# 딥러닝으로 구현하는 자연어 처리

# 목차

1. 신경망 복습

2. 자연어와 단어의 분산 표현

3. word2vec

4. word2vec 속도 개선

5. 순환 신경망(RNN)

6. 게이트가 추가된 RNN

7. RNN을 사용한 문장 생성

8. 어텐션

9. 트랜스 포머 모델

# 1. 신경망 복습

**1.1 수학과 파이썬 복습**

1.2 신경망 추론

1.3 신경망의 학습

1.4 신경망으로 문제를 풀다.

### 백터와 행렬의 예

$$\begin{bmatrix}1\\2\\3\end{bmatrix} \qquad \begin{bmatrix}1 & 2\\3 & 4\\5 & 6\end{bmatrix}$$

행

열

### 벡터의 표현법

$$\begin{bmatrix}1\\2\\3\end{bmatrix} \qquad [1 \quad 2 \quad 3]$$

열벡터 행벡터

브로드캐스트의 예1 : 스칼라 값인 10이 2x2 행렬로 처리된다.

| 1 | 2 |
|---|---|
| 3 | 4 |

* | 10 | = | 1 | 2 | * | 10 | 10 | = | 10 | 20 |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | 3 | 4 | | 10 | 10 | | 30 | 40 |

브로드캐스트의 예2 : 일차원 배열인 10,20이 2x2 행렬로 처리된다.

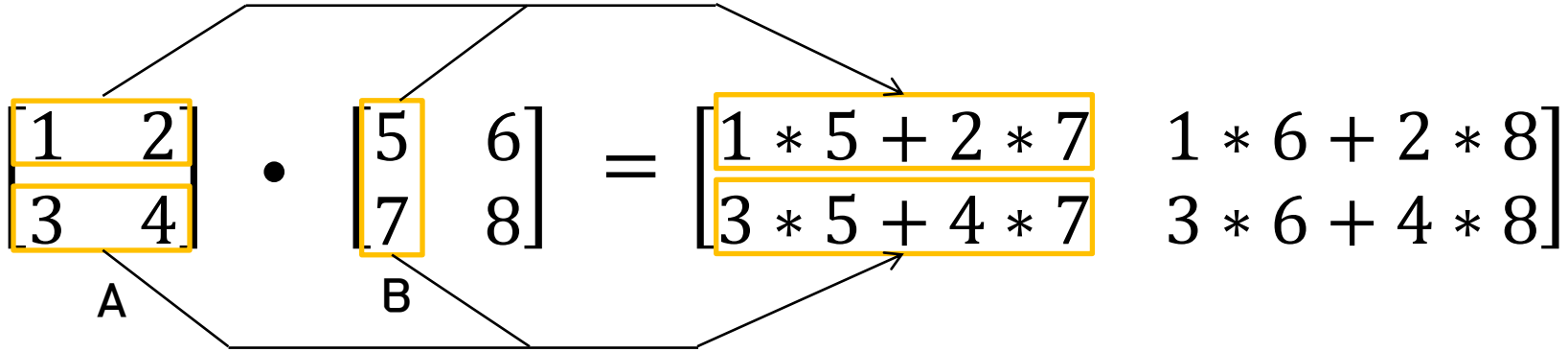
| 1 | 2 |
|---|---|
| 3 | 4 |

* | 10 | 20 | = | 1 | 2 | * | 10 | 20 | = | 10 | 40 |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | 3 | 4 | | 10 | 20 | | 30 | 80 |

### 벡터의 내적

$$x \cdot y = x_1 y_1 + x_2 y_2 + \cdots + x_n y_n$$

### 행렬의 곱



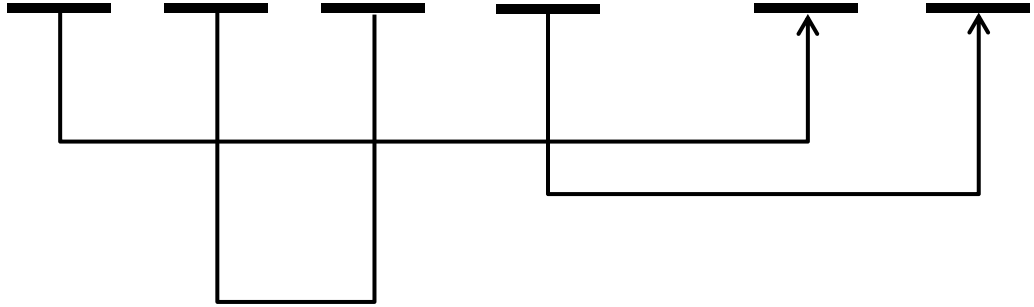

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \cdot \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} = \begin{bmatrix} 1*5+2*7 & 1*6+2*8 \\ 3*5+4*7 & 3*6+4*8 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \cdot \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} = \begin{bmatrix} 19 & 22 \\ 43 & 50 \end{bmatrix}$$

형상 확인 : 행렬의 곱에서는 대응하는 차원의 원소 수를 일치시킨다.

$$A \cdot B = C$$

$$3 \times 2 \quad 2 \times 4 \qquad 3 \times 4$$



일치시킨다.

# 1. 신경망 복습

1.1 수학과 파이썬 복습

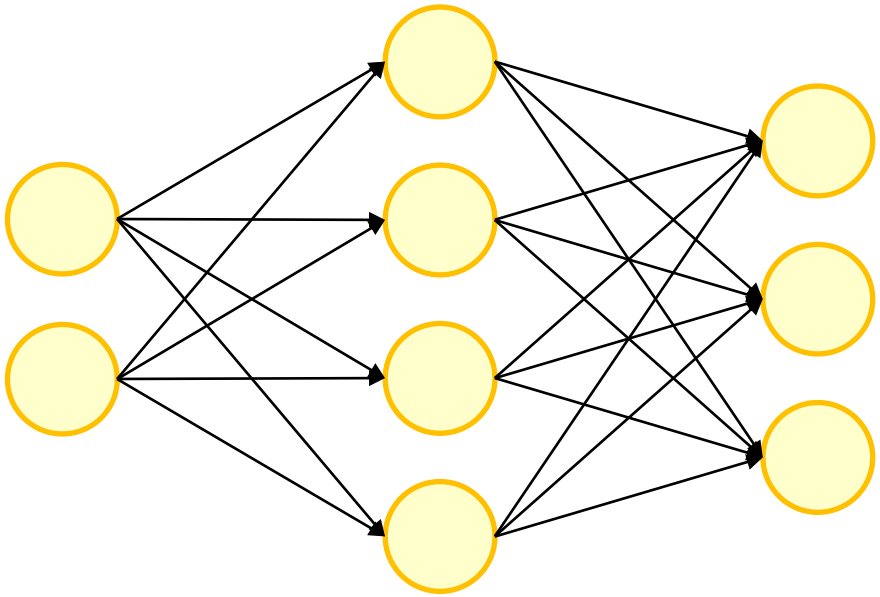
**1.2 신경망 추론**

1.3 신경망의 학습

1.4 신경망으로 문제를 풀다.

# 신경망 추론 전체 그림

## 신경망의 예



# 신경망 추론 전체 그림

신경망이 수행하는 계산 수식(단일층)

$$h_1 = x_1 w_{11} + x_2 w_{21} + b_1$$

신경망이 수행하는 계산 수식(여러층)

$$(h_1, h_2, h_3, h_4) = (x_1, x_2) \begin{pmatrix} w_{11} & w_{12} & w_{13} & w_{14} \\ w_{21} & w_{22} & w_{23} & w_{24} \end{pmatrix} + (b_1, b_2, b_3, b_4)$$

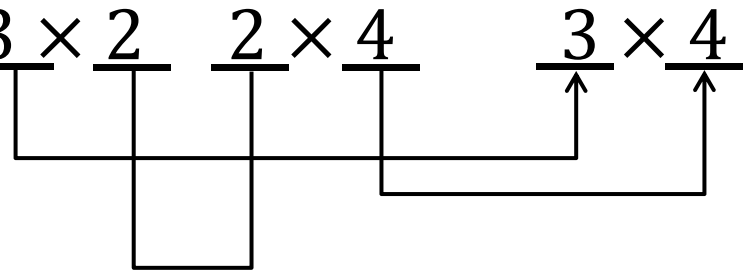
$$h = xW + b$$

# 신경망 추론 전체 그림

형상 확인 : 대응하는 차원의 원소 수가 일치함(편향은 생략)

A · B = C

$3 \times 2 \quad 2 \times 4 \quad 3 \times 4$

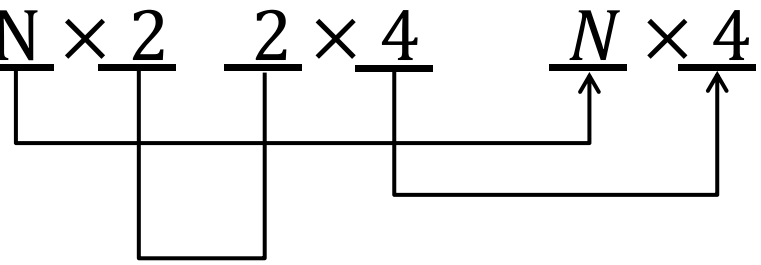


일치시킨다.

형상 확인 : 미니배치 버전의 행렬 곱(편향은 생략)

x · W = h

$N \times 2 \quad 2 \times 4 \quad N \times 4$

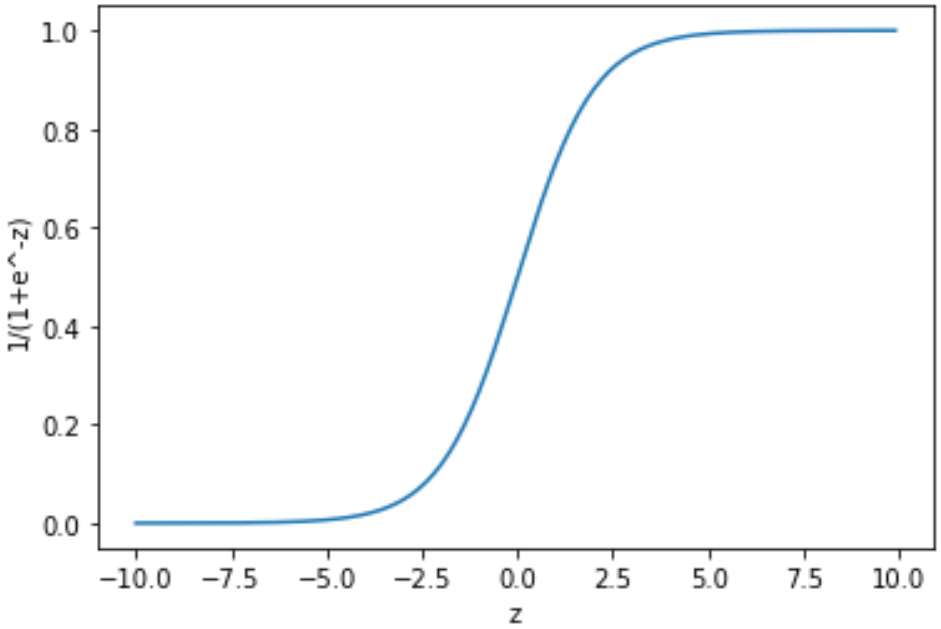


일치시킨다.

# 신경망 추론 전체 그림

**시그모이드 함수**

$$\sigma(\mathrm{x}) = \frac{1}{1 + \exp(-\mathrm{x})}$$

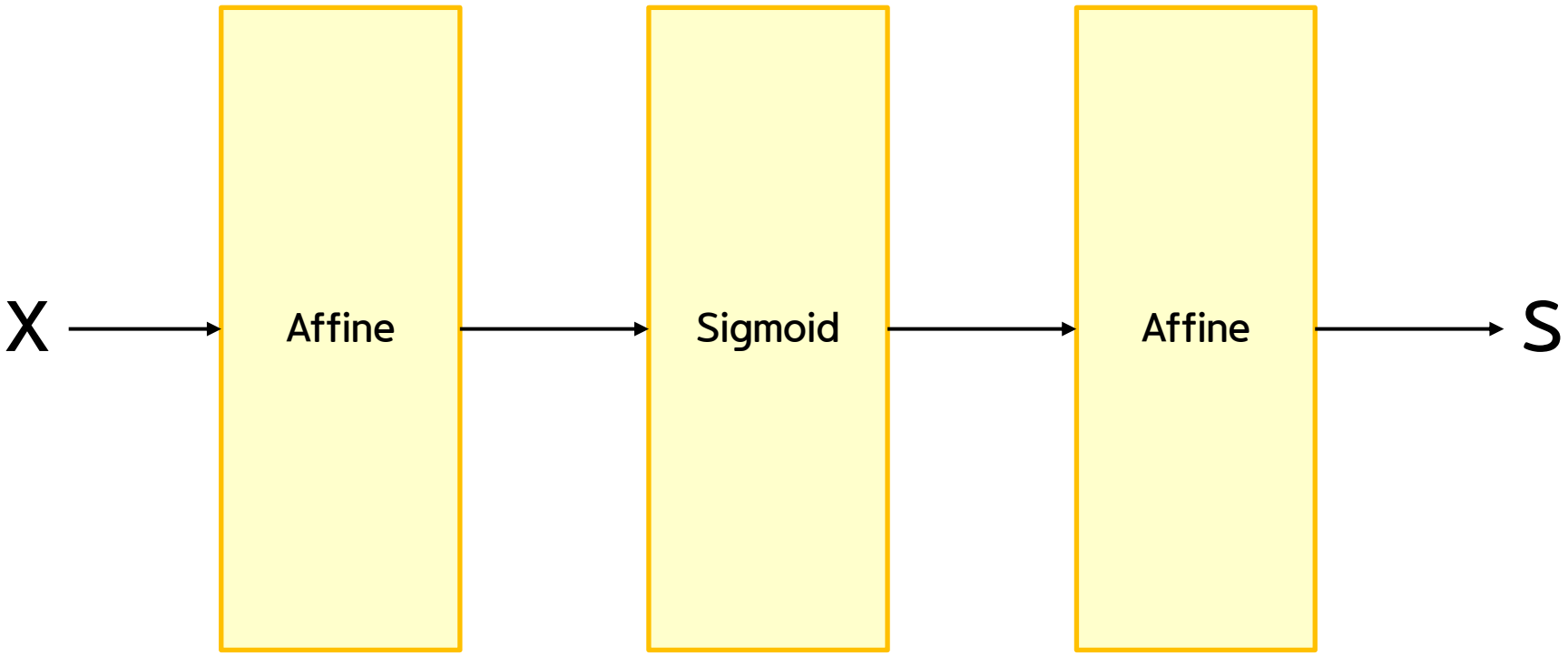


# 계층으로 클래스화 및 순전파 구현

- 모든 계층은 forward()와 backward() 메서드를 가진다.
- 모든 계층은 인스턴스 변수인 param와 grads를 가진다.

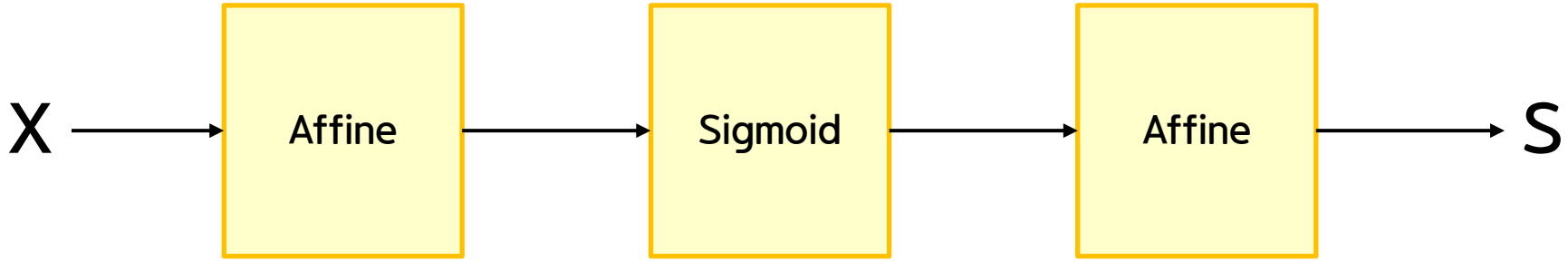
※ 소스 참조

구현해볼 신경망의 계층 구성



# 계층으로 클래스화 및 순전파 구현

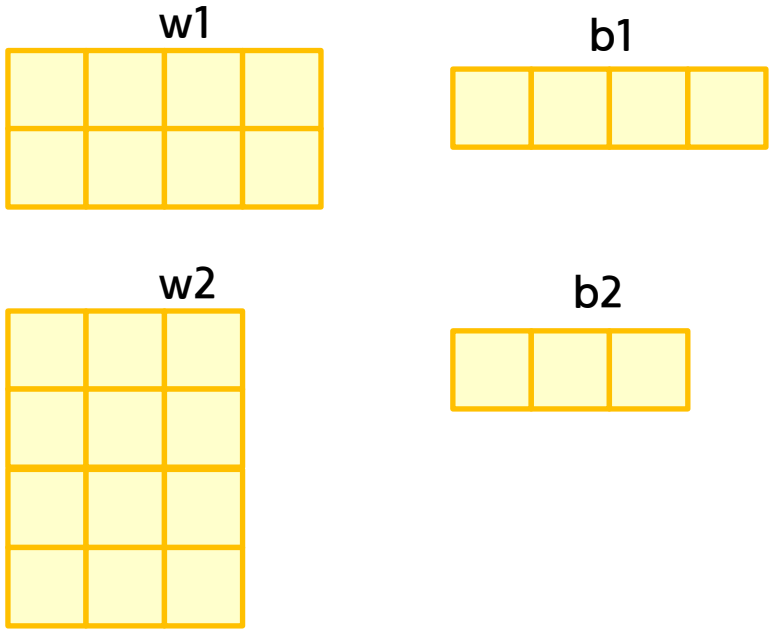
구현해볼 신경망의 계층 구성




```
class TwoLayerNet:
    def __init__(self, input_size, hidden_size, output_size):
        I, H, O = input_size, hidden_size, output_size

        # 가중치와 편향 초기화
        W1 = np.random.randn(I, H)
        b1 = np.random.randn(H)
        W2 = np.random.randn(H, O)
        b2 = np.random.randn(O)

model = TwoLayerNet(2, 4, 3)
```

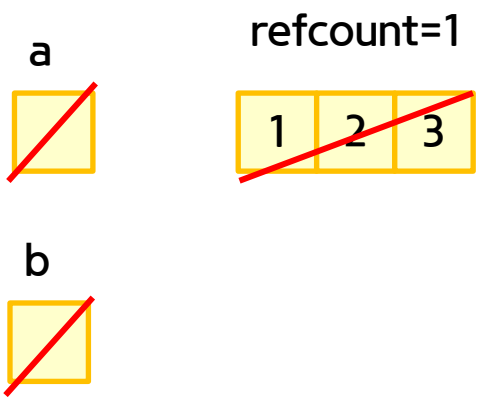
## 구현해볼 신경망의 계층 구성

```
class TwoLayerNet:
    def __init__(self, input_size, hidden_size, output_size):
        I, H, O = input_size, hidden_size, output_size

        # 가중치와 편향 초기화
        W1 = np.random.randn(I, H)
        b1 = np.random.randn(H)
        W2 = np.random.randn(H, O)
        b2 = np.random.randn(O)

model = TwoLayerNet(2, 4, 3)
```



## 구현해볼 신경망의 계층 구성

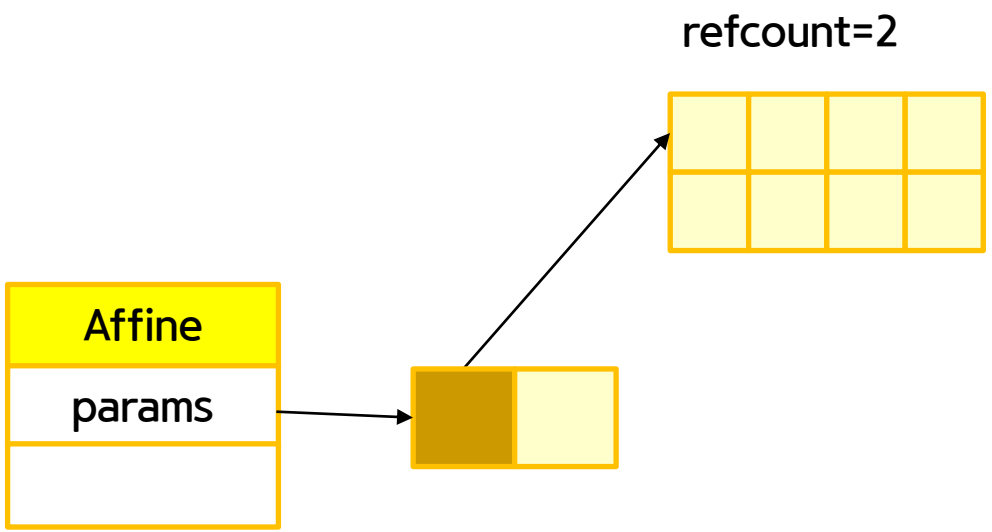
```
import sys
a = np.array([1,2,3])
print( sys.getrefcount(a) )
b = a
print( sys.getrefcount(a) )
print(b)
b[1] = 10
print(b)
print(a)
```




파이썬의 모든 객체는
heap 공간에 동적 할당 된다.

refcount가 다시 1이 되면
파이썬의 gc가 해당
메모리를 해지한다.

```
def __init__(self, input_size, hidden_size, output_size):
      I, H, O = input_size, hidden_size, output_size
    ...
    W1 = np.random.randn(I, H)
    ...
    Affine(W1, b1)
    ...
```

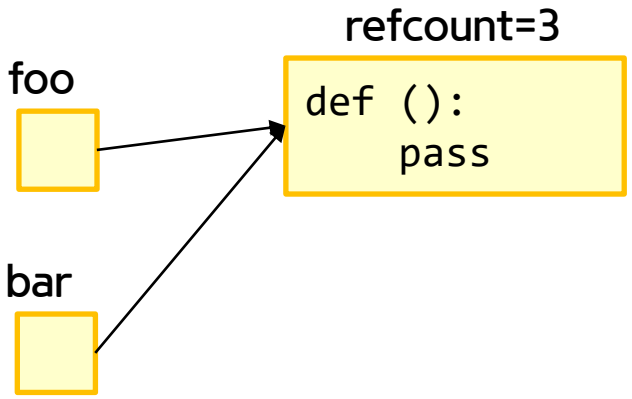



파이썬의 모든 객체는
heap 공간에 동적 할당 된다.

refcount가 다시 1이 되면
파이썬의 gc가 해당
메모리를 해지한다.

# 파이썬 레퍼런스 카운팅

```
import sys

def foo():
    pass

print( sys.getrefcount(foo) )
```



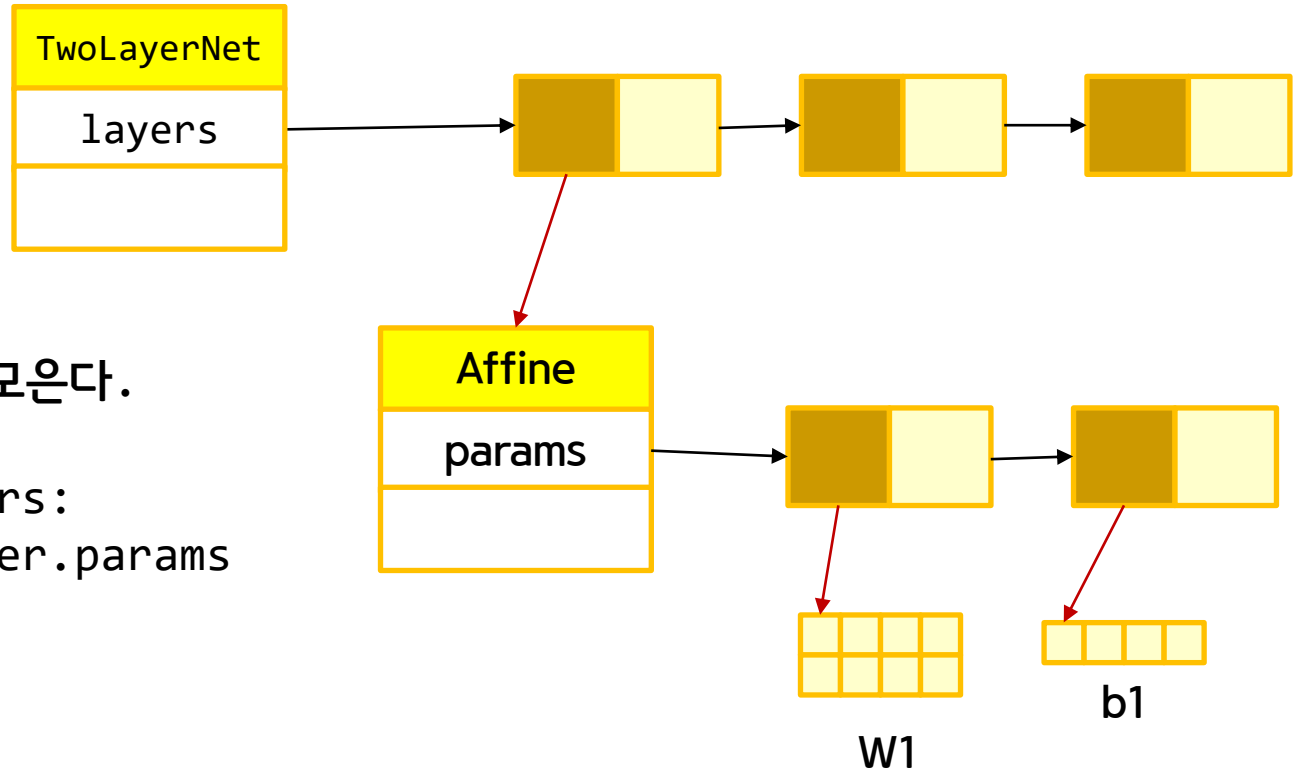

파이썬의 모든 객체는
heap 공간에 동적 할당 된다.

refcount가 다시 1이 되면
파이썬의 gc가 해당
메모리를 해지한다.

# 계층으로 클래스화 및 순전파 구현

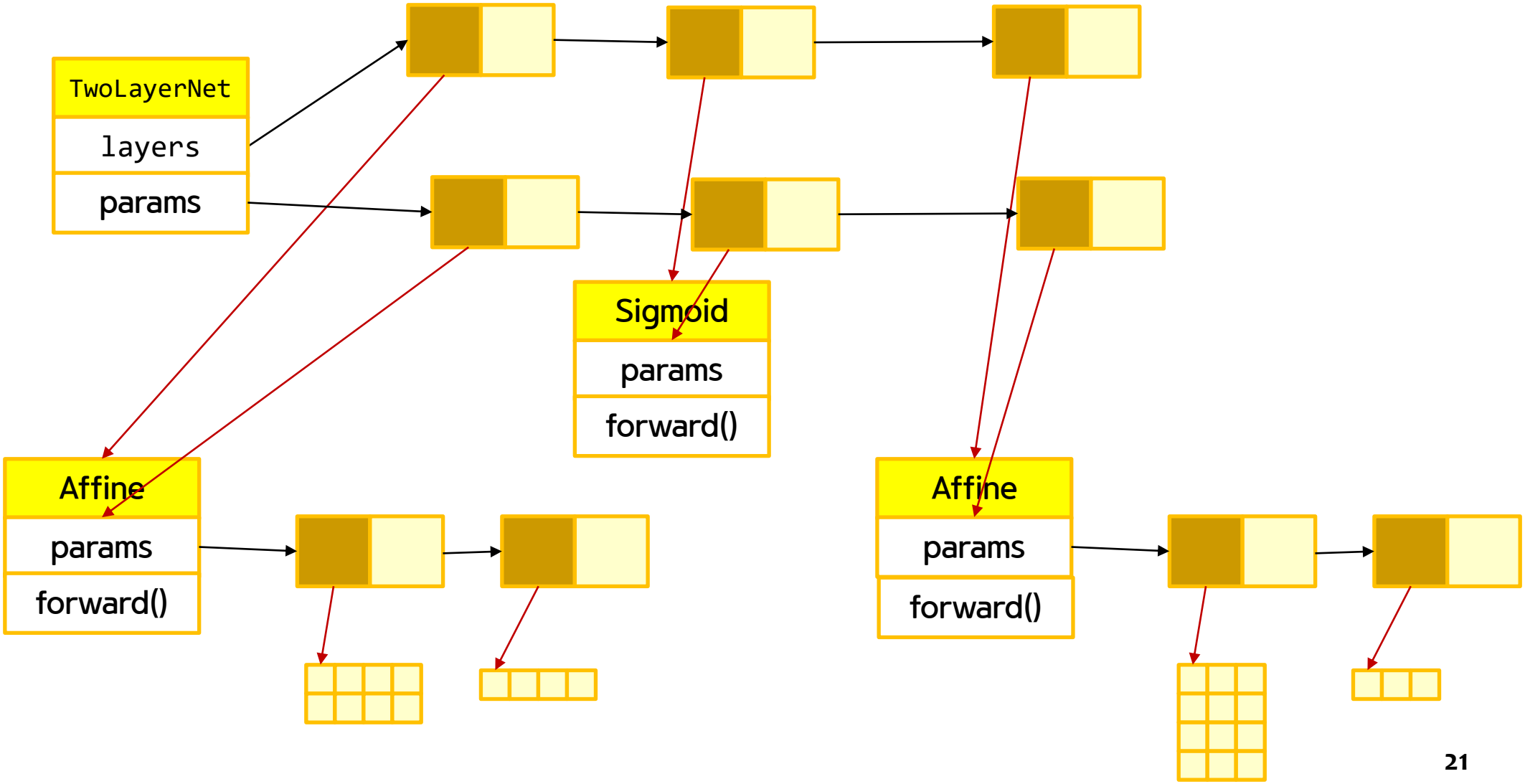
## 구현해볼 신경망의 계층 구성

```
class TwoLayerNet:
    def __init__(self, input_size, hidden_size, output_size):
        ...
        # 계층 생성
        self.layers = [
            Affine(W1, b1),
            Sigmoid(),
            Affine(W2, b2)
        ]

        # 모든 가중치를 리스트에 모은다.
        self.params = []
        for layer in self.layers:
            self.params += layer.params


model = TwoLayerNet(2, 4, 3)
...
```



# 계층으로 클래스화 및 순전파 구현

## 구현해볼 신경망의 계층 구성



# 1. 신경망 복습

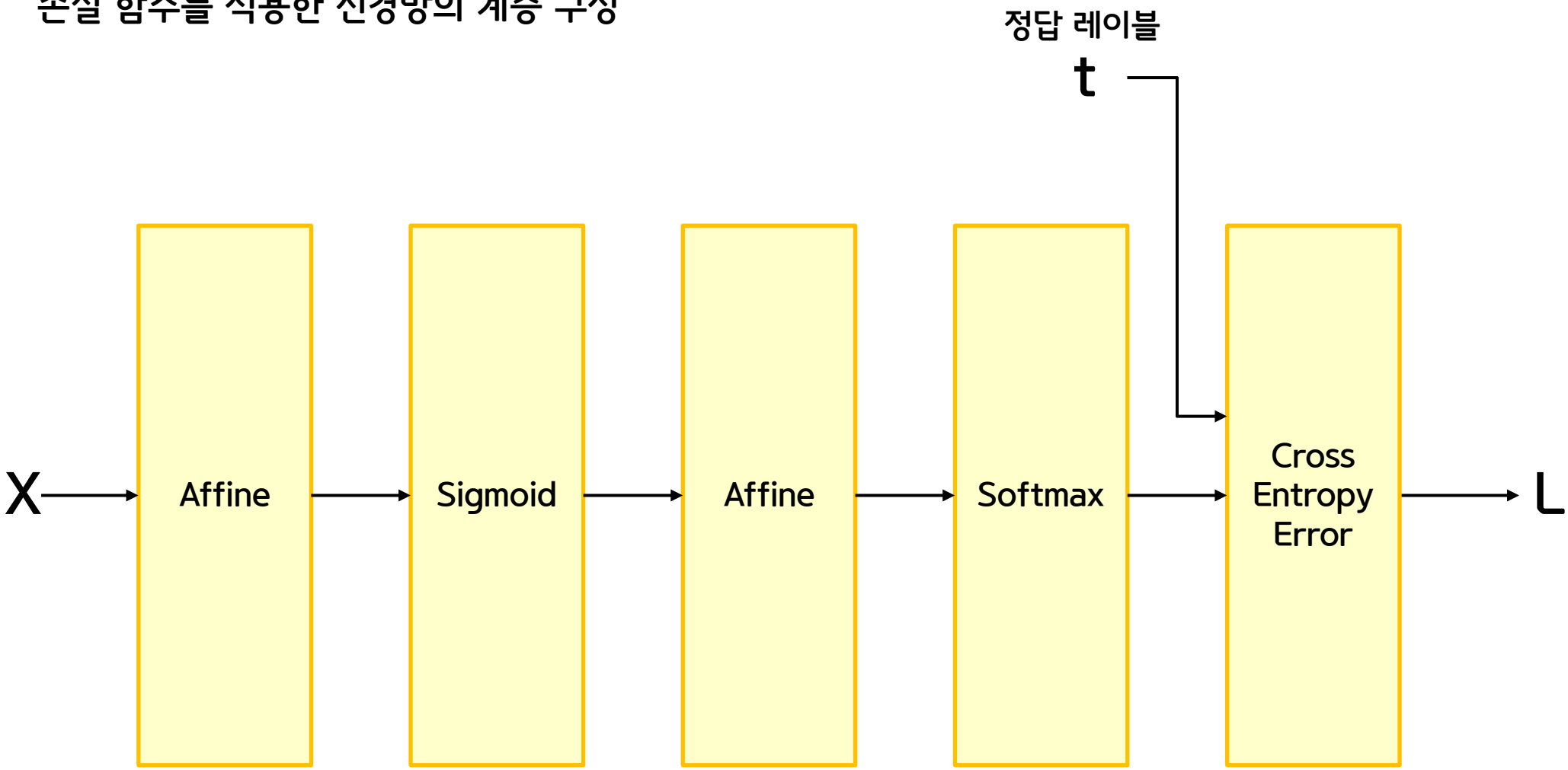
1.1 수학과 파이썬 복습

1.2 신경망 추론

**1.3 신경망의 학습**

1.4 신경망으로 문제를 풀다.

## 손실 함수를 적용한 신경망의 계층 구성



소프트맥스 함수 수식

$$y_k = \frac{\exp(s_k)}{\sum_{i=1}^{n} \exp(s_i)}$$

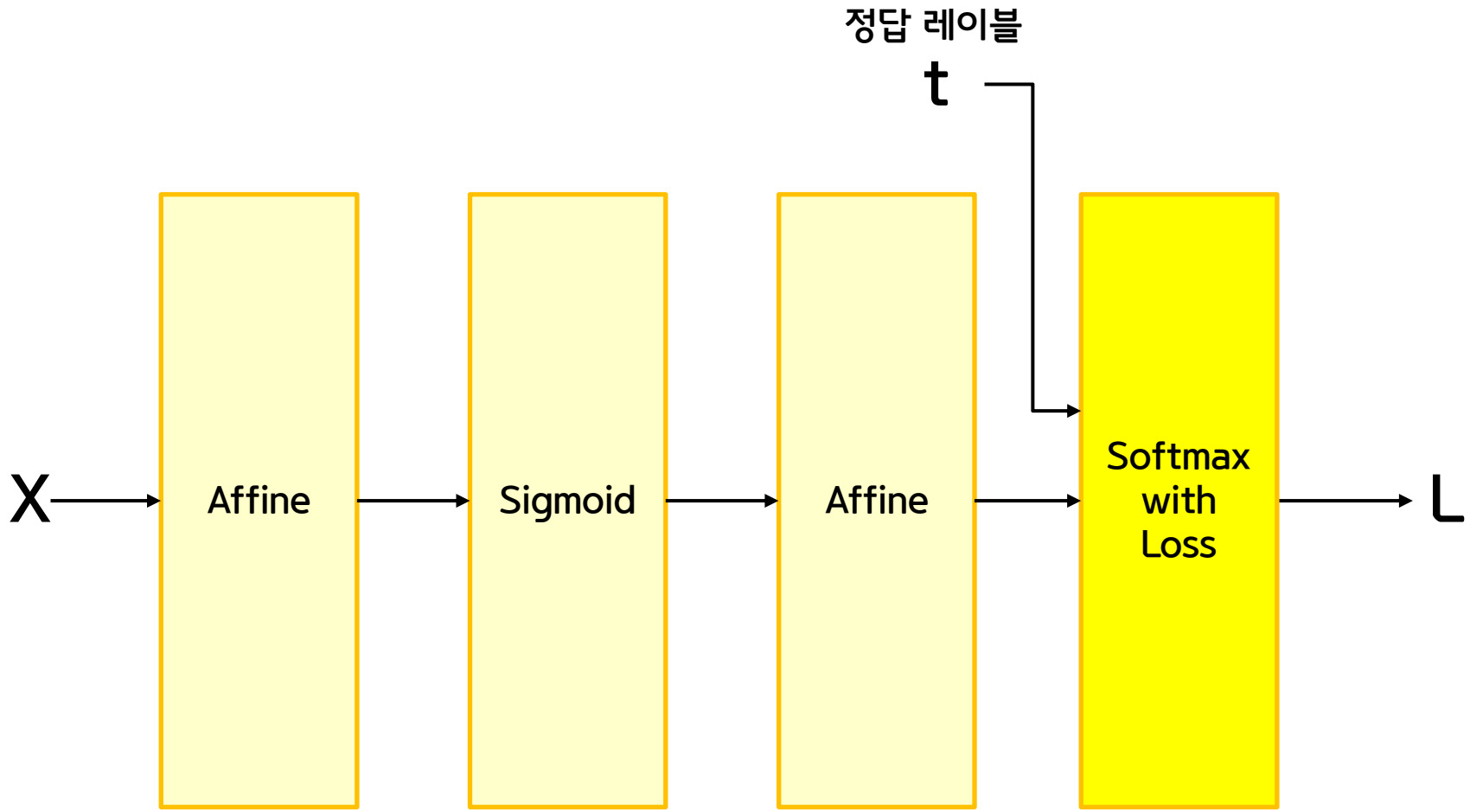
크로스 엔트로피 오차의 수식

$$L = -\sum_{k} t_k log y_k$$

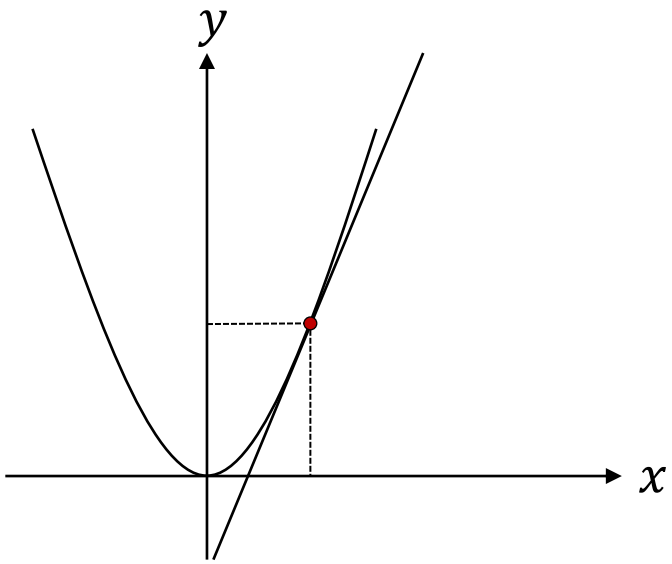
미니배치 처리를 고려한 크로스 엔트로피 오차의 수식

$$L = -\frac{1}{N}\sum_{n}\sum_{k} t_{nk} log y_{nk}$$

Softmax with Loss 계층을 이용하여 손실을 출력



$y = x^2$ 의 미분은 각 x 에서의 기울기를 나타낸다.




벡터인 경우 미분

$$\frac{\partial L}{\partial x} = \left( \frac{\partial L}{\partial x_1}, \frac{\partial L}{\partial x_2}, \cdots, \frac{\partial L}{\partial x_n} \right)$$

**행렬인 경우 미분**

$$\frac{\partial L}{\partial W} = \begin{pmatrix} \frac{\partial L}{\partial W_{11}} & \cdots & \frac{\partial L}{\partial W_{1n}} \\ \vdots & \ddots & \vdots \\ \frac{\partial L}{\partial W_{m1}} & \cdots & \frac{\partial L}{\partial W_{mn}} \end{pmatrix}$$

## 연쇄 법칙

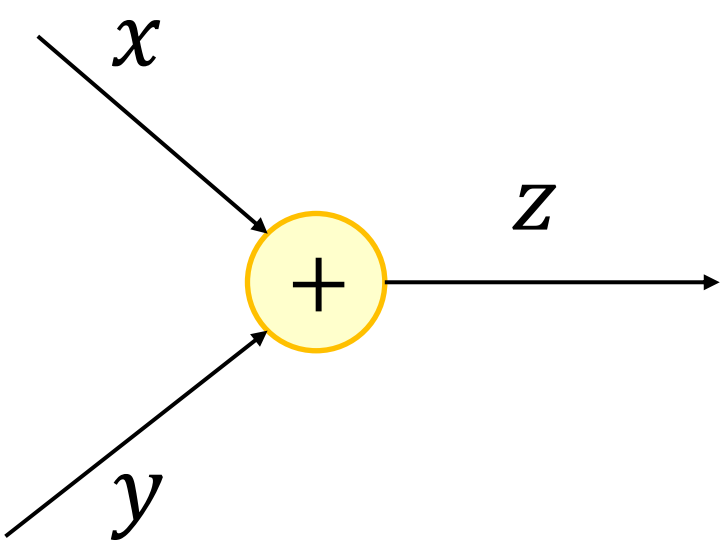
$$y = f(x)$$
$$z = g(y)$$

$$z = g(f(x))$$

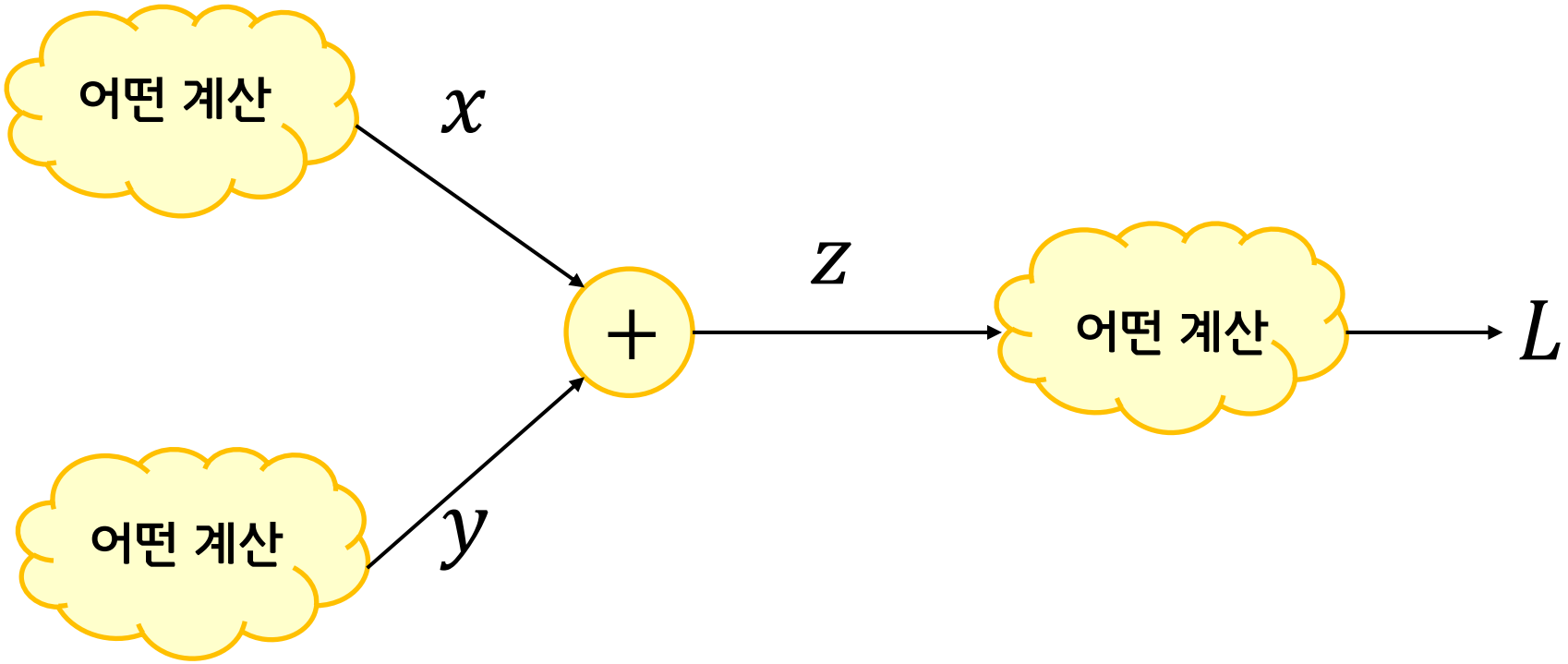
### 합성함수의 미분

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x}$$

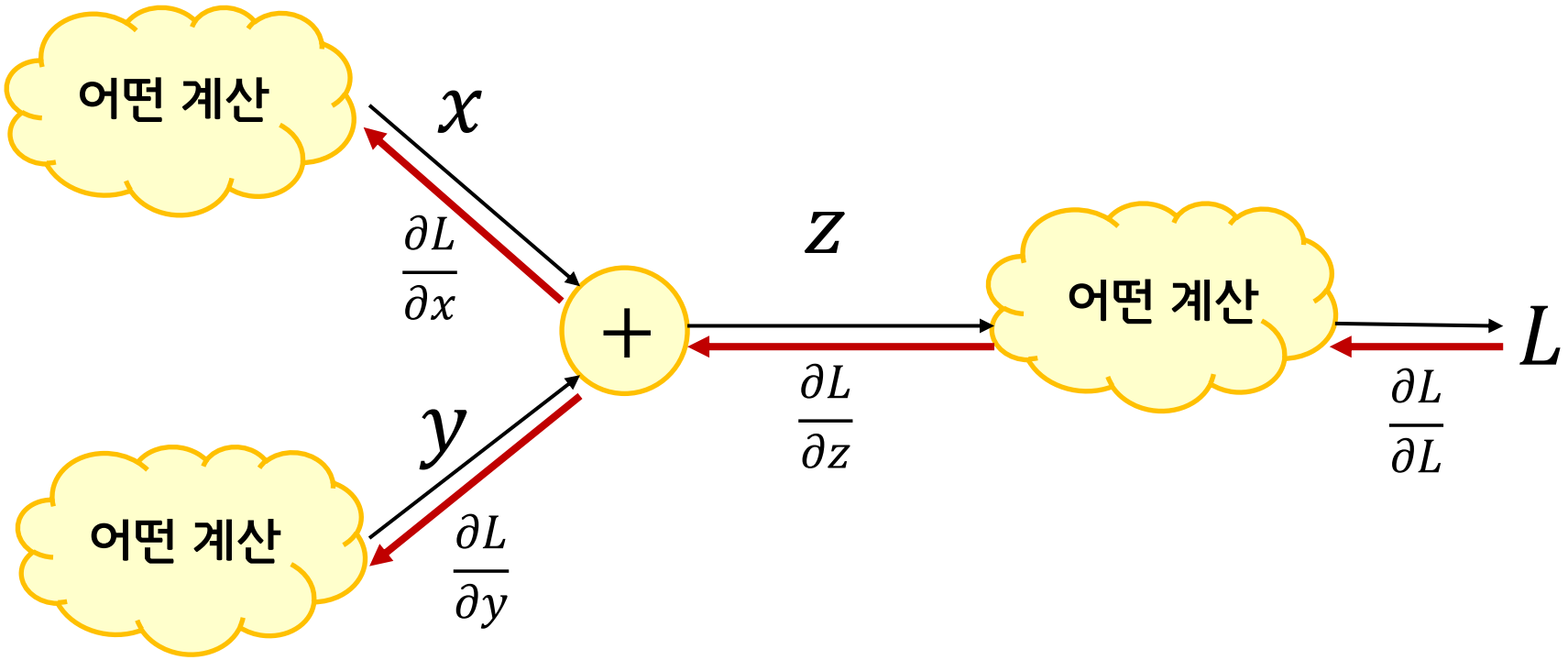
$z = x + y$ 를 나타내는 계산 그래프



앞뒤로 추가된 노드는 '복잡한 전체 계산'의 일부를 구성한다.

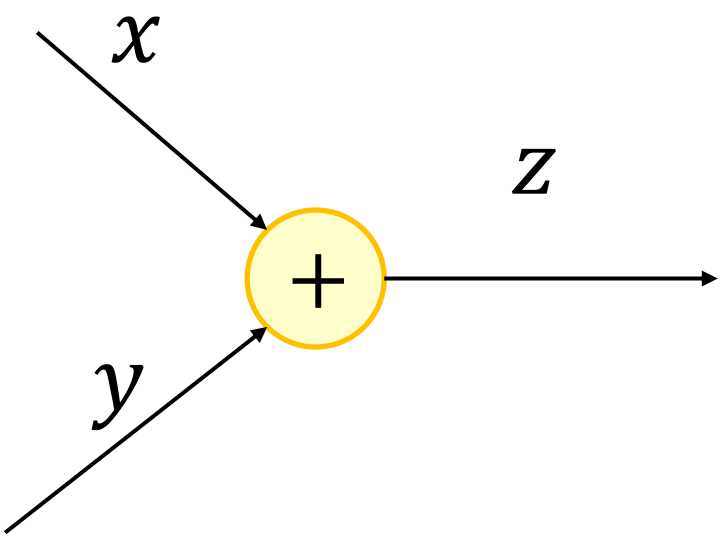


## 계산 그래프의 역전파

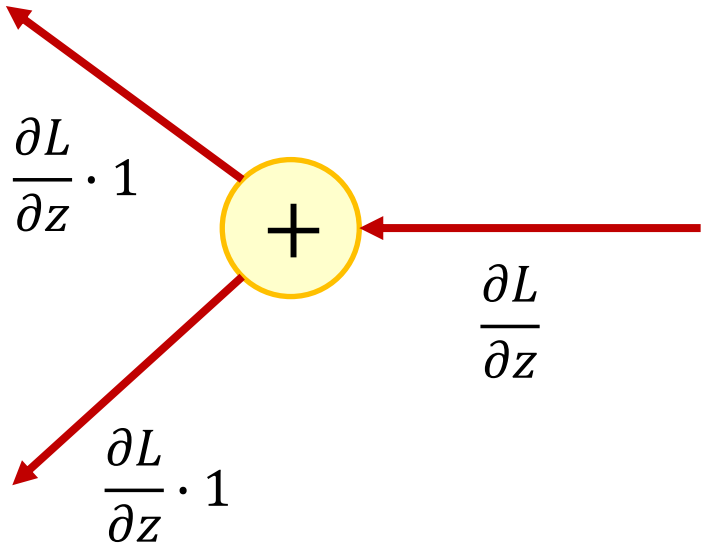


## 덧셈 노드의 순전파와 역전파

$$z = x + y$$



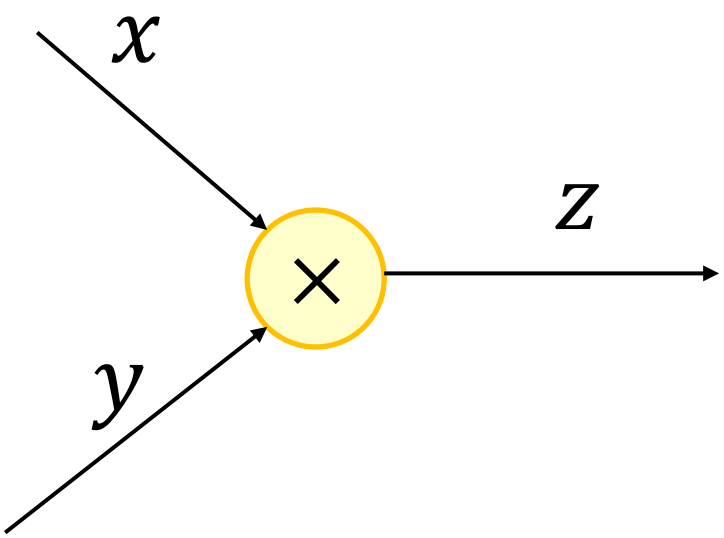

순전파



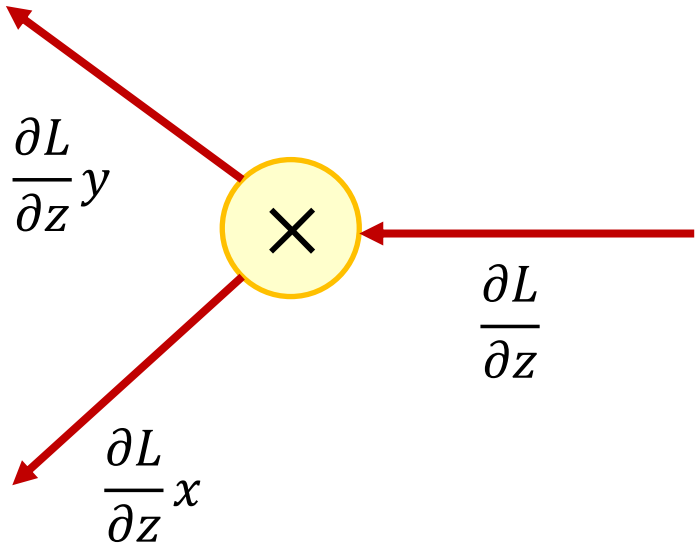

역전파

## 곱셈 노드의 순전파와 역전파

$$z = x * y$$

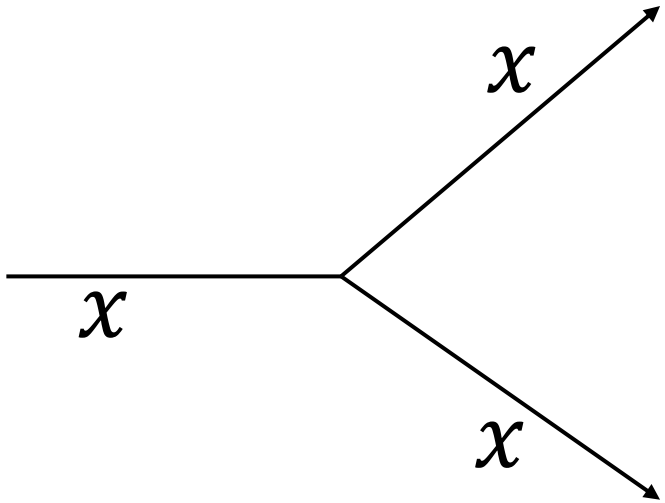



순전파

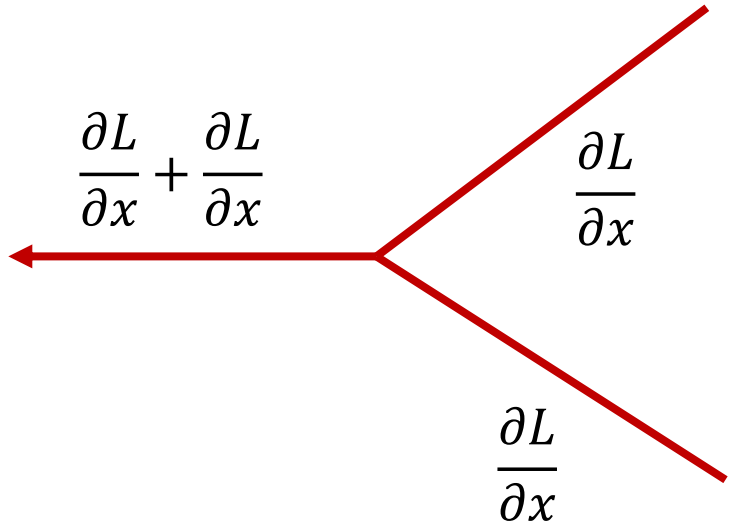



역전파

## 분기 노드의 순전파와 역전파

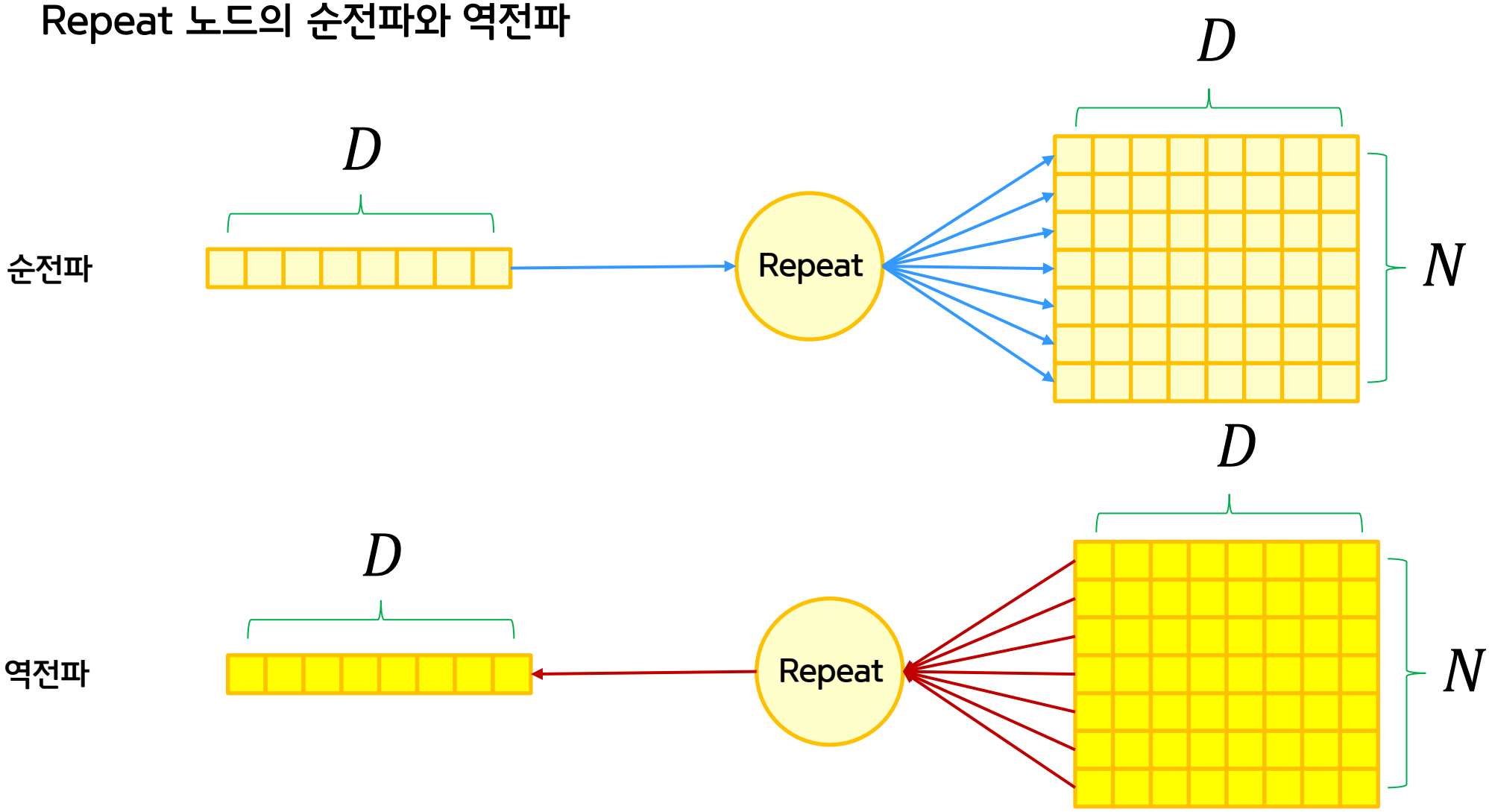



순전파

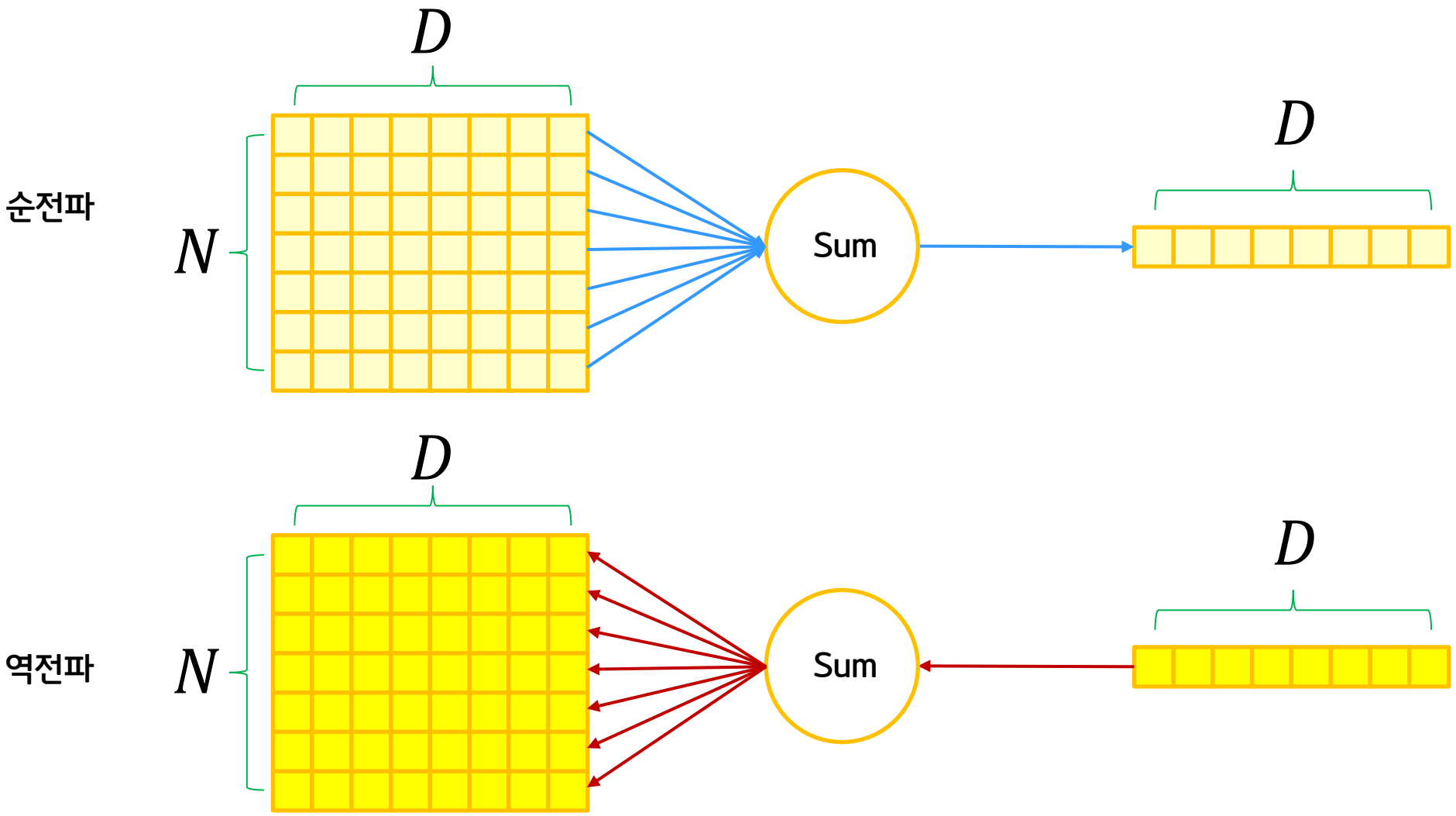



역전파

## Repeat 노드의 순전파와 역전파

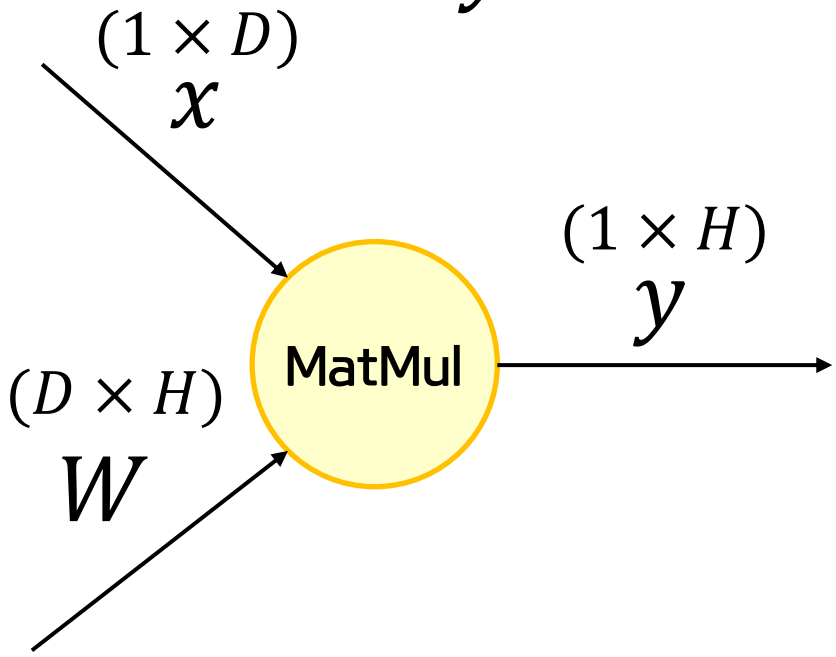


## Sum 노드의 순전파와 역전파



MatMul 노드의 순전파

$$y = xW$$




$$\frac{\partial L}{\partial x_i} = \sum_j \frac{\partial L}{\partial y_j}\frac{\partial y_j}{\partial x_i}$$

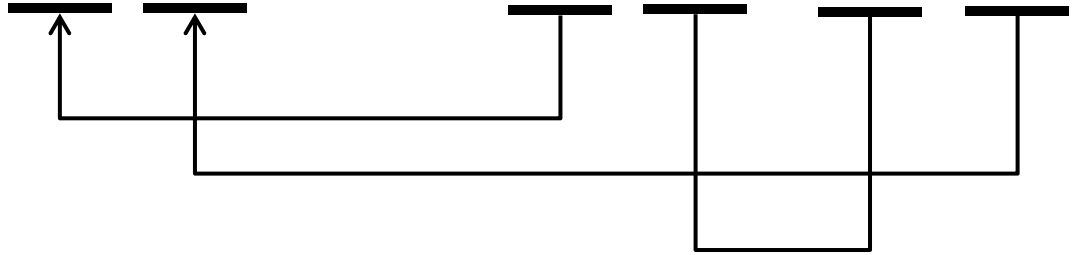
$$= \sum_j \frac{\partial L}{\partial y_j} W_{ij}$$

$$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial y} W^T$$

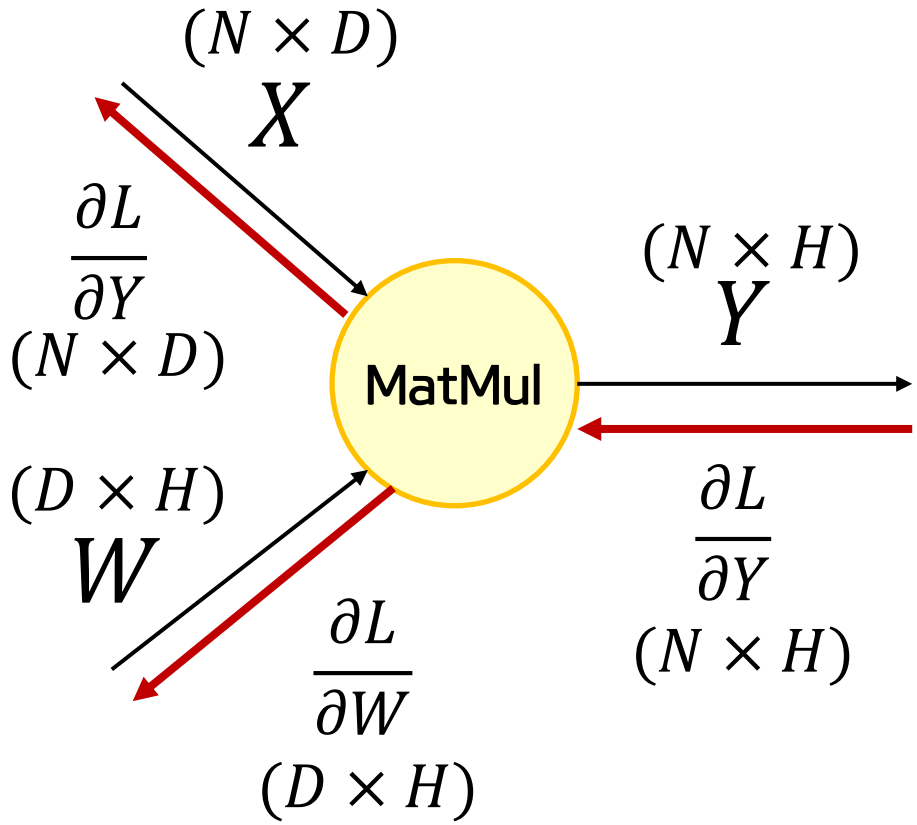
## MatMul 미분

$$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial y} W^T$$

$1 \times D$ $1 \times H$ $H \times D$



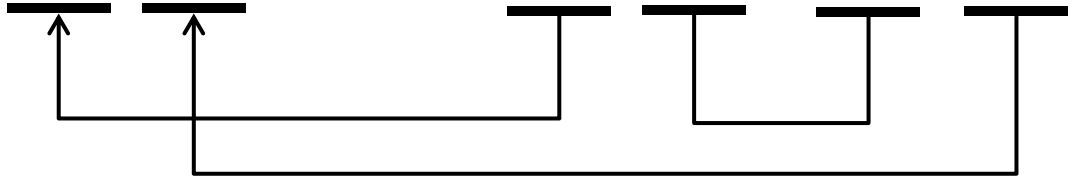
## MatMul 노드의 역전파



행렬의 형상을 확인하여 역전파 식을 유도 한다.

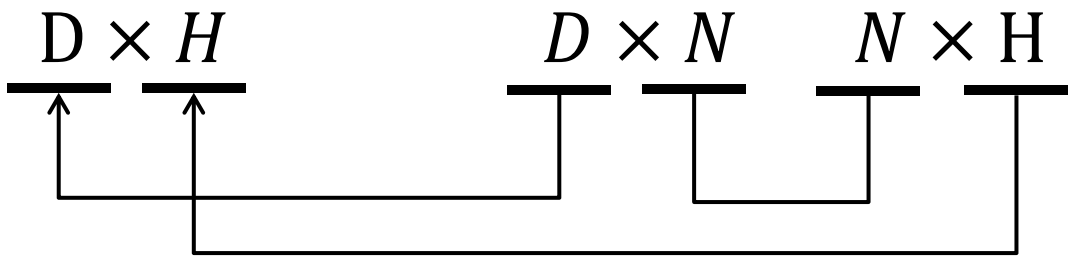
$$\frac{\partial L}{\partial X} = \frac{\partial L}{\partial Y} W^T$$

형상: $\mathrm{N} \times D \quad N \times H \quad H \times D$

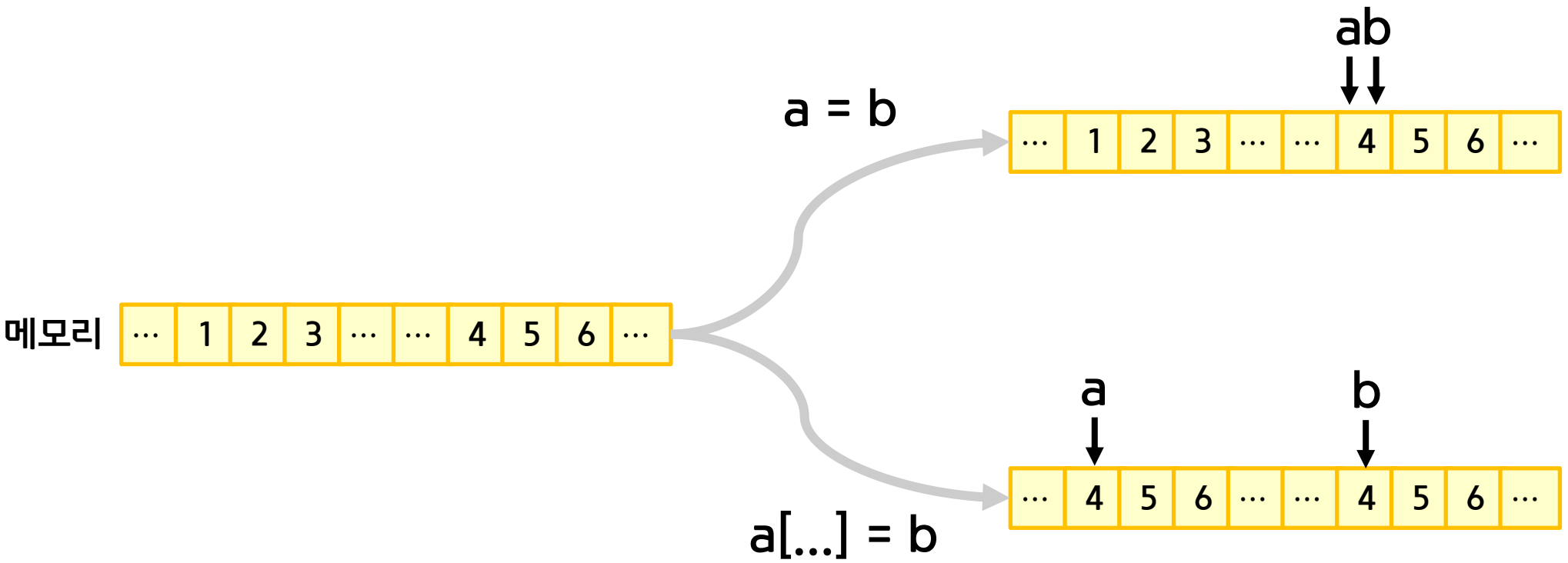


$$\frac{\partial L}{\partial W} = X^T \frac{\partial L}{\partial Y}$$

형상: $\mathrm{D} \times H \quad D \times N \quad N \times \mathrm{H}$



```
a = np.array([1, 2, 3])
b = np.array([1, 2, 3])
```



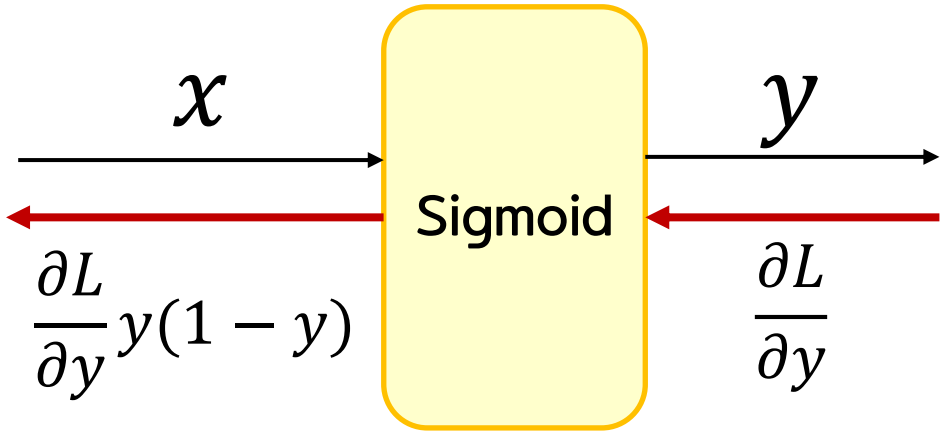
## Sigmoid 계층

Sigmoid 식

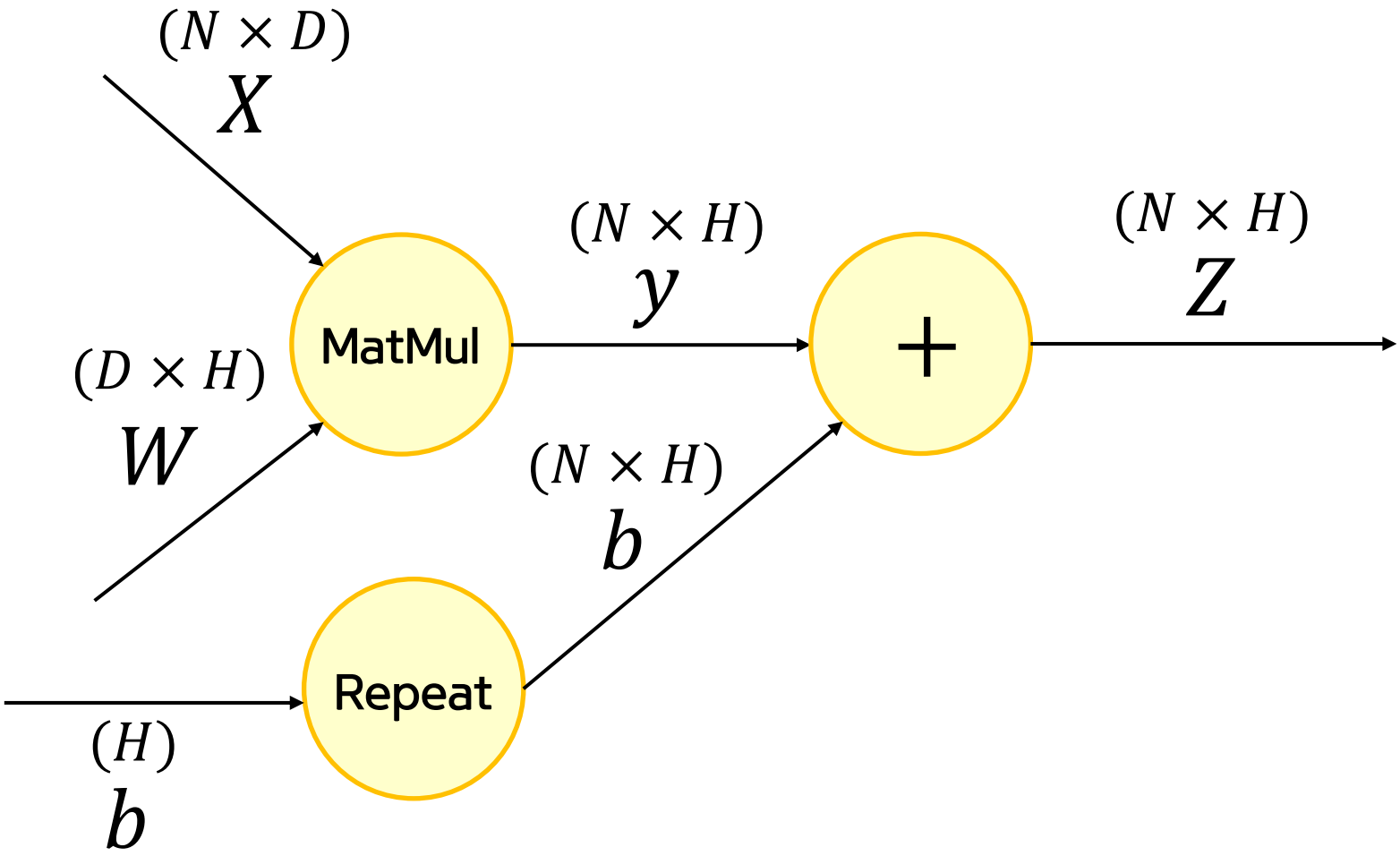
$$y = \frac{1}{1 + e^{-x}}$$

Sigmoid 미분식

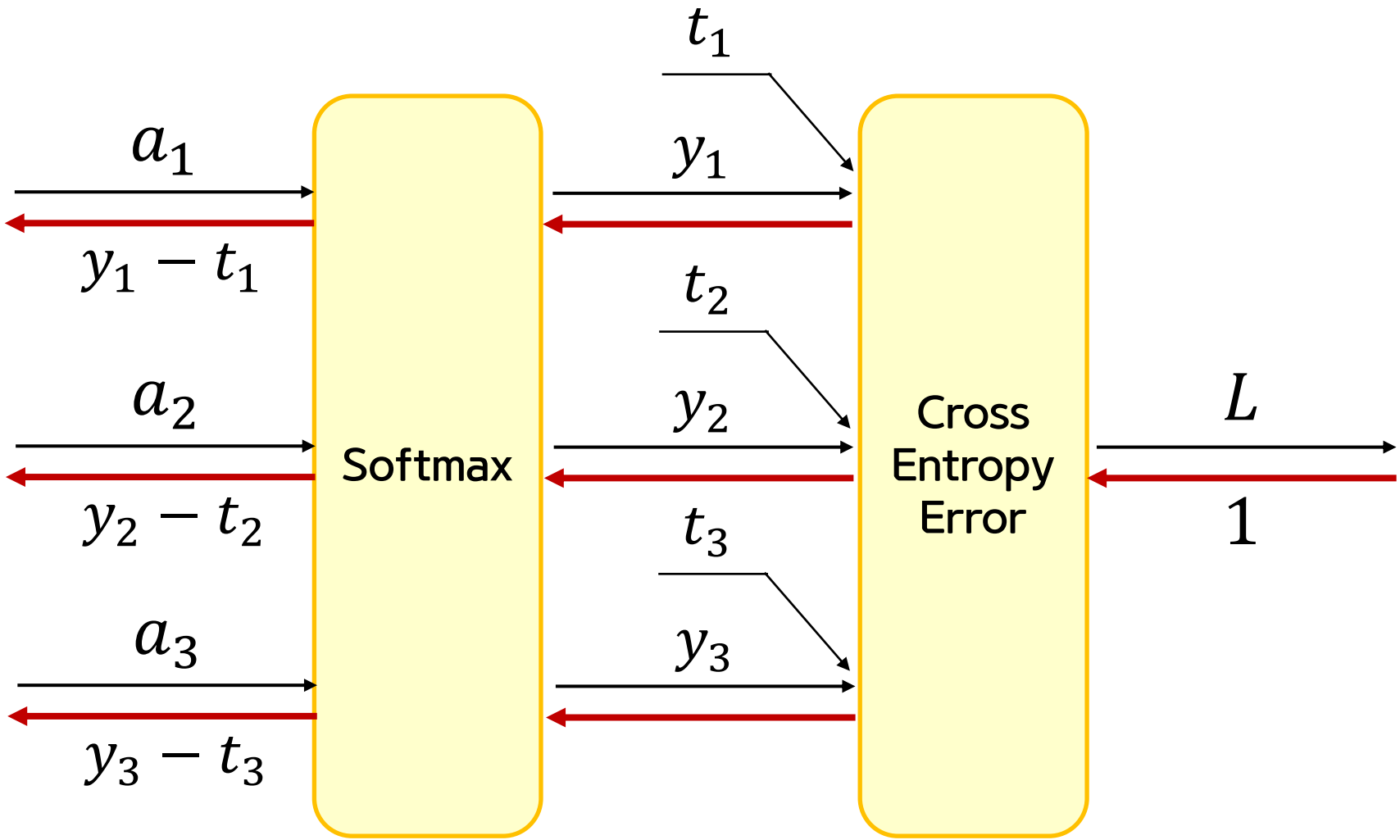
$$\frac{\partial y}{\partial x} = y(1 - y)$$



## Affine 계층


$(N \times D)$
$X$
$(D \times H)$
$W$
$(H)$
$b$
MatMul
Repeat
$(N \times H)$
$y$
$(N \times H)$
$b$
+
$(N \times H)$
$Z$

## Softmax with Loss 계층



- **1단계: 미니배치**
  훈련 데이터 중에서 무작위로 다수의 데이터를 골라낸다.

- **2단계: 기울기 계산**
  오차역전파법으로 각 가중치 매개변수에 대한 손실 함수의 기울기를 구한다.

- **3단계: 매개변수 갱신**
  기울기를 사용하여 가중치 매개변수를 갱신한다.

- **4단계: 반복**
  1~3 단계를 필요한 만큼 반복한다.

$$W \leftarrow W - \alpha \frac{\partial L}{\partial W}$$

# 1. 신경망 복습

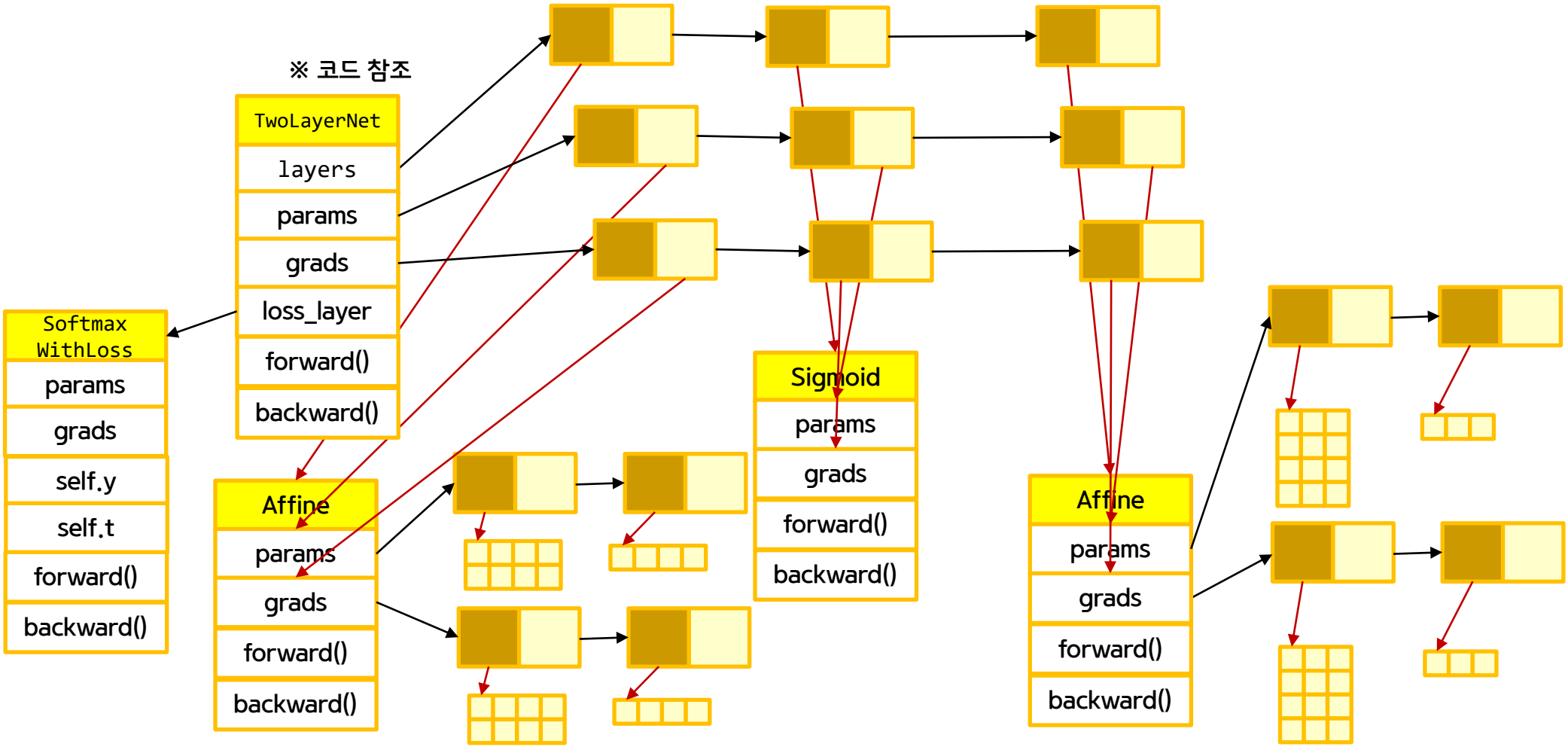
1.1 수학과 파이썬 복습

1.2 신경망 추론

1.3 신경망의 학습

**1.4 신경망으로 문제를 풀다.**

※ 코드 참조


TwoLayerNet
layers
params
grads
loss_layer
forward()
backward()
Softmax
WithLoss
params
grads
self.y
self.t
forward()
backward()
Affine
params
grads
forward()
backward()
Sigmoid
params
grads
forward()
backward()
Affine
params
grads
forward()
backward()

※ 코드 참조

x △ ○

0 1 2

1 0 0 0 1 0 0 0 1

t = 2

0.8 0
0.1 0
0.1 1

0.8 0.1 0.1

2.302585092994045684017991454684 4

0.1 0.1 0.8

0.22314355131420975576629509030983

크로스 엔트로피 손실 함수

$$L = -\sum_{c=1}^{c} y_c \log(a_c) = -(y_1 \log(a_1) + y_2 \log(a_2) + \cdots + y_c \log(a_c))$$

$$-y_3 \log(a_3)$$

-np.sum(np.log(y[np.arange(batch_size), t] + 1e-7)) / batch_size

※ 코드 참조

| 인수 | 설명 |
|---|---|
| x | 입력 데이터 |
| t | 정답 레이블 |
| max_epoch(=10) | 학습을 수행하는 에폭 수 |
| batch_size(=32) | 미니배치 크기 |
| eval_interval(=20) | 결과(평균 손실 등) 출력하는 간격<br>예컨대 eval_interval=20으로 설정하면, 20번째 반복마다 손실의 평균을 구해 화면에 출력한다. |
| max_grad(=None) | 기울기 최대의 노름<br>기울기 노름이 이 값을 넘어서면 기울기를 중인다(이를 기울기 클리핑이라 한다.) |