전체 흐름 (딥러닝 학습 loop)

1. 순전파 (Forward pass)
   현재 weight로 예측값 계산

2. Loss 계산
   예측값과 정답의 차이를 기반으로 Loss 값 산출

3. 역전파 (Backpropagation)
   Loss를 각 weight에 대해 미분해서
   ∂Loss/∂w 같은 gradient(기울기) 계산

4. 경사하강법 (Gradient Descent)
   계산된 gradient를 이용해서
   `w = w - η × ∂Loss/∂w` 로 weight를 업데이트

경사하강법

Loss(w) = 모델이 틀린 정도

최소로 하는것이 학습의 목표

3층짜리 신경망에서 Loss를 간단화해서 표현

x → h1 = f1(w1*x)
  → h2 = f2(w2*h1)
  → y_pred = f3(w3*h2)
  → Loss = L(y_pred, y_true)

합성함수 미분

예:

$$y = f(g(x))$$

- 여기서 y는 x에 직접적으로 안 연결돼 있어.
- x → g(x) → f(g(x)) 로 간접적으로 연결돼 있음.

이걸 미분하려면?

$$dy/dx = f'(g(x)) \cdot g'(x)$$

$$\partial Loss/\partial w1 = \partial Loss/\partial y_p red \times \partial y_p red/\partial h2 \times \partial h2/\partial h1 \times \partial h1/\partial w1$$

이렇게 **계층마다 한 단계씩 거꾸로 미분하며 따라가는 과정이 바로 역전파 (backpropagation)**
그리고 이 미분의 연결 고리가 바로 **체인 룰**이야.

업데이트 : w=w−learningrate×∂L/∂w

Loss를 줄이기 위해서 “무엇을 바꿔야 하는가?”
weight 를 바꿔야 합니다.

“Loss가 최소가 되기 위해서 weight를 바꾸라고 하는데,
그럼 대체 어떤 weight에서 미분을 하고, 그 weight는 처음에 어디서 나오는 걸까요?

1. weight는 처음에 어디서 오는가?

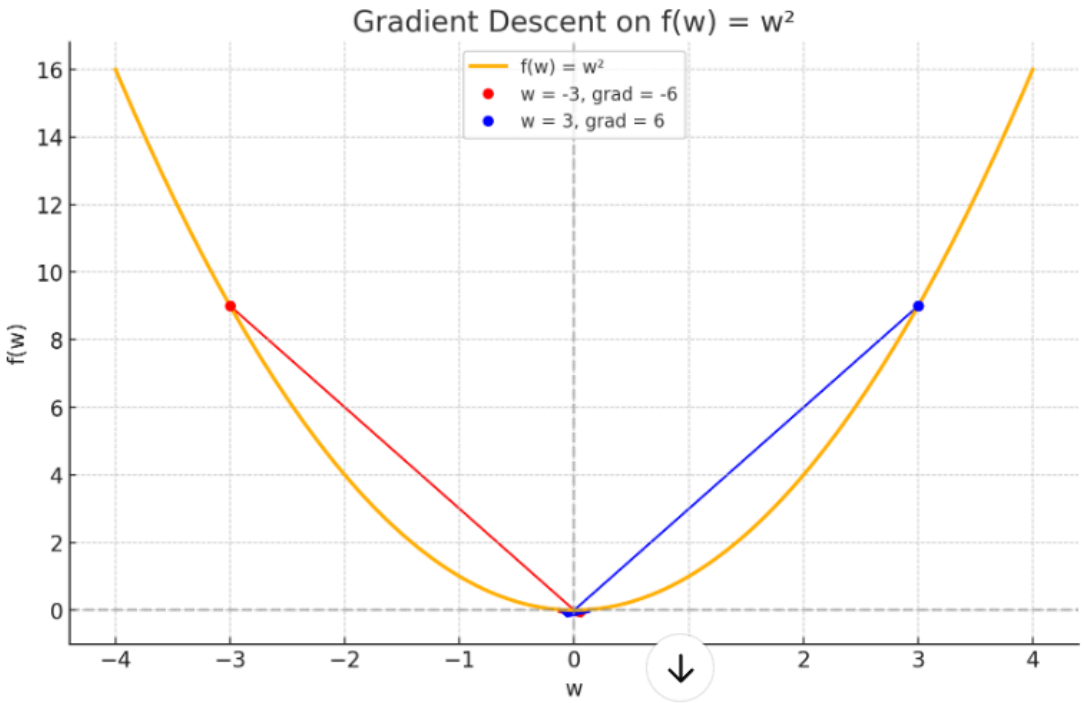
랜덤 초기화에서 옵니다.

- 신경망을 만들면, 각 레이어는 `weight` (가중치)라는 파라미터를 가짐
- 이 `weight`들은 학습 전에는 아무것도 모르기 때문에,
  보통 **무작위(random)**로 시작함

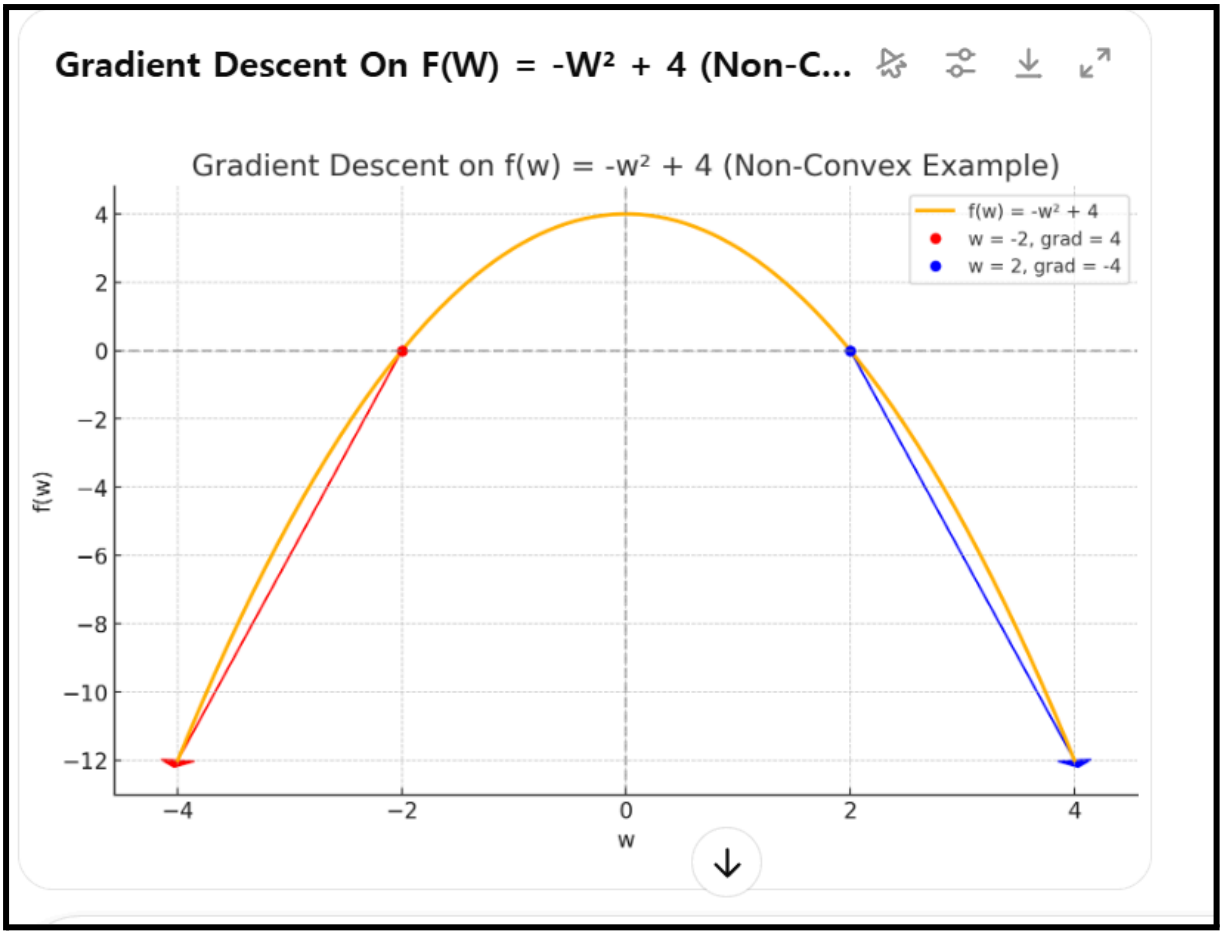
Loss를 미분해서 나온 gradient 방향으로 weight를 바꾼다고 해서,
그게 꼭 Loss를 최소화하는 방향이 맞다는 보장은 없다

기울기(gradient)는 함수가 가장 빠르게 증가하는 방향을 가리킨다.
따라서 그 반대 방향(-gradient)은 함수가 가장 빠르게 감소하는 방향이다.

Gradient Descent On F(W) = W²



무엇이든 물어보세요



이

상적인 경우: convex (아래로 볼록한) loss 함수

이론적으로는 아래와 같은 이유로 **convex (U 형태)**의 loss 함수가 이상적입니다:

- 단 하나의 전역 최솟값(global minimum) 존재
- 경사하강법(gradient descent)이 안정적으로 수렴
- 예:
    - 회귀(regression): MSE (Mean Squared Error)
      → `L(y, ŷ) = (y - ŷ)²` 는 완전한 convex 함수
    - 로지스틱 회귀에서의 Binary Cross Entropy → convex

하지만 실제 딥러닝에서는?

대부분의 딥러닝 모델에서는 loss 함수 자체는 convex가 아니고, 아래로 볼록하지도 않습니다.
이유는 다음과 같습니다:

1. 딥러닝 모델은 비선형 구조 (ReLU, tanh 등) + 수백 개의 weight로 구성
2. 그로 인해 loss 함수는 고차원 비선형 함수
3. 따라서 구불구불한 손실곡선, 즉 non-convex 함수

📌 대표 예:

- CNN, ResNet 등에서의 CrossEntropyLoss
- Transformer에서의 loss surface → saddle point 다수