목차

교육 일정

EliceAcademy

머신러닝을 활용한 나노/탄소 소재 생산공정 최적화

04 금속분말 생산공정 최적화를 위한 비선형회귀 기법

목차

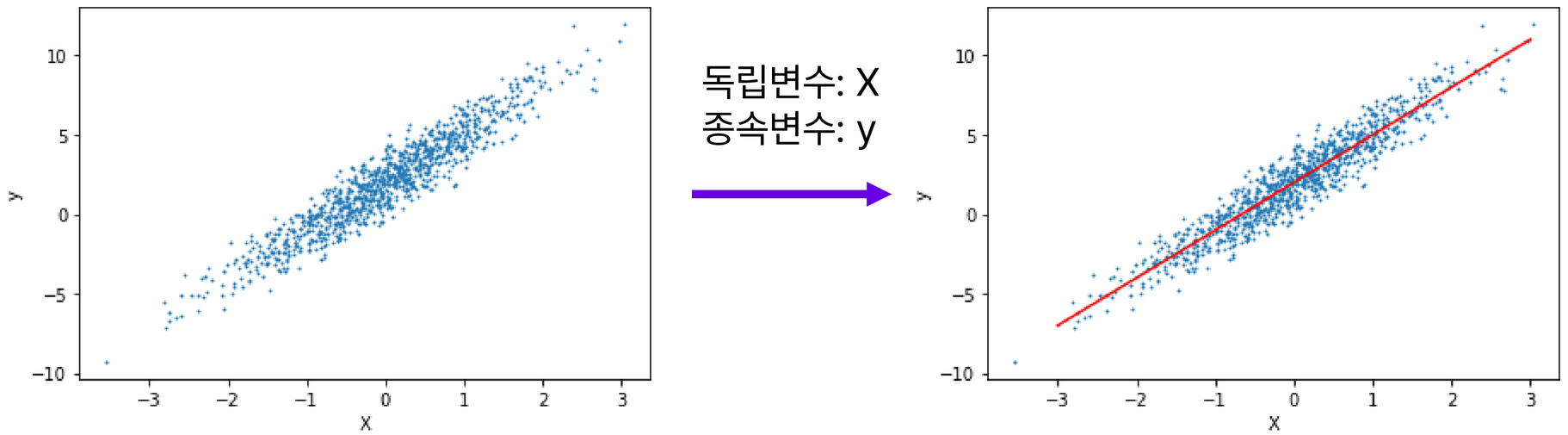
금속분말 생산공정
최적화를 위한 딥러닝 기법

01

선형 회귀 모델의 한계

선형적 관계

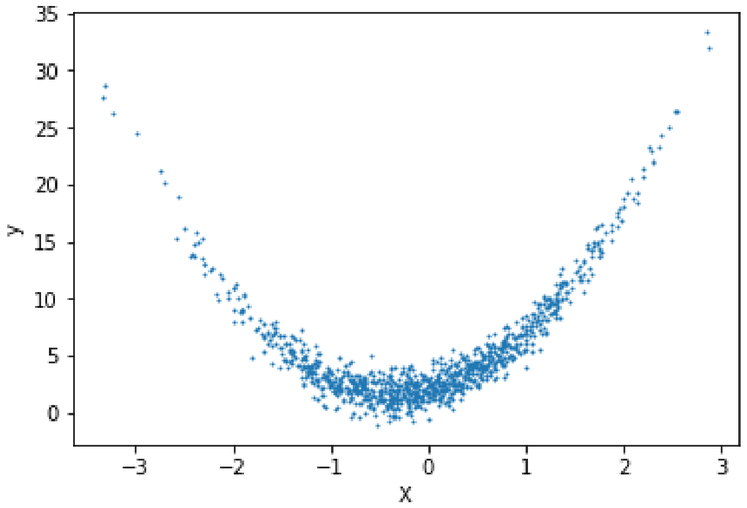
$$y = 3x + 2$$



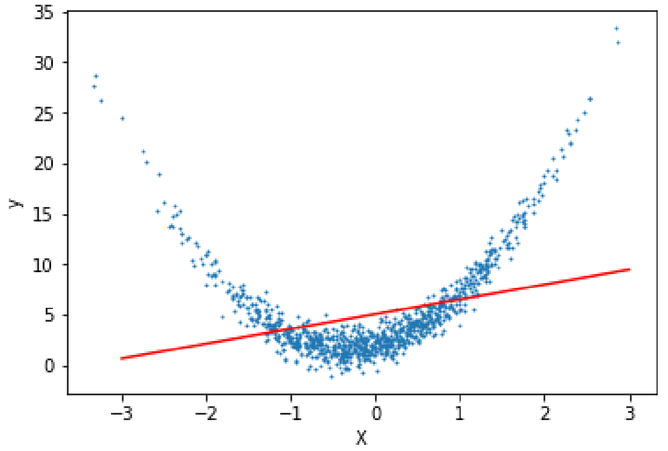
EliceAcademy

선형적 관계

$$y = 3x^2 + 2x + 2$$

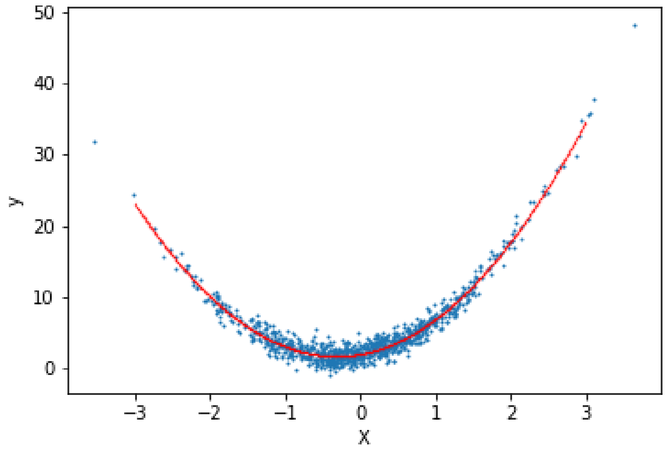


독립변수: X
종속변수: y



독립변수: X, X^2
종속변수: y

다항선형회귀



다항 회귀

- 선형관계처럼 보이지 않는 데이터라도, **다항 선형 회귀**를 통해 분석 가능
  - $Y = \beta_1 X + \beta_0$
  - $Y = \beta_2 X^2 + \beta_1 X + \beta_0$
  - $Y = \beta_3 X^3 + \beta_2 X^2 + \beta_1 X + \beta_0$
  - ...

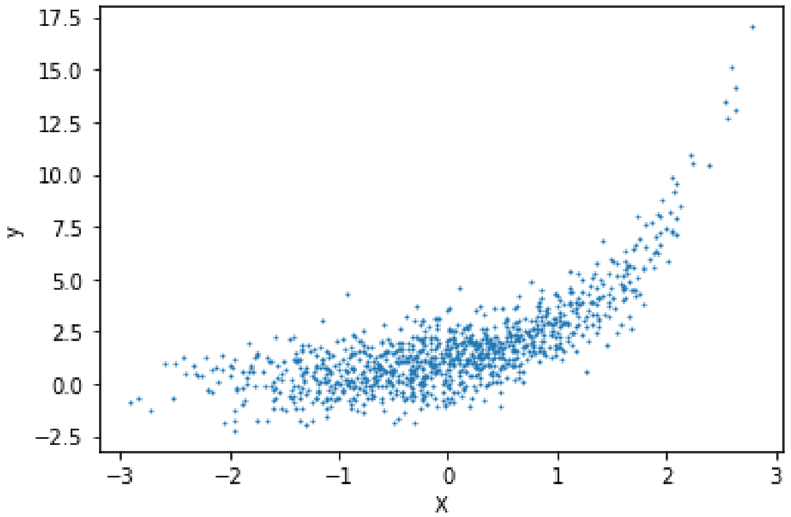
- 장점
  독립변수와 종속변수의 관계를 더 잘 모델링할 수 있다.

- 단점
  이상치에 민감하다.

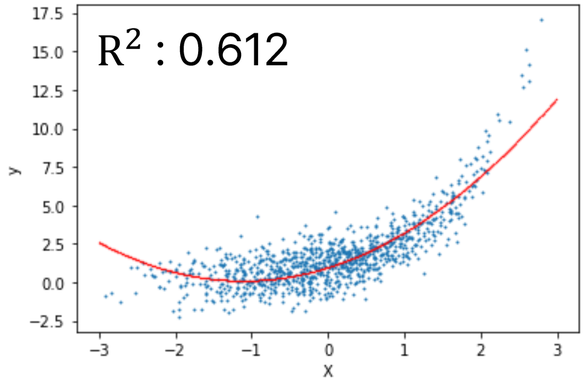
EliceAcademy

비선형적 관계

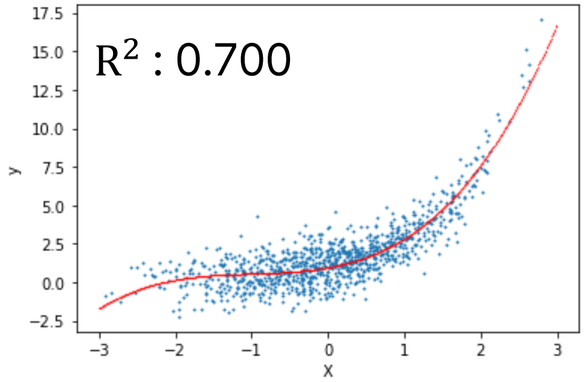
$$y = e^x$$



독립변수: X, X^2
종속변수: y



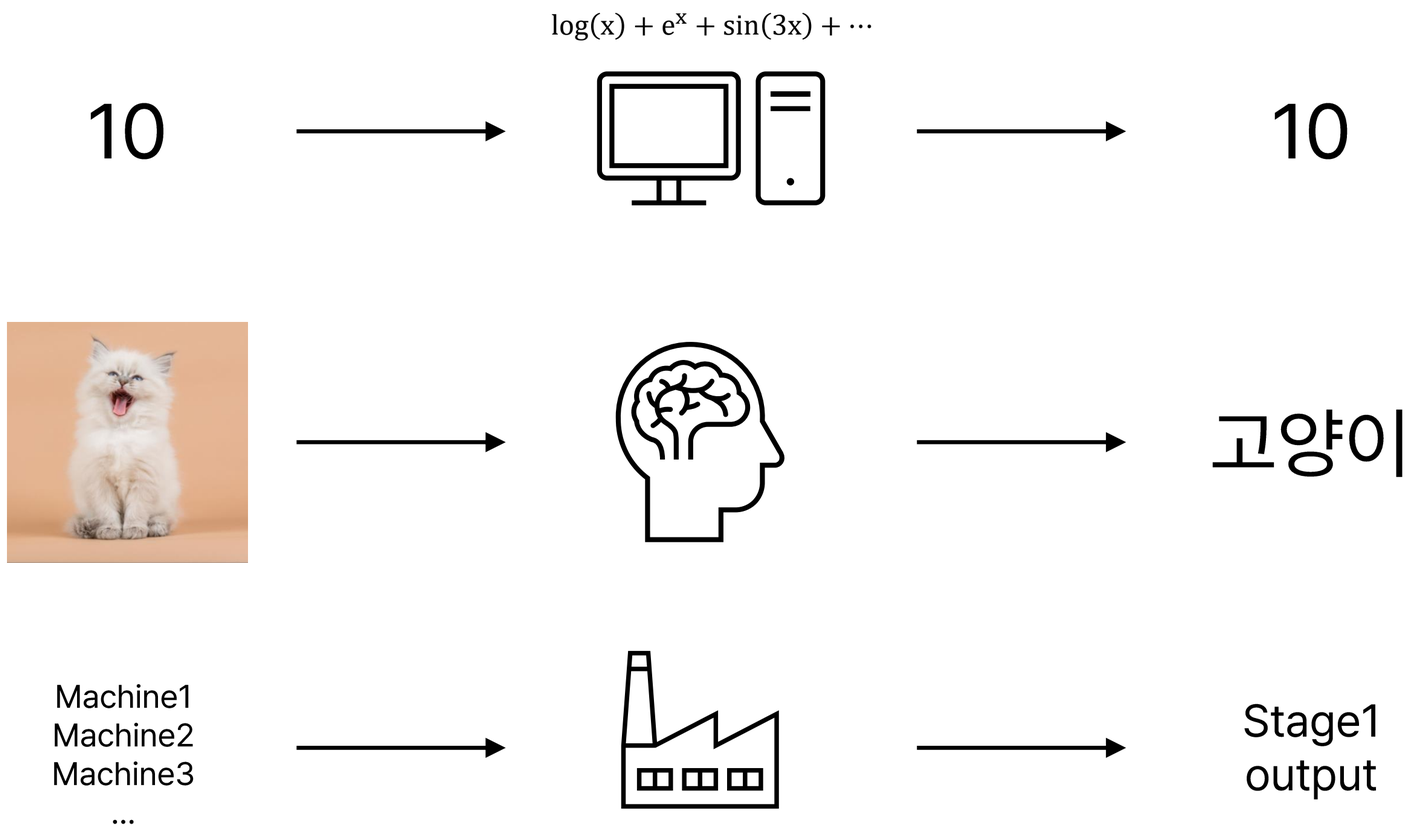
독립변수: X, X^2, X^3
종속변수: y



- 입력값과 출력값이 비선형적 관계일 때, 다항 회귀를 사용하여 근사는 가능하지만, 정확도가 낮음.
- 실제 데이터는 비선형적 관계가 잦음

EliceAcademy

비선형적 관계



02

비선형 회귀 모델

EliceAcademy

비선형 회귀 모델

- 모델의 변수 β_i 와 출력값 Y 가 비선형적 관계
  - $Y = \frac{\beta_0 X + \beta_1}{\beta_2 e^X + \beta_3}$
  - ...
- **복잡한 관계**를 모델링하기 적합
- 선형회귀 모델보다 정확도가 높음
- 그러나, 최적화가 어렵고 **해석이 쉽지 않음**

비선형 회귀 모델

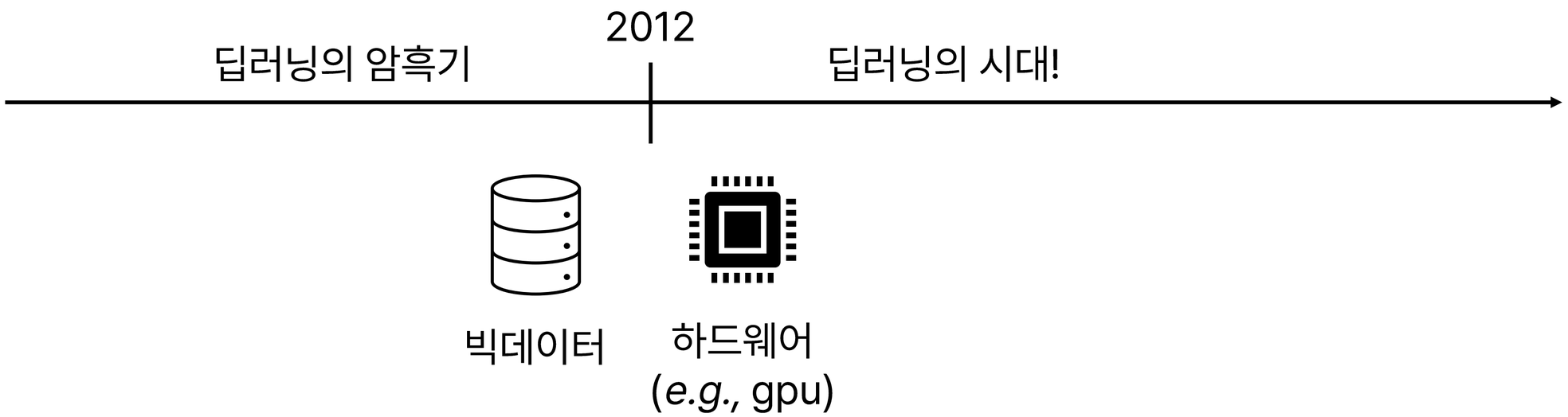
- 선형 회귀 모델
  - 아이스크림 판매량 $= 3 \times$ 기온 $- 2 \times$ 물가
  - 아이스크림 판매량은 기온과는 양의 상관관계를, 물가와는 음의 상관관계를 지니는구나..!

- 비선형 회귀 모델
  - 아이스크림 판매량 $= \frac{3 \times \text{기온} + \text{기온} \times \text{물가}}{e^{\text{물가}} + \log(\text{기온})}$
  - 아이스크림 판매량과 기온, 물가와의 관계가 어떻게 되는거지..?

EliceAcademy

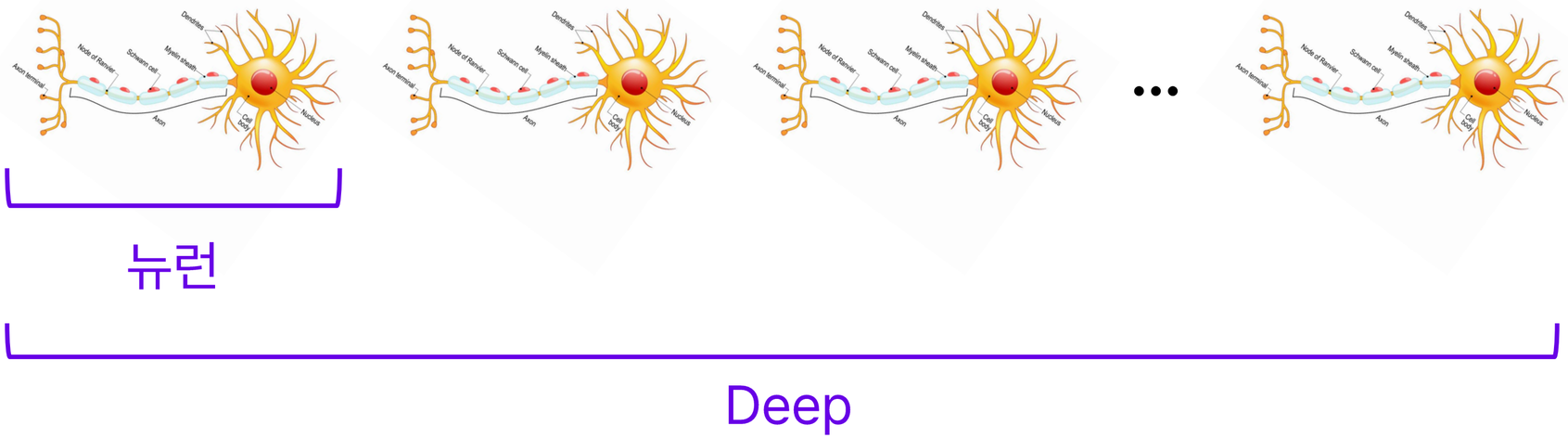
대표적인 비선형 회귀 모델

- 딥러닝



딥 뉴럴 네트워크 (Deep neural network)

- **Deep**
  - 깊다
- **Neural**
  - 뉴런 (신경망)의



EliceAcademy

03

다층 퍼셉트론 모델 (MLP)

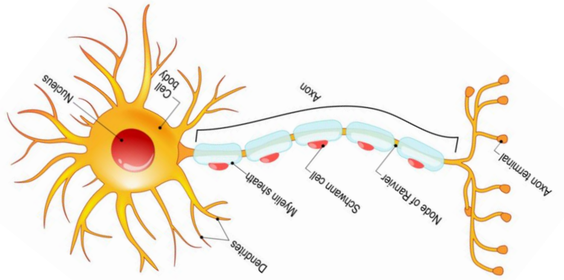
EliceAcademy

퍼셉트론

- 뉴런 세포를 모방

뉴런

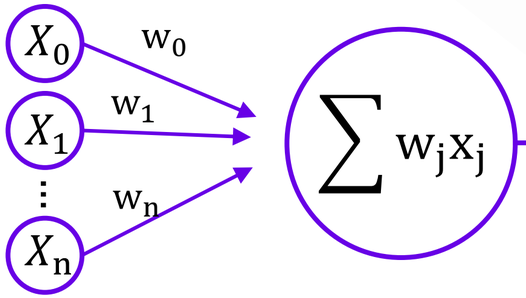
입력 신호



출력 신호

퍼셉트론

입력 값

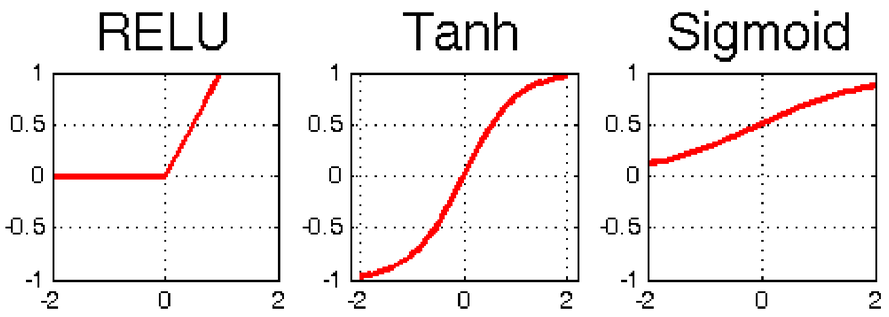


활성 함수

출력 값

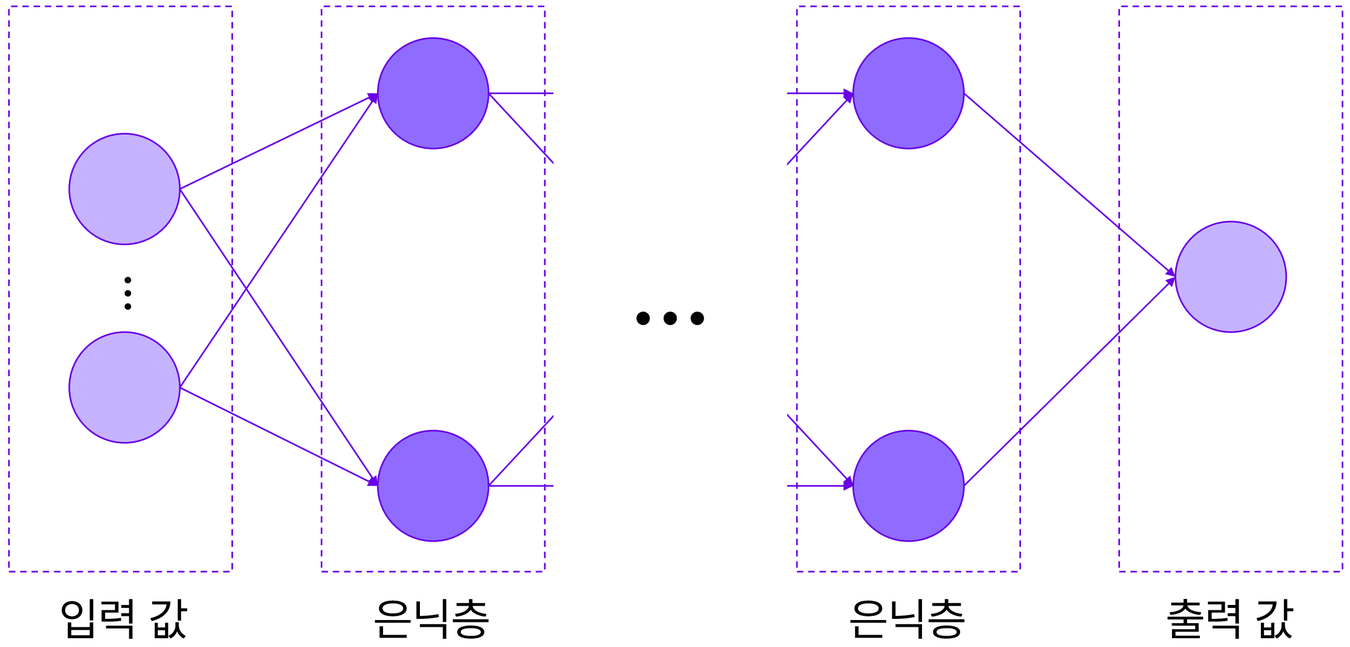
활성 함수

- 퍼셉트론 마지막 계산에 적용
- 주로 비선형 함수를 적용
- 퍼셉트론 모델이 **비선형적 관계**를 모델링할 수 있게 됨
- *e.g.*, Sigmoid, ReLU, tanh, ...

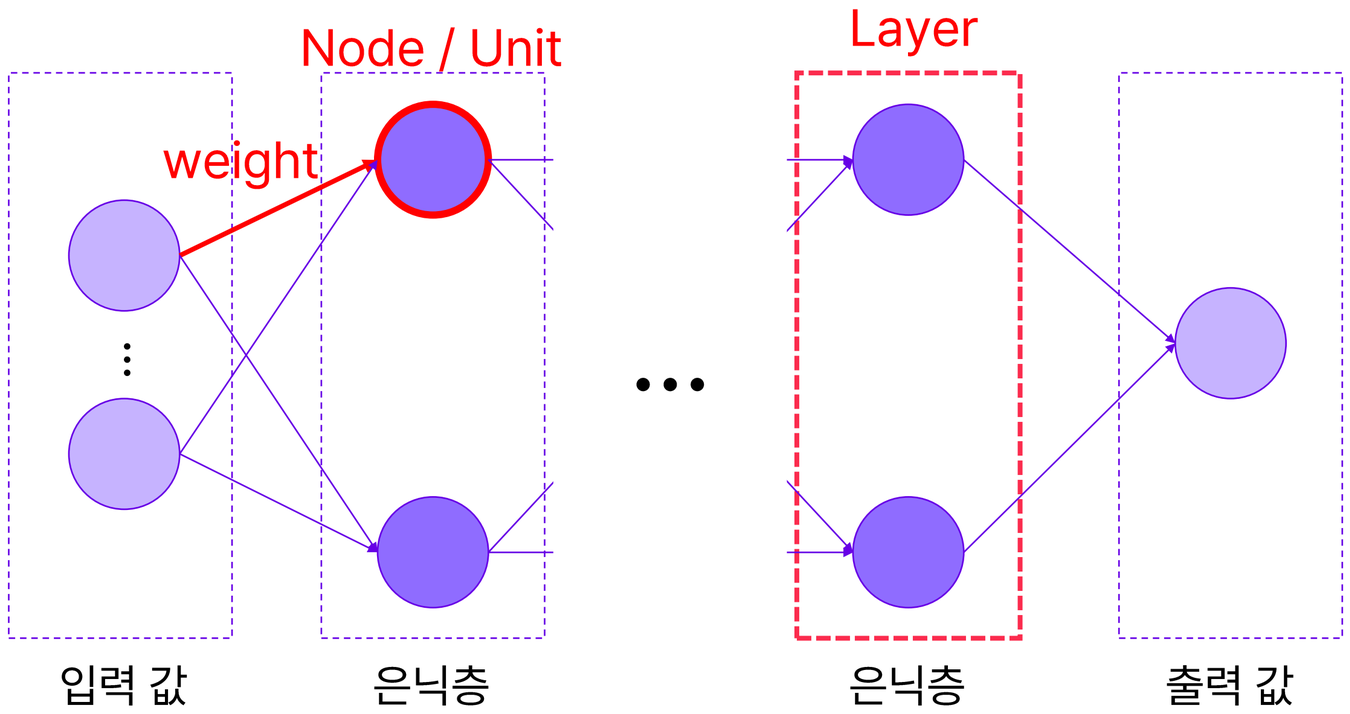


다층 퍼셉트론 모델 (Multi Layer Perceptron; MLP)

- 퍼셉트론을 여러 층으로 쌓은 모델
- 복잡한 관계를 더욱 잘 모델링할 수 있음



다층 퍼셉트론 모델

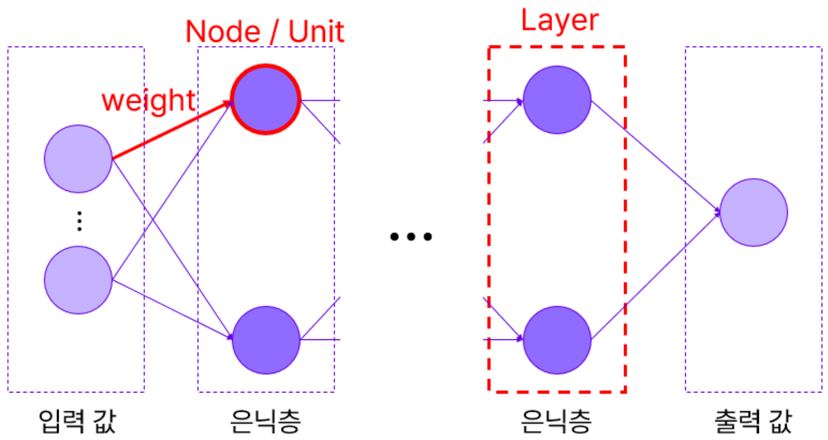


다층 퍼셉트론 모델

- 선형 회귀 모델의 목표
  - 예측값과 실제값의 오차를 최소화 하는 β 를 찾기

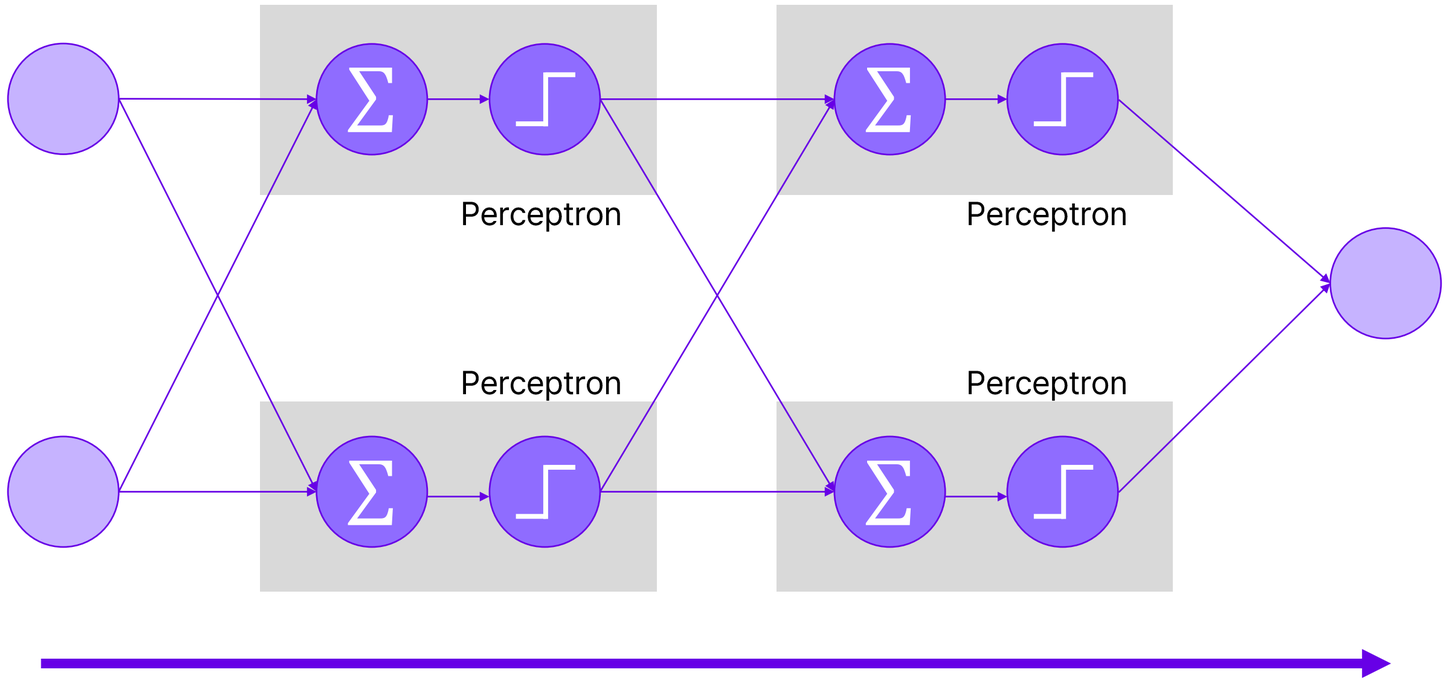
$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots$$

- 다층 퍼셉트론 모델의 목표
  - 예측값과 실제값의 오차를 최소화 하는 weight 를 찾기



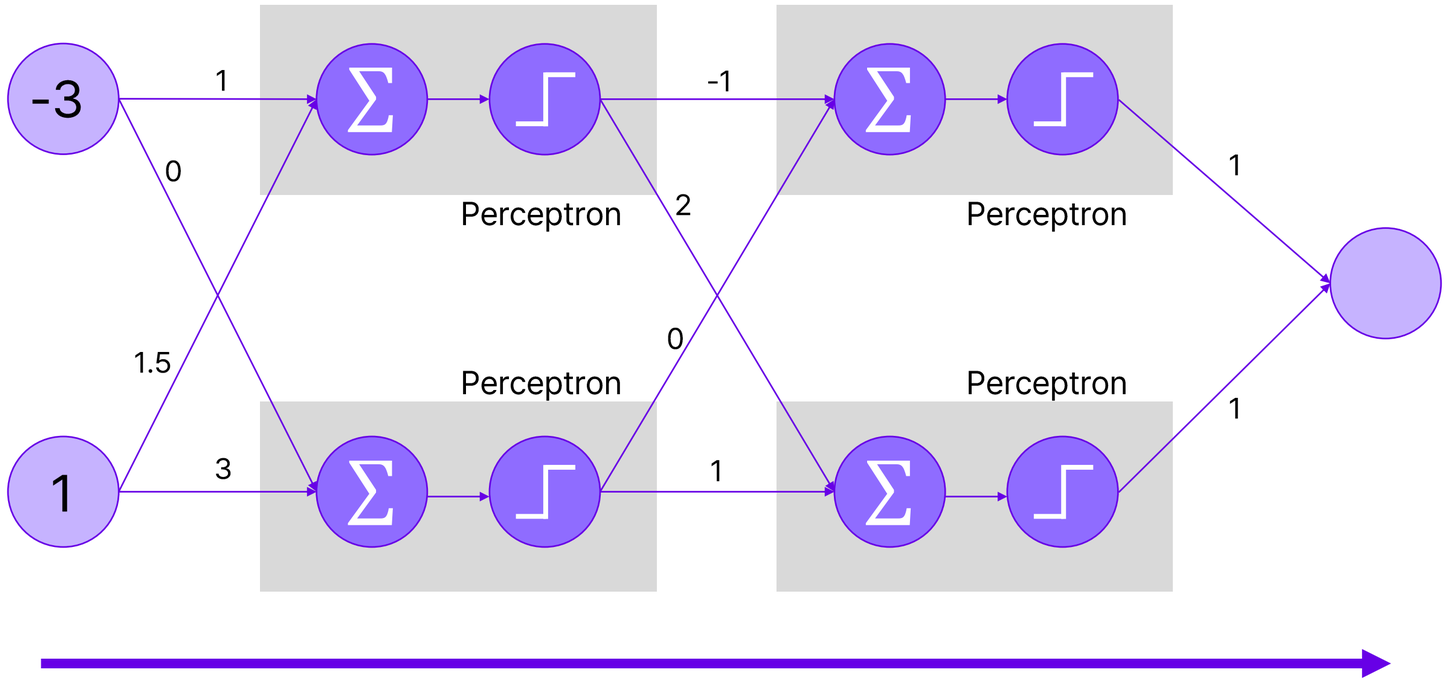
순전파 (Forward propagation)

- MLP 가 입력값으로부터 출력값을 계산하는 방식



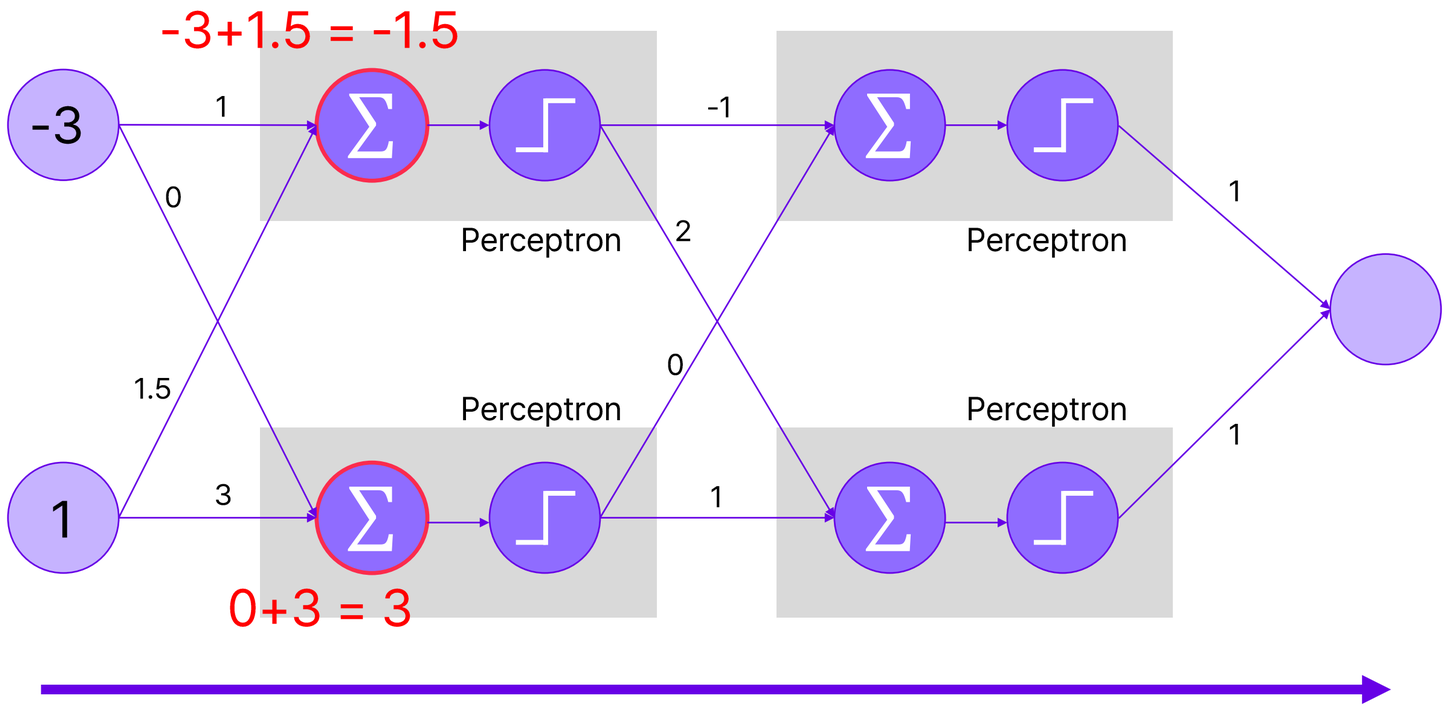
순전파 (Forward propagation)

- 활성 함수가 ReLU 일 때, 출력값을 직접 계산해봅시다.



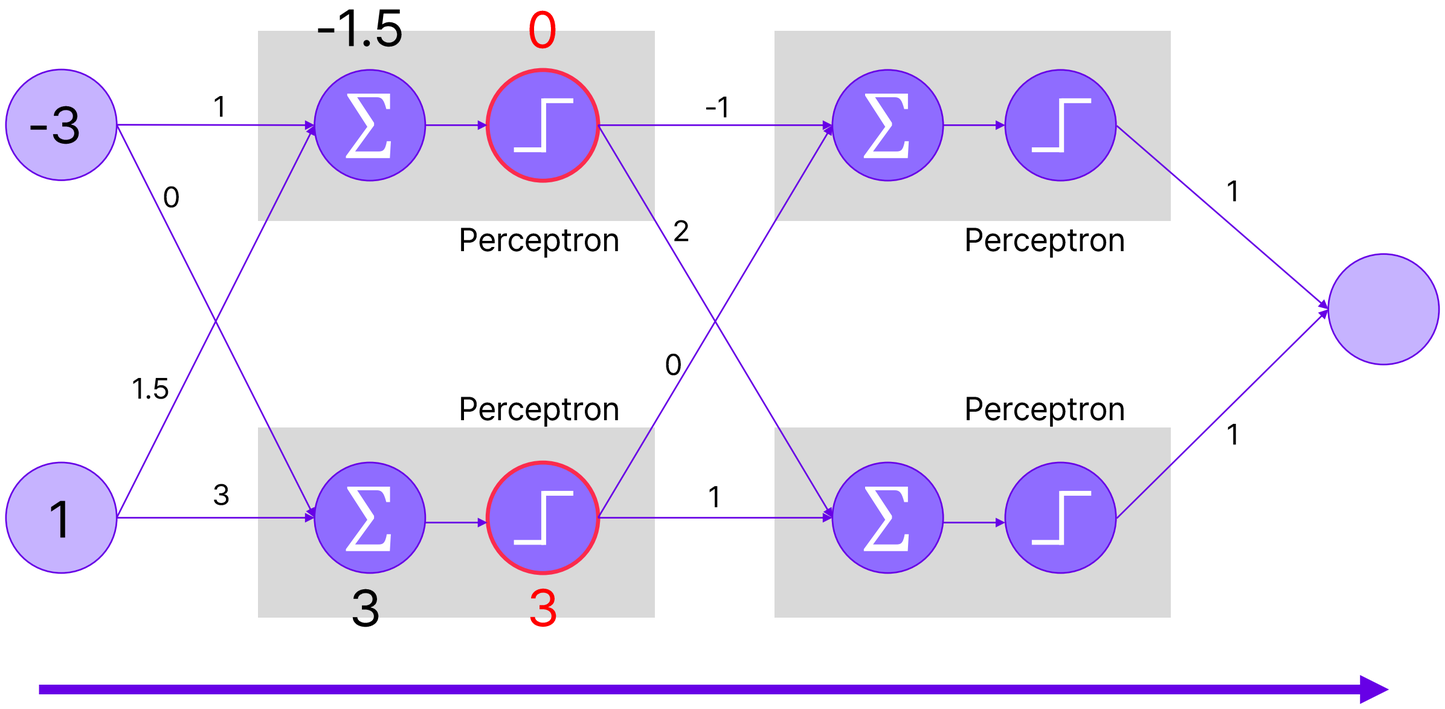
순전파 (Forward propagation)

- 활성 함수가 ReLU 일 때, 출력값을 직접 계산해봅시다.



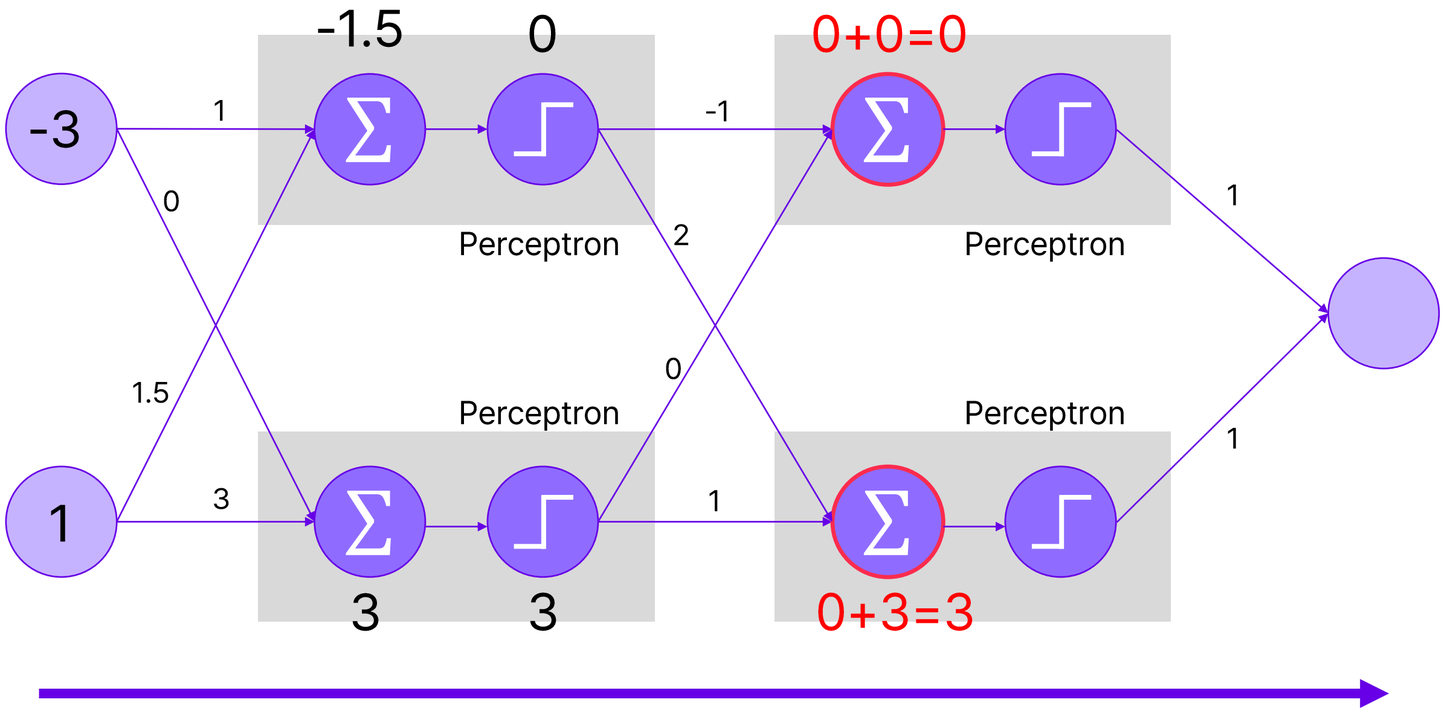
순전파 (Forward propagation)

- 활성 함수가 ReLU 일 때, 출력값을 직접 계산해봅시다.



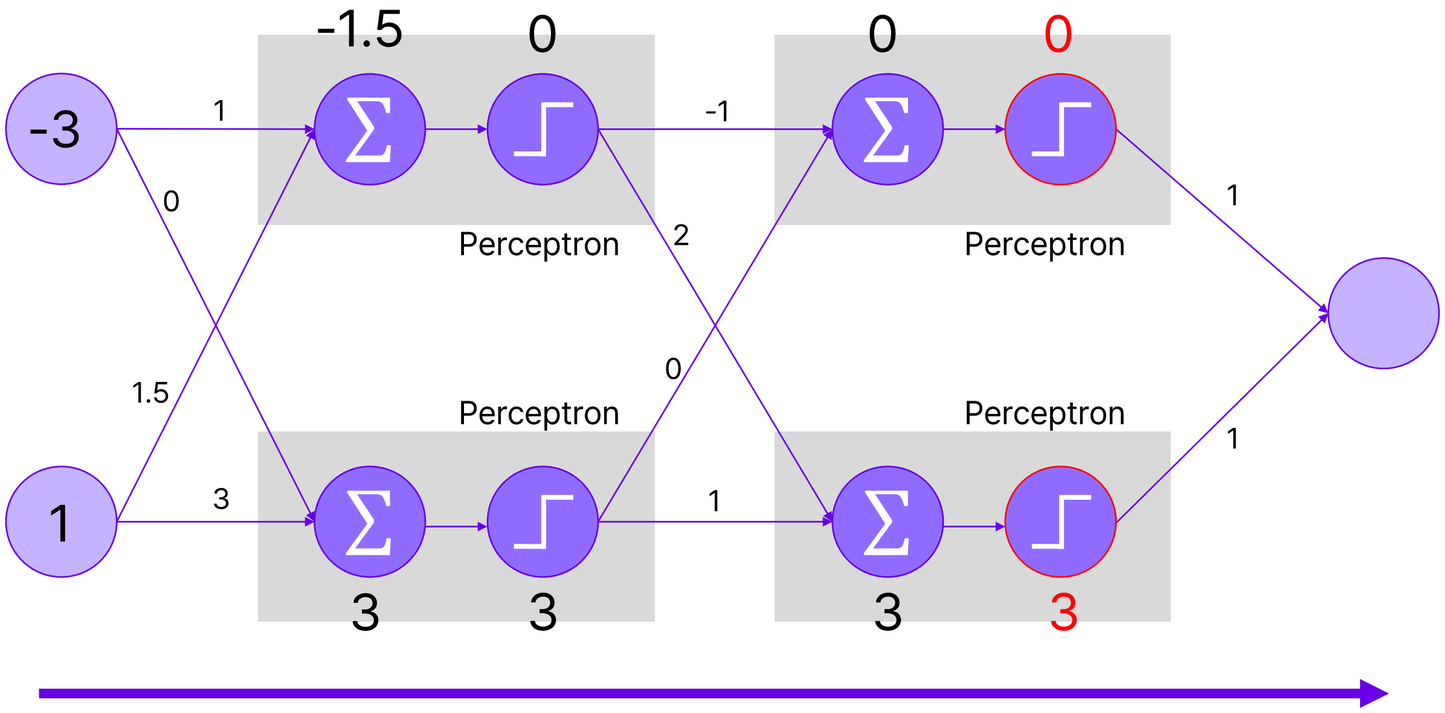
순전파 (Forward propagation)

- 활성 함수가 ReLU 일 때, 출력값을 직접 계산해봅시다.



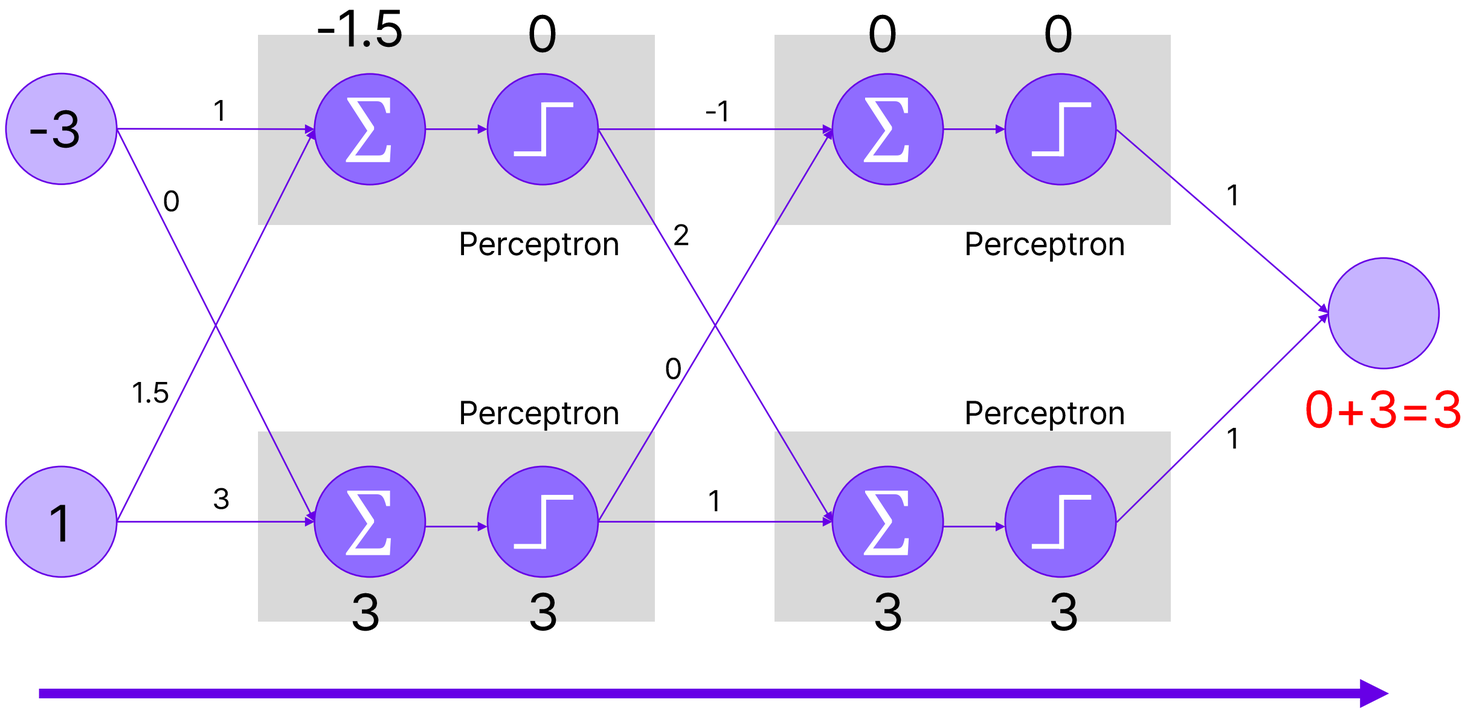
순전파 (Forward propagation)

- 활성 함수가 ReLU 일 때, 출력값을 직접 계산해봅시다.



순전파 (Forward propagation)

- 활성 함수가 ReLU 일 때, 출력값을 직접 계산해봅시다.

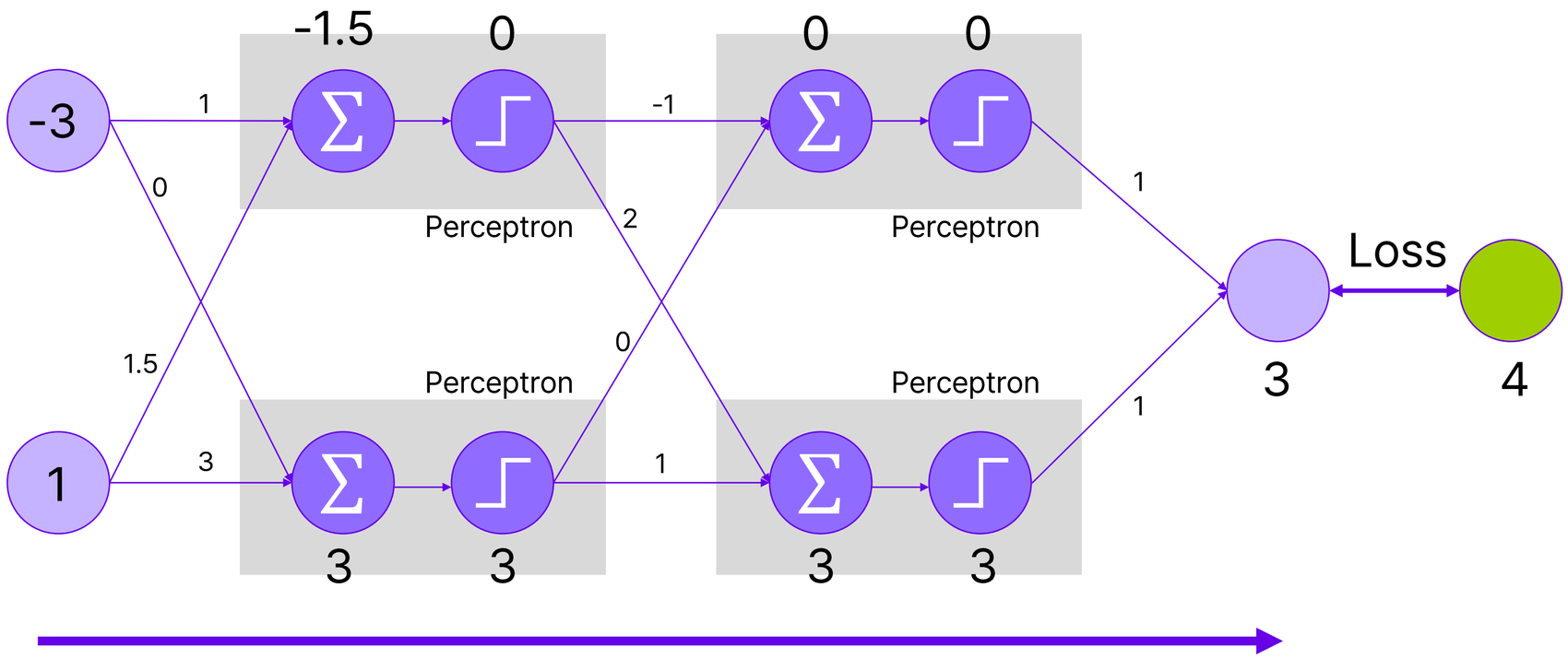


손실 함수 (Loss function)

- 머신러닝의 목표는 **잔차**를 최소화 하는 것
- 딥러닝의 목표는 **Loss function** 을 최소화 하는 것
- 대표적인 Loss function은 MSE (mean squared error)
  - $MSE = (\text{실제값} - \text{예측값})^2$

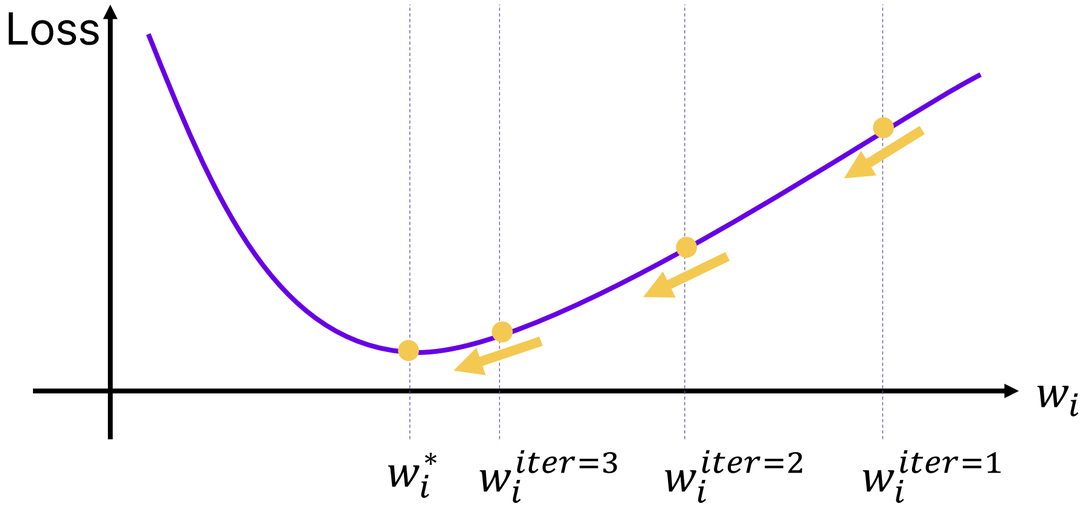
순전파 (Forward propagation)

- 실제 값이 4 였다면, MSE는 1 입니다.



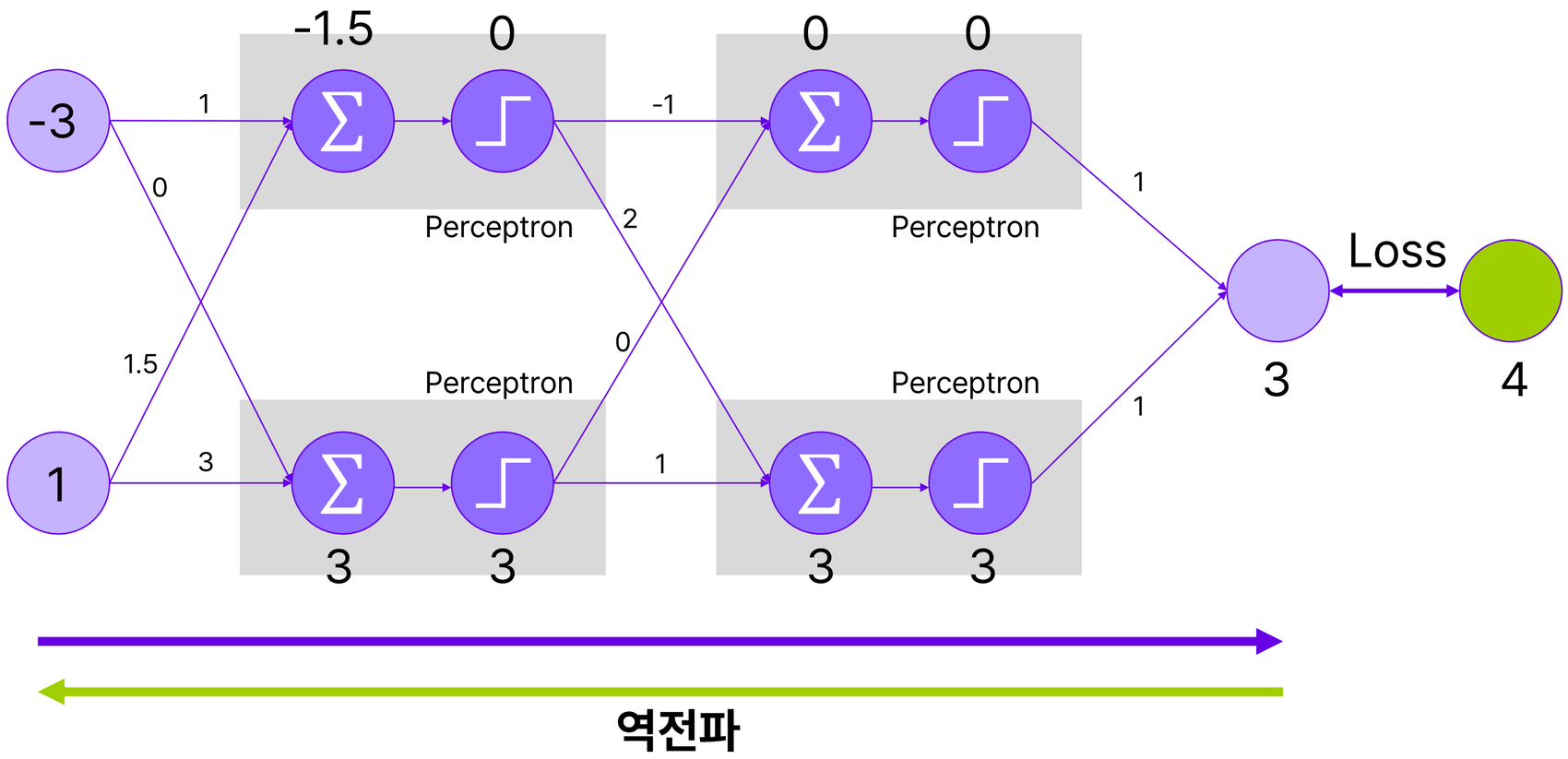
경사 하강법 (Gradient descent)

- Weight 를 Loss function 이 작아지게 업데이트 하는 방법
- 각 weight 마다 gradient 가 정해진다.
- Gradient의 방향으로 조금씩 weight 를 업데이트 한다.



역전파 (Back Propagation)

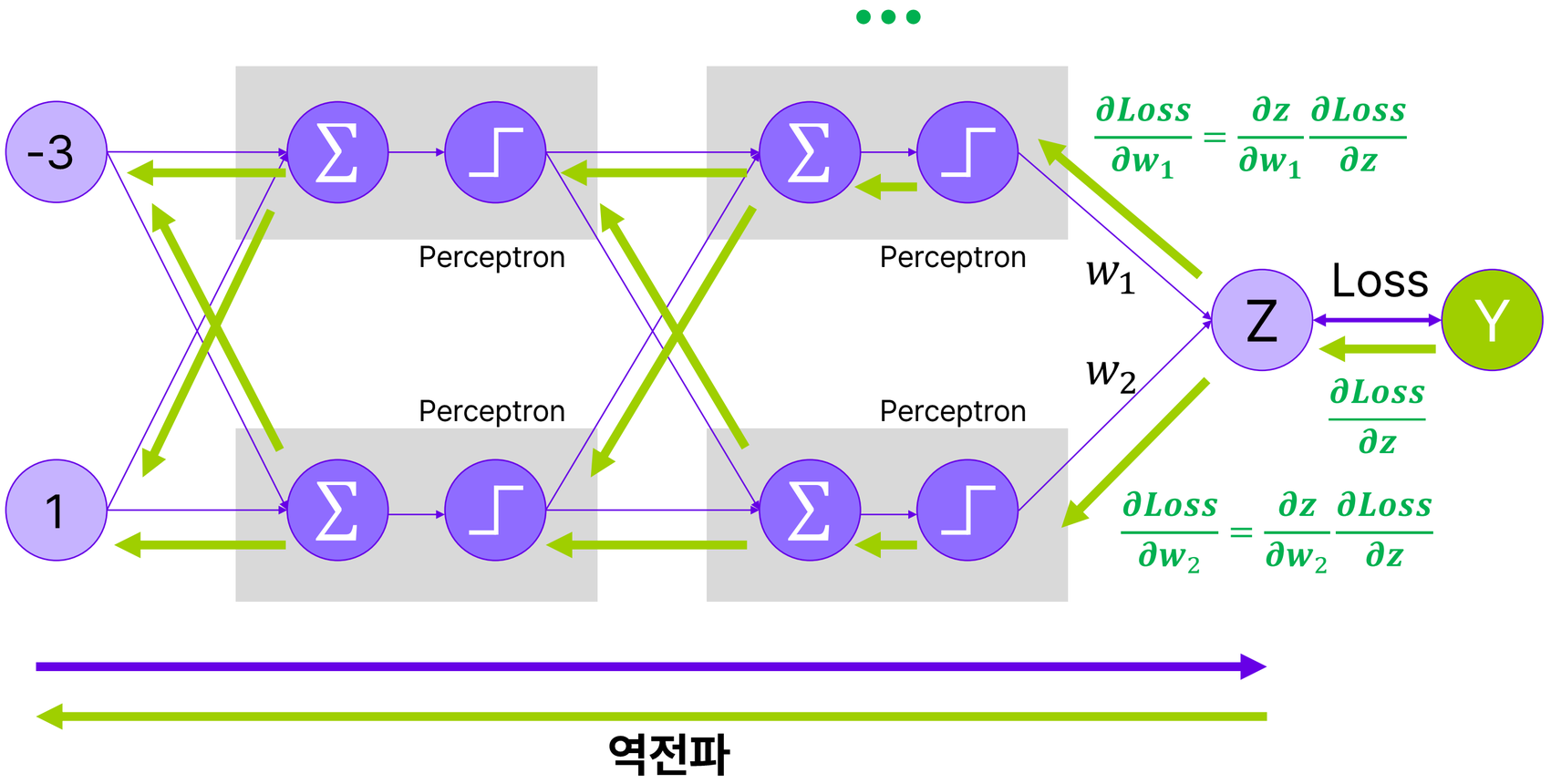
- 순전파의 반대 방향



Chain-Rule

- 각 weight의 gradient를 구하기 위해서, Loss 를 각 weight로 편미분 해야함
  - $\frac{\partial Loss}{\partial w_i}$
- 하지만, 모든 weight 에 대해 바로 편미분 값을 구하는 것은 불가능
- Chain-Rule 을 이용!

Chain-Rule


-3
1
Perceptron
Perceptron
Perceptron
Perceptron
w_1
w_2
Z
Loss
Y
$\frac{\partial Loss}{\partial w_1} = \frac{\partial z}{\partial w_1}\frac{\partial Loss}{\partial z}$
$\frac{\partial Loss}{\partial z}$
$\frac{\partial Loss}{\partial w_2} = \frac{\partial z}{\partial w_2}\frac{\partial Loss}{\partial z}$
역전파

Chain-Rule

- MLP 는 복잡해 보이지만, 사실은 미분 가능한 간단한 연산들의 조합임
  - 더하기
  - 곱하기
  - 활성함수

- Chain-Rule 은 복잡한 연산을 기본 연산으로 나누어 미분하는 방법
- 모든 weight 의 Loss 에 대한 gradient 계산 가능!

다층 퍼셉트론 모델의 학습 방법

- 역전파 알고리즘

순서	행동
0	MLP 모델 초기화
1	첫번째 데이터를 이용해 순전파, 오차 계산, 오차 역전파 계산, 학습 계수 갱신
2	두번째 데이터를 이용해 순전파, 오차 계산, 오차 역전파 계산, 학습 계수 갱신
3	세번째 데이터를 이용해 순전파, 오차 계산, 오차 역전파 계산, 학습 계수 갱신
...	
N	마지막 데이터를 이용해 순전파, 오차 계산, 오차 역전파 계산, 학습 계수 갱신, 1 에폭 끝.

04

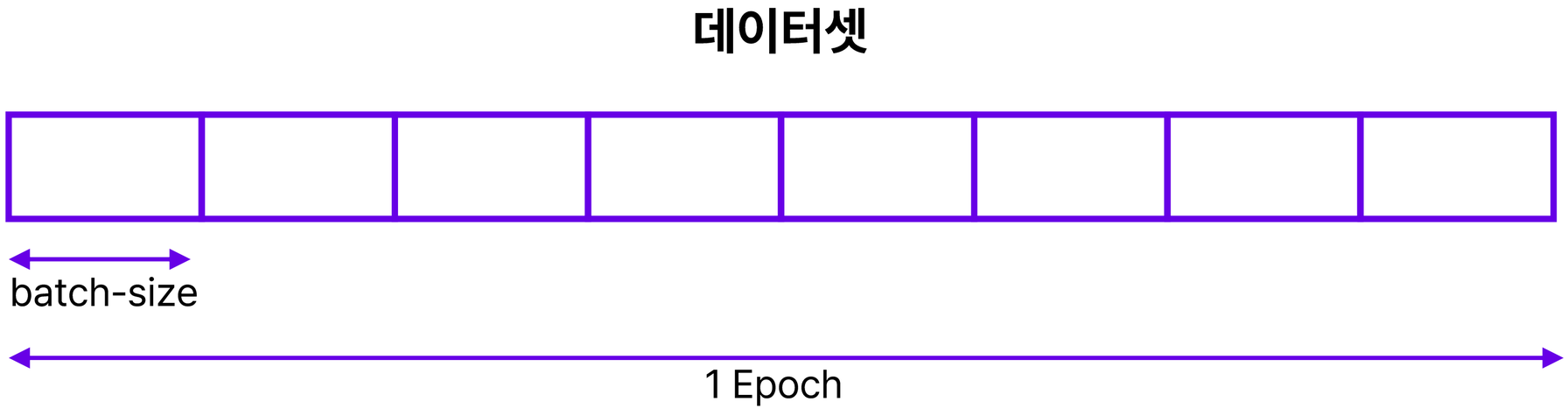
최적화 (Optimization)

최적화 (Optimization)

- 역전파를 계산한 후, **weight 를 업데이트** 하는 방법
- 대표적인 최적화 방법
  - Stochastic Gradient Descent (SGD)
  - Adam
  - RMSProp
  - Momentum
  - Newton method
  - AdaGrad
  - ...

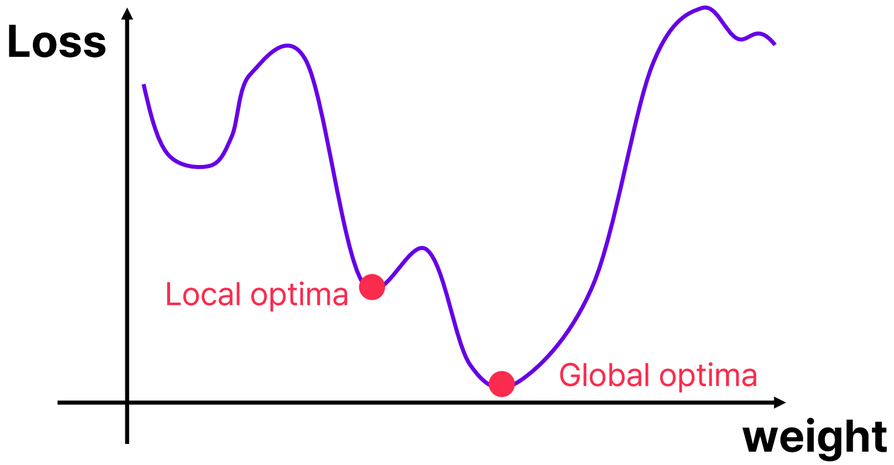
딥러닝 모델의 데이터 전처리

- Mini-batch: 전체 데이터셋 중 한 부분
- Batch-size: mini-batch의 크기
- Iteration: mini-batch 에 대해 역전파 알고리즘 시행 횟수
- Epoch: 전체 데이터셋에 대해 학습 시행 횟수



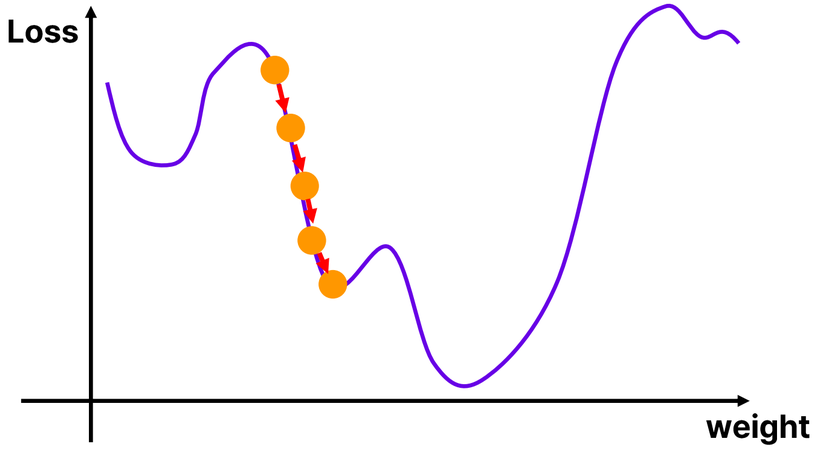
Mini-batch

- 데이터 중 일부를 소규모 집합 단위로 나눈 부분 데이터
- Mini-batch 를 사용하지 않고, 전체 데이터를 한번에 학습하게 되면,
  - 메모리 사용량이 많아 연산이 불가능 할 수 있음
  - **Local Minima** 에 빠지게 될 확률이 높음

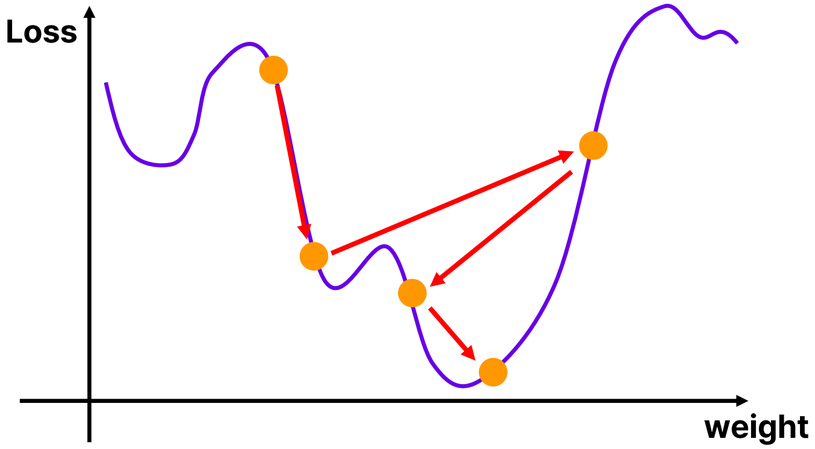


확률적 경사 하강법 (Stochastic Gradient Descent; SGD)

- **Mini-batch** 를 이용하여 학습하는 방법
- 학습 속도가 빠름
- Local minima 에 빠질 확률이 적음
  - 한번에 적은 데이터를 보기 때문에, Gradient 의 변동성이 비교적 큼
  - Overshooting 에 의해 Local minima 에 빠질 확률이 적어짐

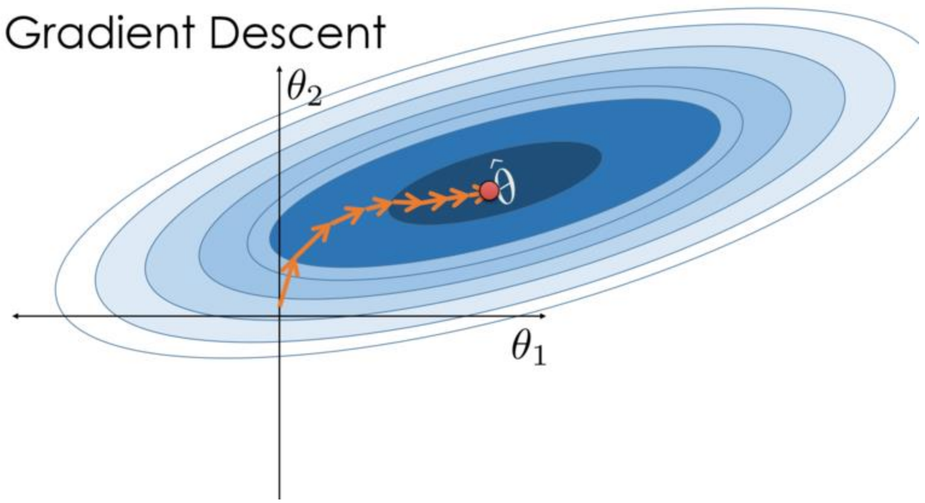


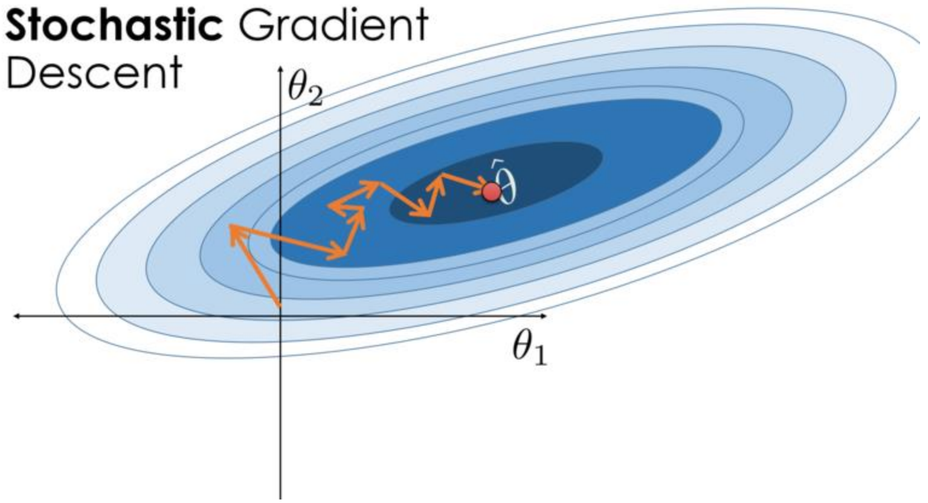
Mini-batch 이용하지 않았을 경우



Mini-batch 이용하였을 경우

확률적 경사 하강법 (Stochastic Gradient Descent; SGD)

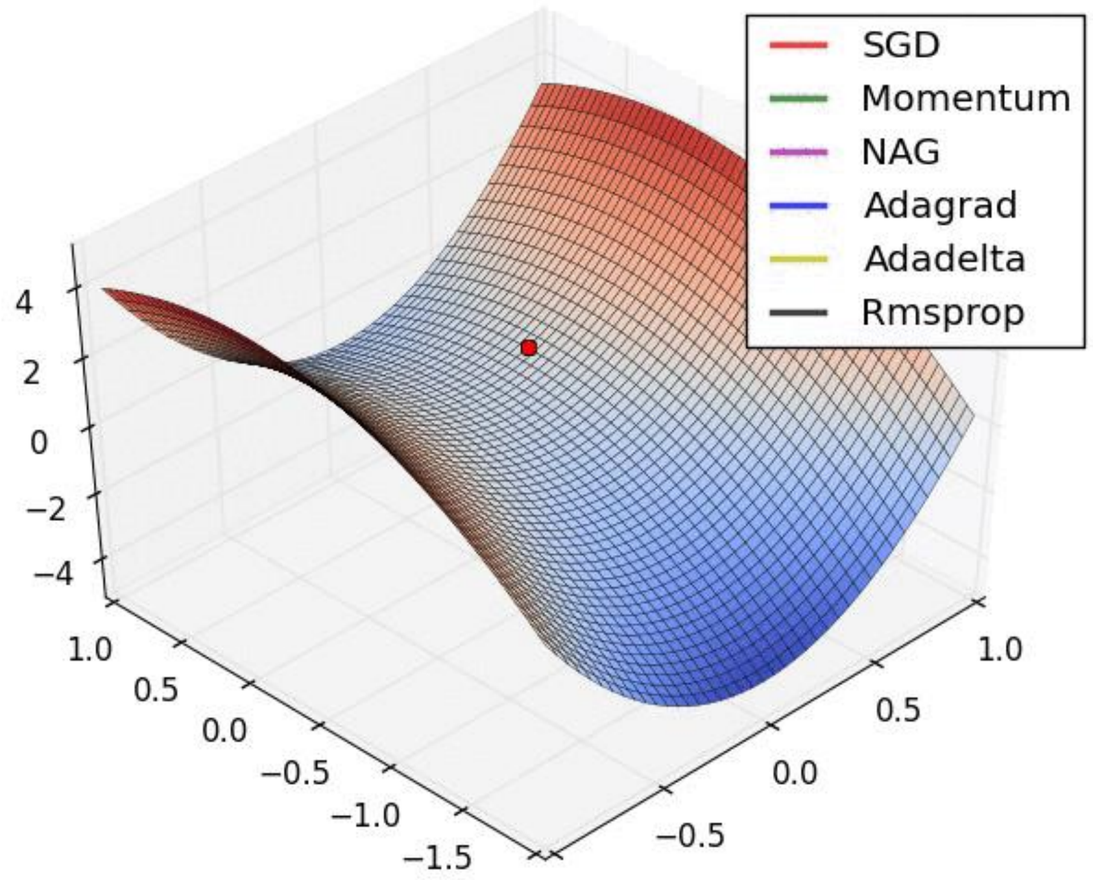




여러가지 최적화 방법들

- **RMSProp**: 지수가중이동평균을 사용하여 최신 기울기를 더 크게 반영하는 학습 방법
- **Momentum**: 이전 step의 학습 정도를 반영하는 학습 방법
- **Adagrad**: 많이 갱신된 가중치는 학습률을 낮추는 학습 방법
- **Adam** : Adagrad, Momentum, RmsProp의 방법론을 종합한 학습 방법
- **Newton Method**: 일차 미분이 아니라 이차 미분을 사용하는 학습 방법

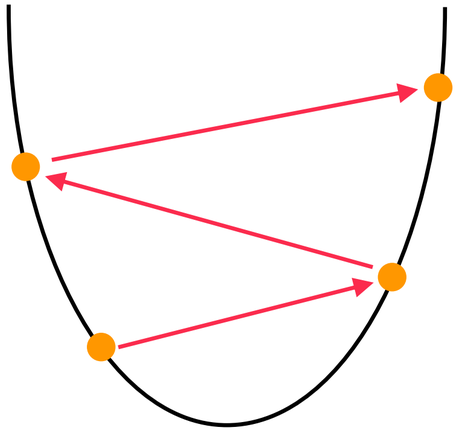
여러가지 최적화 방법들


SGD
Momentum
NAG
Adagrad
Adadelta
Rmsprop
4
2
0
−2
−4
1.0
0.5
0.0
−0.5
−1.0
−1.5
1.0
0.5
0.0
−0.5

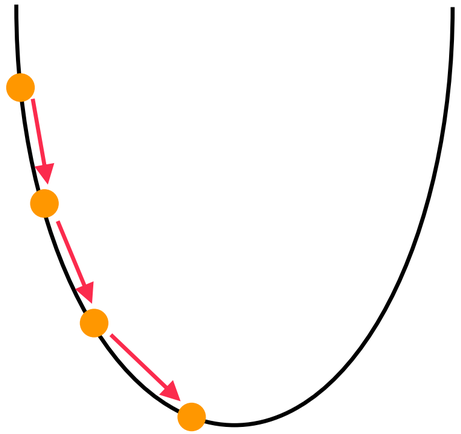
학습률 (Learning rate)

- Learning rate / Step size
- 경사하강법 알고리즘은 계산된 gradient 에 학습률을 곱한 값을 이용하여 weight를 업데이트
- 학습률이 너무 클 경우
  - Loss가 무질서하게 이탈
- 학습률이 너무 작을 경우
  - 학습 시간이 너무 오래 걸림
  - Global minima 에 도달이 어려움

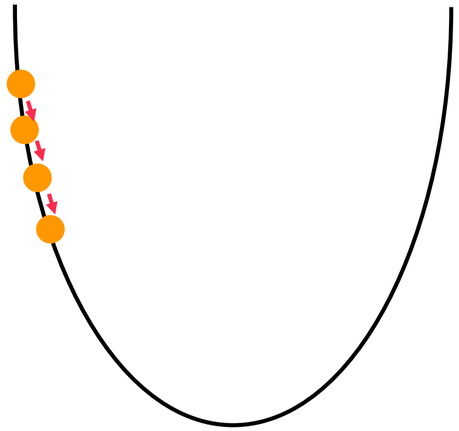
학습률 (Learning rate)



학습률이 너무 클 때

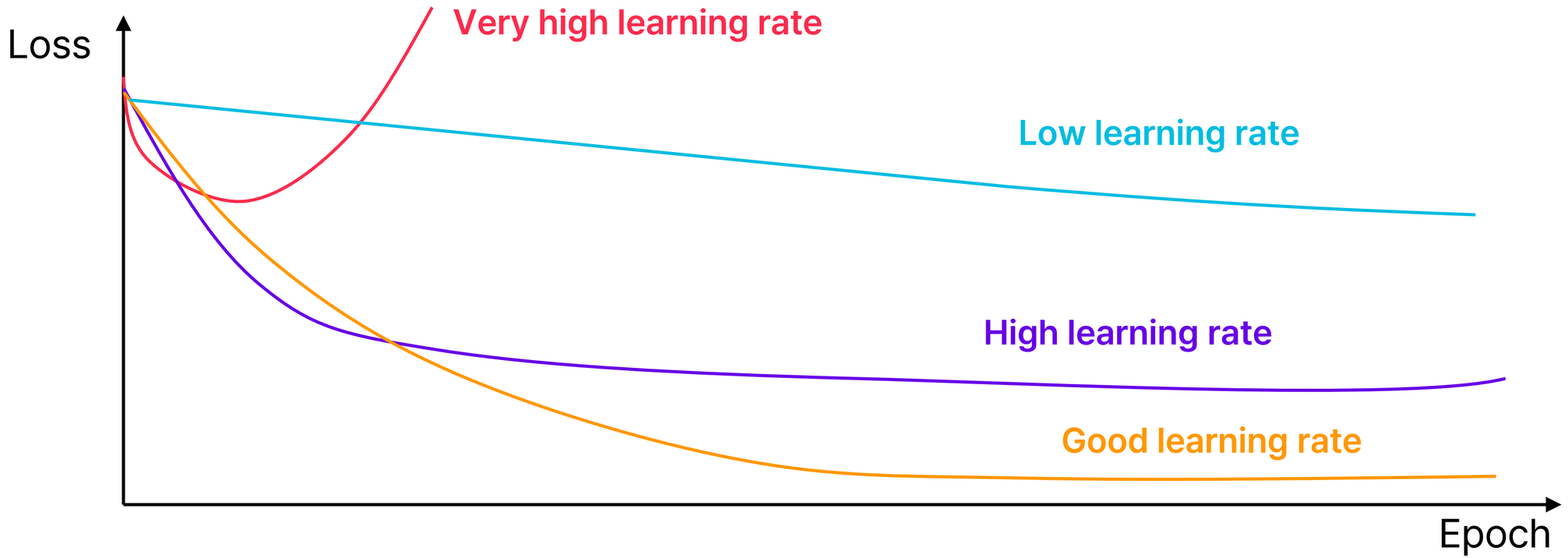


학습률이 너무 적당할 때



학습률이 너무 작을 때

학습률 (Learning rate)



05

MLP 모델의 구현

Keras

- 파이썬으로 작성된 오픈소스 신경망 라이브러리
- Tensorflow 기반
- 신경망 라이브러리 (Pytorch, Tensorflow, ...) 중 가장 간단함

Keras의 흐름

모델 정의

- 은닉층의 개수
- 유닛의 수
- 활성함수

학습방법 설정

- 최적화 방법
- 학습률
- Loss function

학습

- 학습데이터를 이용하여 MLP 모델 학습
- Batch-size
- Epoch 수

예측

- 테스트 데이터를 이용하여 출력값 예측

1. 모델 정의

```
import tensorflow as tf

MLP_model = tf.keras.Sequential([
    tf.keras.layers.Dense(64, activation = 'relu'),
    tf.keras.layers.Dense(32, activation = 'relu'),
    tf.keras.layers.Dense(stage2['train_y'].shape[1])
])
```

2. 학습 방법 설정

```
MLP_model.compile(loss = 'mse',
                  optimizer = tf.keras.optimizers.SGD(learning_rate = 1e-2),
                  metrics = ['mse'])
])
```

3. 모델 학습 및 예측

```
history = MLP_model.fit(X, y, epochs = 50, batch_size = 16)
```

```
pred = MLP_model.predict(test_X)
```

Quiz

- 다항 회귀 분석으로 학습하기에 적합하지 않은 관계를 고르세요.

① $y = 3x^2 + 3 + \log(2)$

② $y = x^{100} + 2$

③ $y = \log(x) + 3x$

④ $y = -3x^2 - e + 1$

Quiz

- MLP 모델의 학습 과정에 포함되지 않는 것을 고르세요.

① 순전파 계산

② 역전파 계산

③ 오차의 표준편차 계산

④ 학습 계수 갱신

Quiz

- 모델의 학습 계수를 갱신할 때, 그 step의 크기를 결정하는 hyper-parameter 는 무엇인가요?

1. 학습률