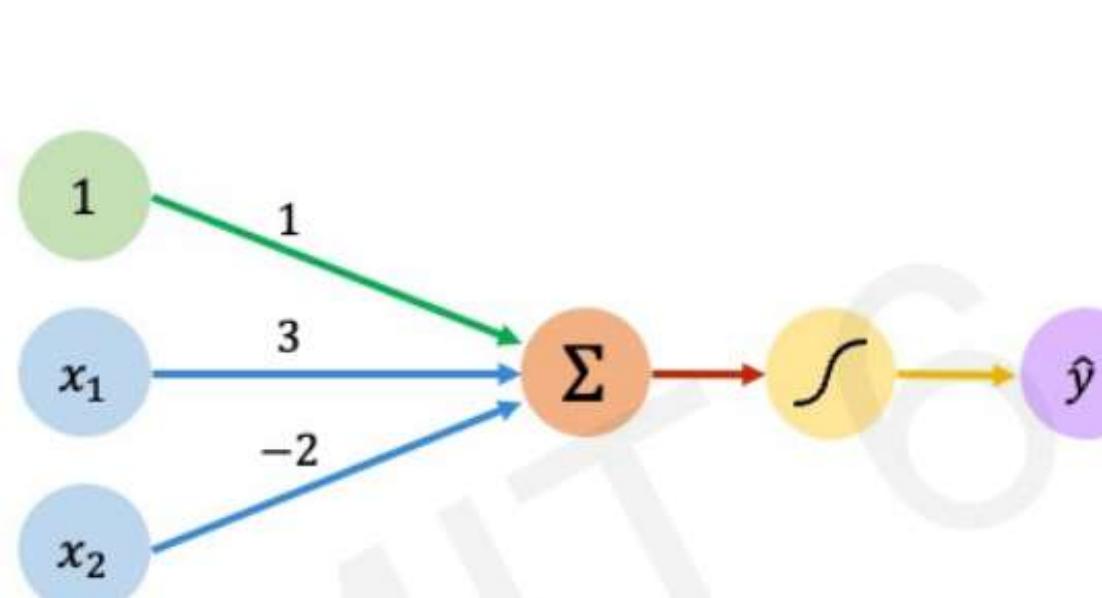
We have: $w_0 = 1$ and $\mathbf{w} = \begin{bmatrix} 3 \\ -2 \end{bmatrix}$

$$\begin{aligned}\hat{y} &= g(w_0 + \mathbf{x}^T \mathbf{w}) \\ &= g\left(1 + \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^T \begin{bmatrix} 3 \\ -2 \end{bmatrix}\right) \\ \hat{y} &= g\left(1 + 3x_1 - 2x_2\right)\end{aligned}$$

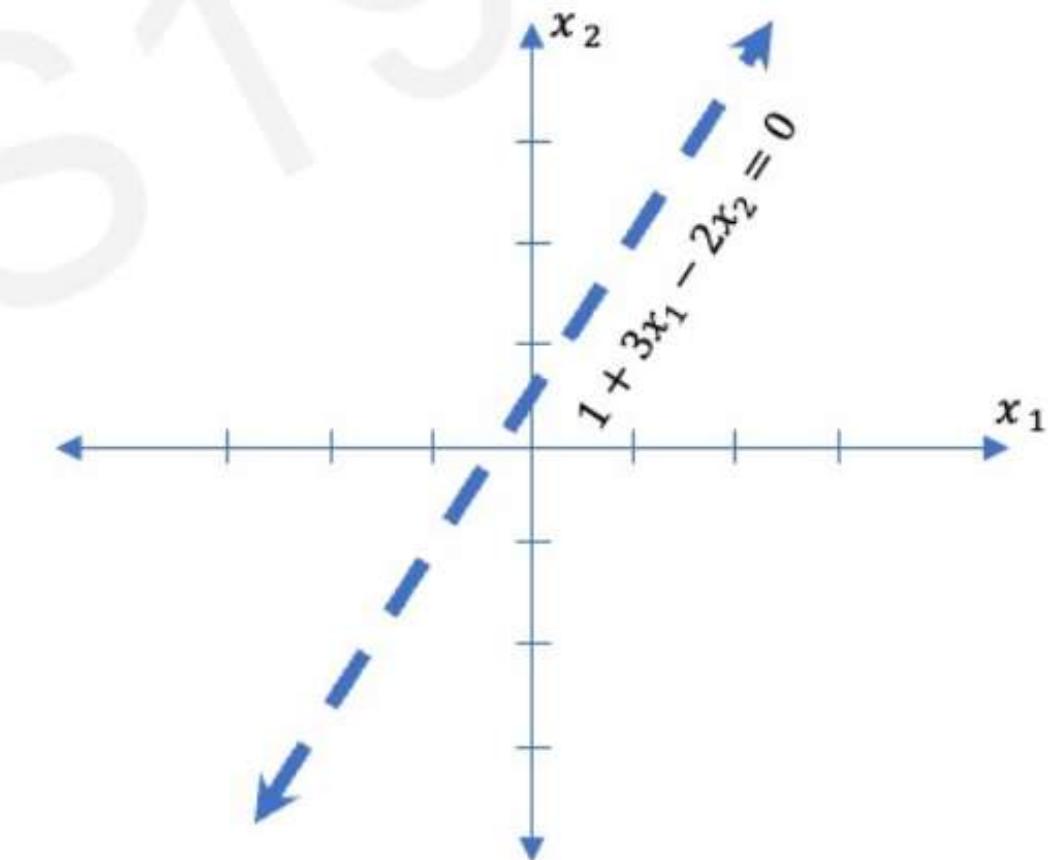
This is just a line in 2D!

PERCEPTRON 02

퍼셉트론(초기 형태 인공신경망)

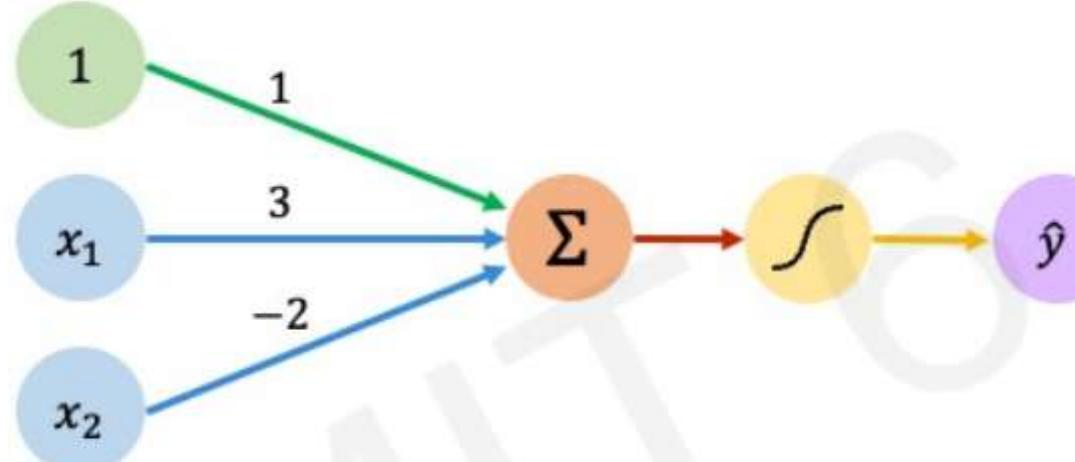


$$\hat{y} = g(1 + 3x_1 - 2x_2)$$



PERCEPTRON 02

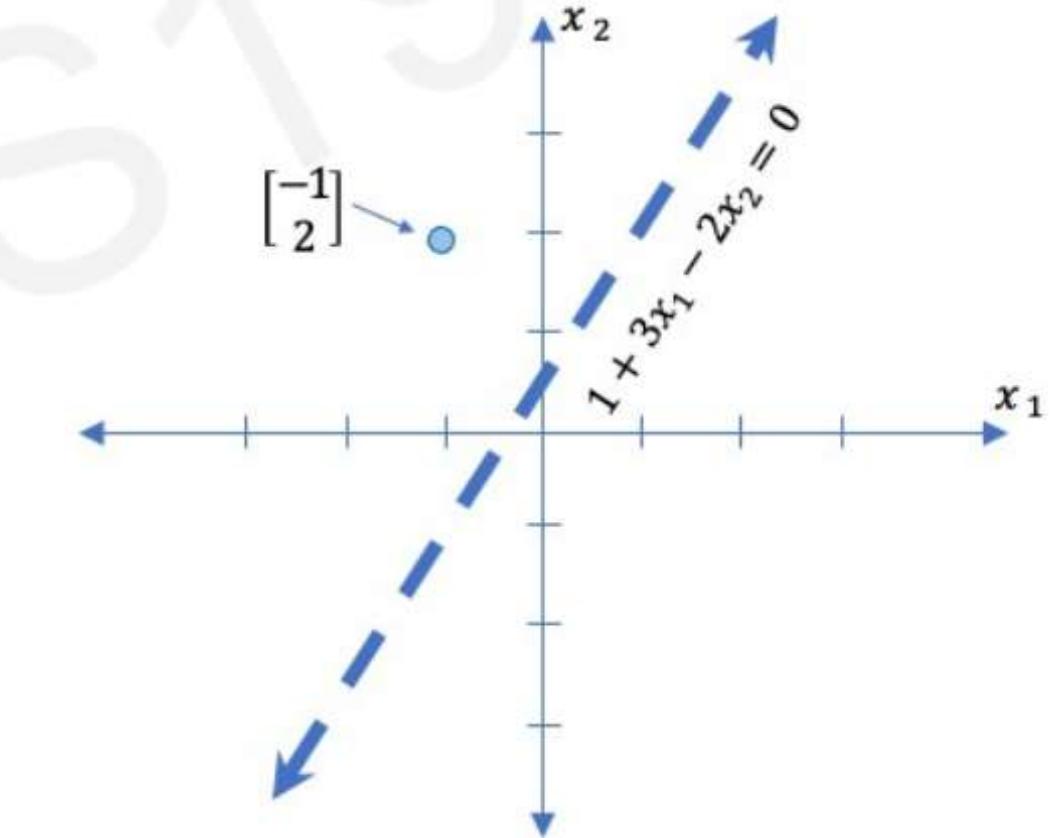
퍼셉트론(초기 형태 인공신경망)



Assume we have input: $\mathbf{x} = \begin{bmatrix} -1 \\ 2 \end{bmatrix}$

$$\begin{aligned}\hat{y} &= g(1 + (3 * -1) - (2 * 2)) \\ &= g(-6) \approx 0.002\end{aligned}$$

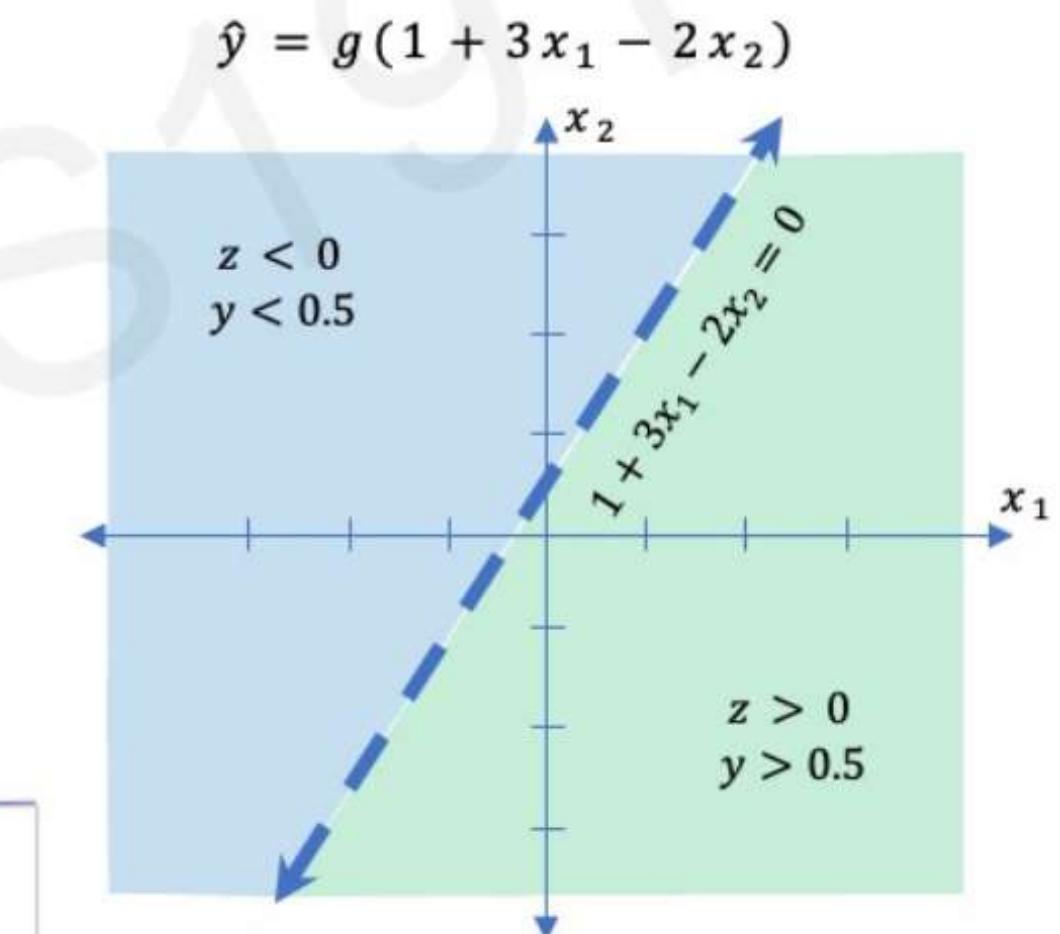
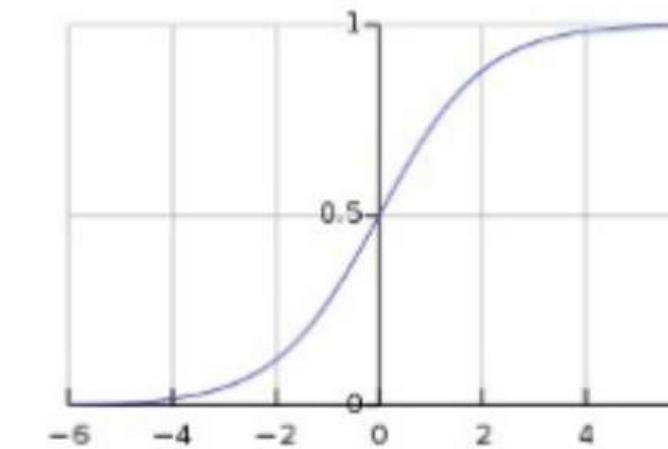
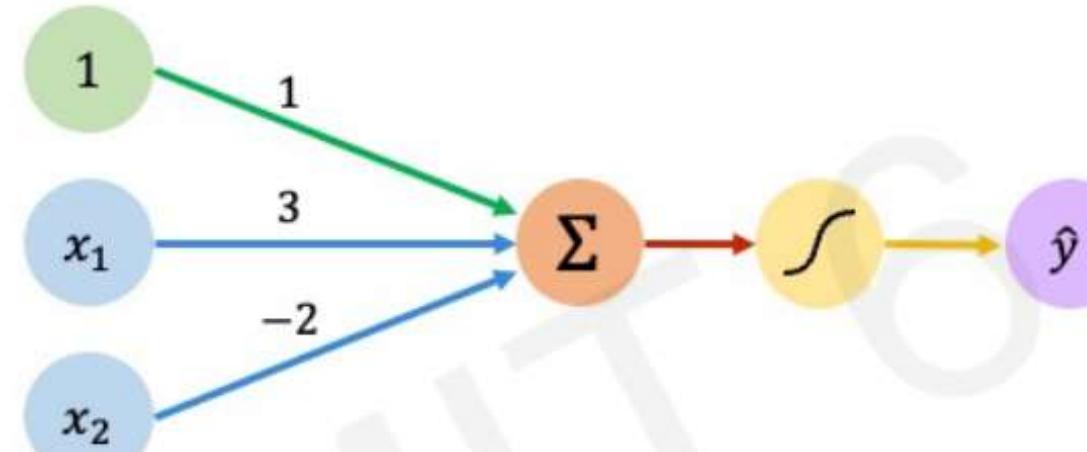
$$\hat{y} = g(1 + 3x_1 - 2x_2)$$



모델에 입력되는 모든 값에 대해 이 선과 관련해서 어디에 위치하는지 볼 수 있음

PERCEPTRON 02

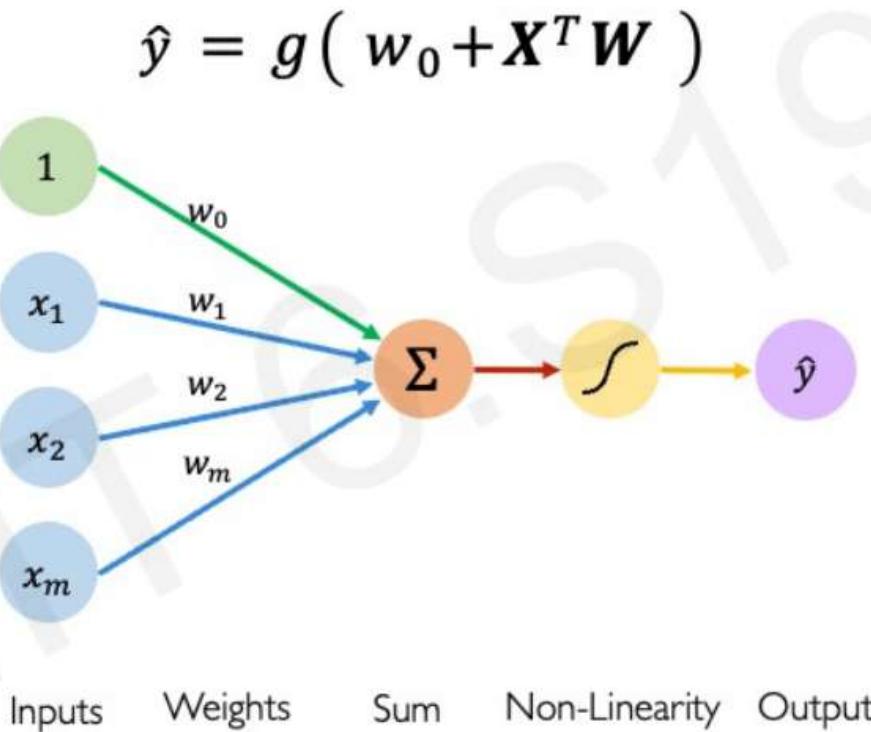
퍼셉트론(초기 형태 인공신경망)



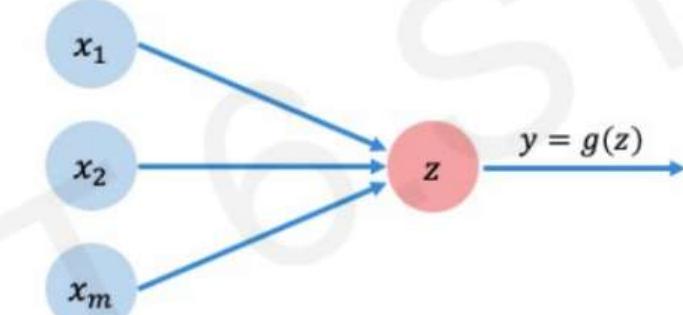
비선형 함수에서 입력된 변수가 0.5
양수면 0.5보다 크고 아니면 반대인 경
우인 것을 알 수 있음
→ 선은 두 지점을 분리한다.

03 인공신경망

파셉트론(초)의 인공신경망



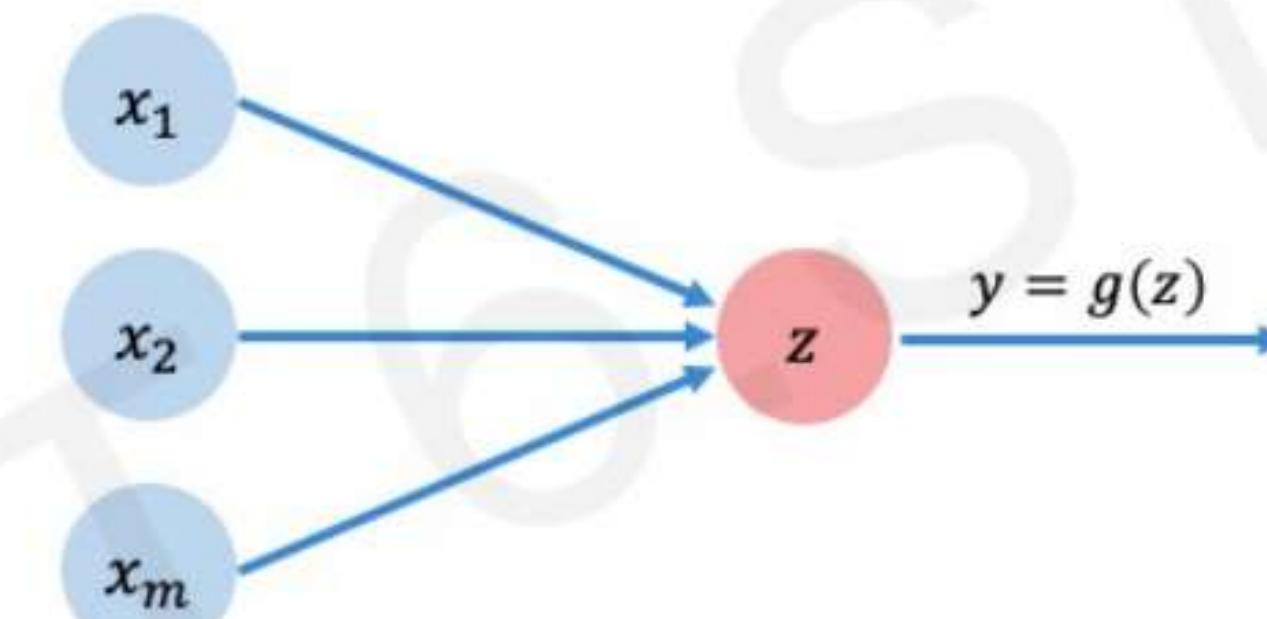
단순화



$$z = w_0 + \sum_{j=1}^m x_j w_j$$

03 인공신경망

파셉트론(초기)의 인공신경망

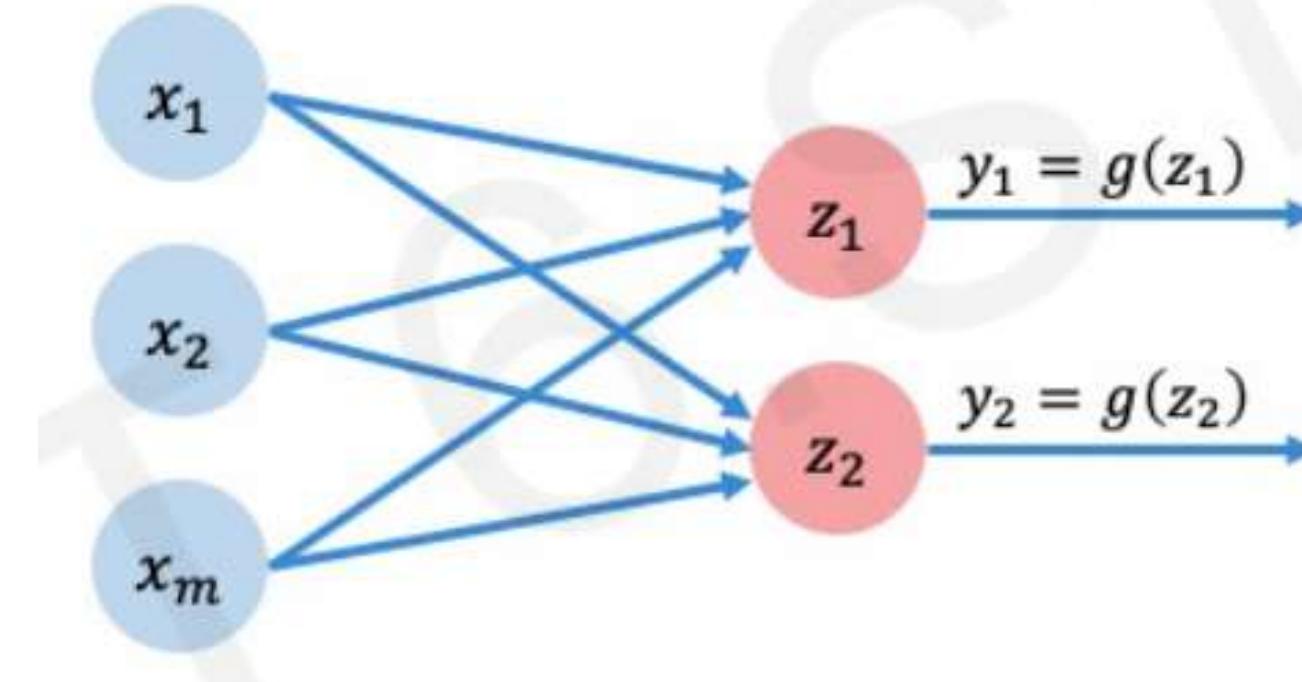


$$z = w_0 + \sum_{j=1}^m x_j w_j$$

03 인공신경망

퍼셉트론(초기)의 인공신경망

다중 출력 신경망



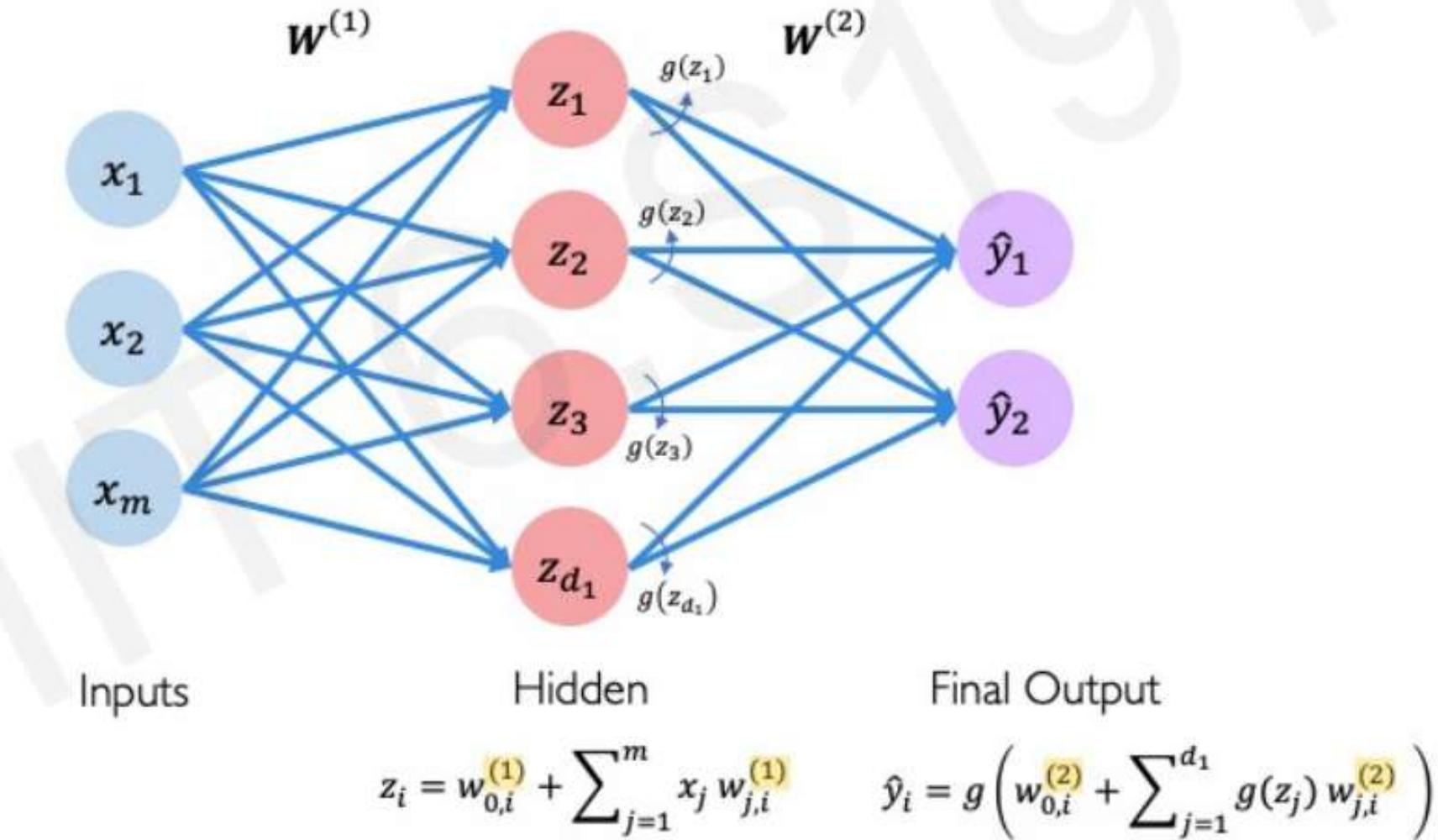
$$z_i = w_{0,i} + \sum_{j=1}^m x_j w_{j,i}$$

i의 의미는 입력은 같지만 Z_1 와 Z_2 에 각자 다른 가중치를 할당하여 두 개의 다른 출력 값을 만든다.
-> 이러한 모든 입력이 모든 출력과 연결되어있는 층을 dense layer라고 부르며 이는 선형 layer이다.

03 인공신경망

퍼셉트론(초)의 인공신경망

단일 계층 신경망 (Single Layer Neural Network or Artificial Neural Network)

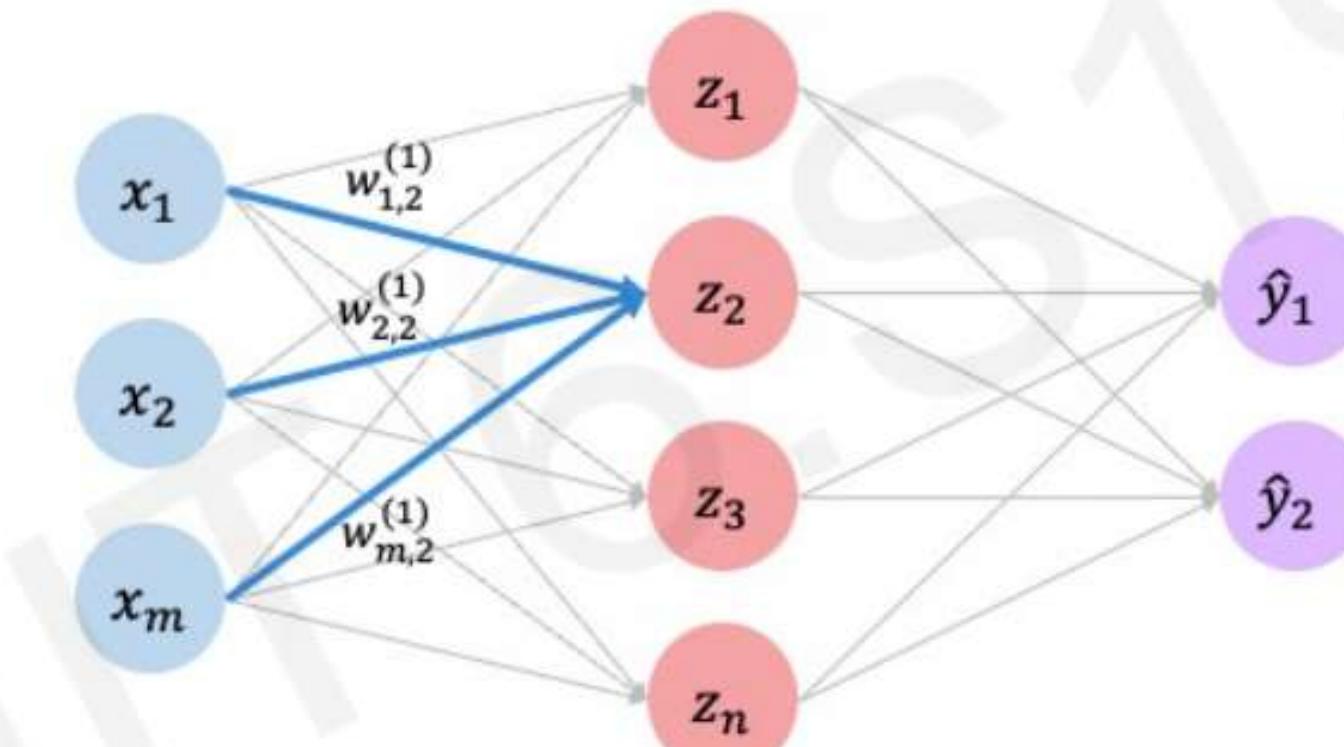


단일 은닉층을 가지며 이는 두 개의 가중치 행렬이 필요하다는 것을 의미한다.
즉 W 라는 벡터가 두 개 있어야 한다는 뜻.

03 인공신경망

퍼셉트론(초)의 인공신경망

단일 계층 신경망 (Single Layer Neural Network or Artificial Neural Network)



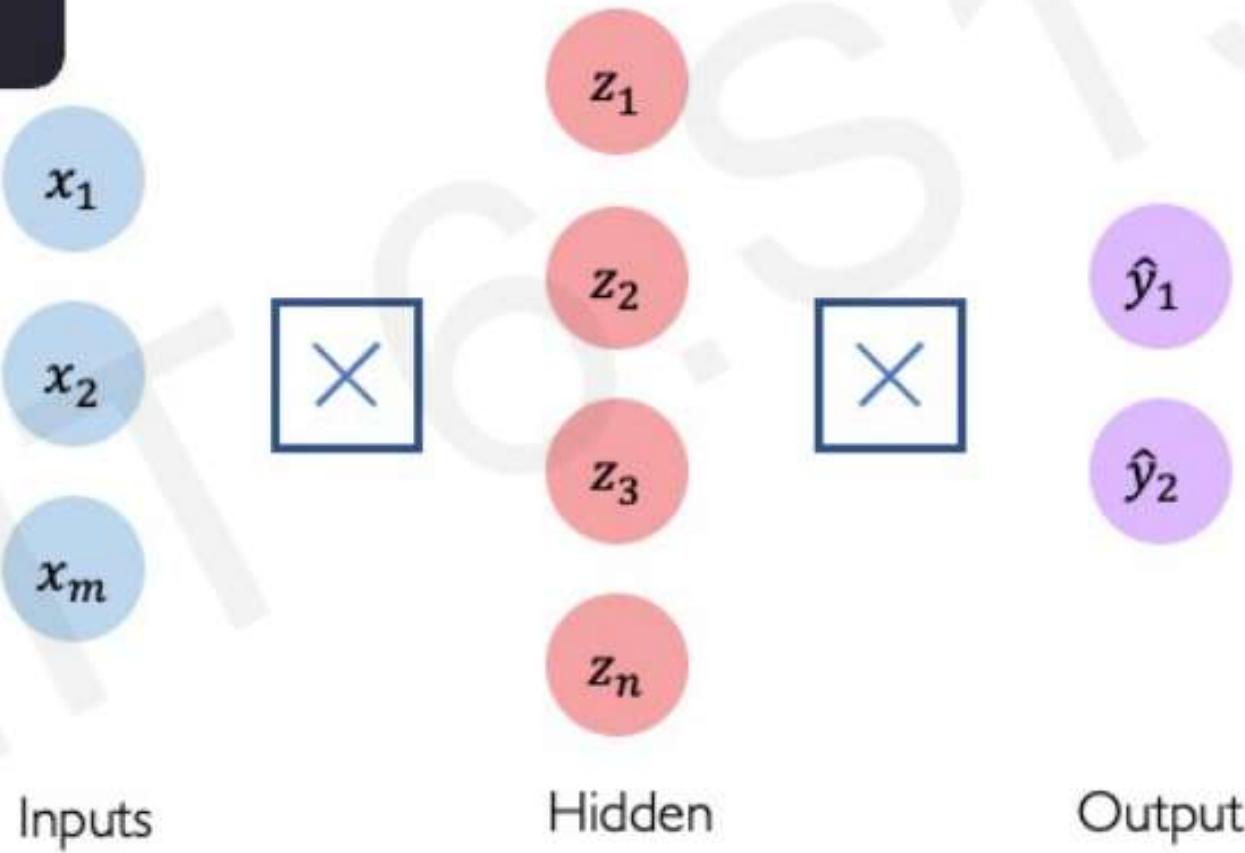
$$\begin{aligned} z_2 &= w_{0,2}^{(1)} + \sum_{j=1}^m x_j w_{j,2}^{(1)} \\ &= w_{0,2}^{(1)} + x_1 w_{1,2}^{(1)} + x_2 w_{2,2}^{(1)} + x_m w_{m,2}^{(1)} \end{aligned}$$

03 인공신경망

파셉트론(초)의 인공신경망

```
import tensorflow as tf   
  
model = tf.keras.Sequential([  
    tf.keras.layers.Dense(n),  
    tf.keras.layers.Dense(2)  
])
```

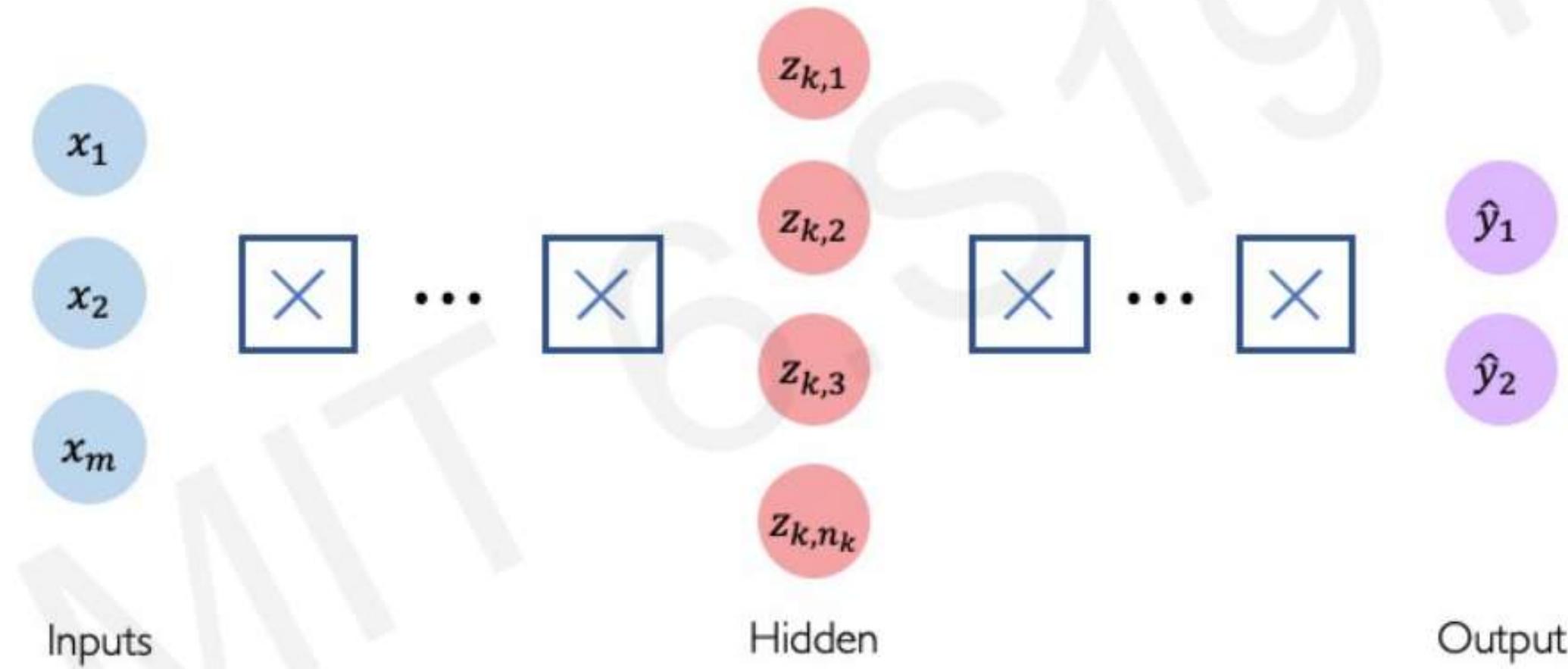
```
from torch import nn   
  
model = nn.Sequential(  
    nn.Linear(m, n),  
    nn.ReLU(),  
    nn.Linear(n, 2)  
)
```



03 인공신경망

파셉트론(초)의 인공신경망

Deep 신경망 (Deep Neural Network)



$$z_{k,i} = w_{0,i}^{(k)} + \sum_{j=1}^{n_{k-1}} g(z_{k-1,j}) w_{j,i}^{(k)}$$

선형 layer 부분의 개수를 늘리고 이에 따라 비선형 layer의 개수도 늘림

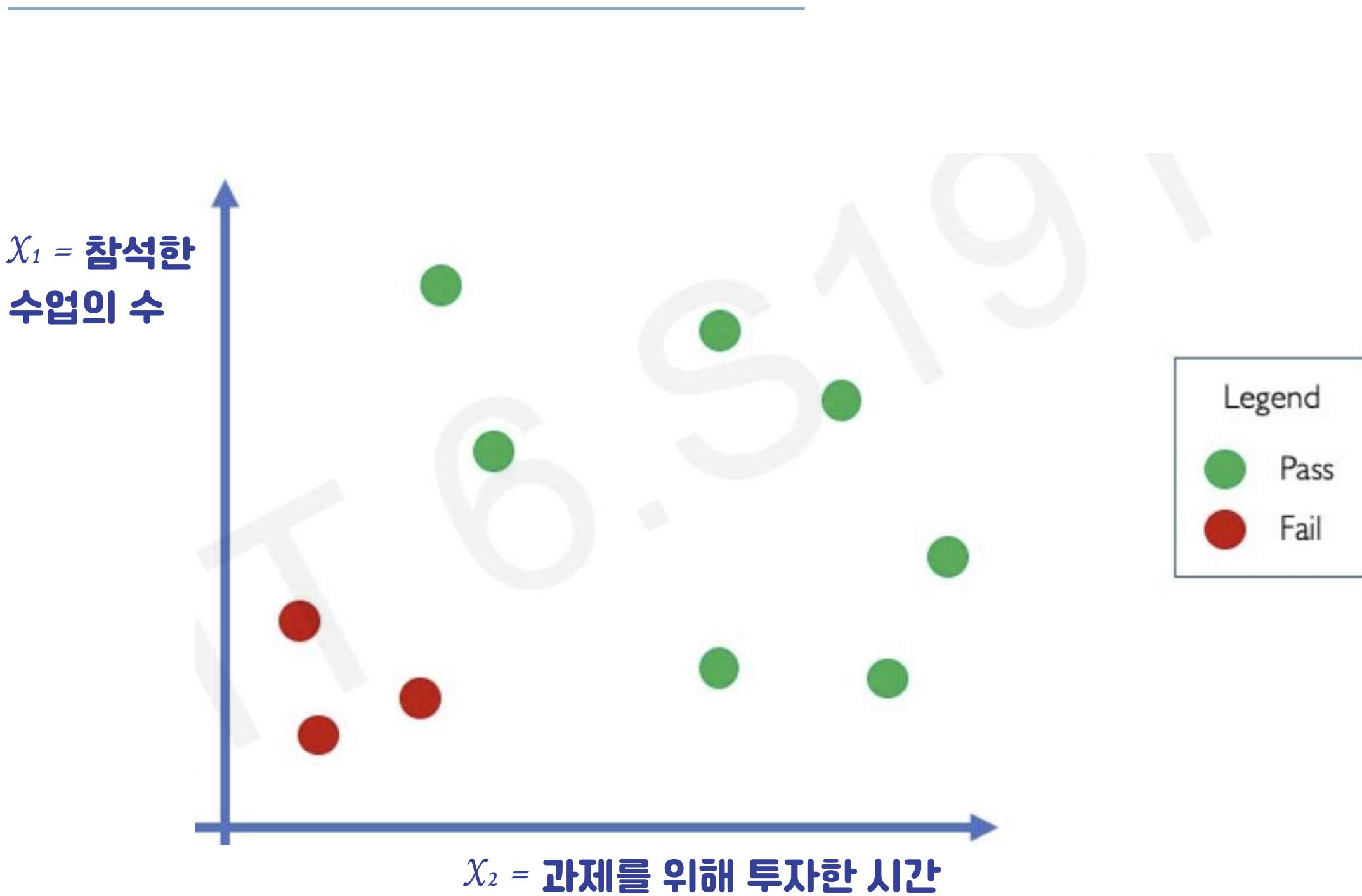
04 수설 | 초|적호

예시 : 시험 통과 할 수 있을까?

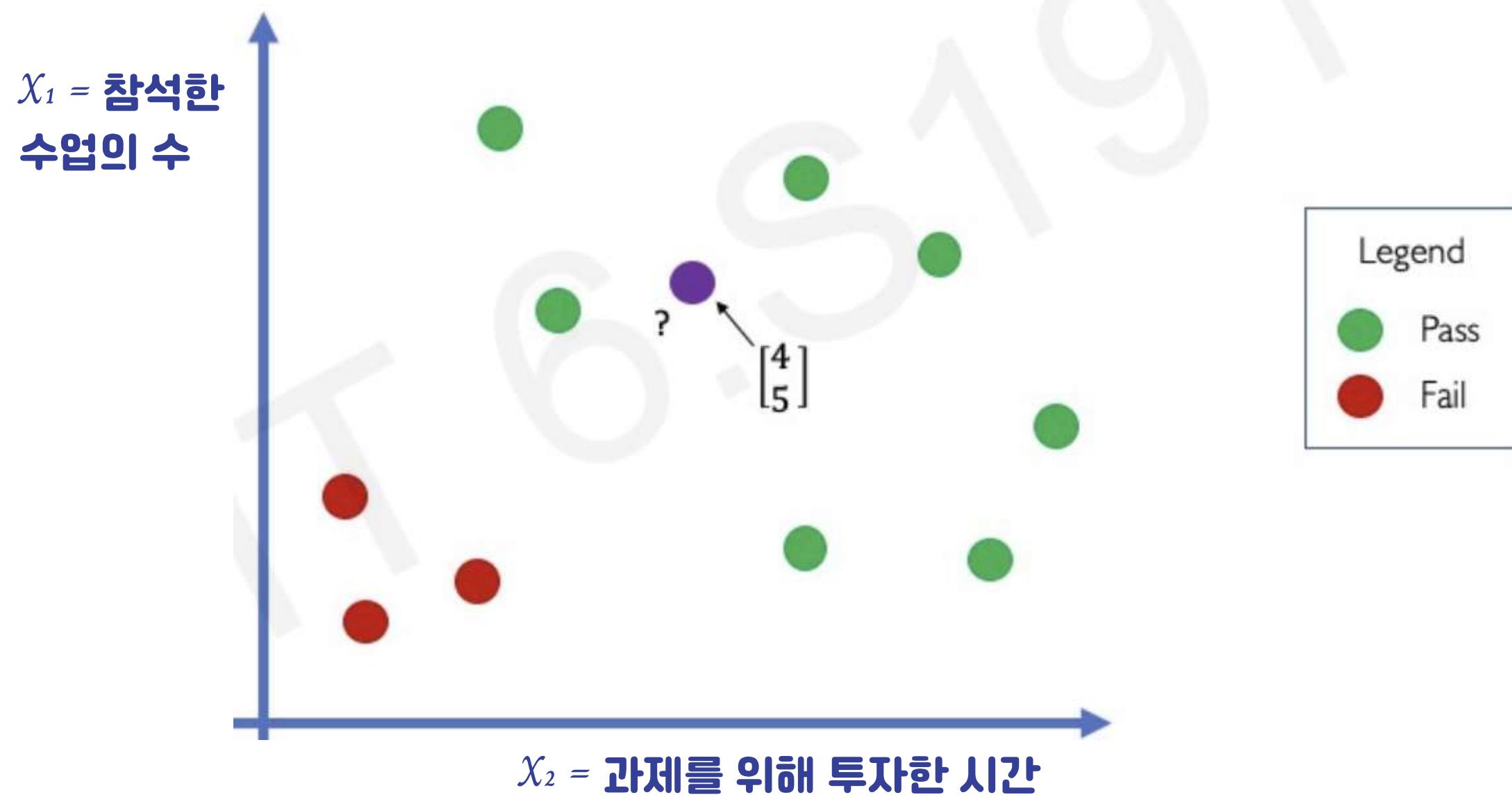
χ_1 = 참석한 수업의 수

χ_2 = 과제를 위해 투자한 시간

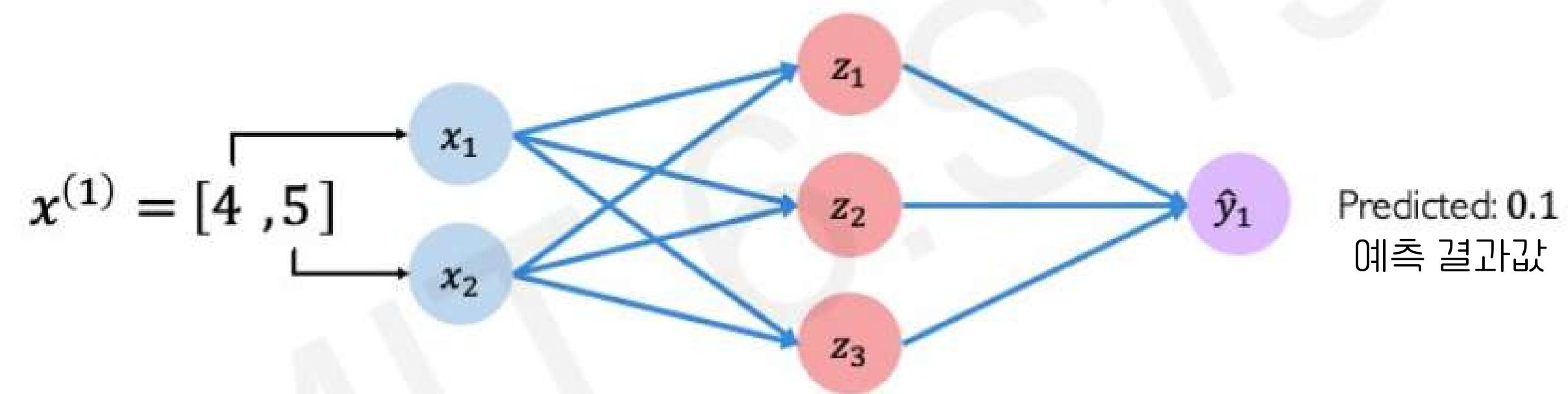
수설초|적호 04



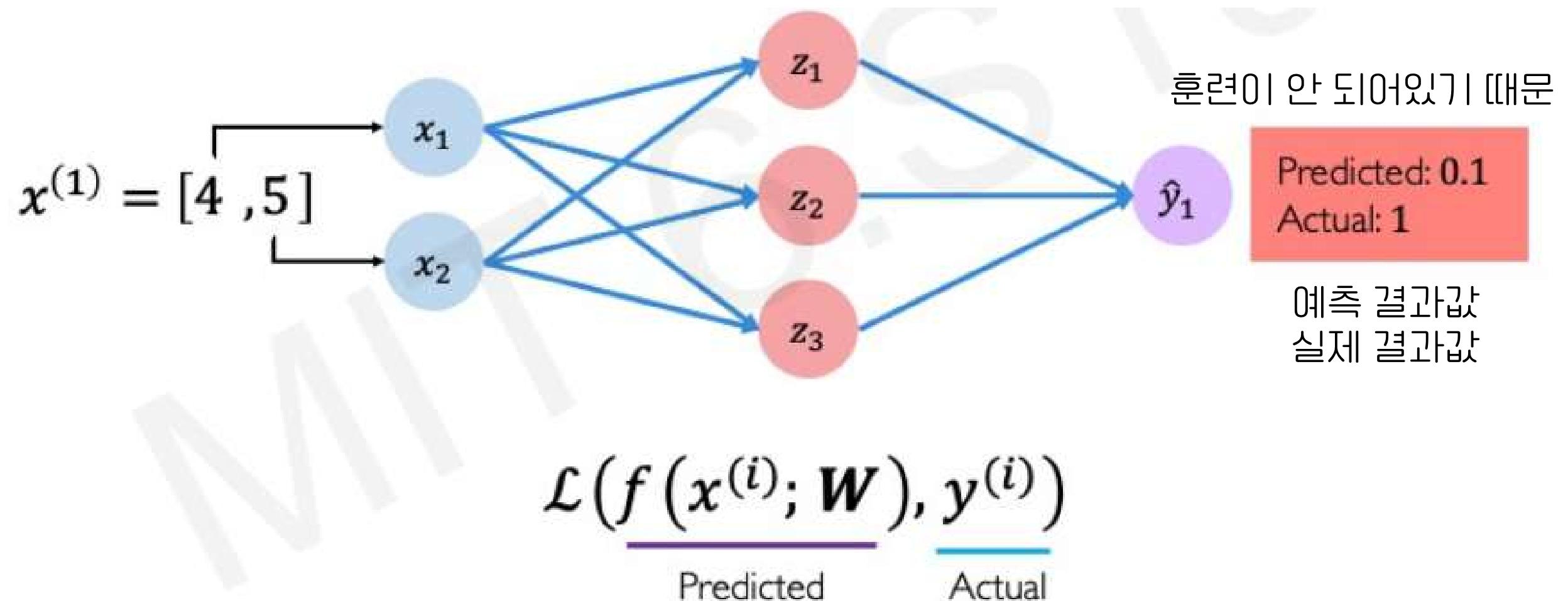
수설 초기화 04



04 | 주제: 손실함수



손실함수 04



학습을 위해 예측이 틀렸을 때에 예측이 얼마나 틀렸는지를 수치화 해야 하며 이를 손실이라고 한다.

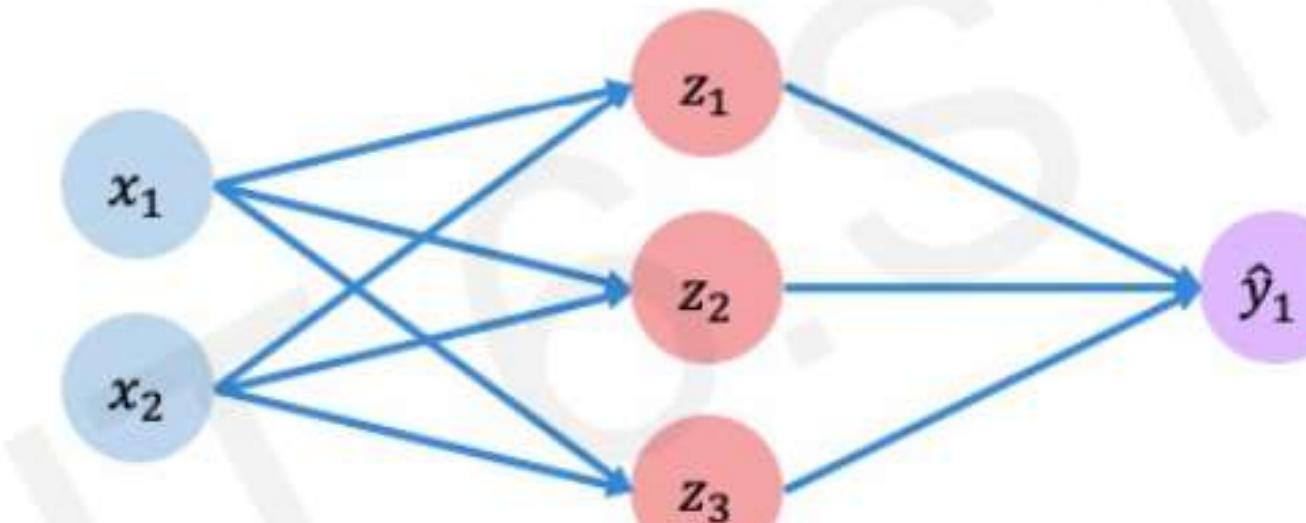
04 | 솔직한 말씀

손실의 최적화를 위해서는 평가지표가 필요함

각각의 데이터의 손실이 아닌 데이터셋 전체의 손실을 측정함

-> 이는 모든 데이터에 대한 손실의 평균

$$X = \begin{bmatrix} 4, & 5 \\ 2, & 1 \\ 5, & 8 \\ \vdots & \vdots \end{bmatrix}$$



$$J(\mathbf{W}) = \frac{1}{n} \sum_{i=1}^n \mathcal{L}(f(x^{(i)}; \mathbf{W}), y^{(i)})$$

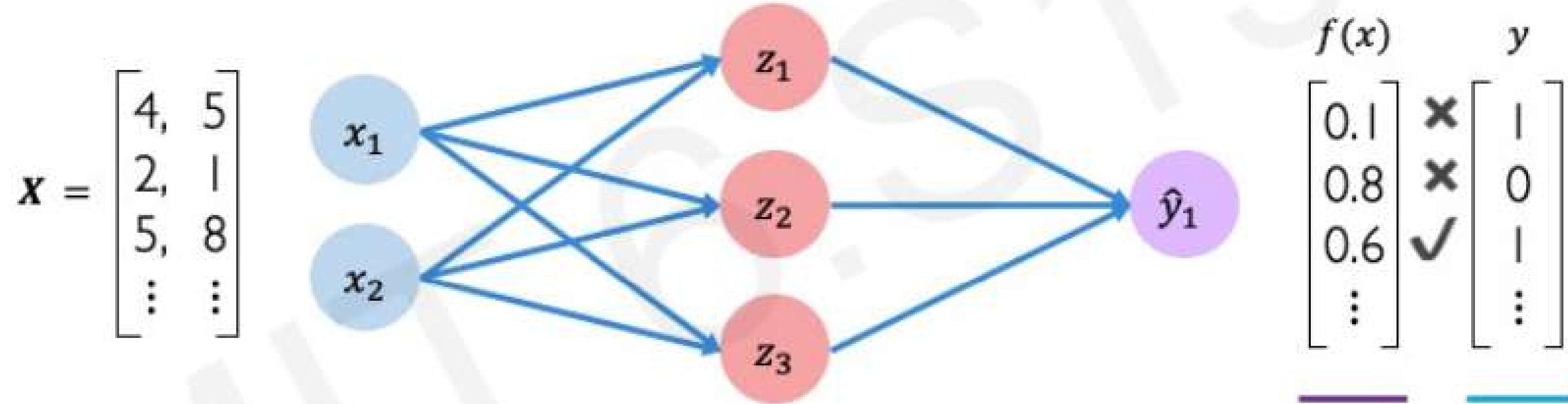
Predicted Actual

- Also known as:
 - Objective function
 - Cost function
 - Empirical Risk

- 같은 말
- 목적함수
- 비용함수
- 경험적 리스
- + 손실함수

04 손실함수

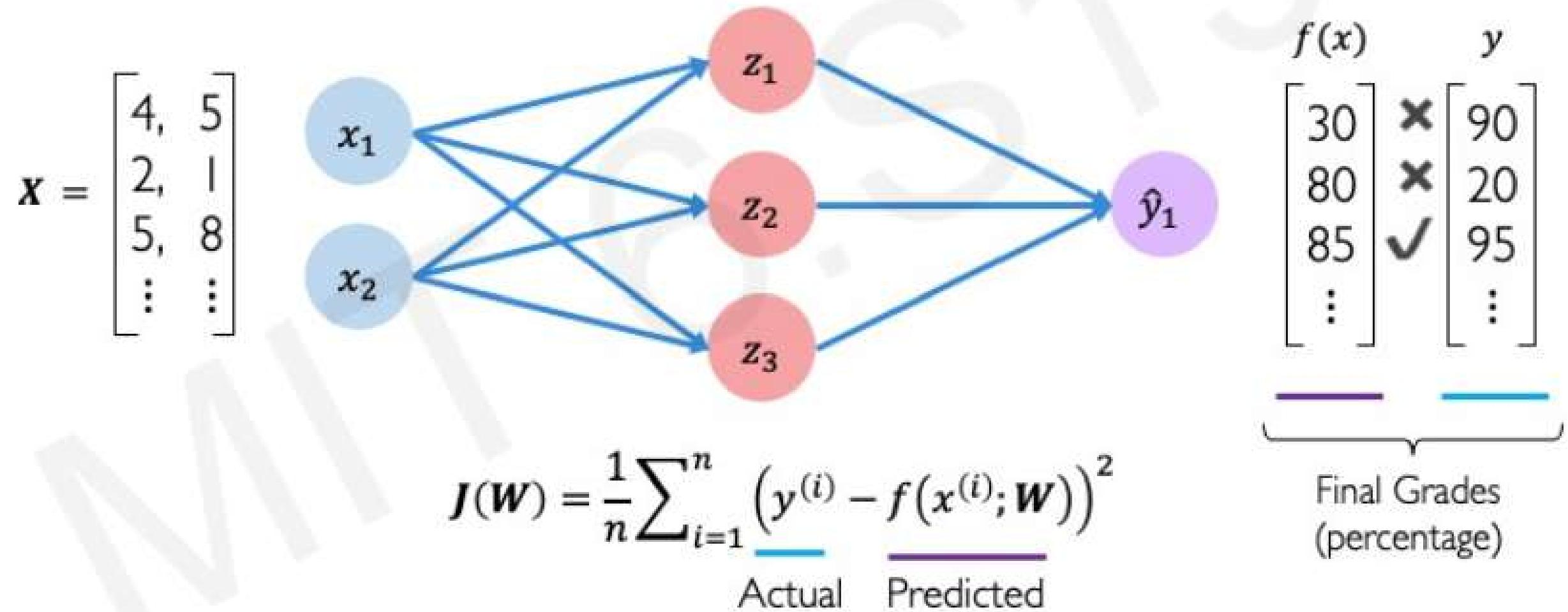
이진 교차 엔트로피 손실은 모델의 예측 확률을 나타내어 얼마나 실제값과 예측값이 일치하는지 평가하는 평가지표이다.



$$J(\mathbf{W}) = -\frac{1}{n} \sum_{i=1}^n \underbrace{y^{(i)} \log(f(\mathbf{x}^{(i)}; \mathbf{W}))}_{\text{Actual}} + \underbrace{(1 - y^{(i)}) \log(1 - f(\mathbf{x}^{(i)}; \mathbf{W}))}_{\text{Actual}} \quad \underbrace{\text{Predicted}}_{\text{Predicted}}$$

04 손실함수

최적화를 위한 손실에는 유형이 있고 목적에 따라 손실을 바꿔야 한다.
예시: 평균 제곱 오차 손실은 연속 실수를 출력하는 회귀 모델과 함께 사용



MSE → 예측등급과 실제 등급을 빼고 제곱하여 차이를 측정
이와 같은 손실의 유형은 MSE와 범주적 이산 손실 등이 있음

04 손실의 최적화

손실의 최적화

$$\mathbf{W}^* = \operatorname{argmin}_{\mathbf{W}} \frac{1}{n} \sum_{i=1}^n \mathcal{L}(f(\mathbf{x}^{(i)}; \mathbf{W}), y^{(i)})$$

$$\mathbf{W}^* = \operatorname{argmin}_{\mathbf{W}} J(\mathbf{W})$$

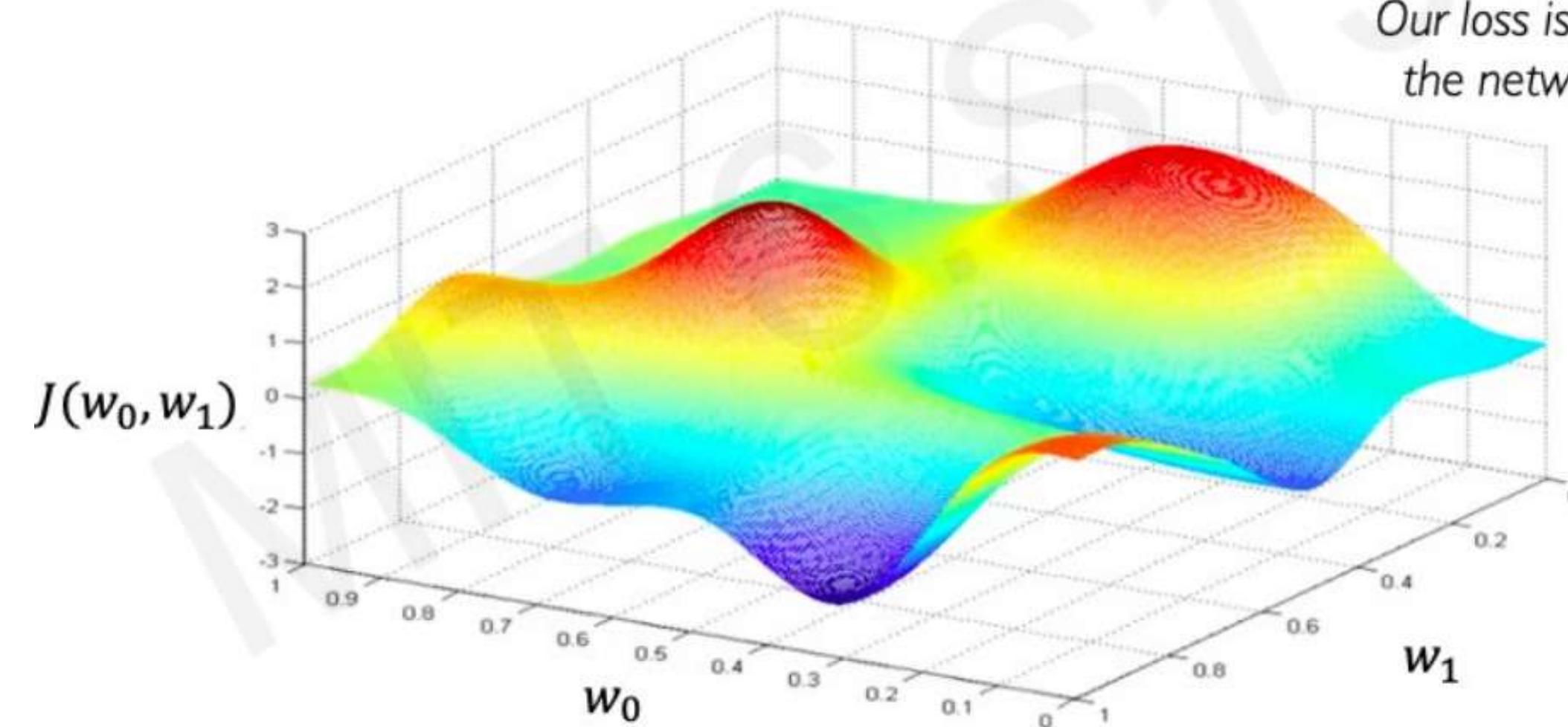
$J(\mathbf{W})$ (손실함수)의 결과값을 최소로 하는 \mathbf{W} 를 찾는 방정식

Remember:
 $\mathbf{W} = \{\mathbf{W}^{(0)}, \mathbf{W}^{(1)}, \dots\}$

04 손실함수

경사 하강법

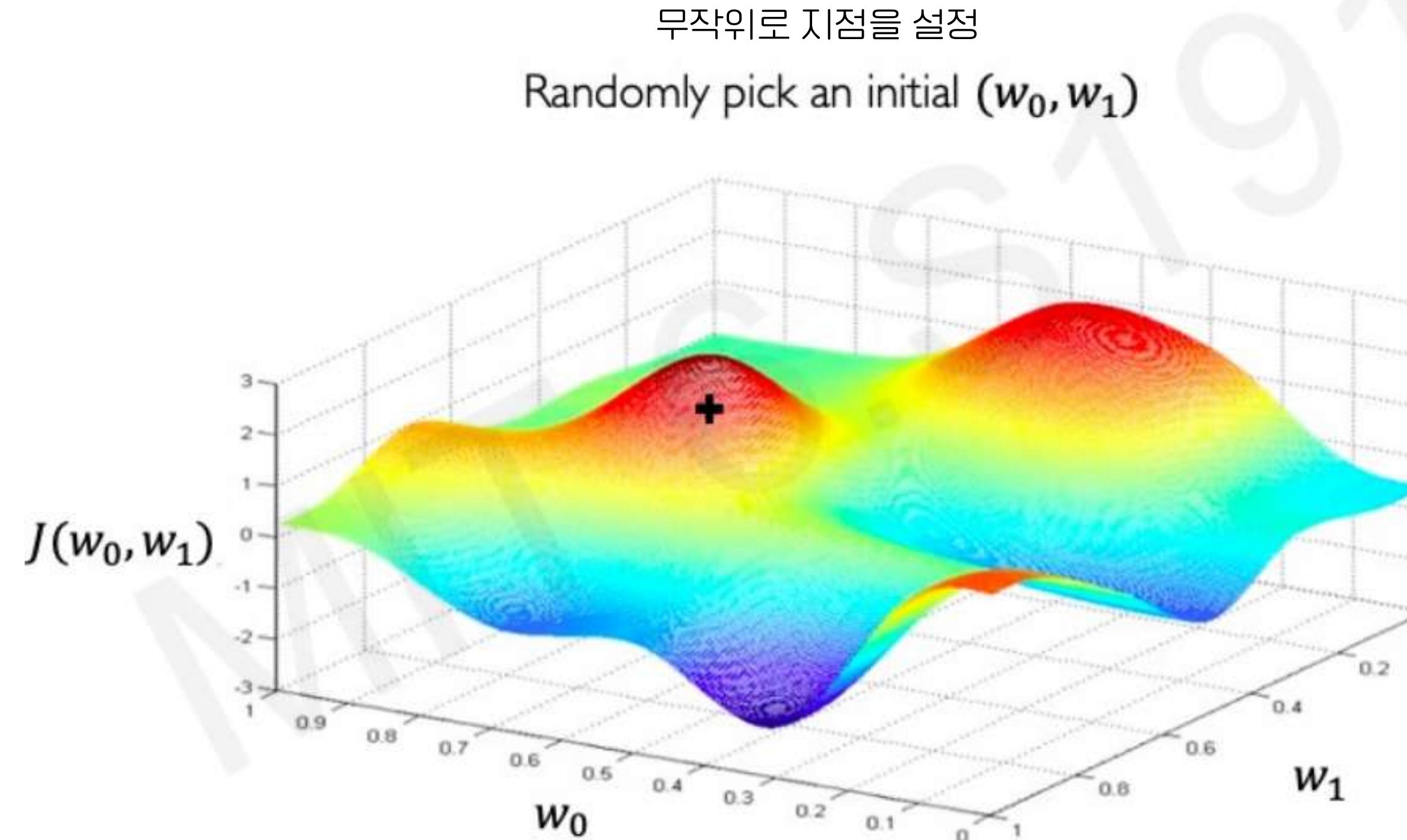
$$\mathbf{W}^* = \underset{\mathbf{W}}{\operatorname{argmin}} J(\mathbf{W})$$



손실은 신경망의 가중치의 함수

Remember:
Our loss is a function of
the network weights!

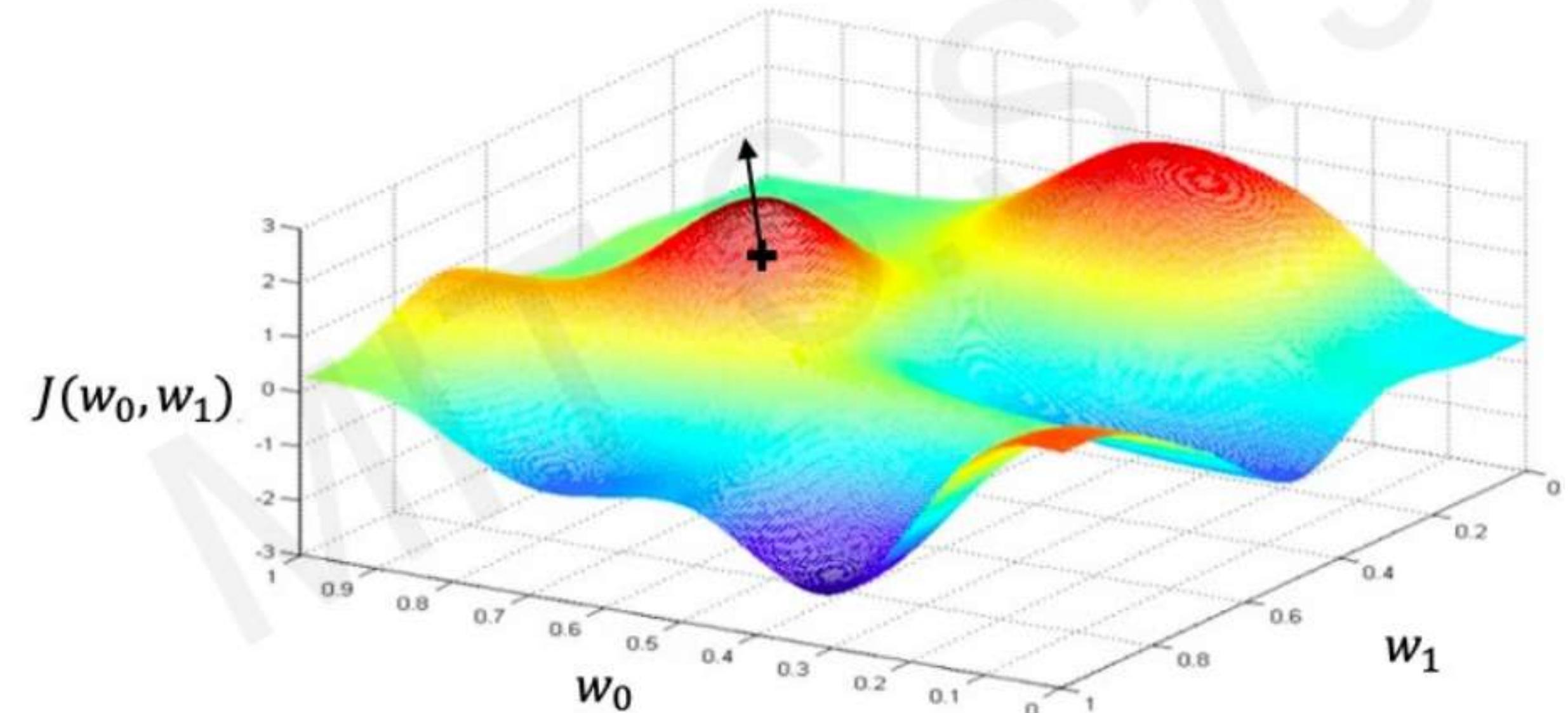
04 손실함수



04 손실함수

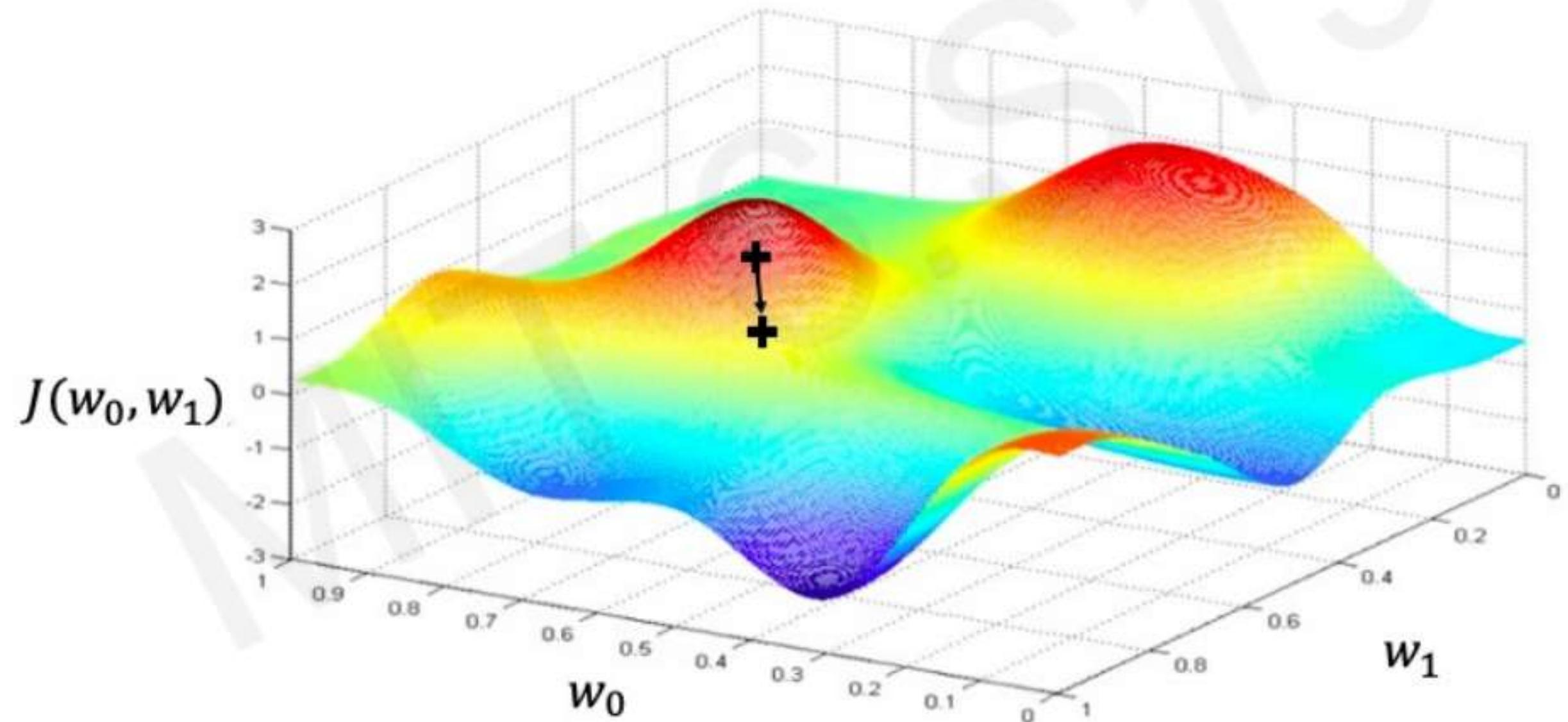
기울기를 구함
기울기는 전체 중에서 어느 쪽이 위쪽인지를 나타냄

Compute gradient, $\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}}$



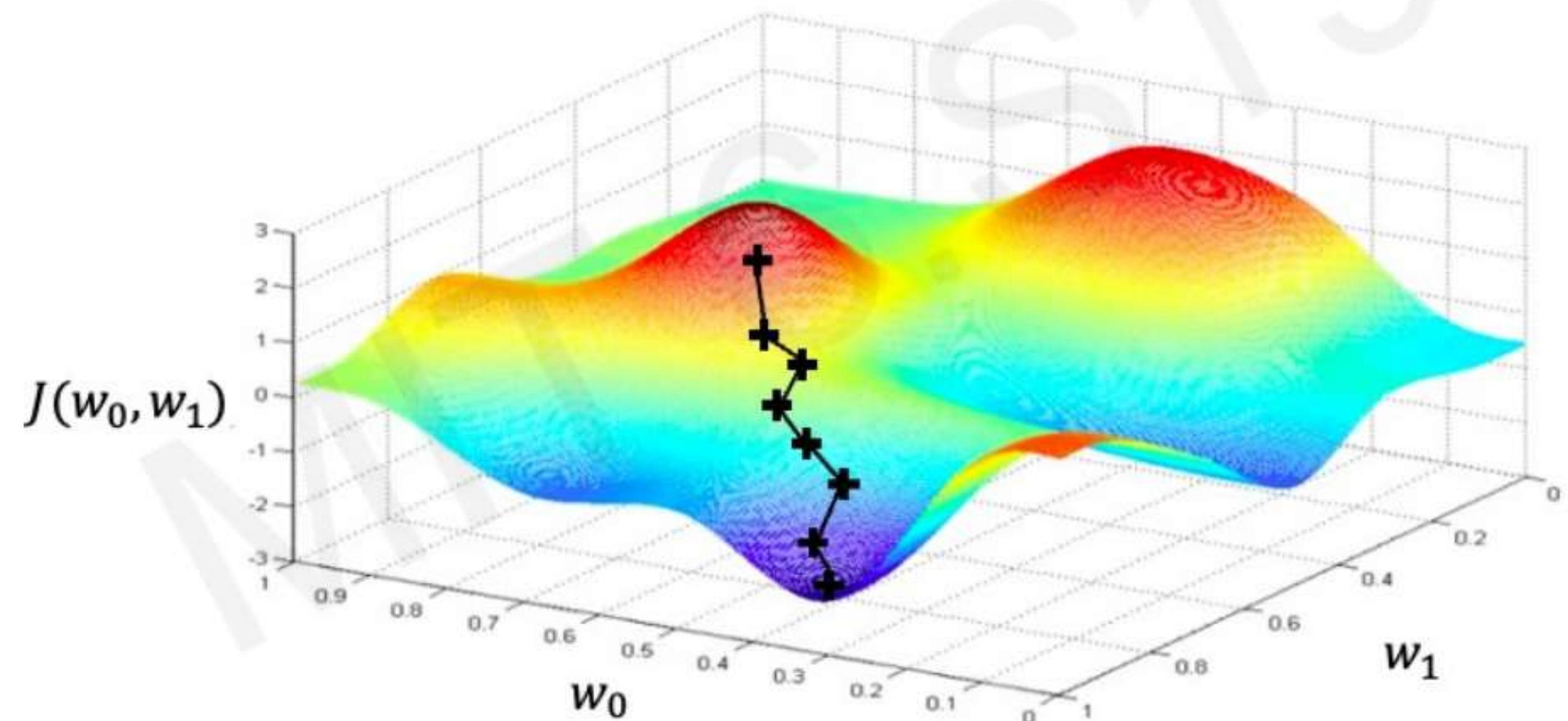
수설 초|적회 04

위쪽 방향의 반대로 이동



04 손실함수

최적의 값에 수렴할 때 까지 반복



04

선형회귀 최적화

경사 하강법 알고리즘 의사코드(파이썬)

Algorithm

1. Initialize weights randomly $\sim \mathcal{N}(0, \sigma^2)$ 무작위로 가중치 초기화
2. Loop until convergence:
수렴할 때 까지 반복:
 기울기 측정(역전파)
 가중치 업데이트
3. Compute gradient, $\frac{\partial J(\mathbf{W})}{\partial \mathbf{W}}$ 해당 가중치의 기울기
4. Update weights, $\mathbf{W} \leftarrow \mathbf{W} - \eta \frac{\partial J(\mathbf{W})}{\partial \mathbf{W}}$
5. Return weights

04 손실함수

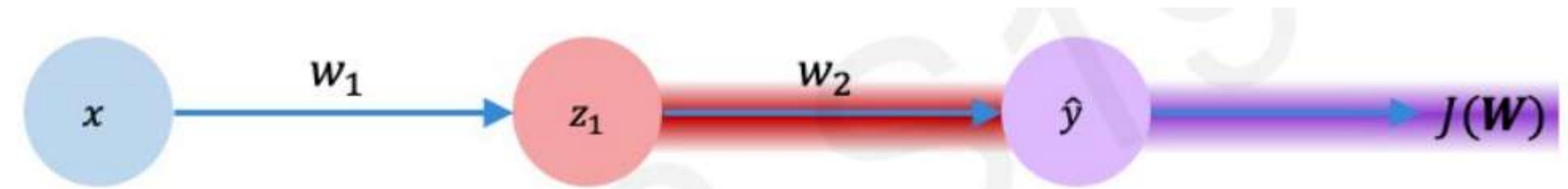
역전파



가중치의(예시 : w_2) 작은 변화가 최종 손실 $J(W)$ 에 어떤 영향을 미칠까?
→ 연쇄법칙

04 손실함수

역전파

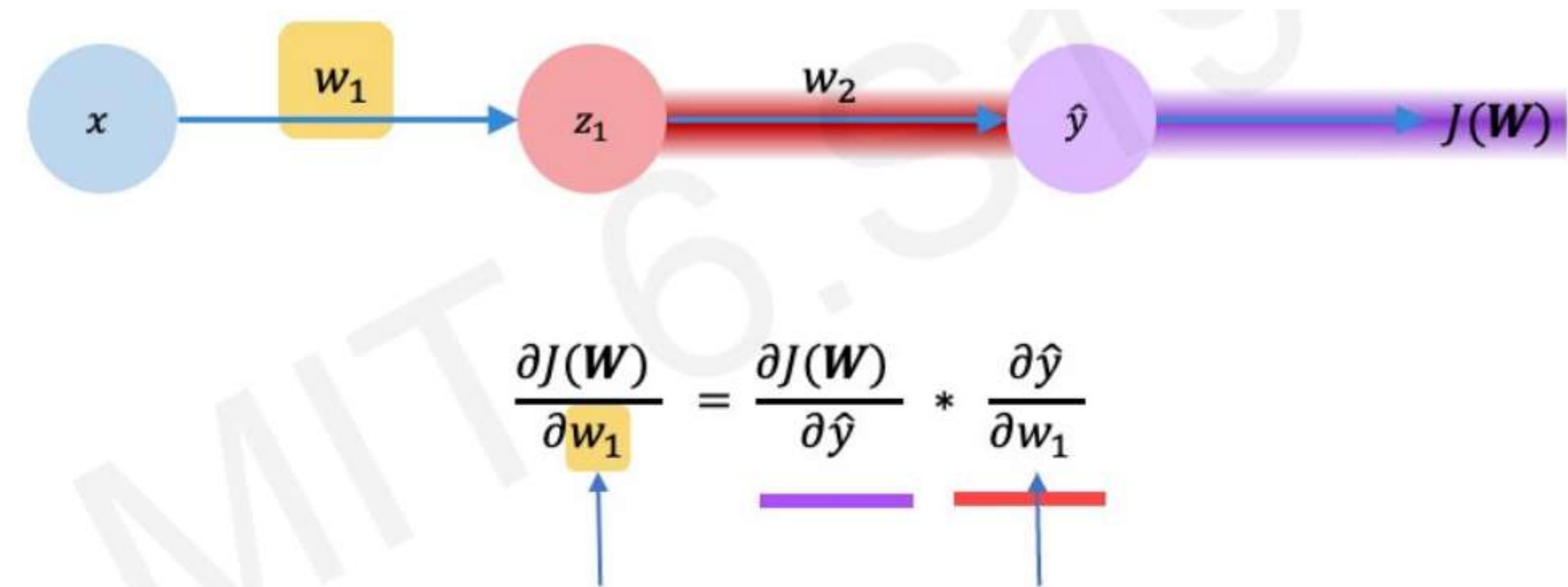


$$\frac{\partial J(\mathbf{W})}{\partial w_2} = \underline{\frac{\partial J(\mathbf{W})}{\partial \hat{y}}} * \overline{\frac{\partial \hat{y}}{\partial w_2}}$$

기울기 계산을 위해 연쇄 법칙을 적용하여 분해
분해하는 이유는 Y가 이전 값에 따라 좌우되기 때문

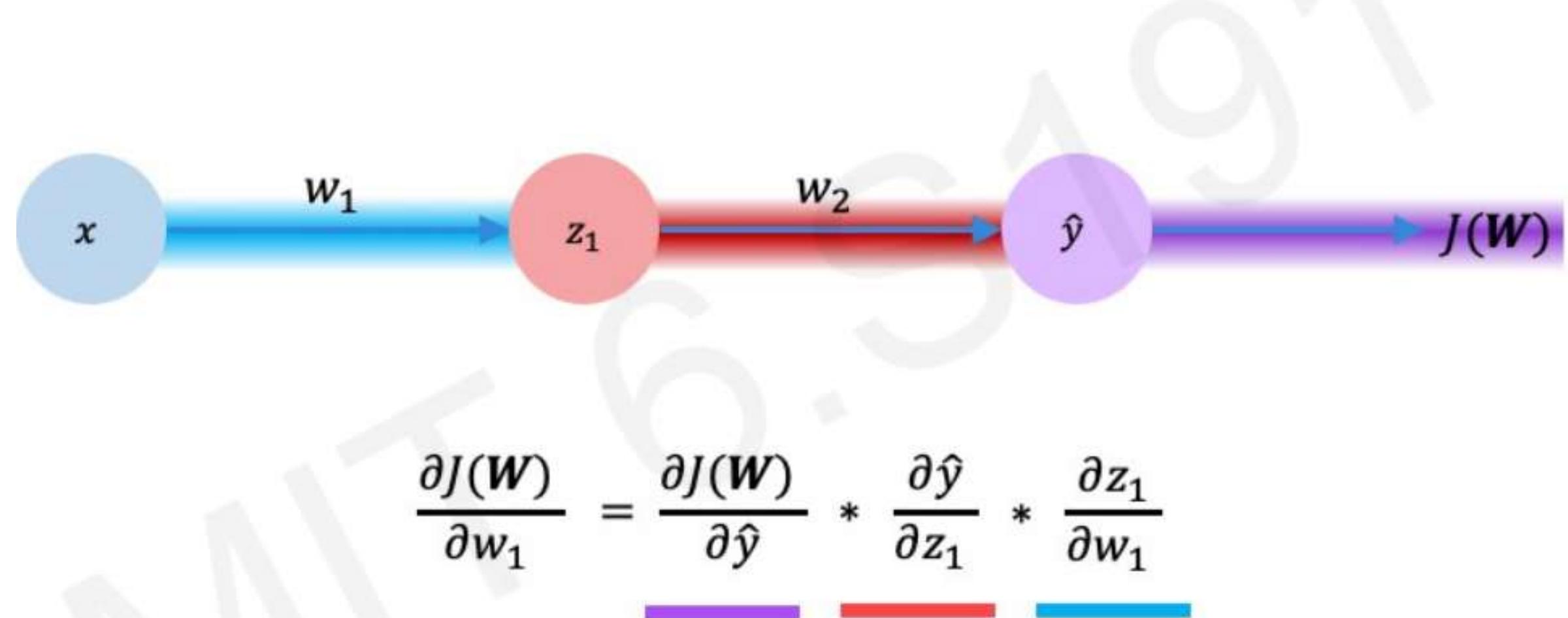
04 손실함수

역전파



04 손실함수

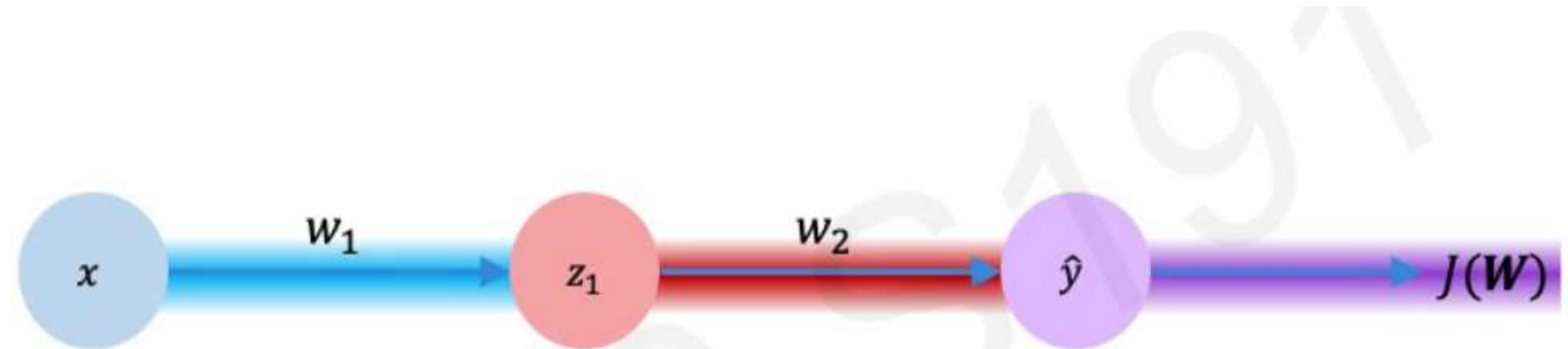
역전파



이전 식에서 w_{10} 이 정의되지 않았기 때문에 한번 더 적용

04 손실함수

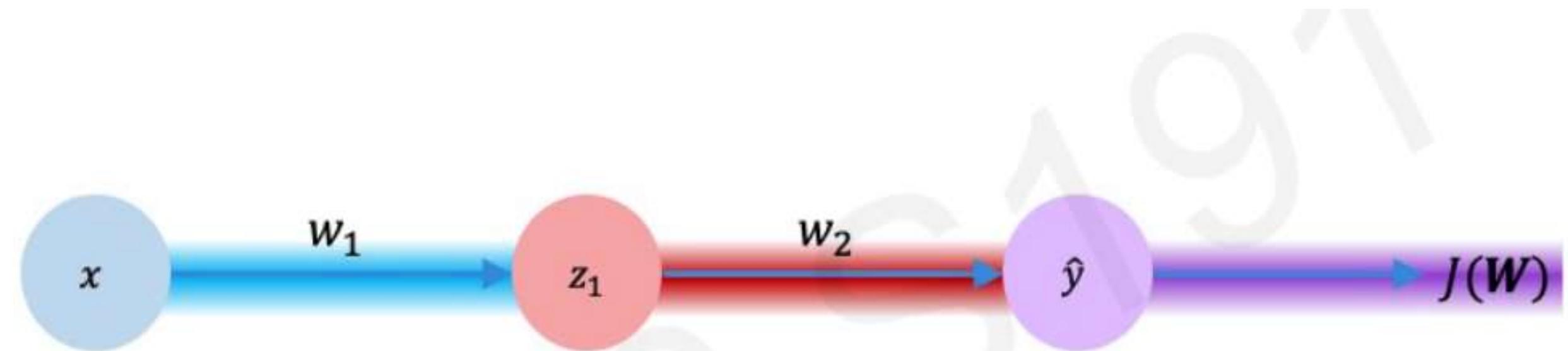
역전파



$$\frac{\partial J(\mathbf{W})}{\partial w_1} = \underbrace{\frac{\partial J(\mathbf{W})}{\partial \hat{y}}}_{\text{purple}} * \underbrace{\frac{\partial \hat{y}}{\partial z_1}}_{\text{red}} * \underbrace{\frac{\partial z_1}{\partial w_1}}_{\text{blue}}$$

출력에서 입력까지 이전 layer의 기울기를 사용하여 신경망의 모든 가중치에 대해 반복하여 이전 기울기들을 구할 수 있음

손실 최적화 04



$$\frac{\partial J(\mathbf{W})}{\partial w_1} = \underline{\frac{\partial J(\mathbf{W})}{\partial \hat{y}}} * \underline{\frac{\partial \hat{y}}{\partial z_1}} * \underline{\frac{\partial z_1}{\partial w_1}}$$

최적화를 위해 가중치를 늘리거나 경사의 반대 방향으로 가면 손실이 줄음

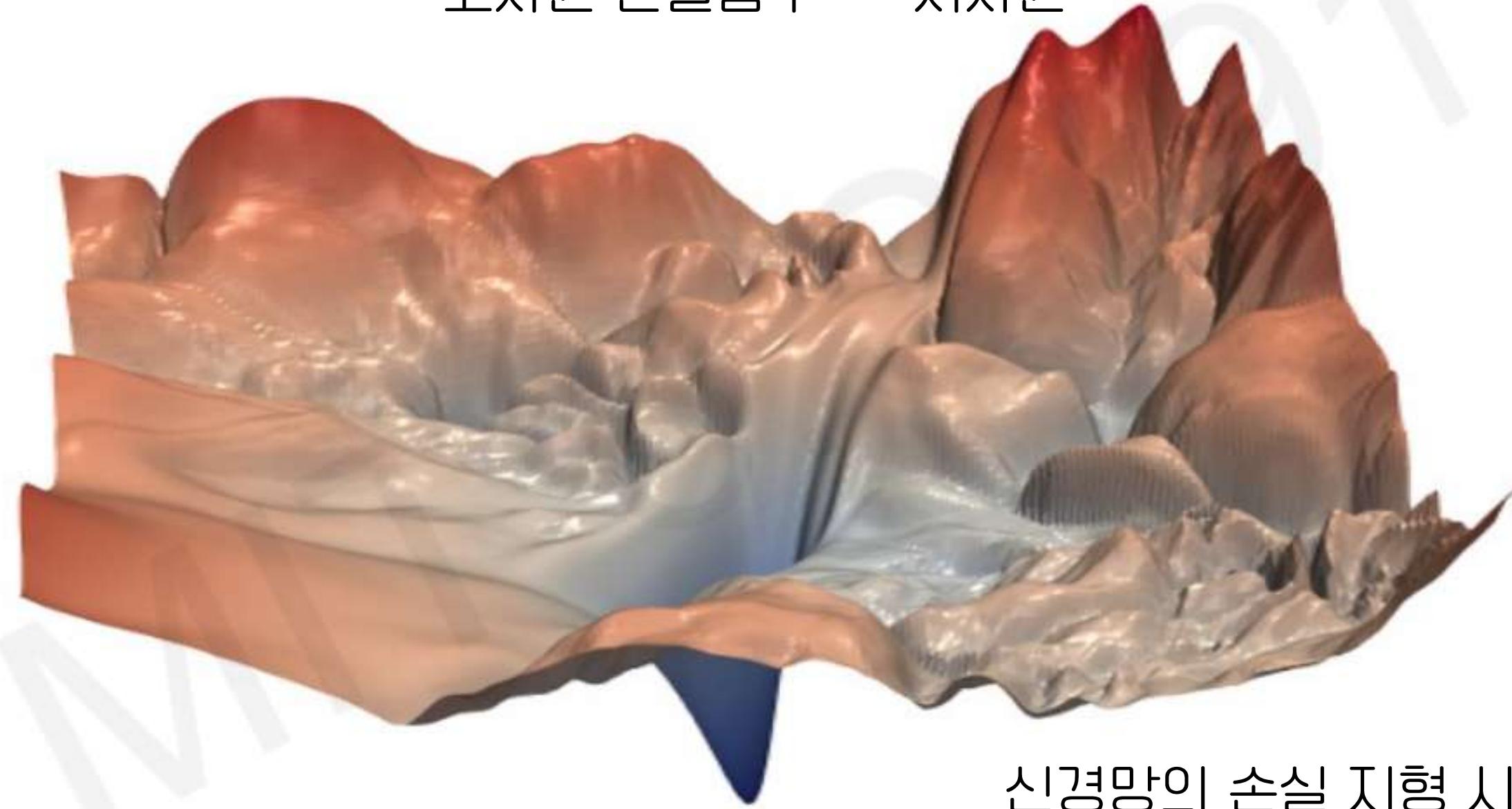
04 손실함수 | 최적화



역전파 프로세스는 미분적분학의 연쇄법칙을 적용하여 계산하기 굉장히 복잡하고 어렵지만 TensorFlow 와 PYTORCH에서 자동으로 수행함

04 손실함수

고차원 손실함수 → 저차원



신경망의 손실 지형 시각화
("Visualizing the loss landscape
of neural nets". Dec 2017.)

04 | 최적화

경사 하강법을 통한 최적화

$$\mathbf{W} \leftarrow \mathbf{W} - \eta \frac{\partial J(\mathbf{W})}{\partial \mathbf{W}}$$

04 | 최적화

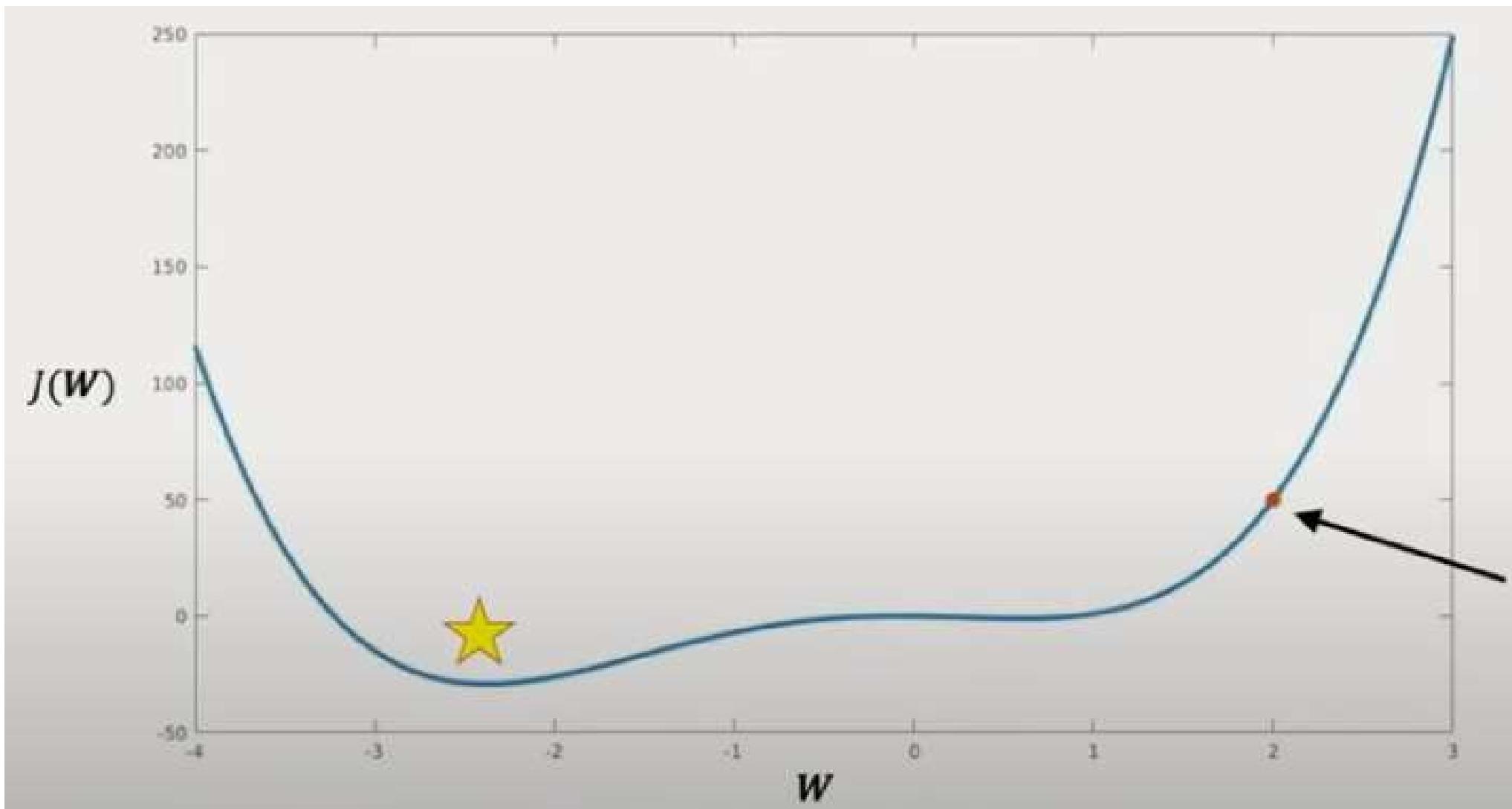
경사 하강법을 통한 최적화

$$\mathbf{W} \leftarrow \mathbf{W} - \eta \frac{\partial J(\mathbf{W})}{\partial \mathbf{W}}$$

학습률은 가중치 업데이트의 크기를 결정함
→ 얼마나 하강할지 결정함

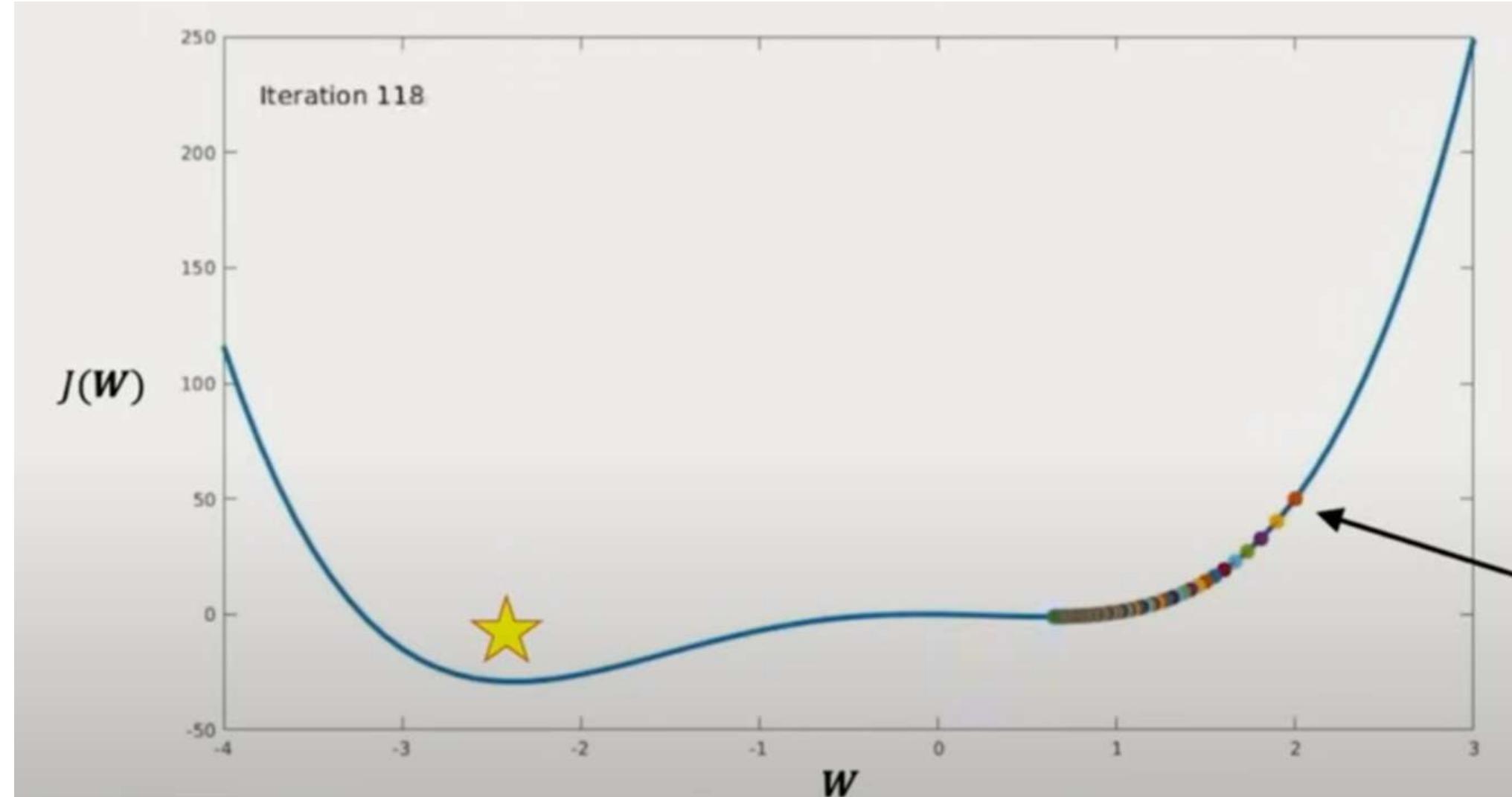
04 손실함수

학습률을 너무 작게 설정하면 잘못된 최소값에 도달하거나 느린 속도로 수렴하게 된다.



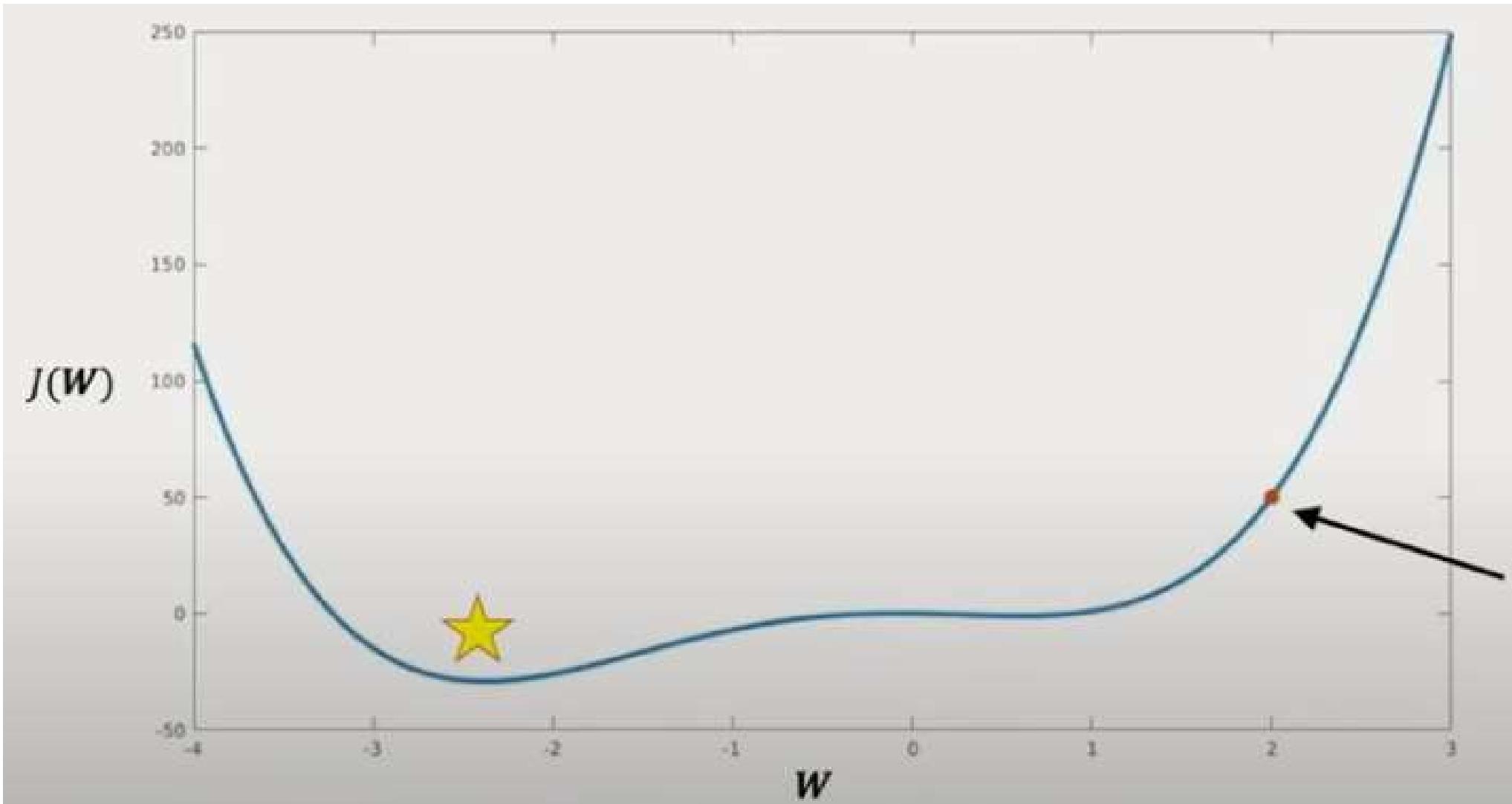
04 수학적 최적화

잘못된 최소값에 도달



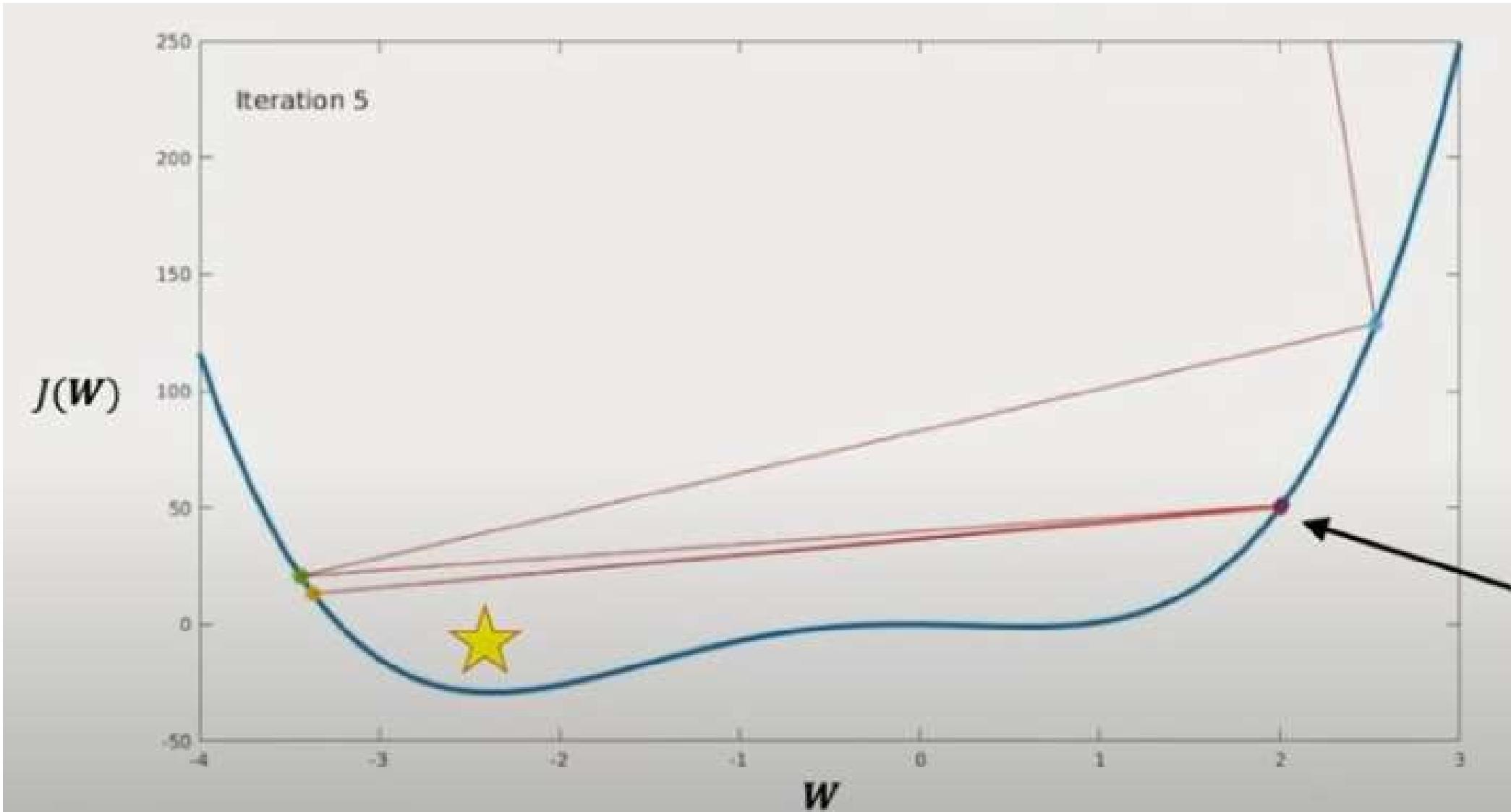
04 손실함수

학습률을 너무 크게 설정하면 초과, 불안정해지고 발산하게 된다.



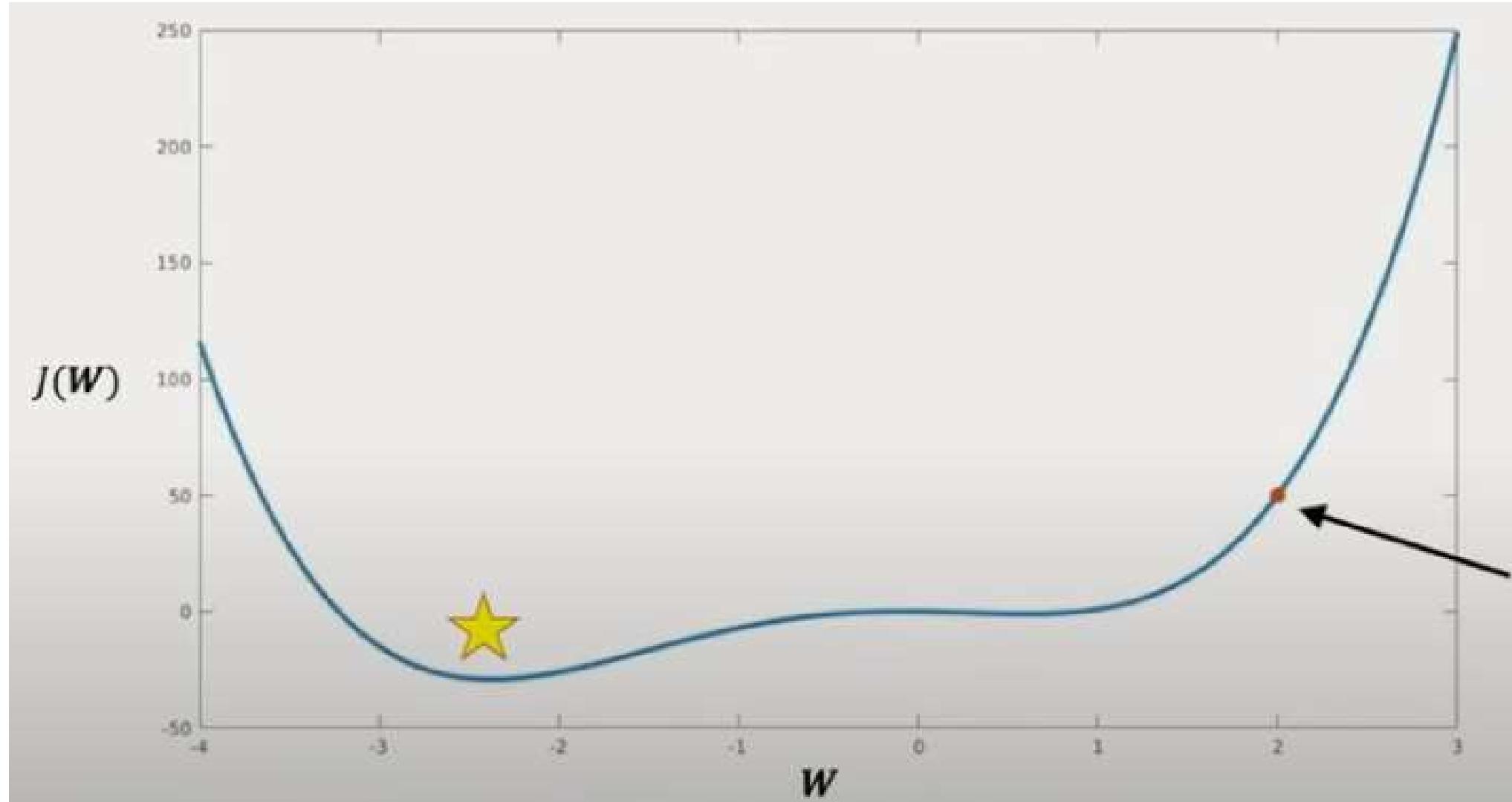
04 손실함수

학습률을 너무 크게 설정하면 초과, 불안정해지고 발산하게 된다.

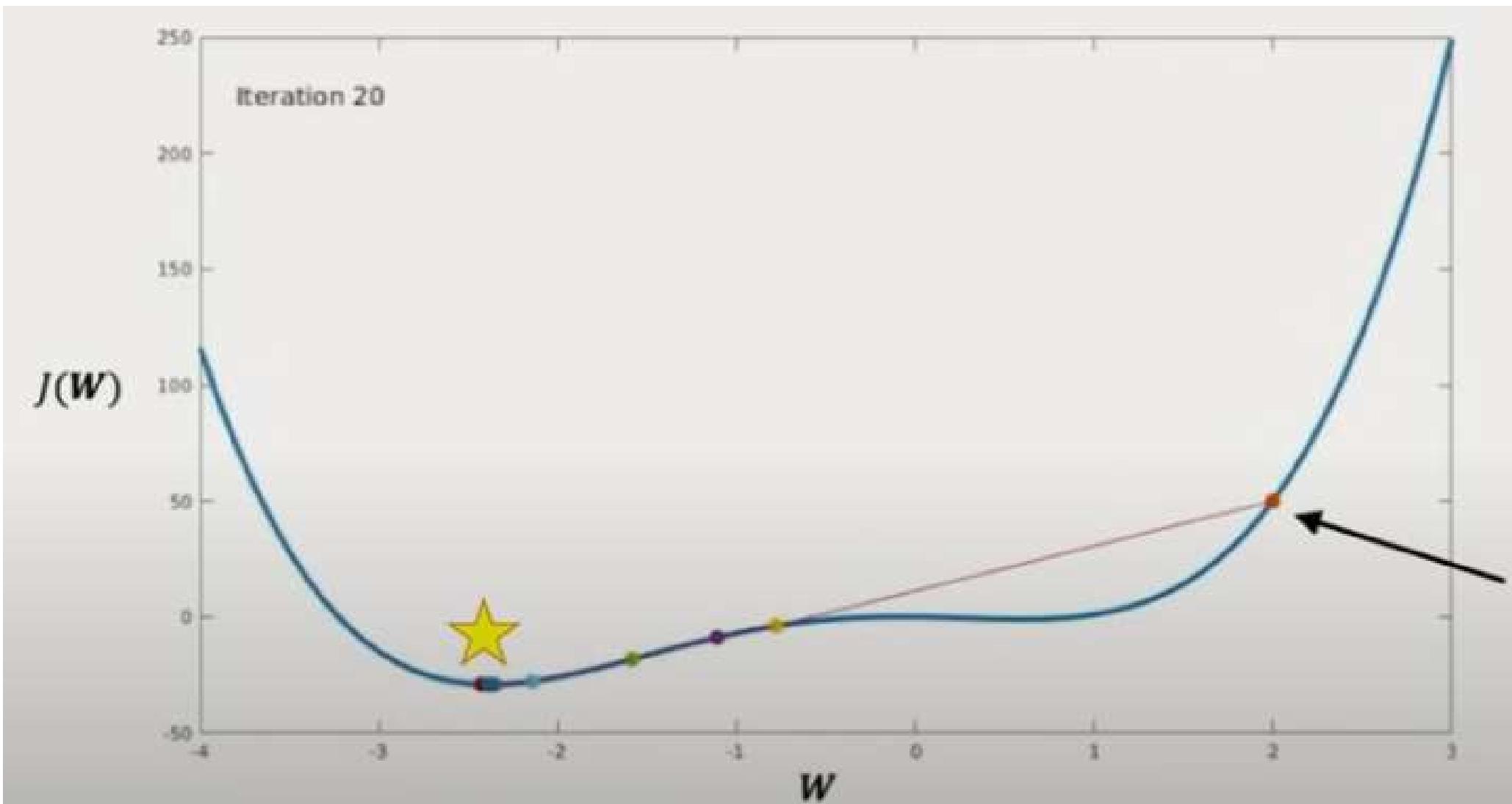


04 손실함수

학습률을 적당하게 설정하면 발산하거나 잘못된 최소값을 건너뛰는 것을 방지하며 이상적인 값에 수렴할 수 있다.



04 | 주제



04 손실 최적화

학습률을 어떻게 설정 해야할까?

아이디어1. 여러 학습률을 일일이 대입하여 최적의 값을 찾는다

아이디어2. 모든 손실 지형에 적응하는 알고리즘 설계

04 수학적 학습률

적응적 학습률

학습률은 고정되지 않음
많은 요인들로 인해서 학습률이 커지거나 작아짐

얼마나 기울기가 큰지
얼마나 학습이 빠른지
특정 가중치의 사이즈에 따라
등등...

적응적 학습률에 대한 연구가 넓은 범위로 진행되었고
이에 따라 많은 유형의 적응적 학습률 스케줄러가 생기면서
지금은 코드 한 줄 딸깍으로 해결할 수 있다.

04 | 솔루션 최적화

경사 하강법 알고리즘

- SGD
- Adam
- Adadelta
- Adagrad
- RMSProp

 tf.keras.optimizers.SGD

 tf.keras.optimizers.Adam

 tf.keras.optimizers.Adadelta

 tf.keras.optimizers.Adagrad

 tf.keras.optimizers.RMSProp

 torch.optim.SGD

 torch.optim.Adam

 torch.optim.Adadelta

 torch.optim.Adagrad

 torch.optim.RMSProp

세부사항: <https://brunch.co.kr/@chris-song/50?ref=ruder.io>

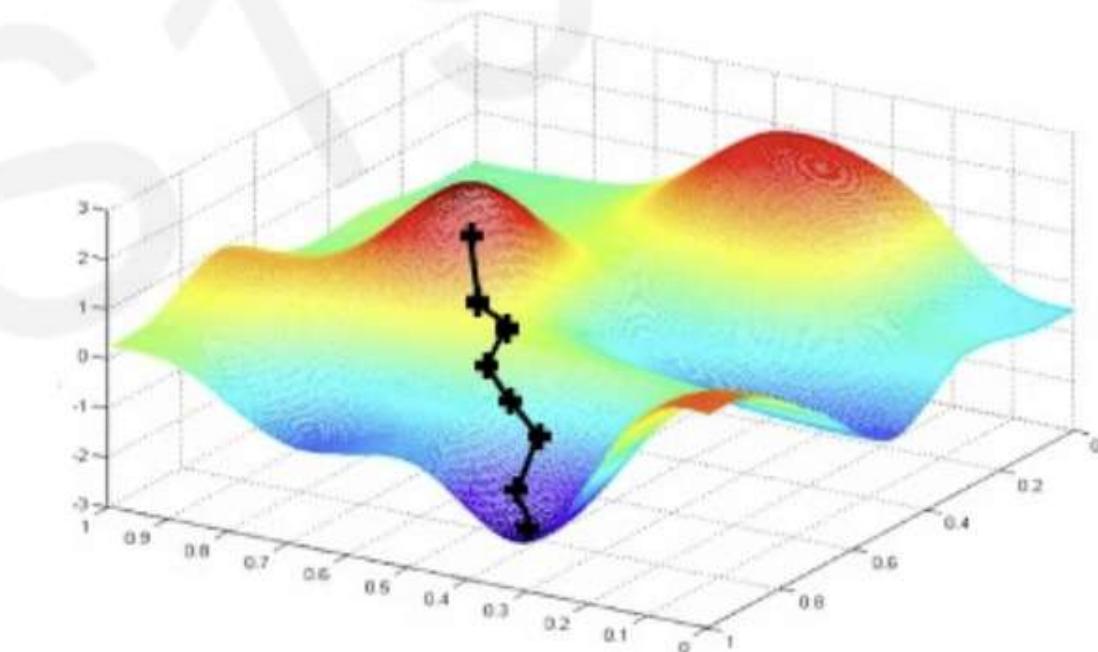
04 수설 | 최적화

경사하강법 VS 확률적 경사하강법

Algorithm

1. Initialize weights randomly $\sim \mathcal{N}(0, \sigma^2)$
2. Loop until convergence:
3. Compute gradient, $\frac{\partial J(\mathbf{W})}{\partial \mathbf{W}}$
4. Update weights, $\mathbf{W} \leftarrow \mathbf{W} - \eta \frac{\partial J(\mathbf{W})}{\partial \mathbf{W}}$
5. Return weights

경사 하강법의 기울기는 역전파로 계산되며
데이터셋의 모든 데이터에 합계 또는 평균을
계산해야함
→ 효율성이 떨어짐

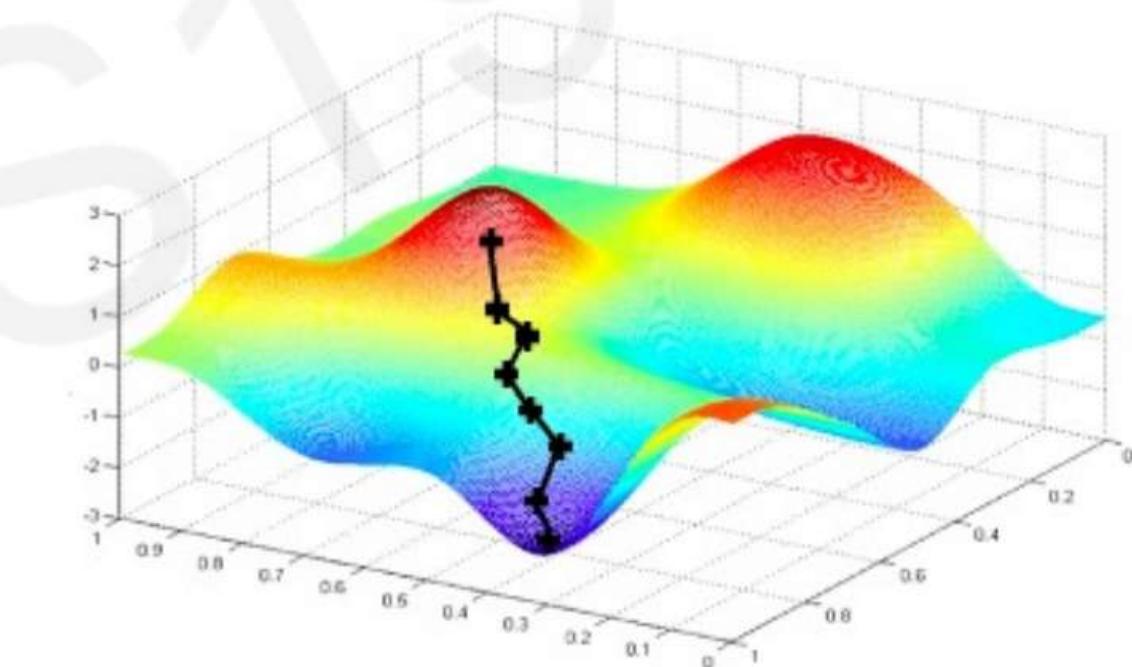


경사하강법 VS 확률적 경사하강법

Algorithm

1. Initialize weights randomly $\sim \mathcal{N}(0, \sigma^2)$
2. Loop until convergence:
3. Pick single data point i
4. Compute gradient, $\frac{\partial J_i(\mathbf{W})}{\partial \mathbf{W}}$
5. Update weights, $\mathbf{W} \leftarrow \mathbf{W} - \eta \frac{\partial J(\mathbf{W})}{\partial \mathbf{W}}$
6. Return weights

데이터 포인트 i 를 무작위로 선택하고 전체 데이터셋에 대한 기울기가 아닌 하나의 데이터에 대한 기울기를 구한다.
→ 확률적이지만 빠르게 답을 얻을 수 있음



04 손실함수

학률적 경사하강법

Algorithm

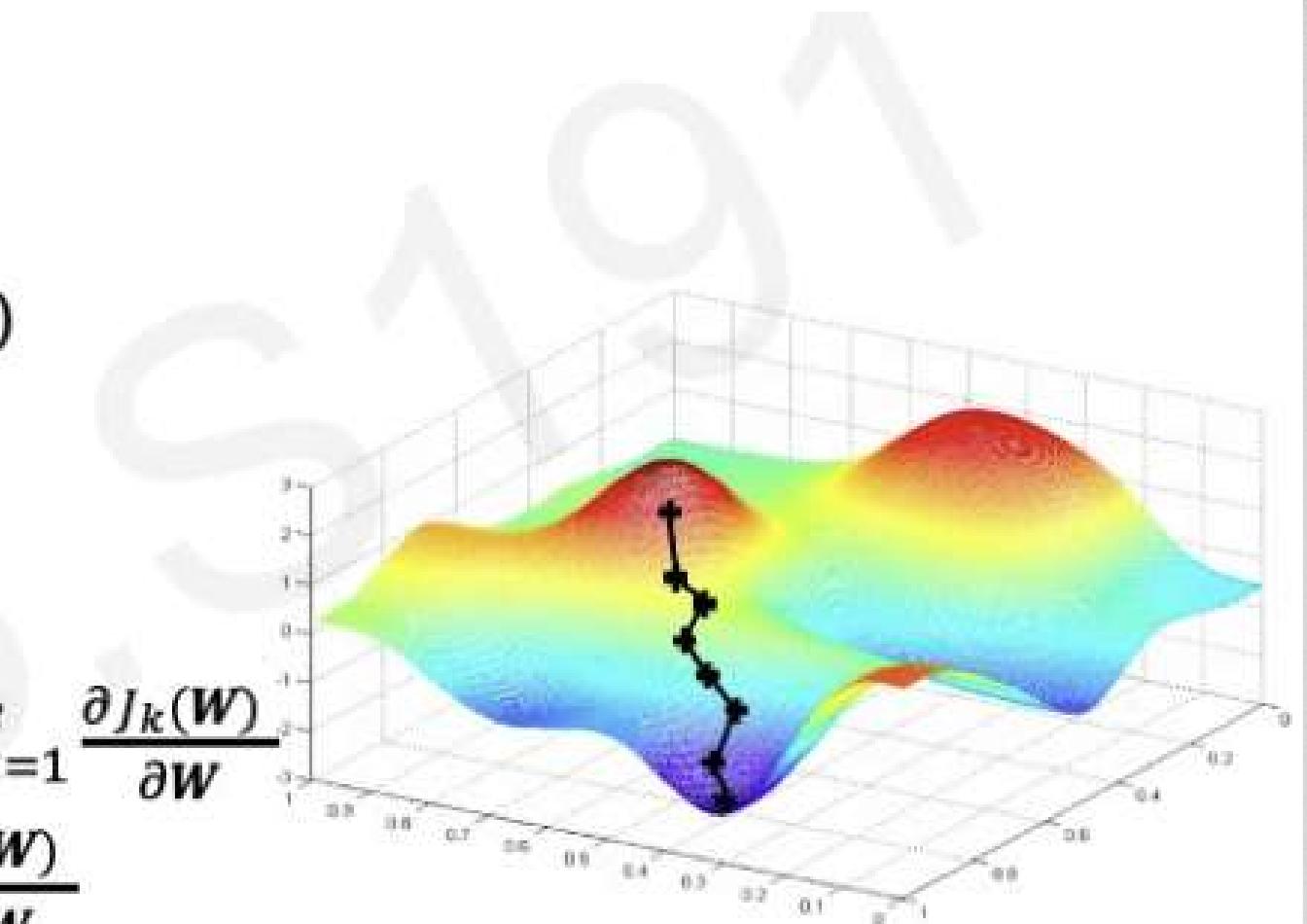
1. Initialize weights randomly $\sim \mathcal{N}(0, \sigma^2)$
2. Loop until convergence:
3. Pick batch of B data points
4. Compute gradient, $\frac{\partial J(\mathbf{W})}{\partial \mathbf{W}} = \frac{1}{B} \sum_{k=1}^B \frac{\partial J_k(\mathbf{W})}{\partial \mathbf{W}}$
5. Update weights, $\mathbf{W} \leftarrow \mathbf{W} - \eta \frac{\partial J(\mathbf{W})}{\partial \mathbf{W}}$
6. Return weights

배치 사이즈를 정한 후 K 개의 데이터 포인트의 기울기를 구하고 평균화 함

epoch: 전체 데이터셋을 학습한 횟수

batch: 나눈 데이터셋의 일부분

iteration : 나눈 데이터셋을 학습한 횟수



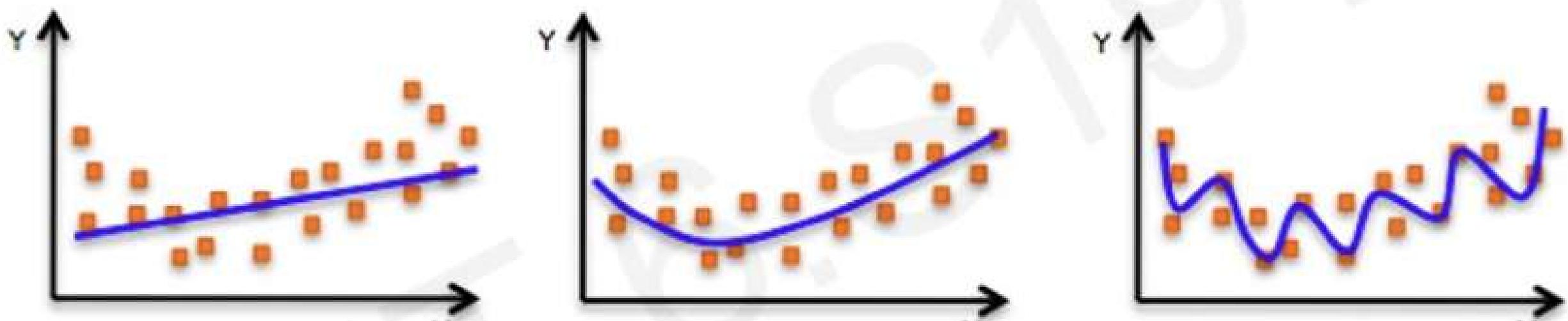
손실 조작학 04



04 손실함수

과적합의 문제

모델은 학습 데이터셋뿐 아니라 배포 했을 때 혹은 다른 데이터셋에서도 잘 작동해야 함
→ 학습 데이터와 테스트 데이터 모두에서 일반화 되기를 원함



Underfitting

Model does not have capacity
to fully learn the data

과소적합

: 데이터셋의 복잡성을 완전히
작용하지 못함

Ideal fit

Overfitting

Too complex, extra parameters,
does not generalize well

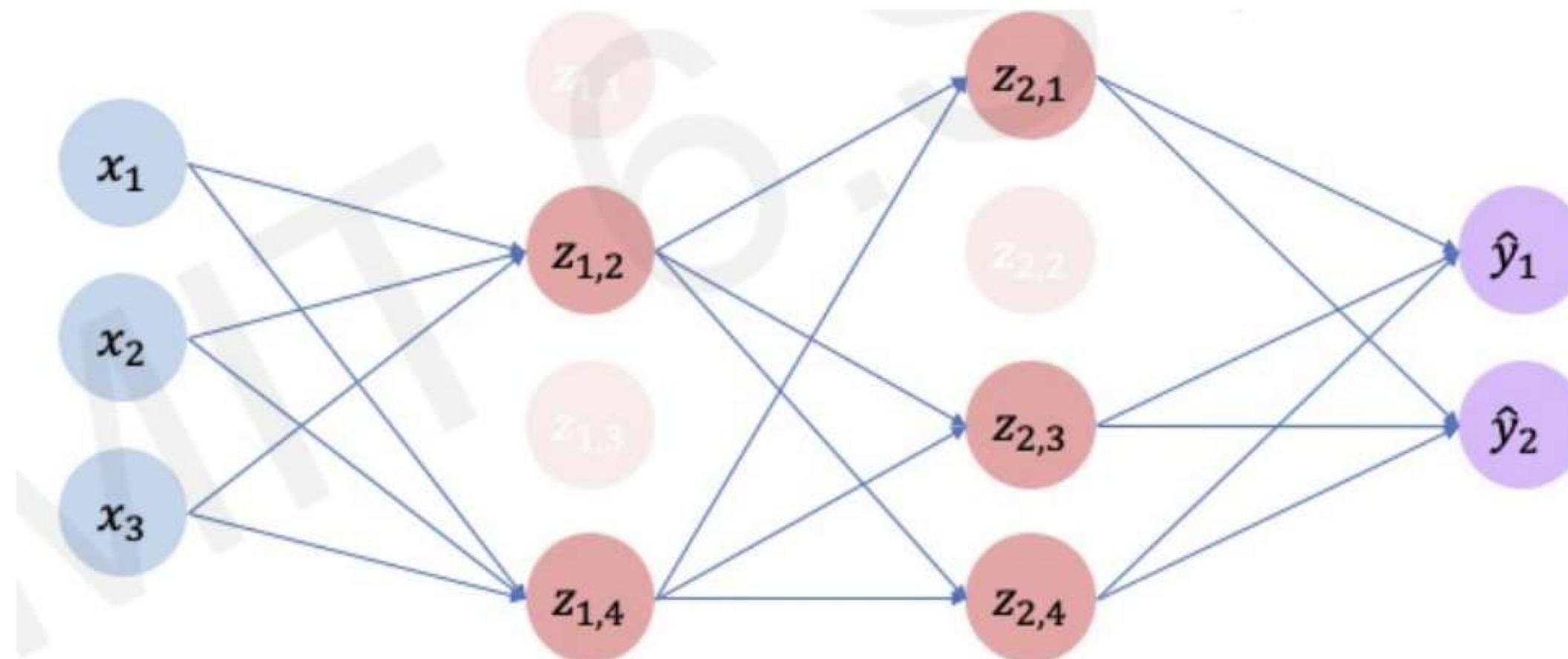
과적합

: 학습 데이터를 너무 많이 기억하여
너무 복잡하고, 성능이 떨어짐

04 손실 최적화

정규화

Dropout : 확률적으로 신경망의 은닉 뉴런의 활성화를 0으로 설정하여 인공신경망이 어떤 노드에 의존하지 않도록 하는 것 -> 많은 것들을 기억할 수 없게 된다.



04 손실함수와 정규화

정규화

조기 중단: 과적합 이전에 멈추는 것



