신경망 학습 & 오차 역전파법

4강 & 5강

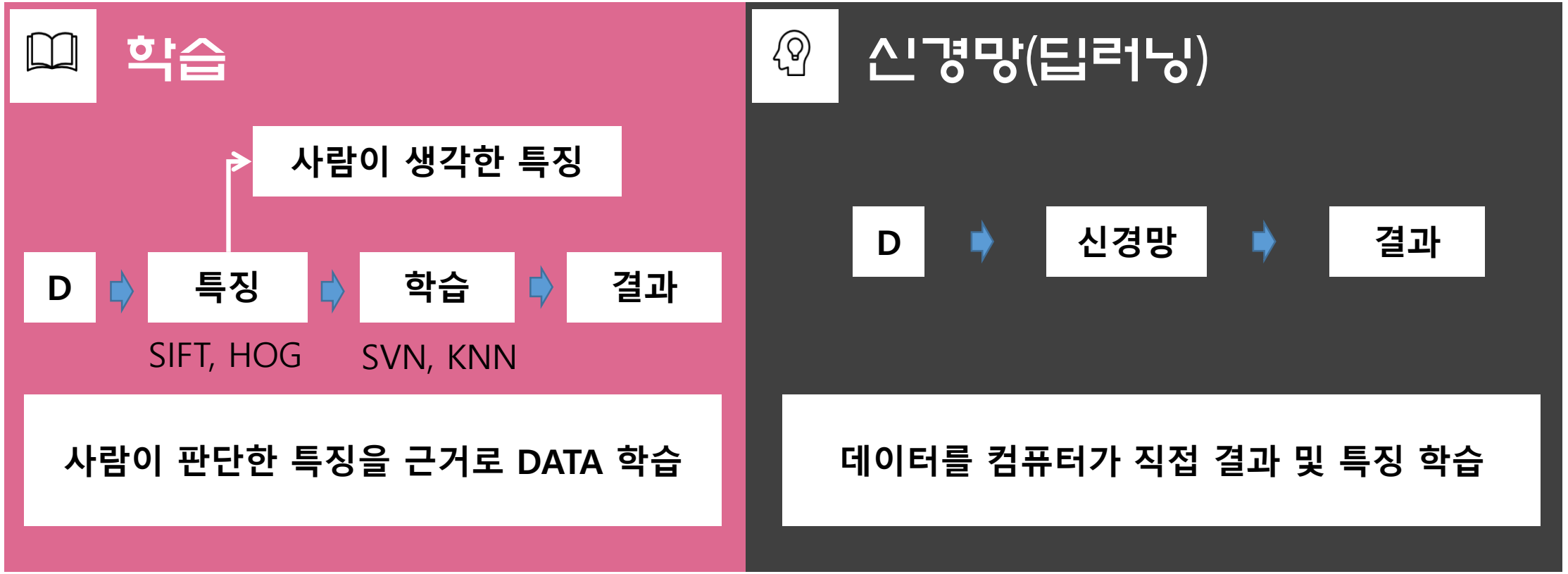
INDEX

4. 신경망 학습

5. 오차 역전파 법

01 개요

학습이란? : 훈련데이터로부터 가중치의 매개변수 최적값을 자동으로 획득 하는것



02 손실함수

학습이란? : 훈련데이터로부터 가중치의 매개변수 최적값을 자동으로 획득 하는것

손실함수

최적의 매개변수 값을 탐색하기 위한 지표

오차제곱합

$$E = \frac{1}{2}\sum_{k}(y_k - t_k)^2$$

교차 엔트로피 오차

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

02 손실함수

학습이란? : 훈련데이터로부터 가중치의 매개변수 최적값을 자동으로 획득 하는것

손실함수

최적의 매개변수 값을 탐색하기 위한 지표

오차제곱합

```python
# 4.2.1 평균 제곱 오차
# E = 1/2 * ∑ _k (yk-tk)²
# yk : 신경망의 출력
# tk : 정답 레이블
# k : 데이터의 차원 수
def mean_squared_error(y, t):
    return 0.5 * np.sum((y-t)**2)
```

교차 엔트로피 오차

```python
# 4.2.2 교차 엔트로피 오차
# E = -∑ _k (tk * log(yk))
# log : 자연로그
# yk : 신경망의 출력
# tk : 정답 레이블(one-hot encoding)
# k : 데이터의 차원 수
# 실질적으로 정답일때의 추정의 자연로그를 계산하는 식이 됨
def cross_entropy_error(y, t):
    delta = 1e-7  # 0일때 -무한대가 되지 않기 위해 작은 값을 더함
    return -np.sum(t * np.log(y + delta))
```

02 손실함수 예시

오차제곱합 예제

```
# 정답은 '2'
t = [0, 0, 1, 0, 0, 0, 0, 0, 0, 0]
#-----------------------------#
# '2'일 확률이 가장 높다고 추정한 값(0.6)
y = [0.1, 0.05, 0.6, 0.0, 0.05, 0.1, 0.0, 0.1, 0.0, 0.0]
mse = mean_squared_error(np.array(y), np.array(t))
print(mse)  # 0.0975
#-----------------------------#
# ex2 '7'일 확률이 가장 높다고 추정함(0.6)
y = [0.1, 0.05, 0.1, 0.0, 0.05, 0.1, 0.0, 0.6, 0.0, 0.0]
mse = mean_squared_error(np.array(y), np.array(t))
print(mse)  # 0.5975
#-----------------------------#
```

손실함수
2 < 7
2일 확률이 더 높다

02 손실함수 예시

교차 엔트로피 예제

```
# 정답은 '2'
t = [0, 0, 1, 0, 0, 0, 0, 0, 0, 0]
#------------------------------#
# '2'일 확률이 가장 높다고 추정한 값(0.6)
y = [0.1, 0.05, 0.6, 0.0, 0.05, 0.1, 0.0, 0.1, 0.0, 0.0]
cee = cross_entropy_error(np.array(y), np.array(t))
print(cee)  # 0.510825457099
#------------------------------#
# '7'일 확률이 가장 높다고 추정함(0.6)
y = [0.1, 0.05, 0.1, 0.0, 0.05, 0.1, 0.0, 0.6, 0.0, 0.0]
cee = cross_entropy_error(np.array(y), np.array(t))
print(cee)  # 2.30258409299
#------------------------------#
```

손실함수
2 < 7
2일 확률이 더 높다

03 미니배치

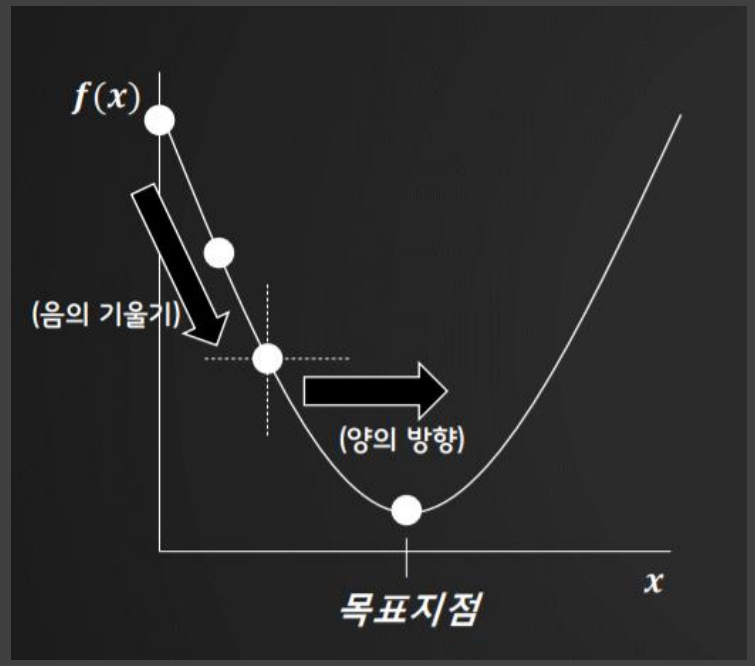
※ 미니배치

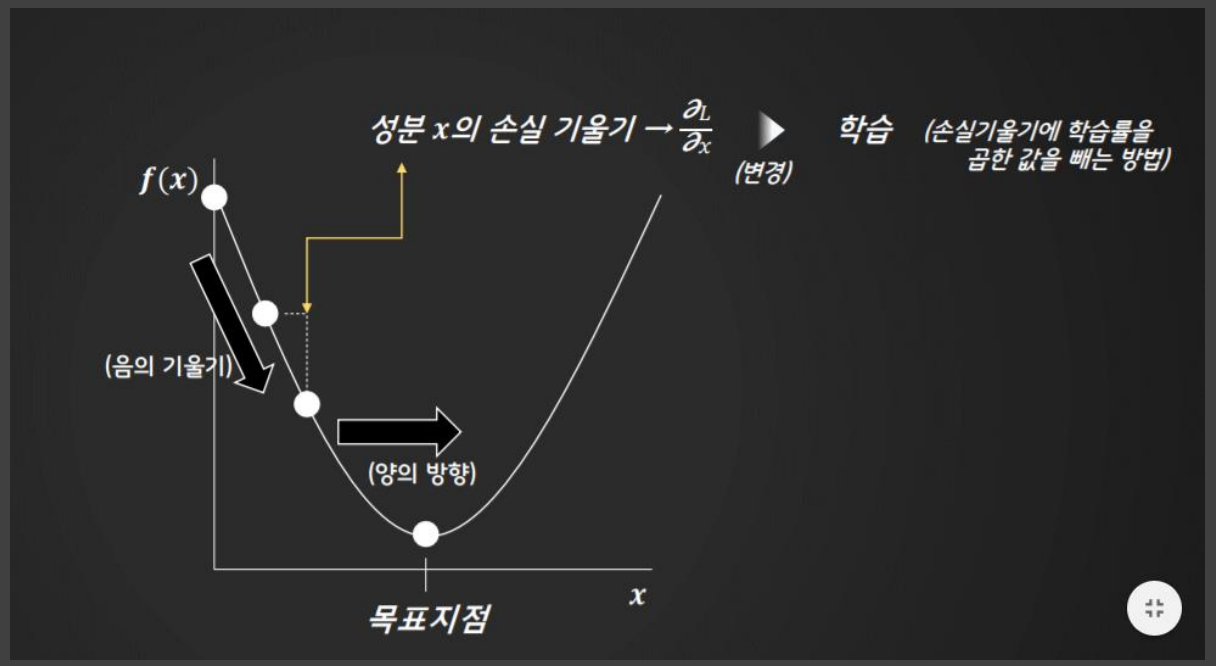
많은 수의 훈련 데이터 중 임의의 데이터셋을 추출한것

```
(x_train, t_train), (x_test, t_test) = load_mnist(normalize=True, one_hot_label=False)
print(x_train.shape)  # (60000, 784)
print(t_train.shape)  # 원-핫 인코딩 된 정답 레이블 (60000, 10)

# 무작위 10개 추출
train_size = x_train.shape[0] #60000
batch_size = 10
batch_mask = np.random.choice(train_size, batch_size) #60000이하의 수 랜덤
x_batch = x_train[batch_mask]
t_batch = t_train[batch_mask]
```

04 경사하강법

※ 경사하강법 : 기울기에 따라 함숫값이 낮아지는 방향으로 이동하는 기법.





05 정리

1단계 - 미니배치훈련 데이터 중 일부를 무작위로 가져온다.
선정된 무작위 데이터를 미니배치라 하며,
그 미니배치의 손실함수 값을 줄이는 것이 목표이다

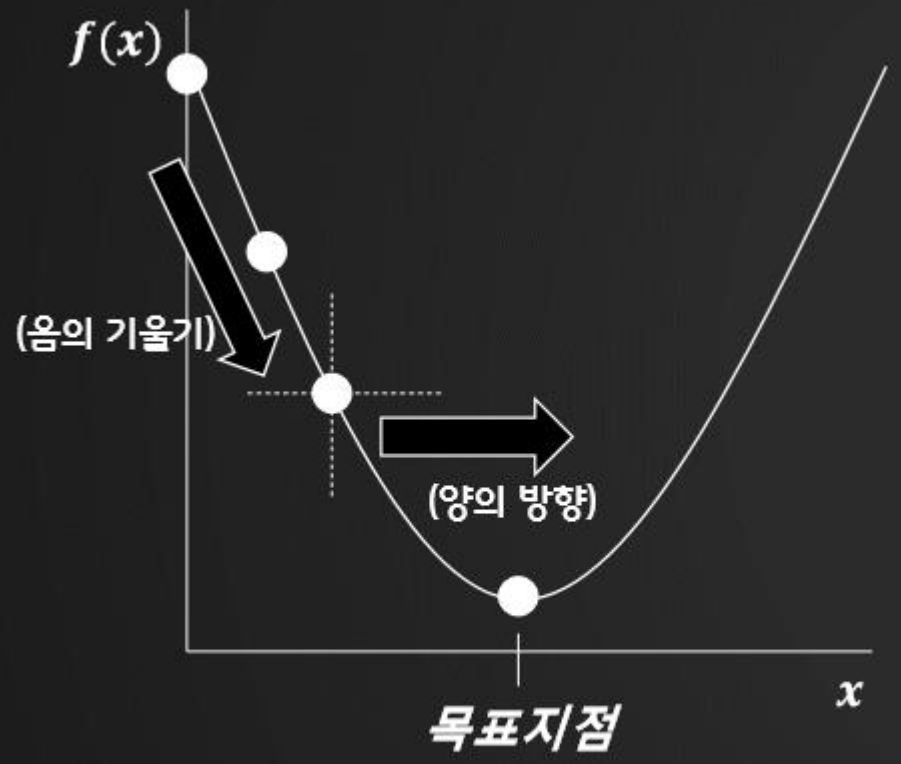
2단계 - 기울기 산출미니배치의 손실 함수 값을 줄이기 위 해 각 가중치 매개변수의 기울기를 구한다.
기울기는 손실 함수의 값을 가장 작게 하는 방향을 제시한다.

3단계 - 매개변수 갱신가중치 매개변수를 기울기 방향으로 갱신한다.

4단계 - 반복1~3단계를 반복한다.

```python
# 1에폭당 반복 수
iter_per_epoch = max(train_size / batch_size, 1)

for i in range(iters_num):
    print(i)
    # 미니배치 획득
    batch_mask = np.random.choice(train_size, batch_size)
    x_batch = x_train[batch_mask]
    t_batch = t_train[batch_mask]

    # 기울기 계산
    grad = network.numerical_gradient(x_batch, t_batch)

    # 매개변수 갱신
    for key in ('W1', 'b1', 'W2', 'b2'):
        network.params[key] -= learning_rate * grad[key]

    # 학습 경과 기록
    loss = network.loss(x_batch, t_batch)
    train_loss_list.append(loss)

    # 1에폭 당 정확도 계산
    if i % iter_per_epoch == 0:
        train_acc = network.accuracy(x_train, t_train)
        test_acc = network.accuracy(x_test, t_test)
        train_acc_list.append(train_acc)
        test_acc_list.append(test_acc)
        print("train acc, test acc | "
              + str(train_acc) + ", " + str(test_acc))
```

01 역전파



2차원 그래프로 표현한 경사하강법의 원리

경사하강법(gradient descent algorithm)?

기울기에 따라 함숫값이 낮아지는 방향으로 이동하는 기법.

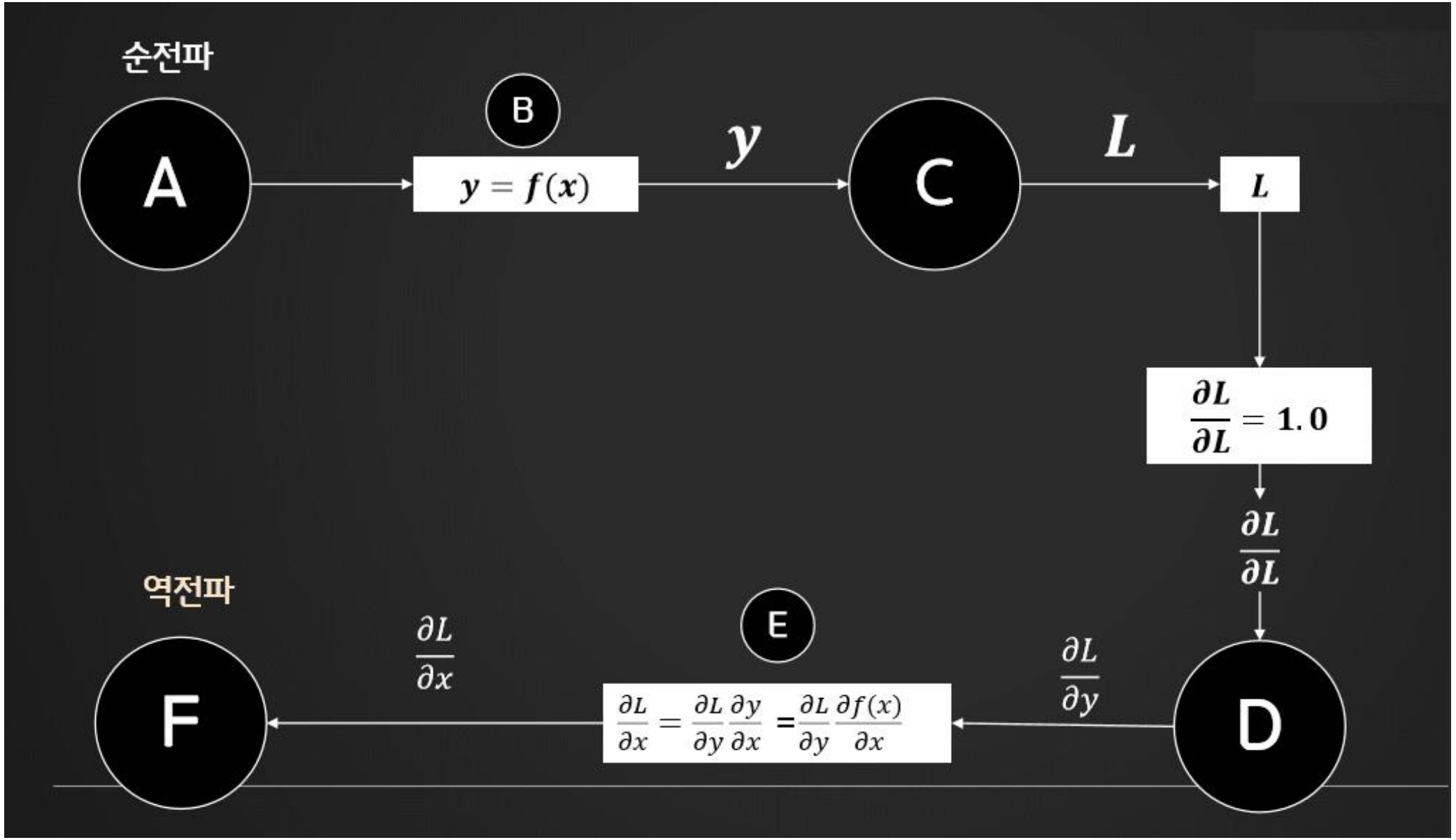
순전파(forward propagation)?

입력 데이터에 대해 신경망 구조를 따라가면서 현재의
파라미터값들을 이용해 손실함숫값을 계산하는 과정

역전파(backward propagation)?

순전파의 계산과정을 역순으로 거슬러 가면서 손실 함숫값에
직간접적으로 영향을 미친 모든 성분에 대해 손실 기울기를 계산하는 과정

01 역전파

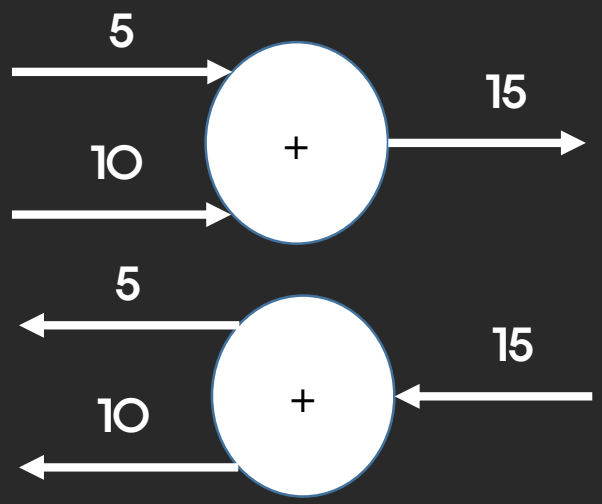

순전파
A
B
$y = f(x)$
y
C
L
L
$\frac{\partial L}{\partial L} = 1.0$
$\frac{\partial L}{\partial L}$
역전파
F
$\frac{\partial L}{\partial x}$
E
$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial y}\frac{\partial y}{\partial x} = \frac{\partial L}{\partial y}\frac{\partial f(x)}{\partial x}$
$\frac{\partial L}{\partial y}$
D

02 규칙

※ 연쇄 법칙

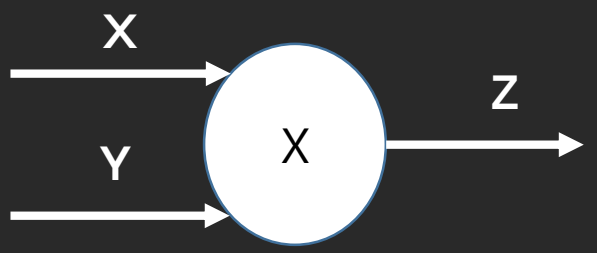
$$\frac{dy}{dx} = \frac{dt}{dx}\frac{dy}{dt}$$

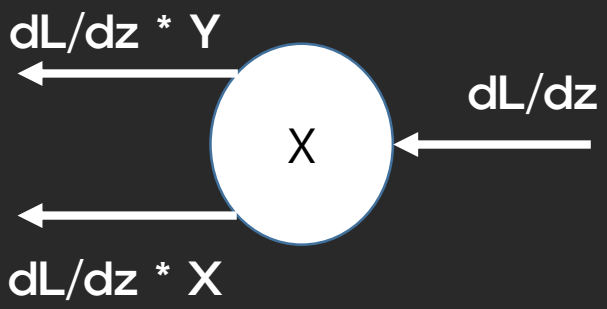
※ 노드 덧셈 (입력 그대로)




```python
# 5.4.2 덧셈 계층
class AddLayer:
    def __init__(self):
        pass

    def forward(self, x, y):
        out = x + y
        return out

    def backward(self, dout):
        dx = dout * 1
        dy = dout * 1
        return dx, dy
```

※ 노드 곱셈

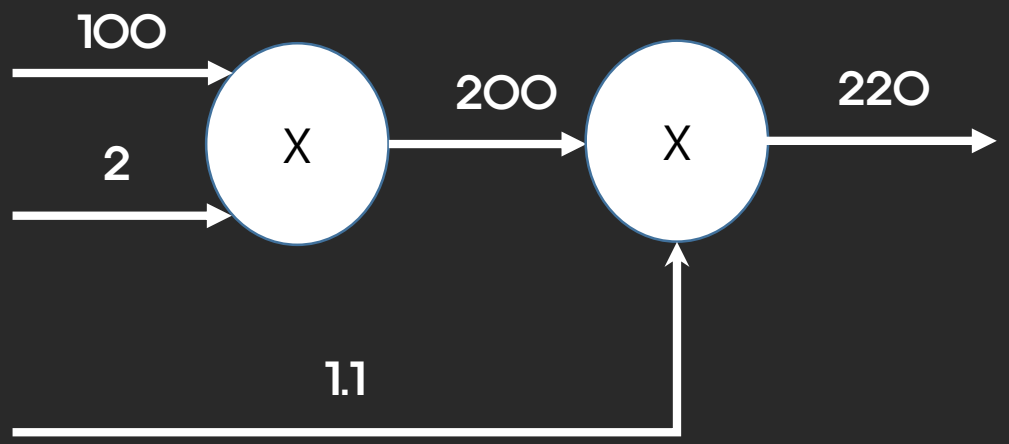




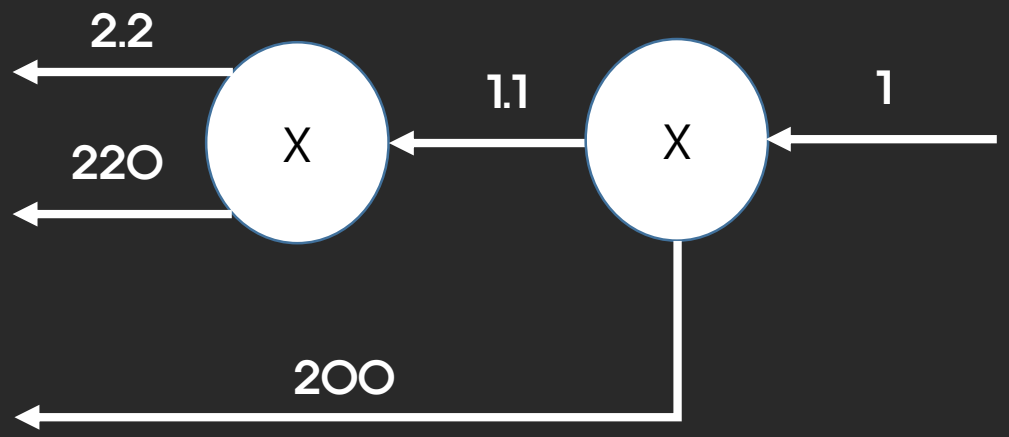

```python
# 5.4.1 곱셈 계층
class MulLayer:
    def __init__(self):
        self.x = None
        self.y = None

    def forward(self, x, y):
        self.x = x
        self.y = y
        out = x * y
        return out

    def backward(self, dout):
        dx = dout * self.y
        dy = dout * self.x
        return dx, dy
```

03 예시

※ 순전파



※ 역전파



THANK YOU