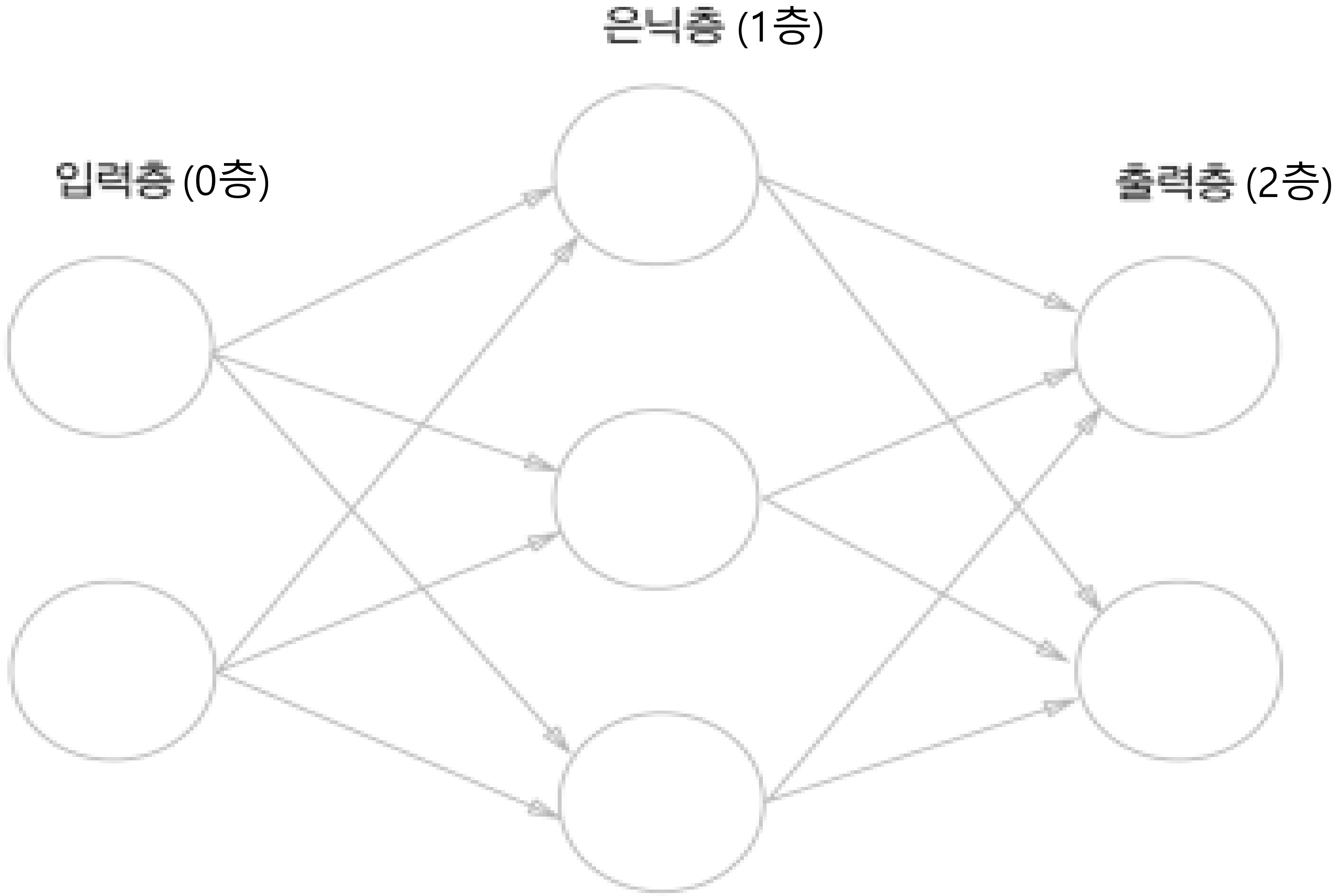
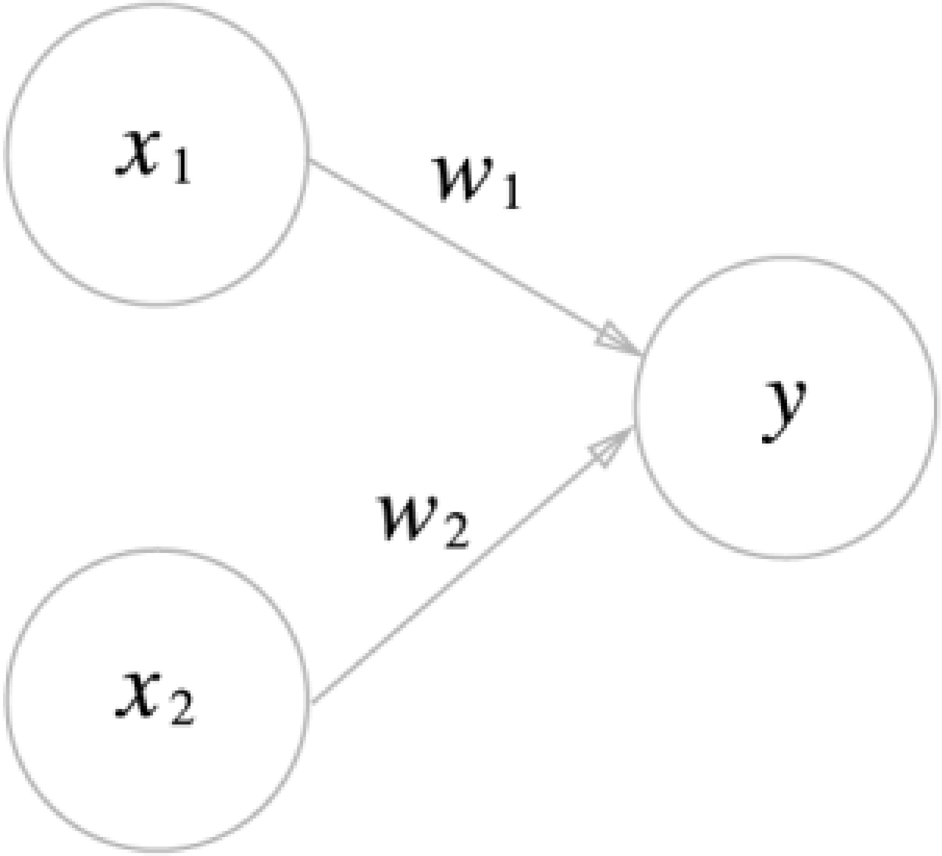
신경망

(Neural Networks)

신경망의 예 - 2층 신경망



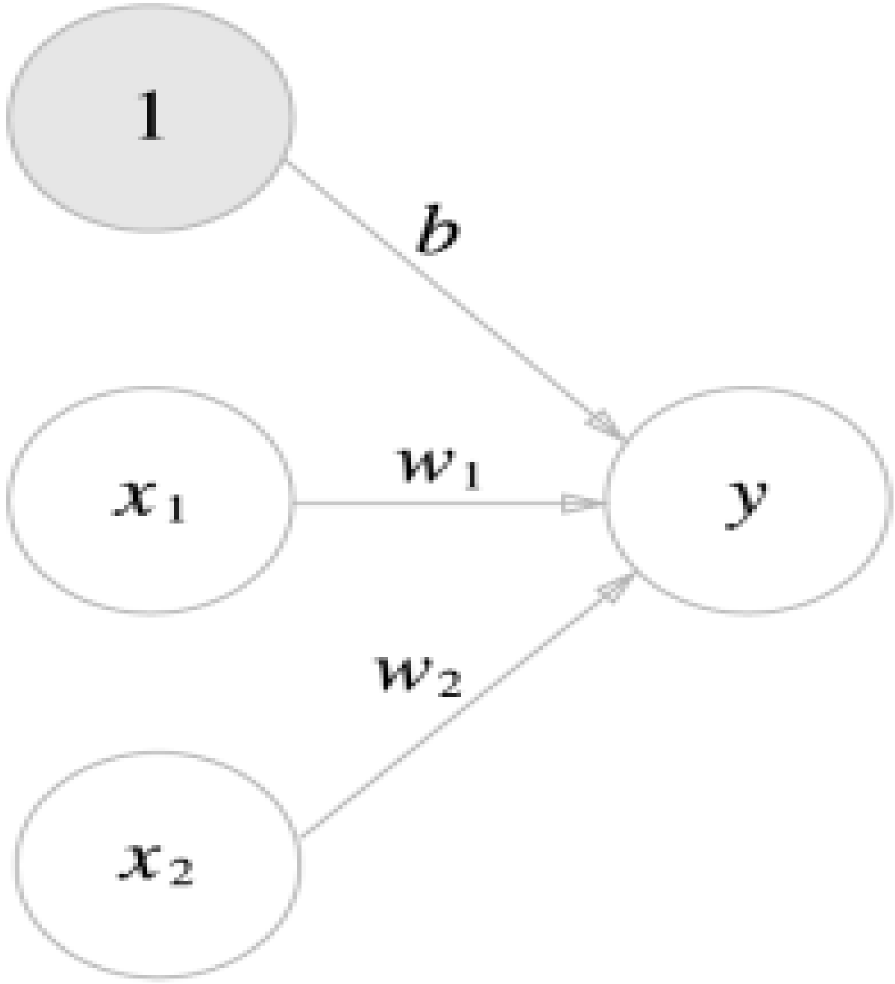
퍼셉트론 복습



$$y = \begin{cases} 0 & (b + w_1x_1 + w_2x_2 \leq 0) \\ 1 & (b + w_1x_1 + w_2x_2 > 0) \end{cases}$$

- y는 출력 신호.
- x1과 x2는 입력 신호.
- w1과 w2는 가중치(weight) : 각 신호가 결과에 주는 영향력을 조절하는 요소로 작용.
- b(편향) : 뉴런이 얼마나 쉽게 활성화되느냐를 제어.
- 뉴런(혹은 노드) : 그림의 원.

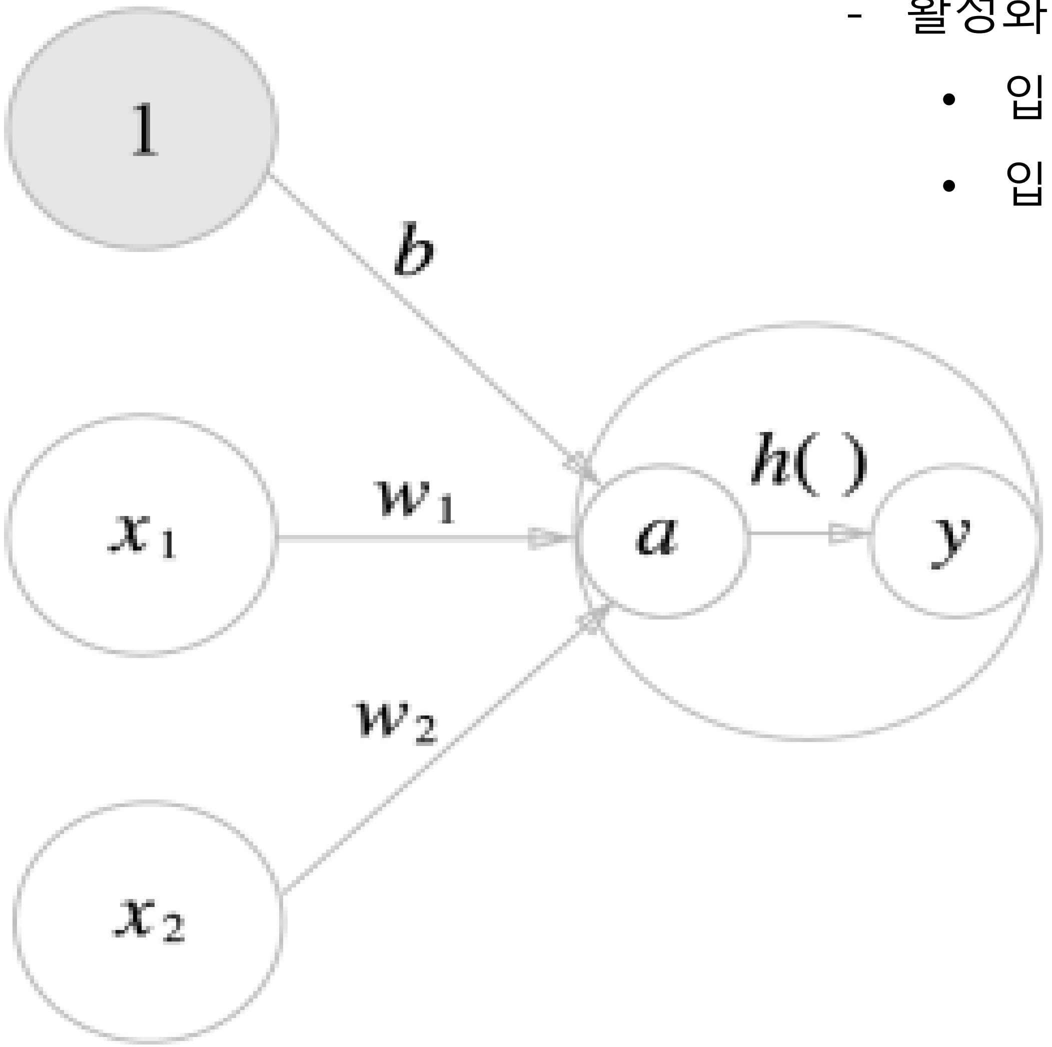
Bias(편향)를 명시한 퍼셉트론



$$y = h(b + w_1x_1 + w_2x_2)$$

$$h(x) = \begin{cases} 0 & (x \le 0) \\ 1 & (x > 0) \end{cases}$$

활성화 함수의 처리 과정

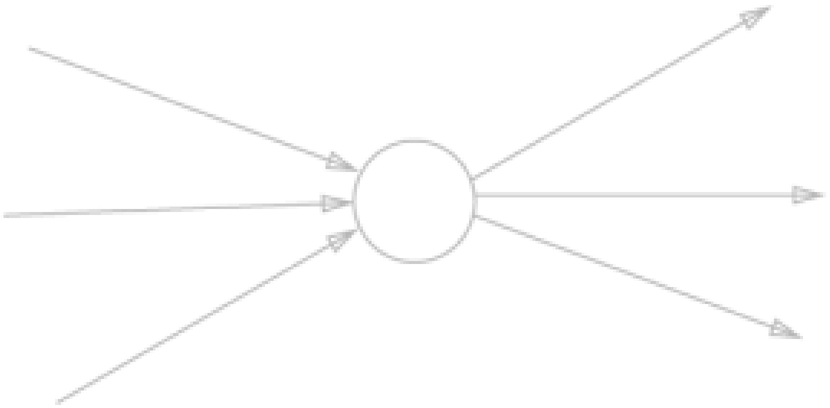


- 활성화 함수(activation function) : h(a)
  - 입력 신호의 총합을 출력 신호로 변환하는 함수.
  - 입력 신호의 총합이 활성화를 일으키는지를 정하는 역할.

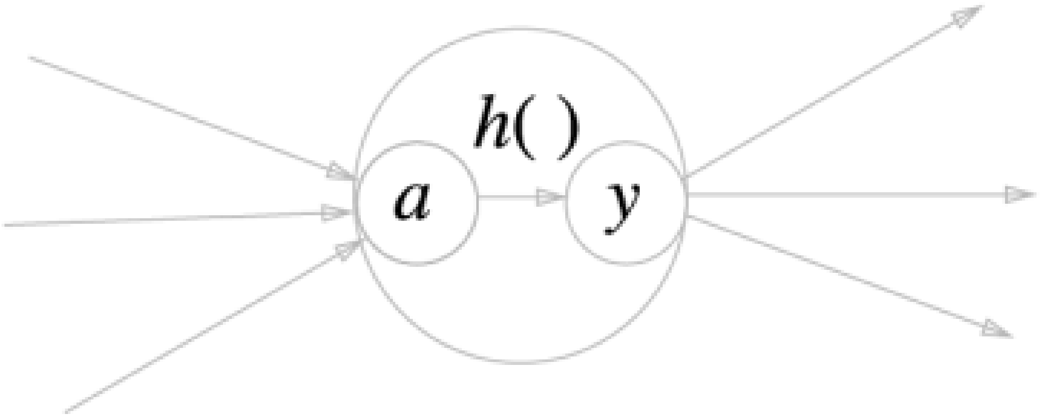
$$a = b + w_1x_1 + w_2x_2$$

$$y = h(a)$$

활성화 함수



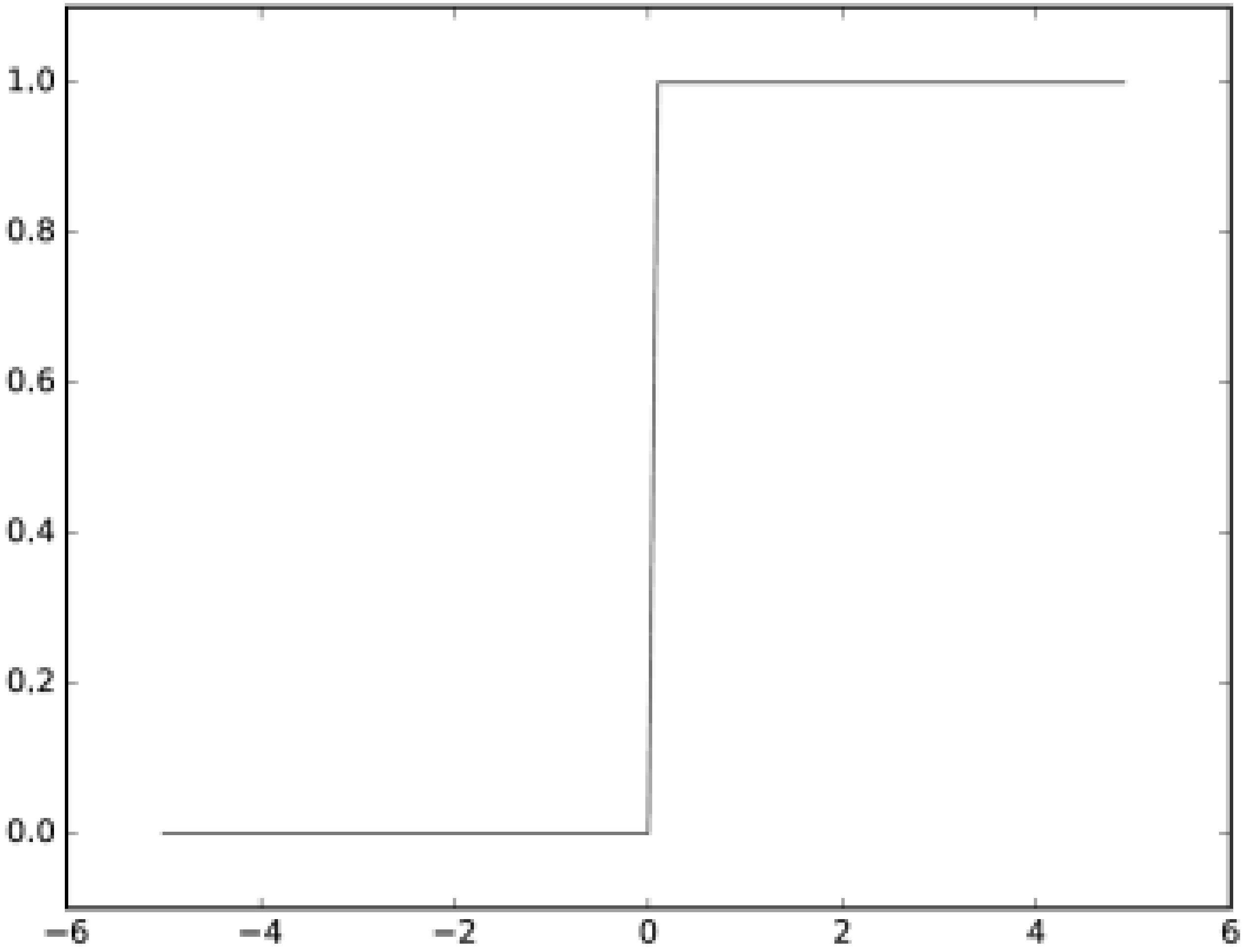
- 일반적인 뉴런



- 활성화 처리 과정을 명시한 뉴런

(a는 입력 신호의 총합, h()는 활성화 함수, y는 출력)

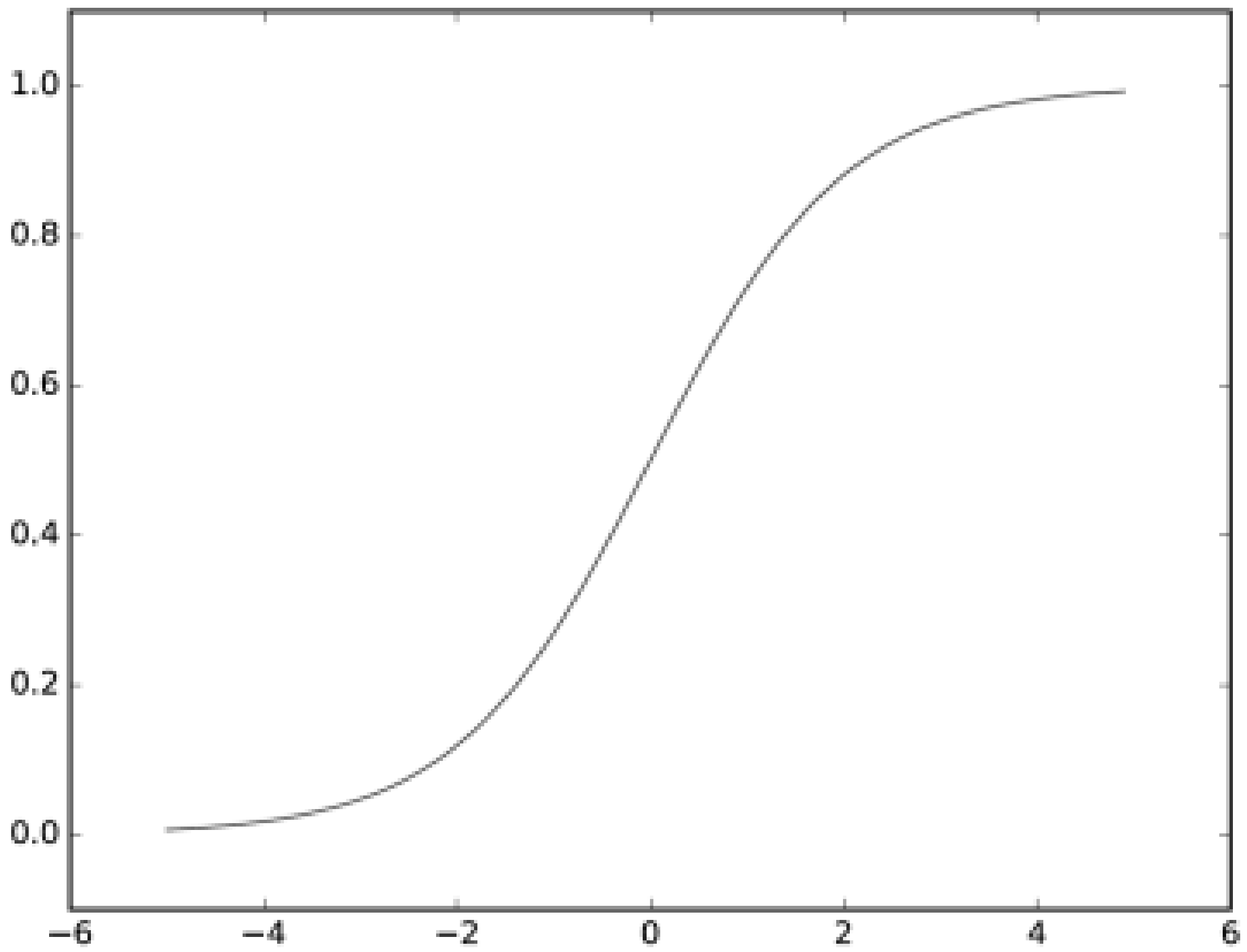
계단 함수(step function)



$$h(x) = \begin{cases} 0 & (x \leq 0) \\ 1 & (x > 0) \end{cases}$$

- 임계값을 경계로 출력이 바뀌는 활성화 함수.
- 퍼셉트론에서는 활성화 함수로 계단 함수를 이용.
- 활성화 함수를 계단함수에서 다른 함수로 변경하는 것이 신경망의 세계로 나아가는 열쇠.

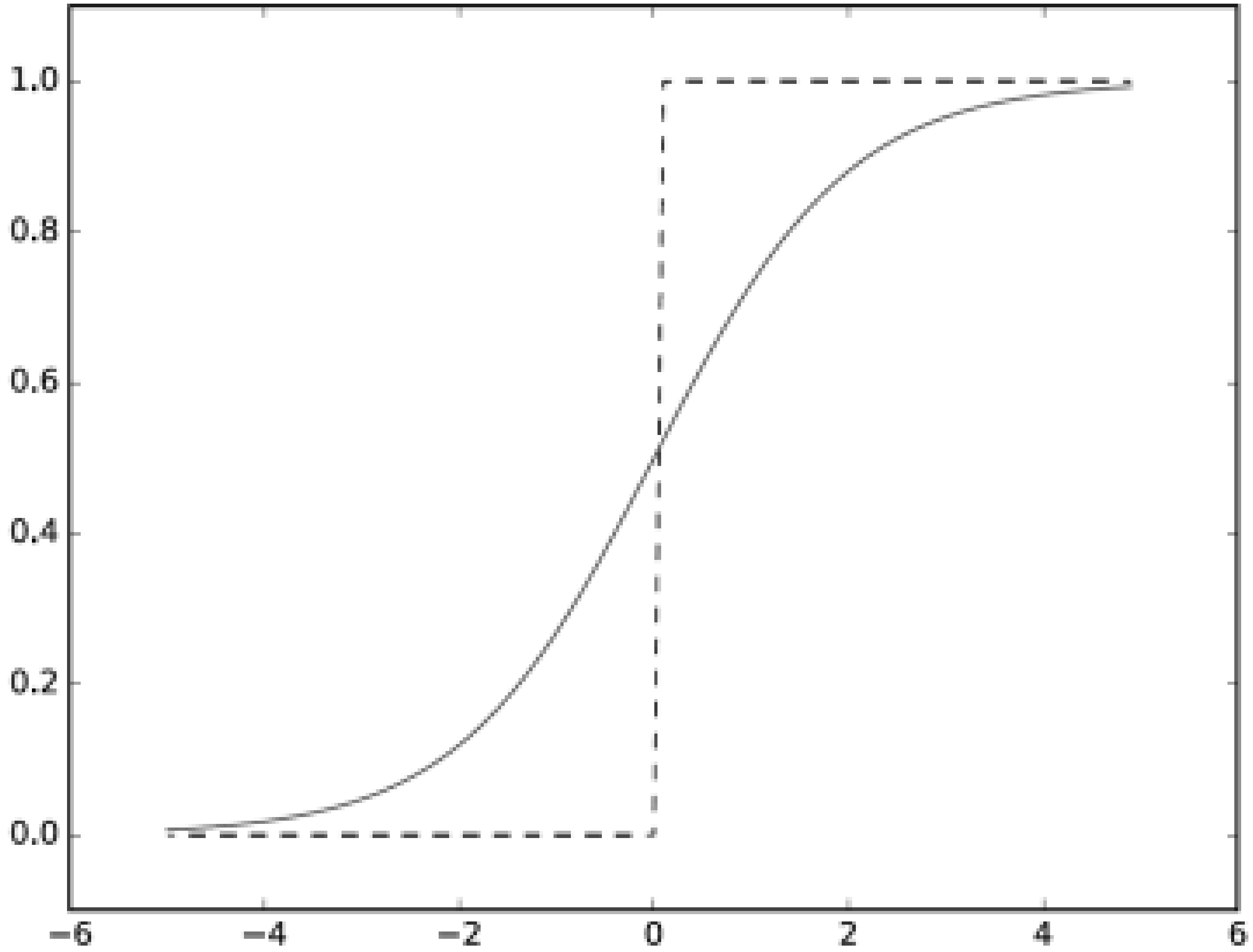
시그모이드 함수



- 시그모이드 함수(Sigmoid function)
  - 신경망에서 자주 이용하는 활성화 함수.
  - 시그모이드 함수를 이용하여 신호를 변환하고, 그 변환된 신호를 다음 뉴런에 전달.

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

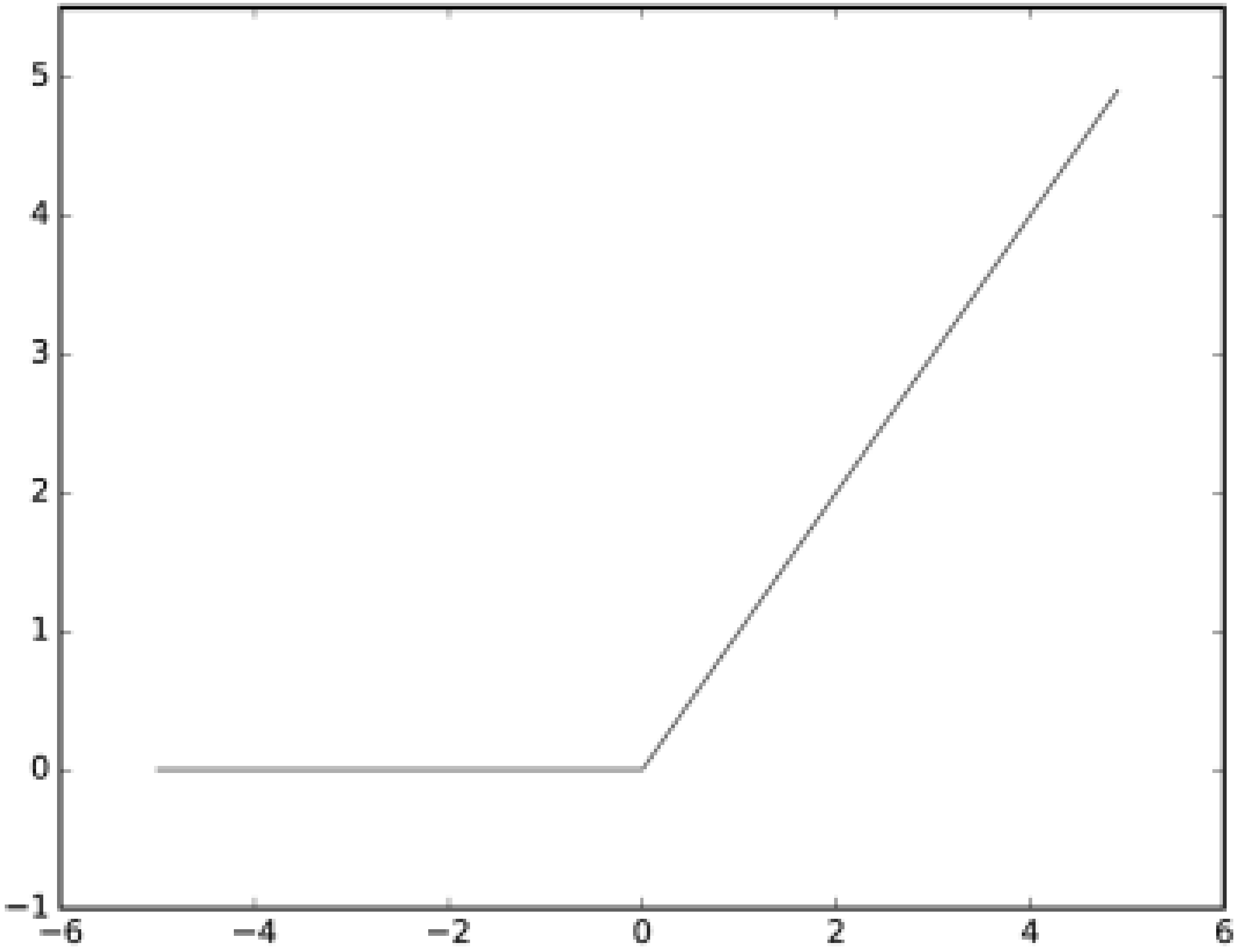
시그모이드 함수와 계단 함수 비교



* 퍼셉트론과 신경망의 주된 차이는 활성화 함수의 차이

- 계단 함수가 0과 1 중 하나의 값만 돌려주는 반면 시그모이드 함수는 실수를 돌려준다.
- 즉, 퍼셉트론에서는 뉴런 사이에 0 혹은 1이 흘렀다면, 신경망에서는 연속적인 실수가 흐른다.
- 두 함수 모두 비선형 함수.
- 신경망에서는 활성화 함수로 비선형 함수를 사용해야 함.

ReLU 함수(Rectified Linear Unit Function)



$$h(x) = \begin{cases} x & (x > 0) \\ 0 & (x \leq 0) \end{cases}$$

- 최근 신경망 분야에서 이용.
- 입력이 0을 넘으면 그대로 출력하고,
  0 이하이면 0을 출력하는 함수.

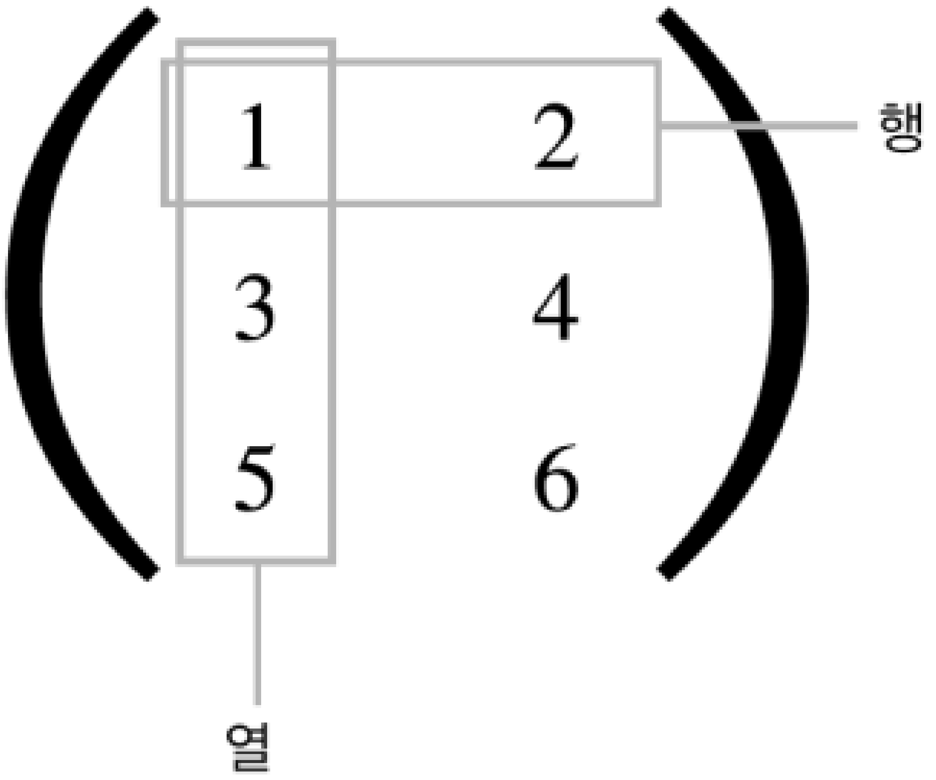
다차원 배열

- 1차원 배열

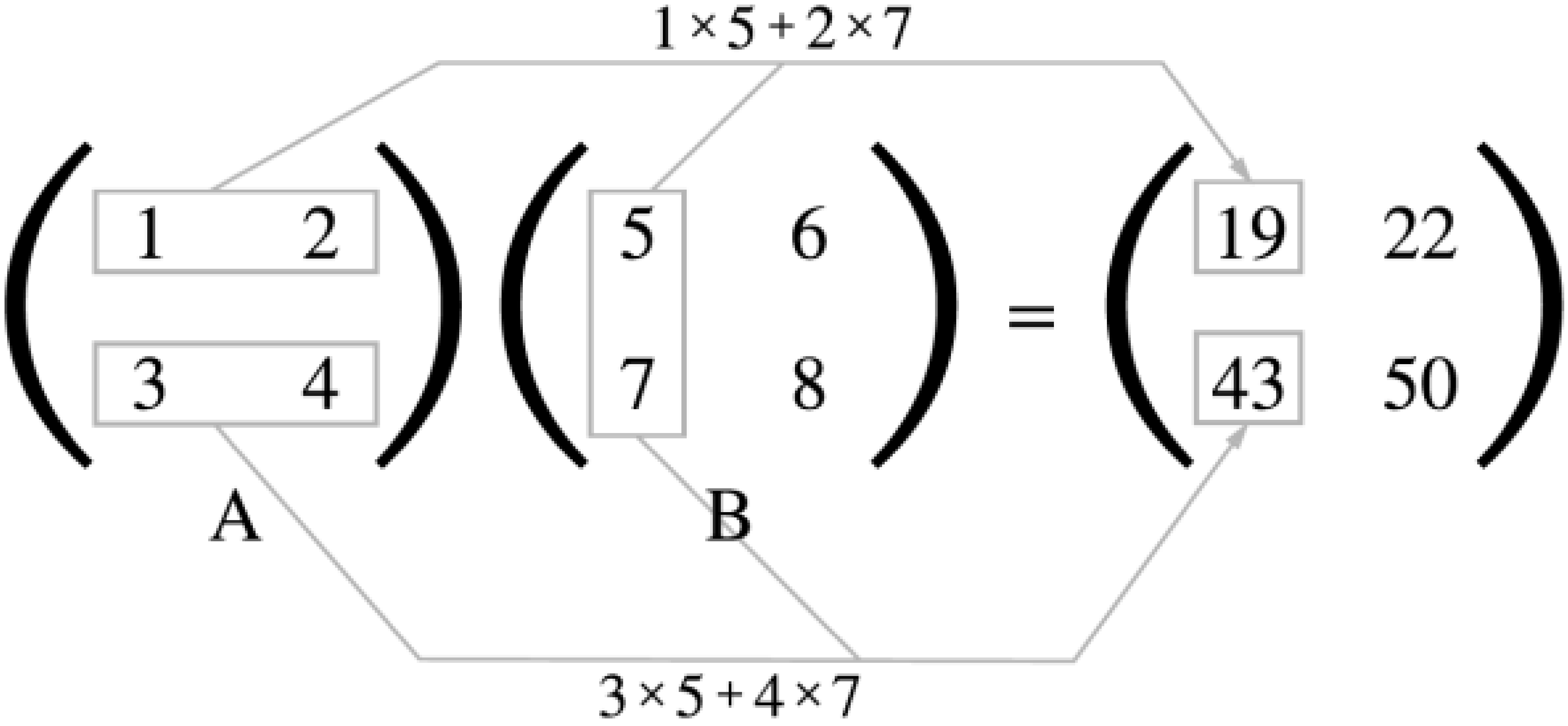
```
import numpy as np

A = np.array([1, 2, 3, 4])
print(A)
np.ndim(A)
A.shape
A.shape[0]
```

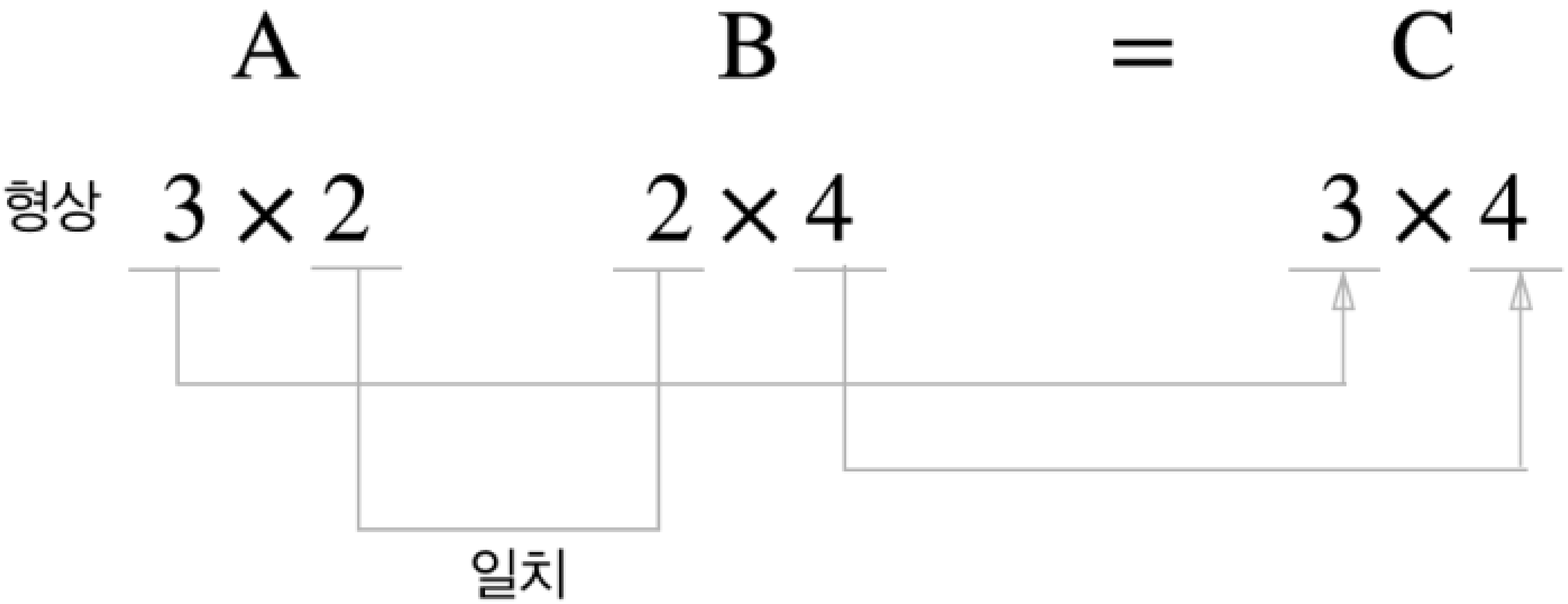
- 2차원 배열 : 행렬(matrix)



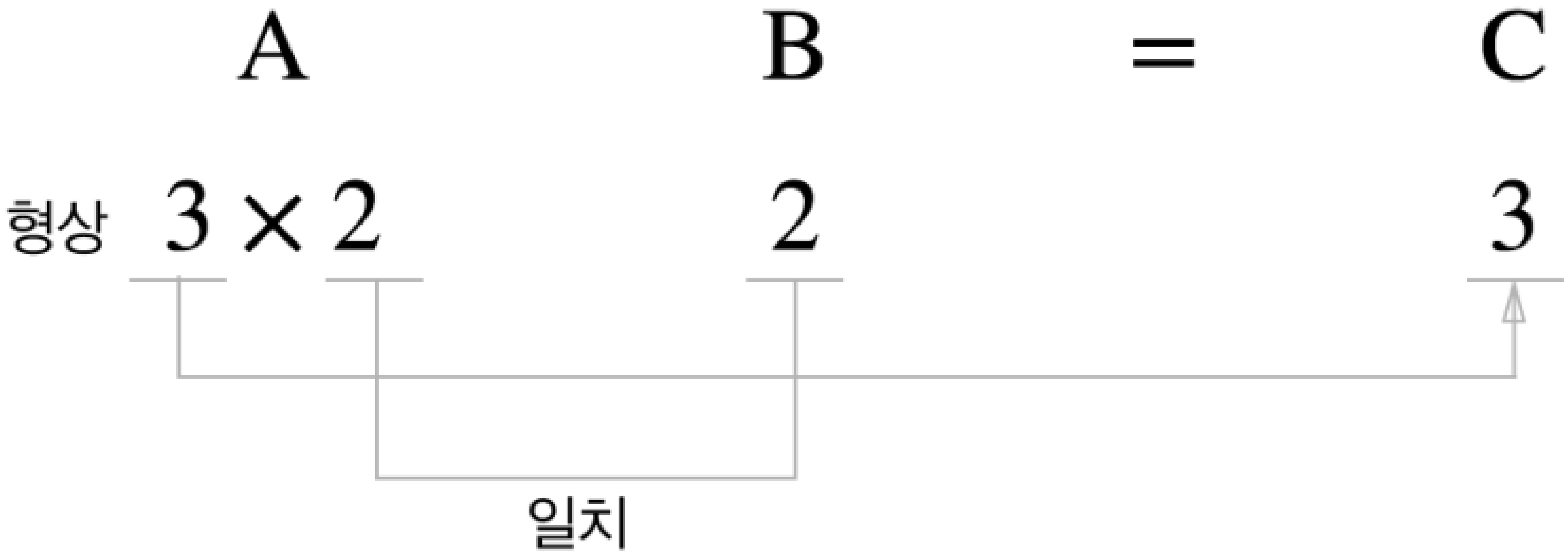
행렬의 내적(행렬 곱)



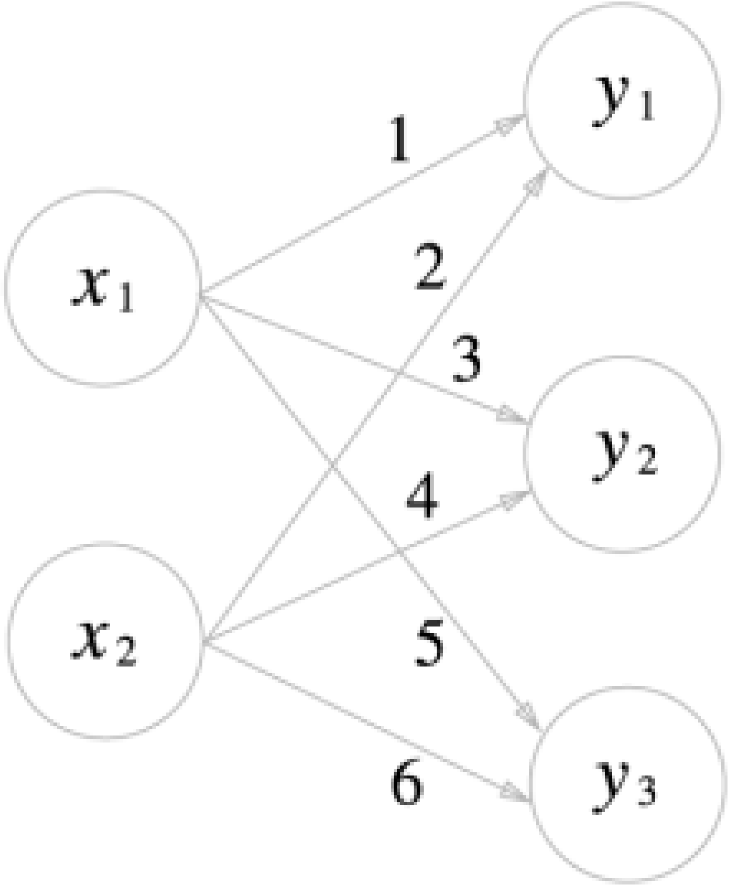
행렬의 내적(행렬 곱)



행렬의 내적(행렬 곱)



신경망의 내적



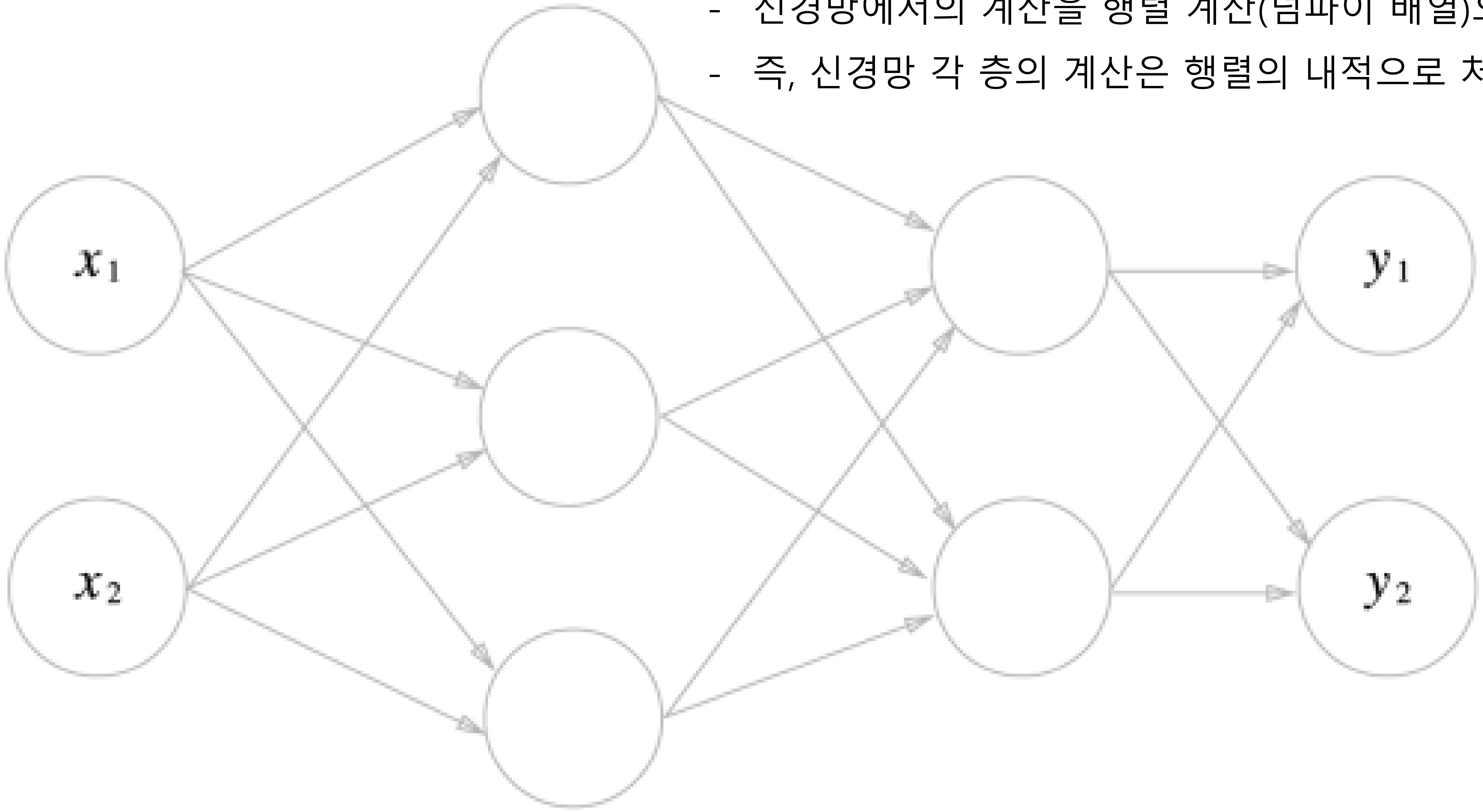
$$\begin{pmatrix}1 & 3 & 5\\ 2 & 4 & 6\end{pmatrix}$$

$$\begin{array}{cccc} \mathbf{X} & \mathbf{W} & = & \mathbf{Y} \\ 2 & 2 \times 3 & & 3 \end{array}$$

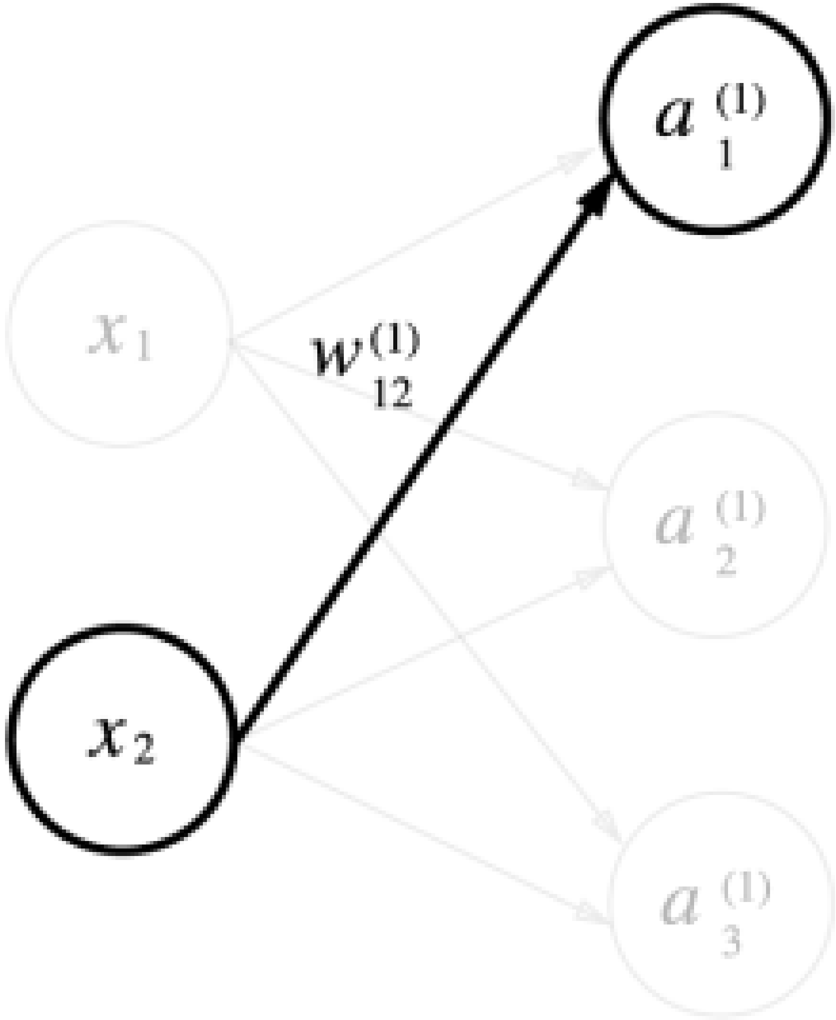
일치

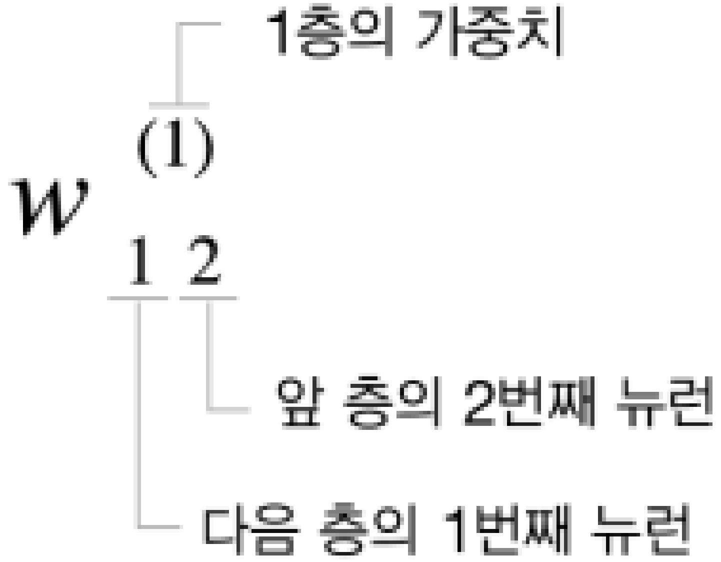
3층 신경망 구현하기

- 신경망에서의 계산을 행렬 계산(넘파이 배열)으로 정리할 수 있다.
- 즉, 신경망 각 층의 계산은 행렬의 내적으로 처리할 수 있다.

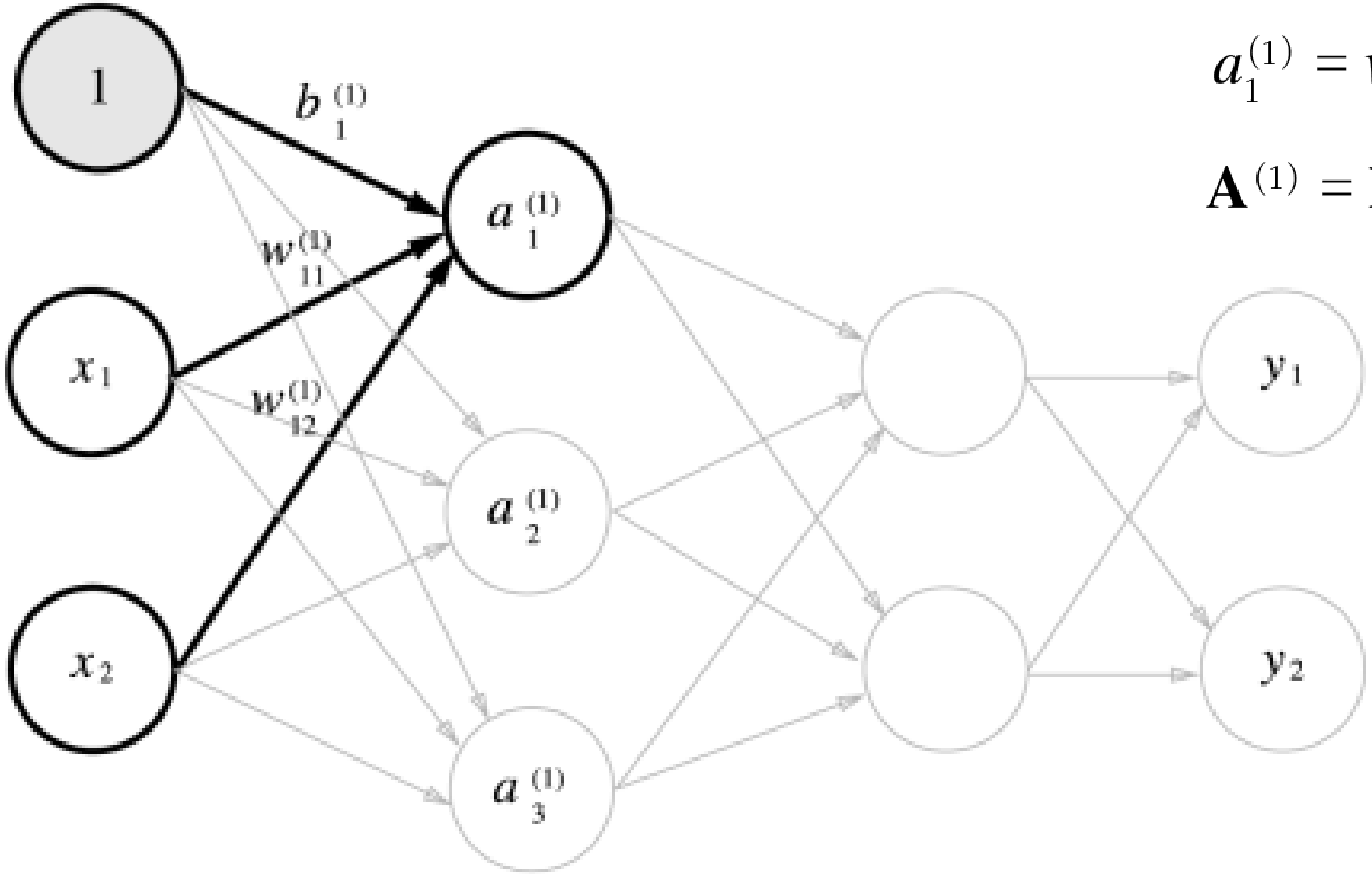


신경망 표기법





각 층의 신호 전달 구현 – 입력층에서 1층으로 신호 전달



$$a_1^{(1)} = w_{11}^{(1)}x_1 + w_{12}^{(1)}x_2 + b_1^{(1)}$$

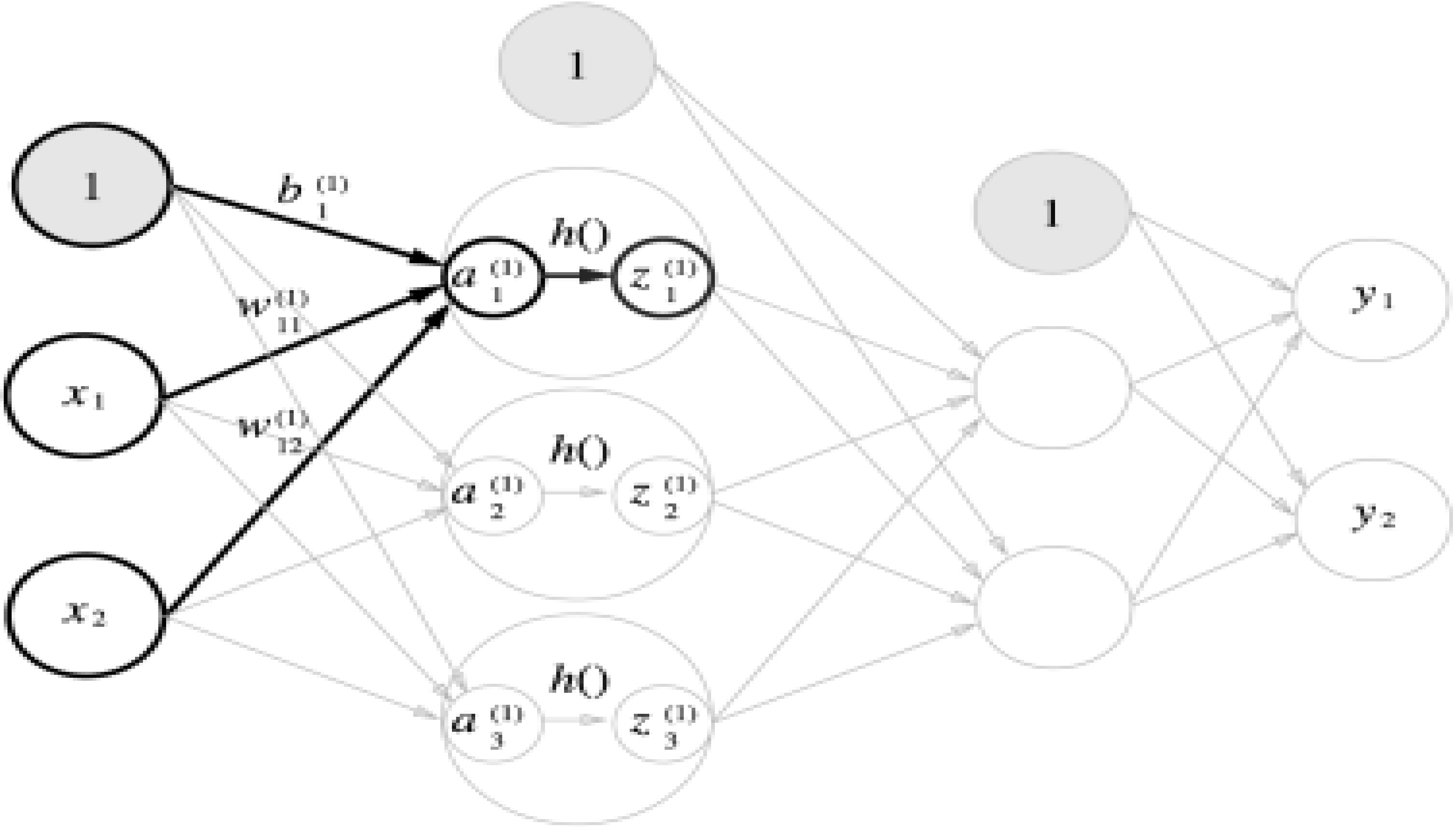
$$\mathbf{A}^{(1)} = \mathbf{XW}^{(1)} + \mathbf{B}^{(1)}$$

$$\boldsymbol{A}^{(1)} = (a_1{}^{(1)}\ a_2{}^{(1)}\ a_3{}^{(1)})$$

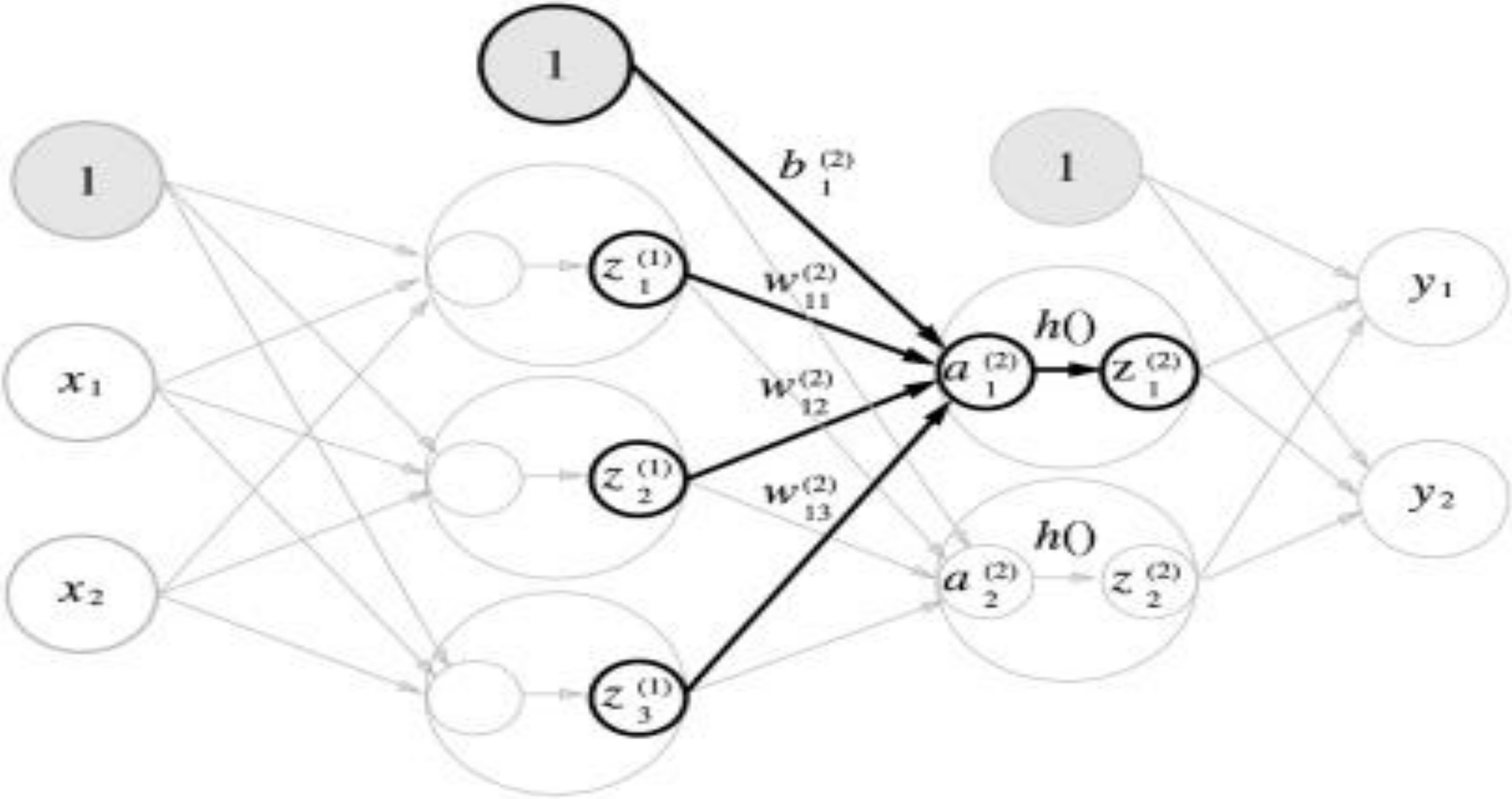
$$\mathbf{X} = (x_1\ \ x_2)$$

$$\boldsymbol{B}^{(1)} = (b_1{}^{(1)}\ b_2{}^{(1)}\ b_3{}^{(1)})$$

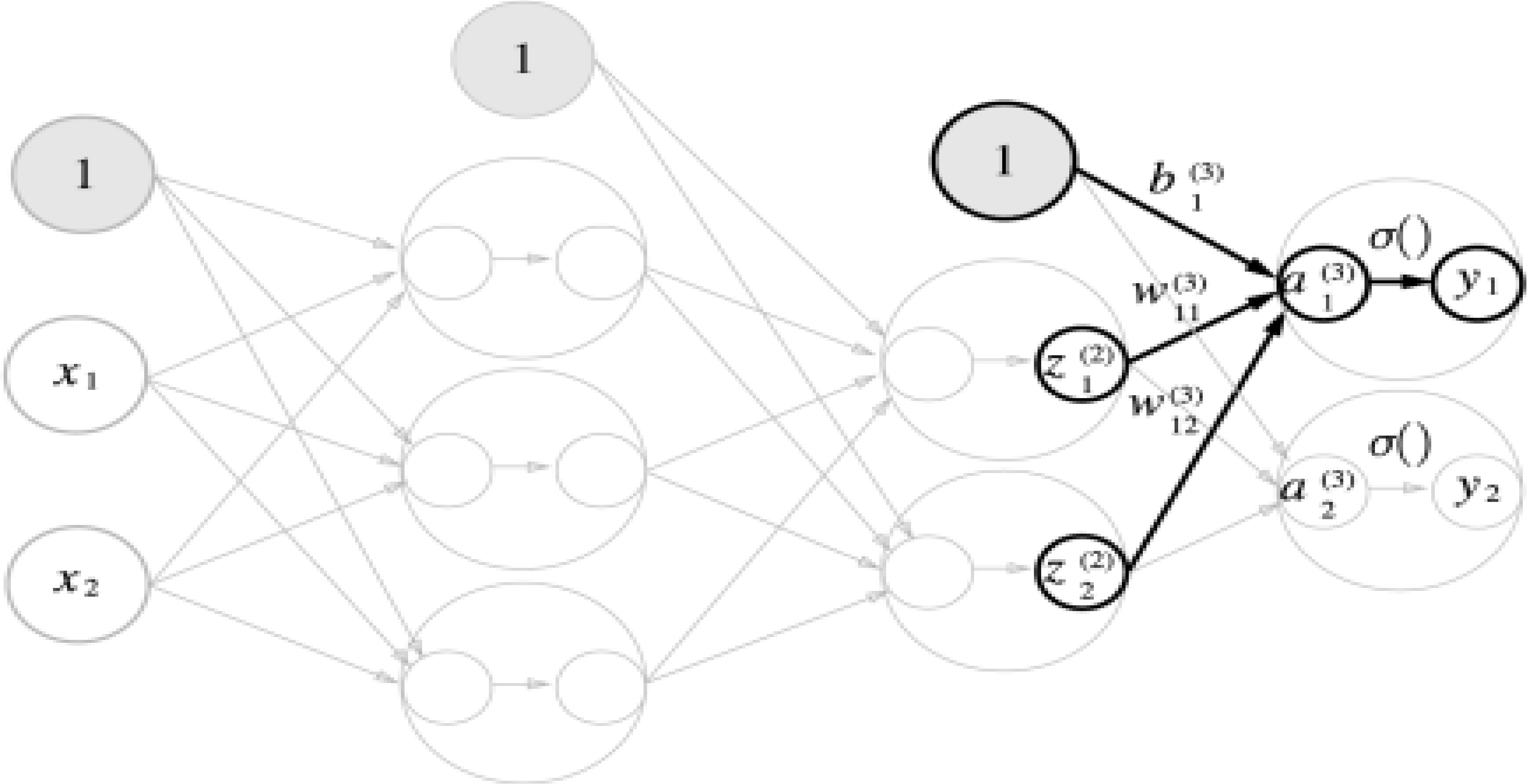
입력층에서 1층으로의 신호 전달



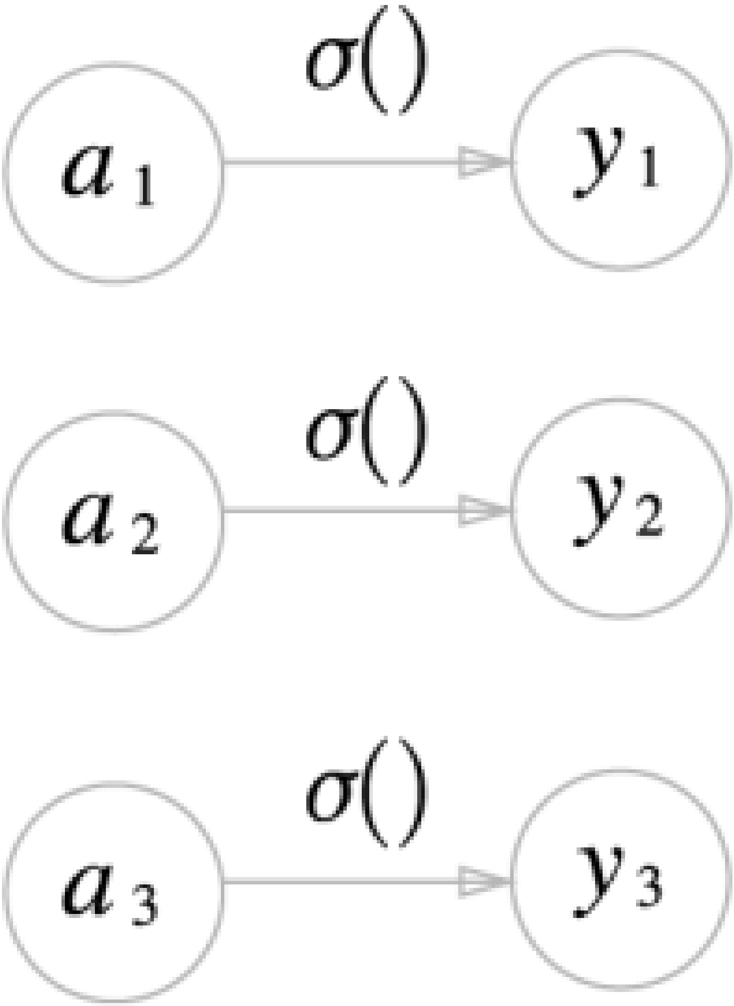
1층에서 2층으로의 신호 전달



2층에서 3층으로의 신호 전달

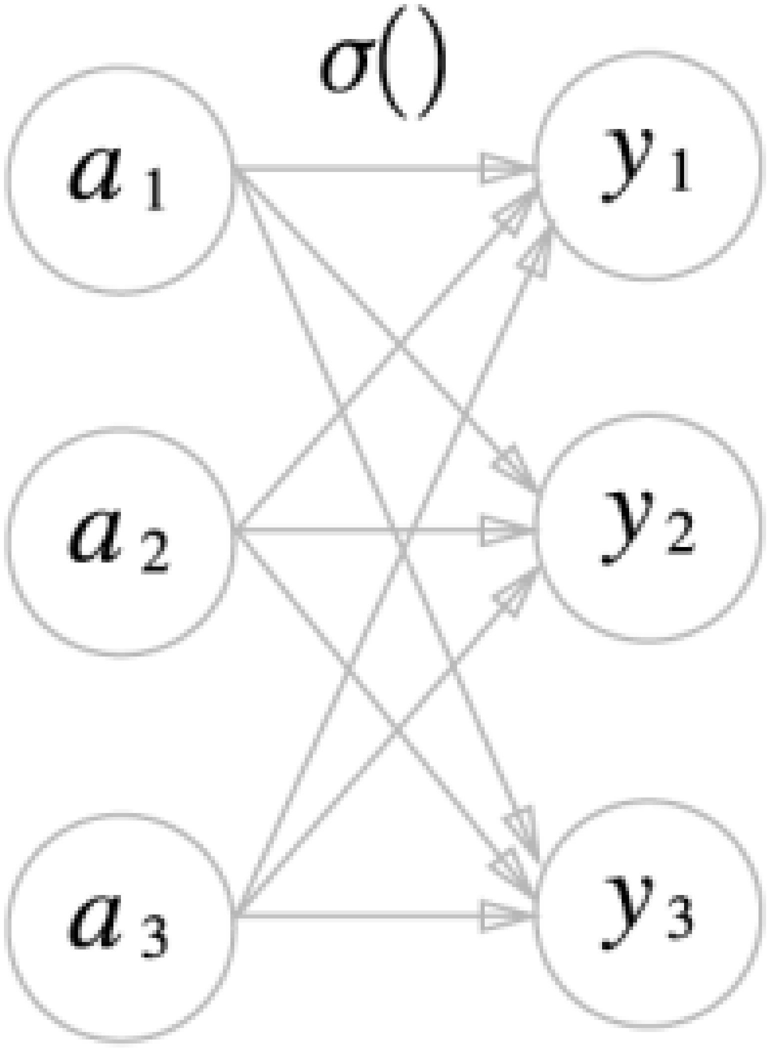


출력층 설계 (1) – 항등함수



- 항등 함수(identity function) : 회귀
  - 입력을 그대로 출력

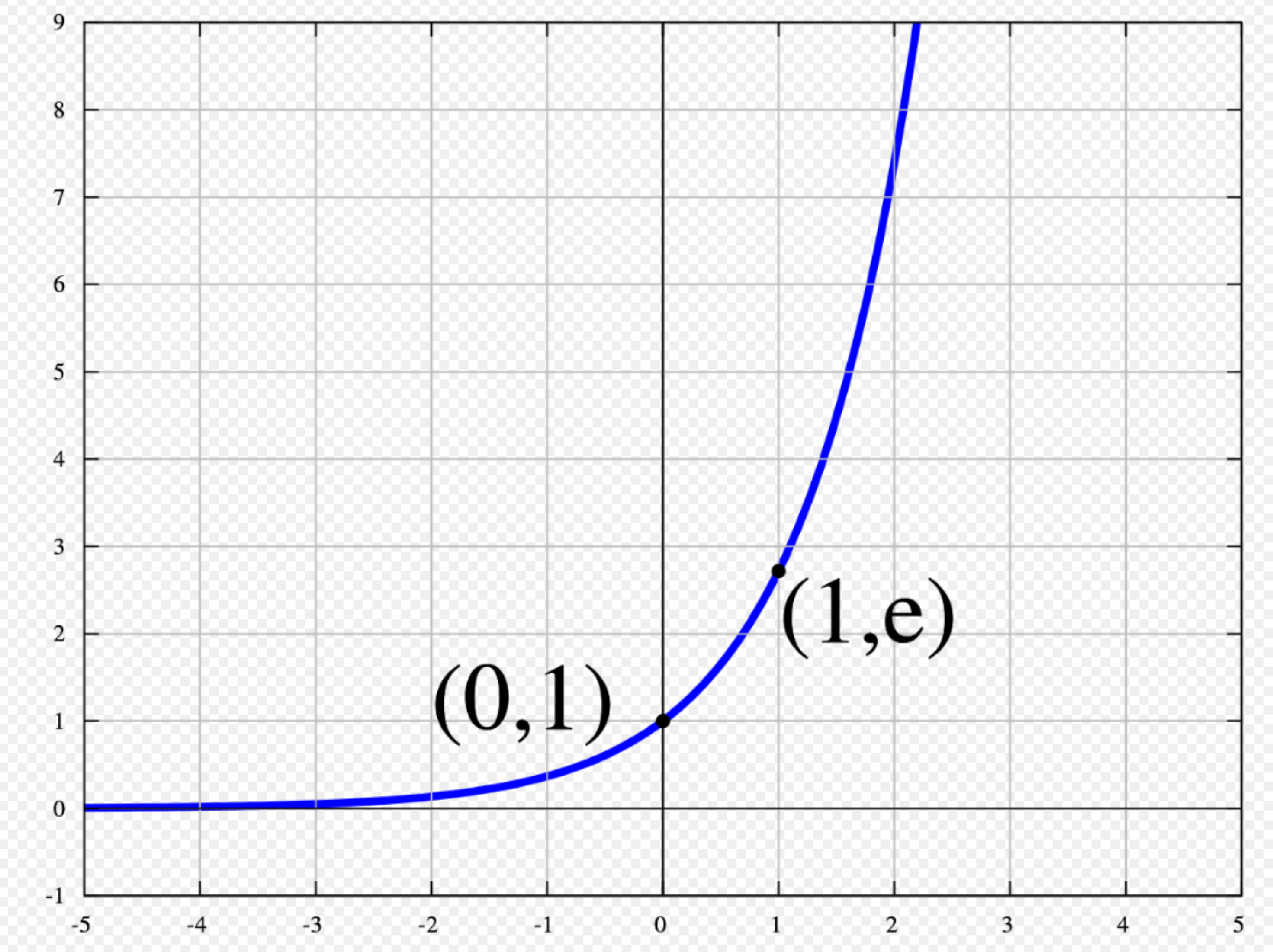
출력층 설계(2) – 소프트맥스 함수



- 소프트맥스(softmax function) : 분류

$$y_k = \frac{\exp(a_k)}{\sum_{i=1}^{n} \exp(a_i)}$$

출력층 설계(3) – 소프트맥스 함수



자연 지수 함수 $y = e^x$

소프트맥스 함수 구현 시 주의점

- 오버플로 문제
  - 소프트맥스의 지수 함수를 계산할 때 어떤 중수를 더하거나 빼도 결과는 바뀌지 않는다는 것을 이용.
  - 오버플로를 막을 목적으로는 입력 신호 중 최댓값을 이용하는 것이 일반적.

$$
\begin{aligned}
y_k = \frac{\exp(a_k)}{\sum_{i=1}^{n}\exp(a_i)} &= \frac{C\exp(a_k)}{C\sum_{i=1}^{n}\exp(a_i)} \\
&= \frac{\exp(a_k + \log C)}{\sum_{i=1}^{n}\exp(a_i + \log C)} \\
&= \frac{\exp(a_k + C')}{\sum_{i=1}^{n}\exp(a_i + C')}
\end{aligned}
$$

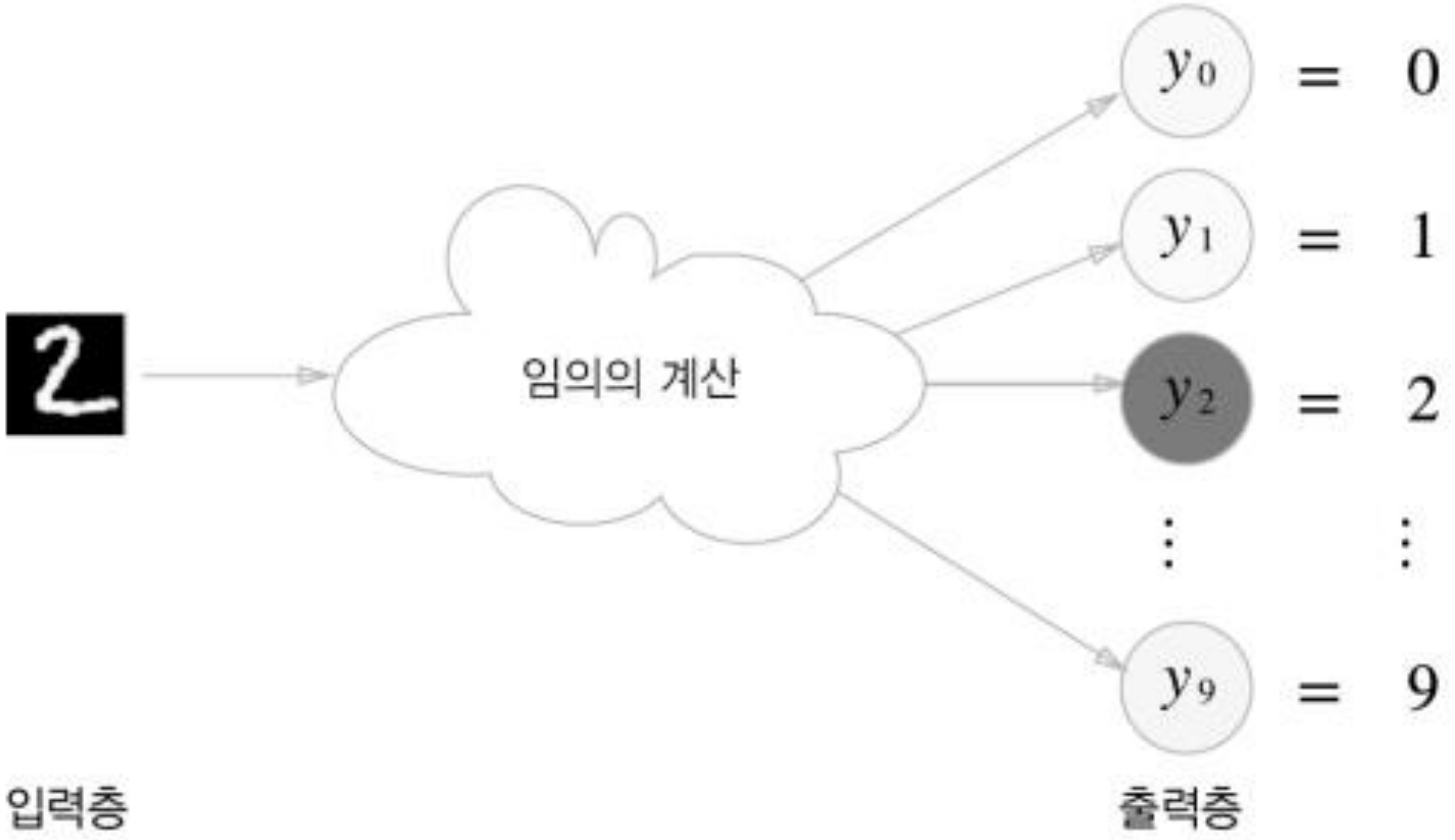
소프트맥스 함수의 특징

- 소프트맥스 함수의 출력은 0에서 1사이의 실수.
- 출력의 총합은 1.
- 출력을 확률로 해석할 수 있음.
- 신경망을 학습시킬 때는 출력층에서 소프트맥스 함수를 사용.
- 추론 단계에서는 출력층의 소프트맥스 함수를 생략하는 것이 일반적.
  - 각 원소의 대소 관계는 변하지 않음.
  - 지수 함수 $y = e^x$ 가 단조 증가 함수이기 때문.

MNIST 데이터셋

- Modified National Institude of Standards and Technology.
- 손으로 직접 쓴 숫자(필기체 숫자)들로 이루어진 데이터 셋.
- 0 ~ 9까지의 숫자 이미지로 구성되며, 60,000개의 트레이닝 데이터와 10,000개의 테스트 데이터로 이루어져 있음.
- 28 x 28 size의 흑백(1ch) 이미지 데이터.

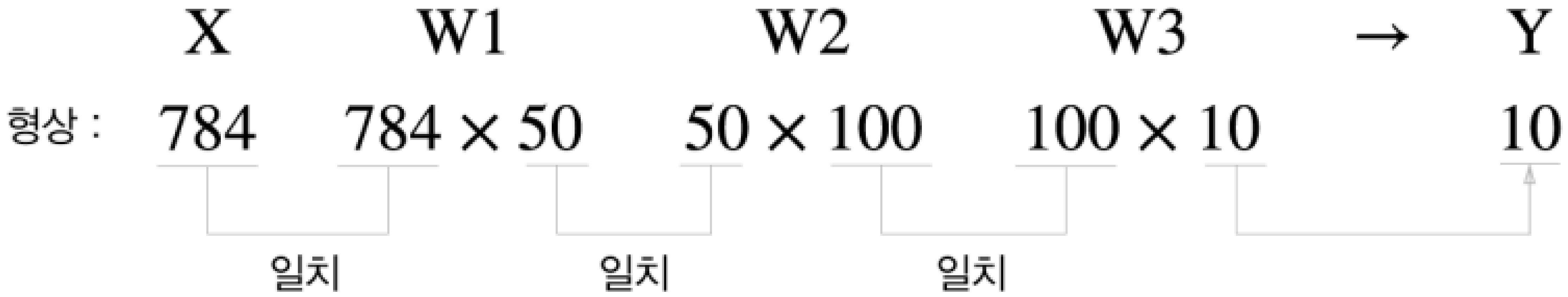
출력층의 뉴런 수 정하기

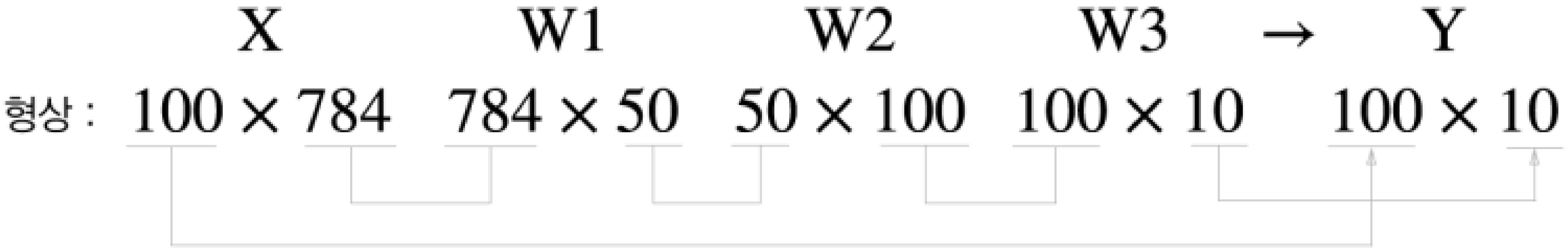


손글씨 숫자 인식


tmp2ojurv7v.BMP
View
Zoom
Share
Highlight
Rotate
Markup
Search...

신경망 각 층의 배열 형상 추이 / 배치 처리를 위한 배열들의 형상 추이





신경망 학습

신경망 학습

- 학습
  - 훈련 데이터로부터 가중치 매개변수의 최적 값을 자동으로 획득하는 것.

- 신경망 학습 : 데이터로부터 매개변수의 값을 정하는 방법.

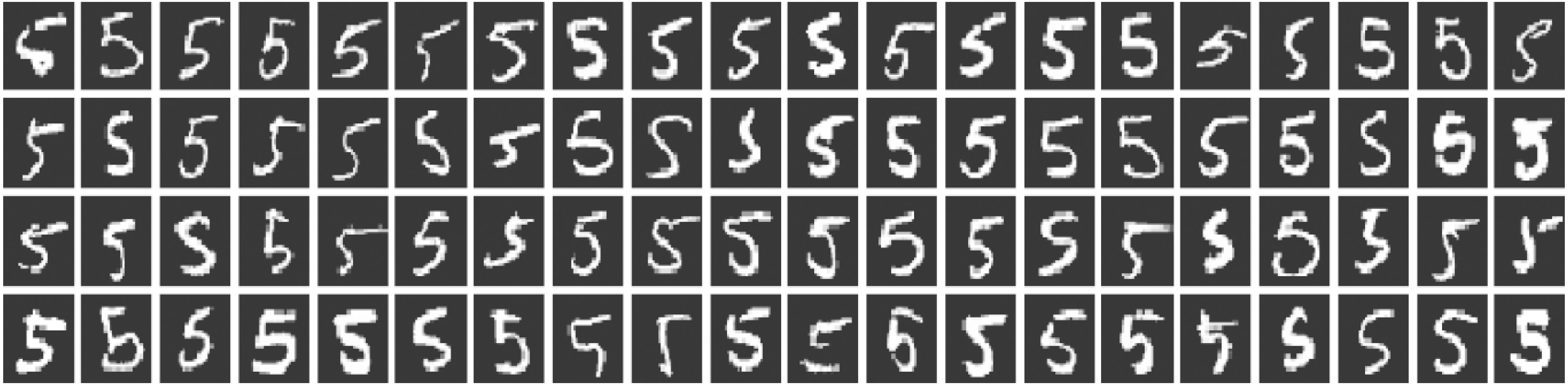
- 손실 함수
  - 신경망이 학습할 수 있도록 해주는 지표.
  - **손실 함수의 결과값을 가장 작게 만드는 가중치 매개 변수를 찾는 것이 학습의 목표.**

데이터 주도 학습(기계학습)

- 신경망의 특징
  - 데이터를 보고 학습할 수 있다.
  - 가중치 매개변수의 값을 데이터를 보고 자동으로 결정한다는 뜻.
- 사람의 개입을 최소화하고, 수집한 데이터로부터 답을 찾고, 패턴을 찾으려는 시도.
- 특히, 신경망과 딥러닝은 기존 기계학습에서 사용하던 방법보다 사람의 개입을 더 욱 배제할 수 있게 해주는 중요한 특성을 지님.

머신 러닝과 신경망(딥러닝) 방식

- MNIST dataset 중 숫자 ‘5’의 예 : 사람마다 자신만의 필체가 있다.



데이터 주도 학습(기계학습)

- 이미지에서 특징(feature)을 추출하고, 그 특징의 패턴을 기계학습 기술로 학습.
  - 특징(feature) : 입력 데이터(입력 이미지)에서 본질적인 데이터(중요한 데이터)를 정확하게 추출할 수 있도록 설계된 변환기(컴퓨터비전 분야:SIFT, SURF, HOG)를 일컬음.
  - 이미지 데이터의 특징을 벡터로 변환하고, 변환된 벡터를 가지고 학습.
- 다만, 이미지를 벡터로 변환할 때 사용하는 특징은 여전히 사람이 설계하는 것임에 주의.

머신 러닝과 신경망(딥러닝) 방식

훈련 데이터와 시험 데이터

- 훈련 데이터(training data)
  - 훈련 데이터만 사용하여 학습하면서 최적의 매개변수를 찾는다.

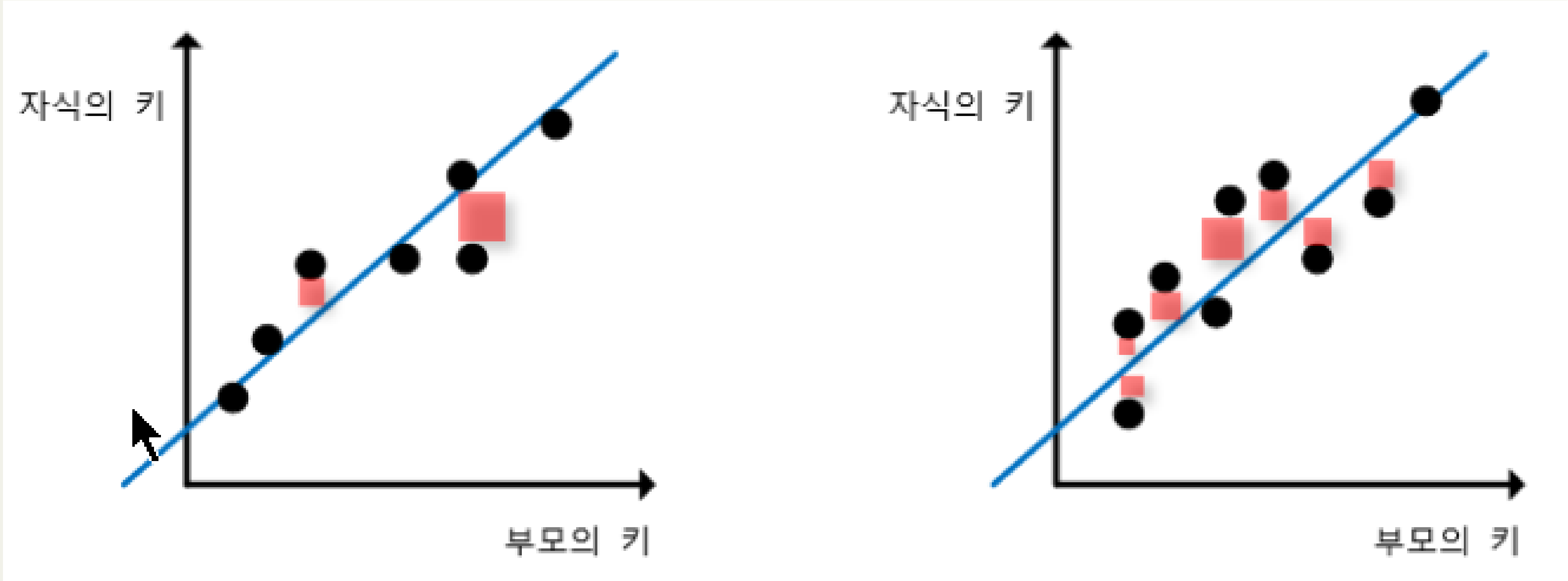
- 시험 데이터(test data)
  - 앞서 훈련한 모델의 실력을 평가하는 것.

- 훈련 / 시험 데이터 분리 이유
  - 우리가 원하는 것은 범용적으로 사용할 수 있는 모델 구현.
  - 범용 능력을 제대로 평가하기 위해 모델을 찾아내는 것이 기계 학습의 최종 목표.

- 오버피팅(overfitting) : 한 데이터 셋에만 지나치게 최적화된 상태.

손실 함수(loss function) 혹은 비용 함수(cost function)

- 평균 제곱 오차(mean squared error, MSE)
  - 가장 많이 쓰이는 손실 함수
  - $$MSE = \frac{1}{N}\sum_{i=1}^{N}(p_i - y_i)^2$$
  - 각 원소의 출력(추정 값)과 정답 레이블(참 값)의 차를 제곱한 후, 그 총합의 평균을 구한다.



손실 함수(loss function) 혹은 비용 함수(cost function)

- 교차 엔트로피 오차(cross entropy error, CEE)

  - $$CEE = -\sum_{i=1}^{n} t_i \times log_e(y_i)$$, 데이터 하나에 대한 손실 함수

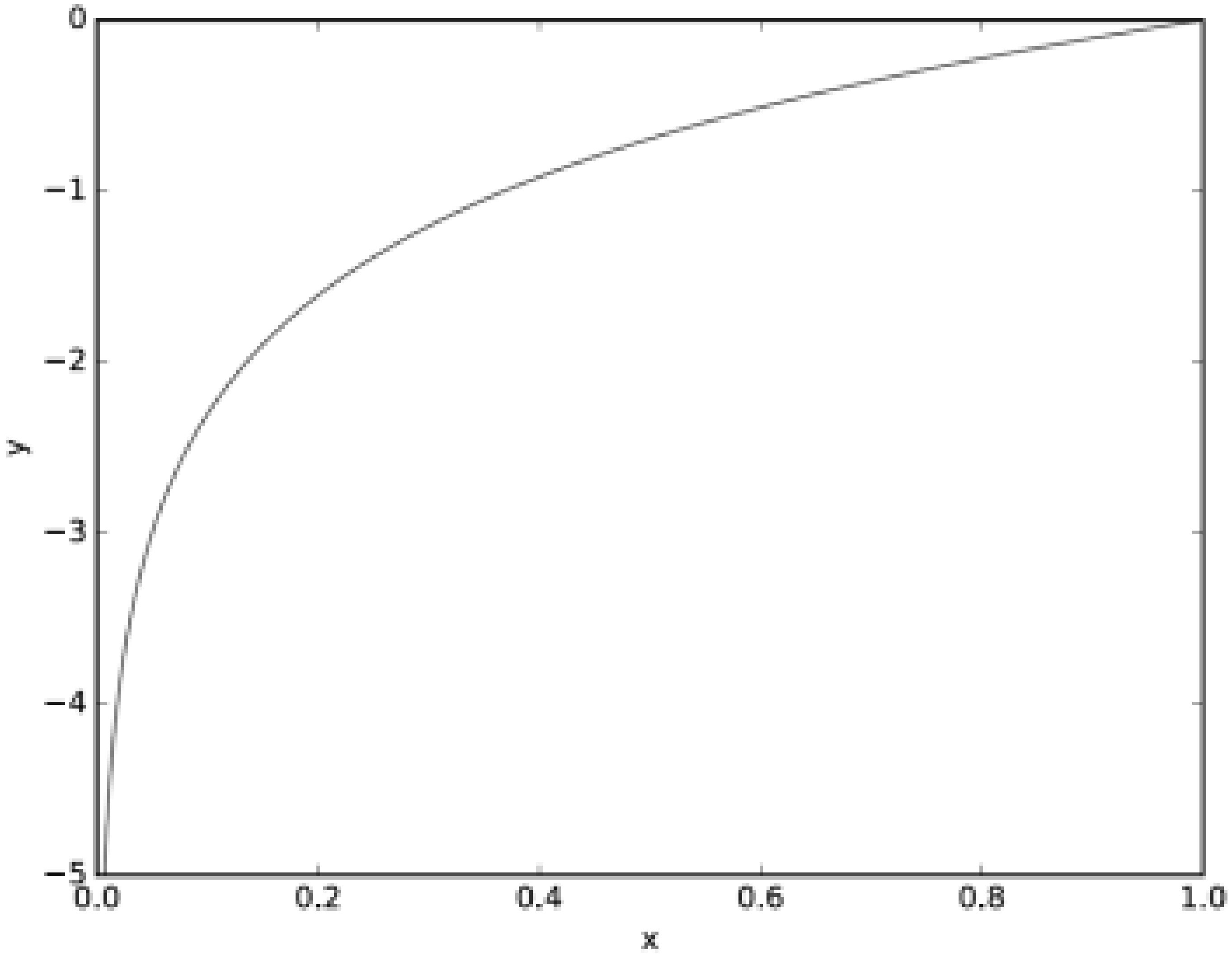
  - 교차 엔트로피 오차는 정답일 때의 출력이 전체 값을 정하게 된다.

- 이제 훈련 데이터 모두에 대한 손실 함수의 합을 구하는 방법을 생각

  - $$E = -\frac{1}{N}\sum_{n}\sum_{k} t_{nk} \log y_{nk}$$ - 교차 엔트로피 오차

  - 데이터 하나에 대한 손실 함수에서 N개의 데이터로 확장. 다만, N으로 나누어 정규화.
  - 평균 손실 함수 – 훈련 데이터 개수와 관계없이 언제든 통일된 지표를 얻을 수 있음.

자연로그 $y = \log x$ 의 그래프

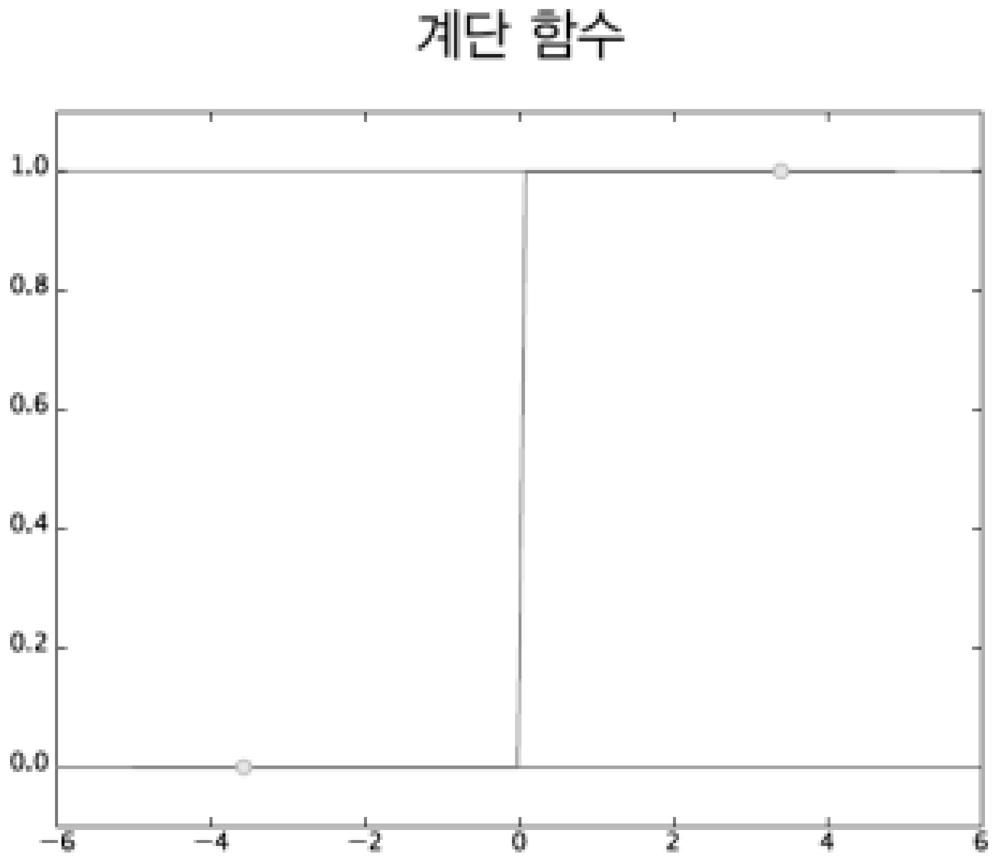


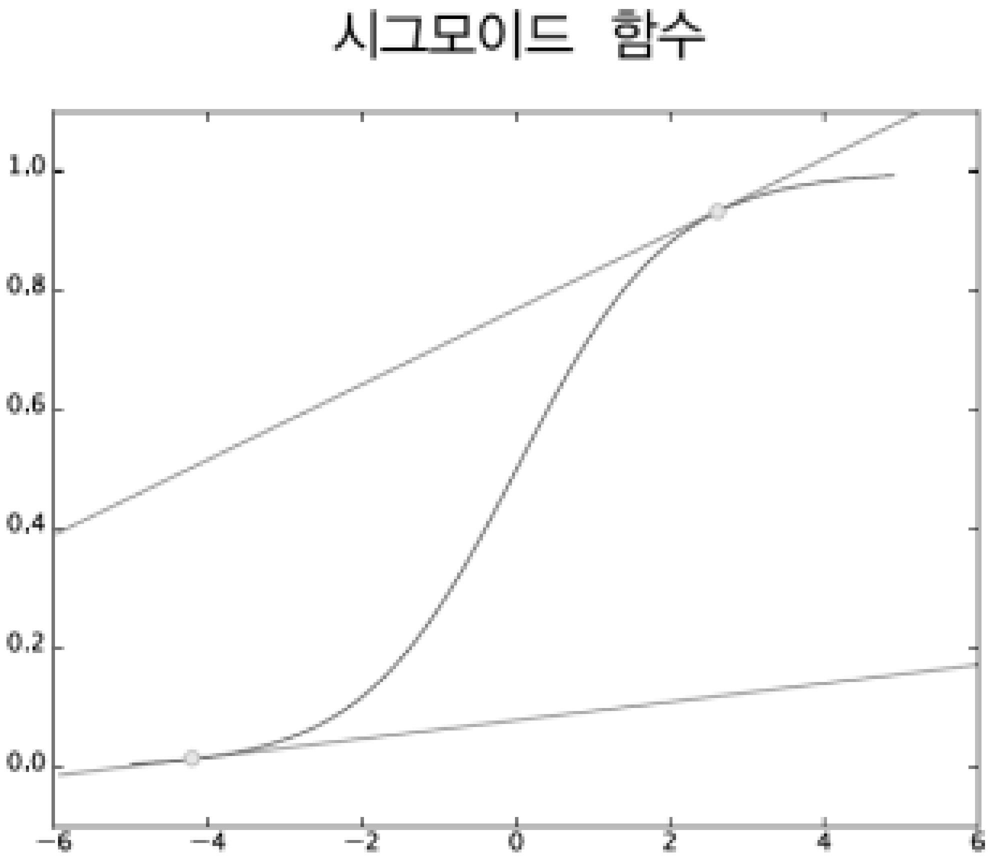
- x가 1일 때 y는 0이 되고, x가 0에 가까워질수록 y의 값은 점점 작아짐.

미니배치 학습

- 미니배치(mini-batch) 학습
  - 모든 훈련 데이터를 대상으로 손실 함수의 합을 구하려면 많은 시간이 걸리고, 현실적이지 않음.
  - 훈련 데이터 중 일부(미니배치)만 골라 학습을 수행.
  - 가령 60,000장의 훈련 데이터 중에서 100장을 무작위로 뽑아 학습(미니배치 학습)하는 것.
- 미니배치의 손실 함수 계측을 통해 전체 훈련 데이터의 '근사치'로 이용.

손실 함수 설정



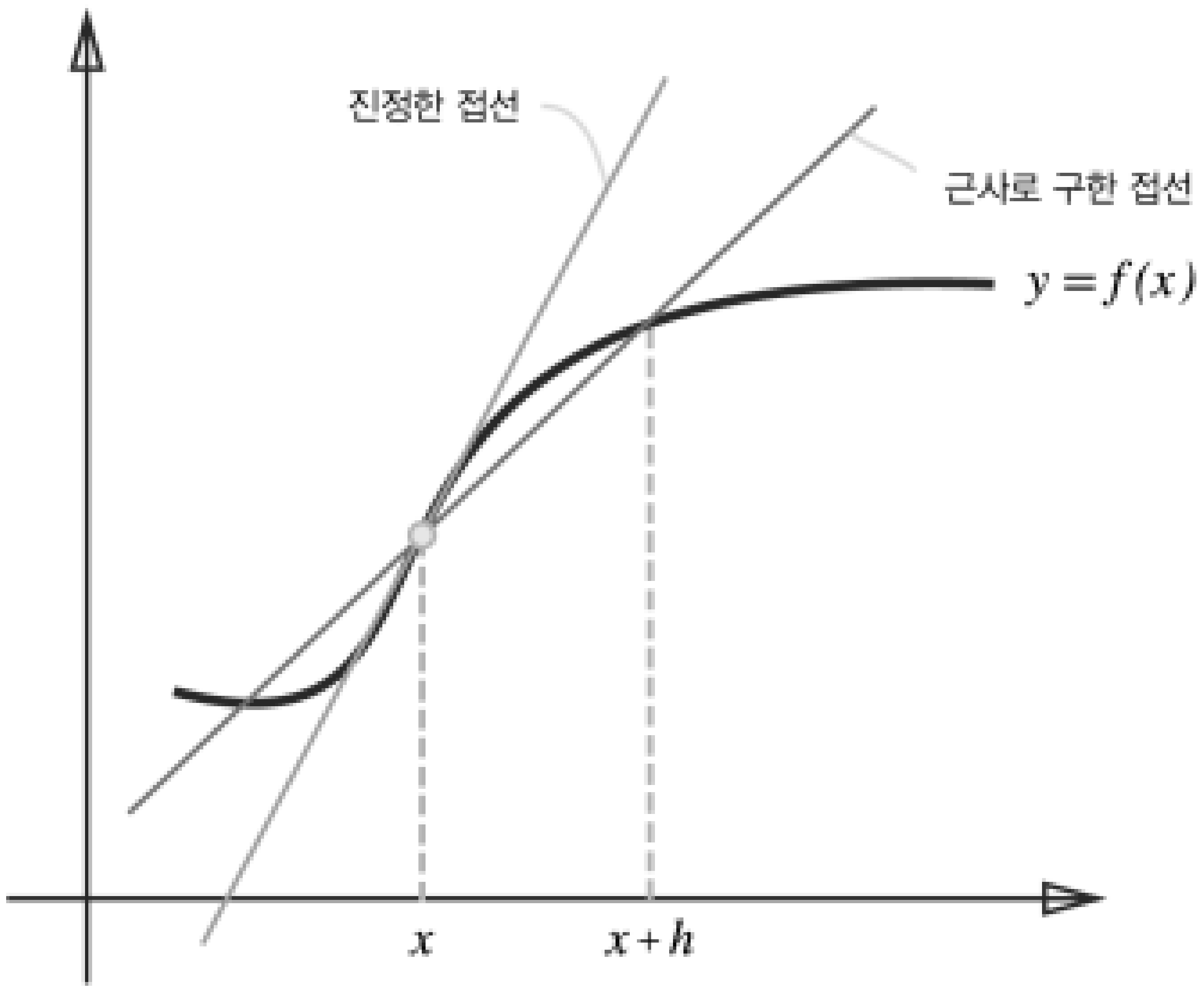


수치 미분

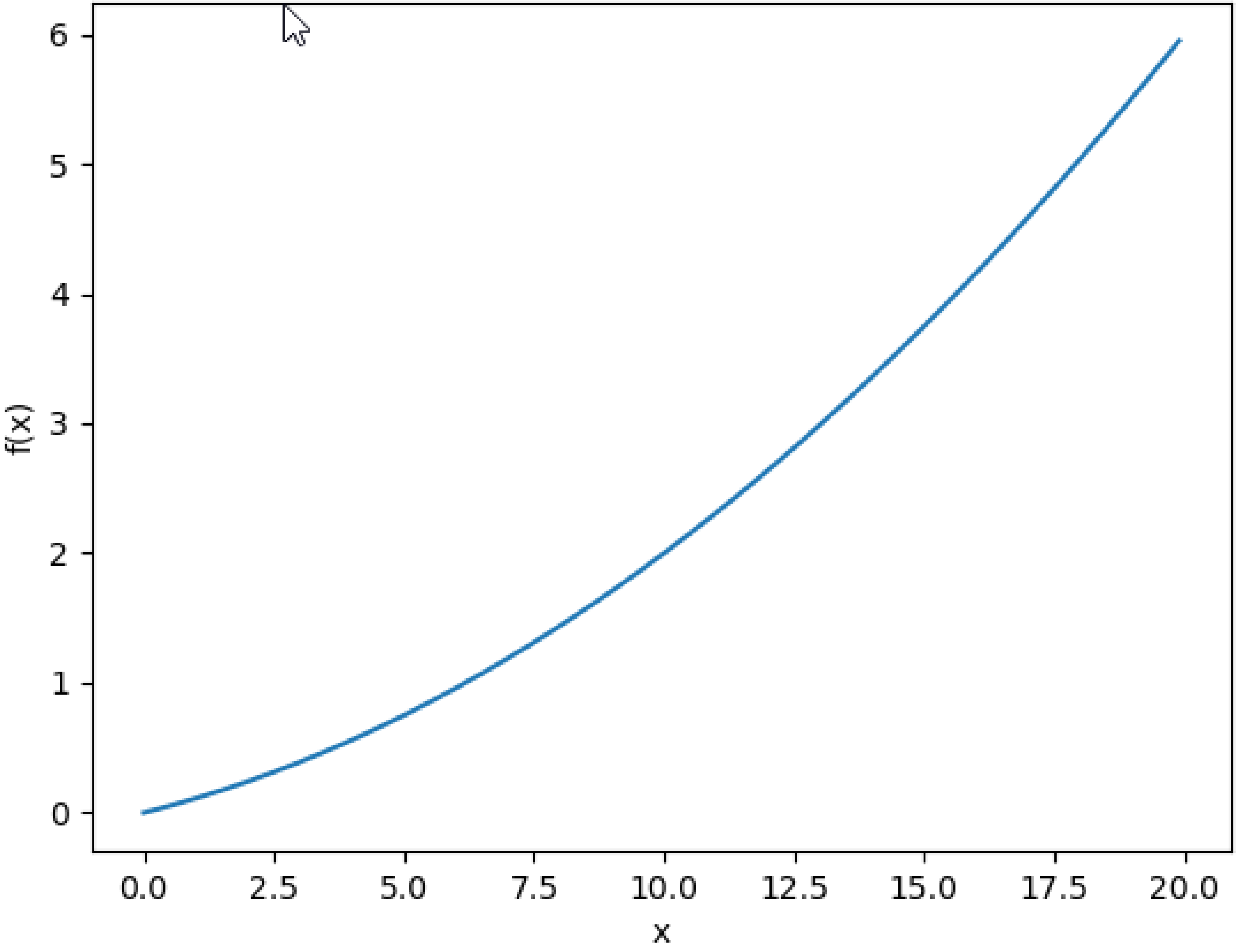
- 미분
  - x의 '작은 변화'가 함수 f(x)를 얼마나 변화시키느냐를 의미.

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

진정한 미분(접선)과 수치 미분(근사로 구한 접선)의 값



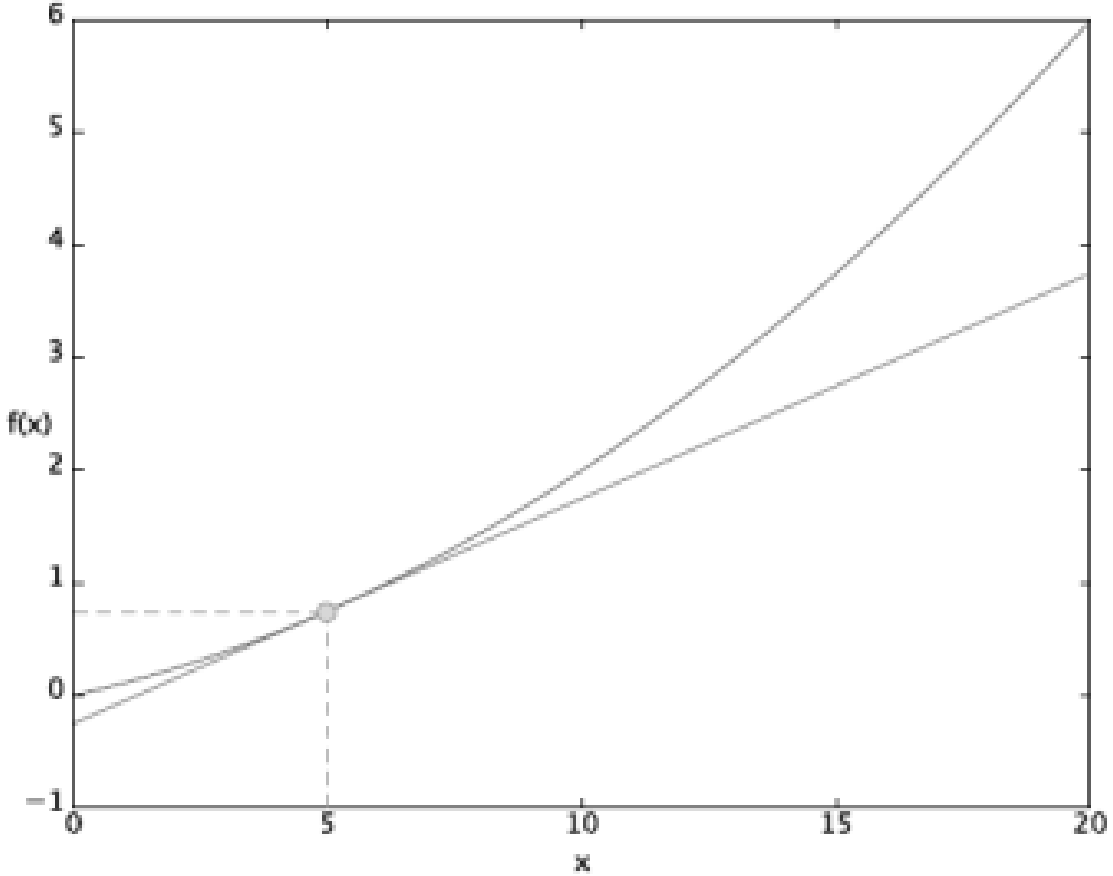
수치 미분의 예



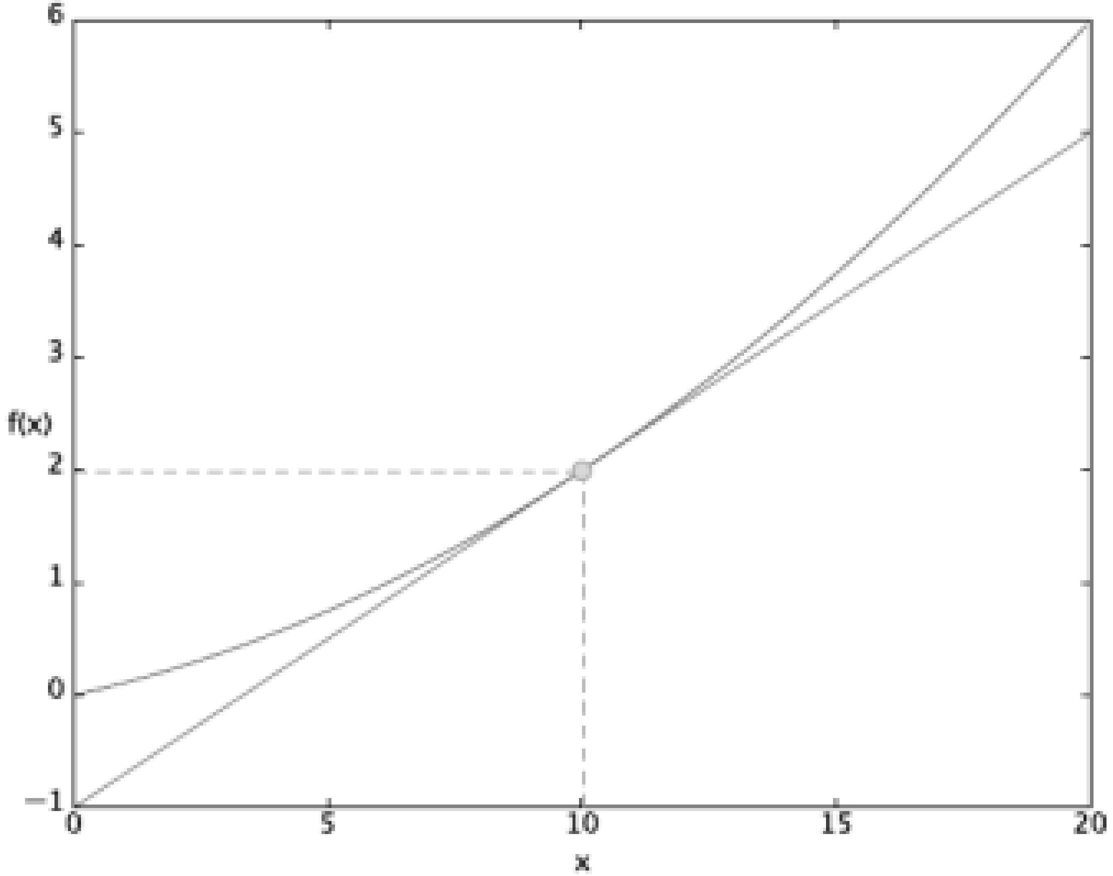
$$y = 0.01x^2 + 0.1x$$

$$y' = 0.02x + 0.1$$

수치 미분의 예



x = 5에서의 접선

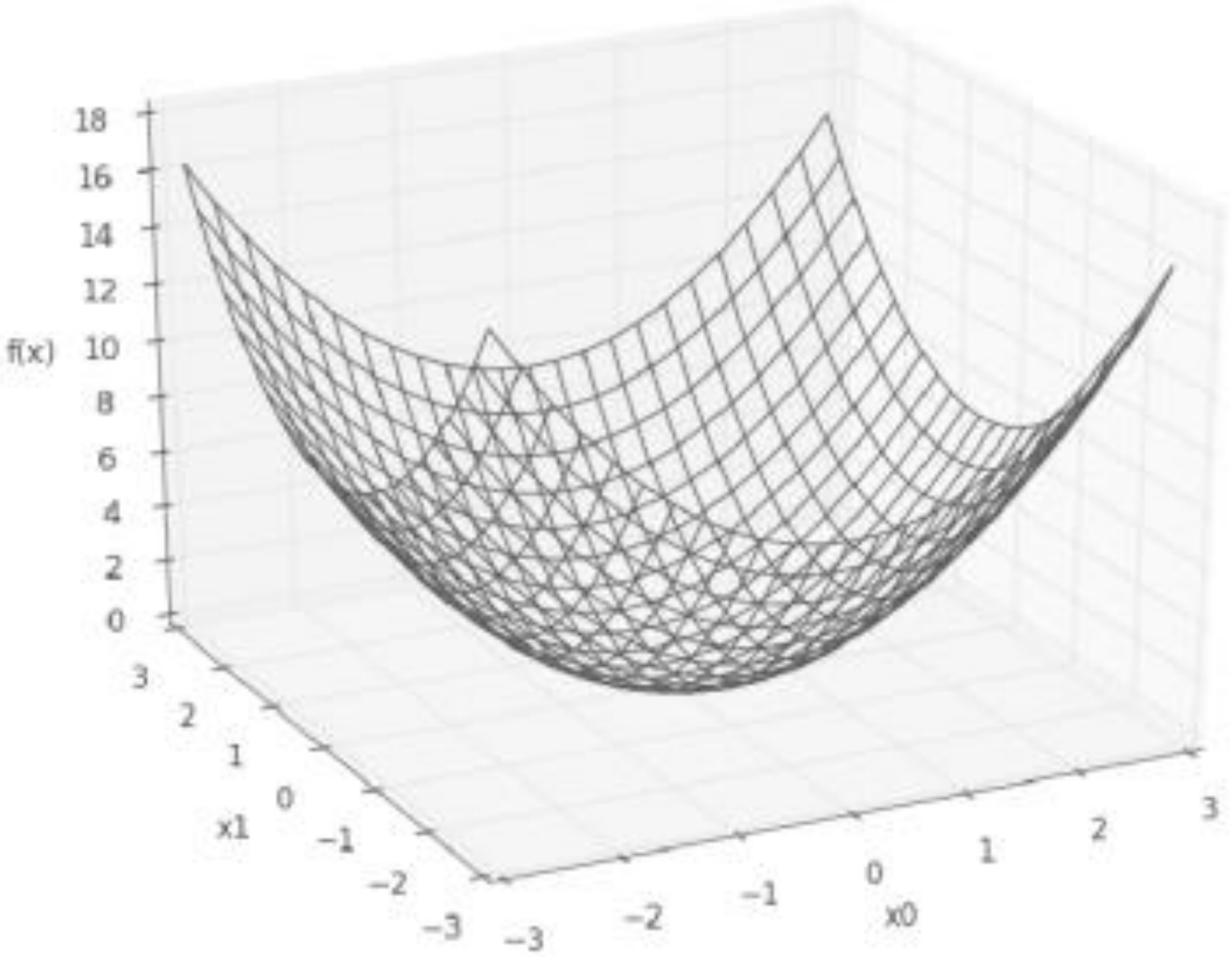


x = 10에서의 접선

편미분

$$f(x_0, x_1) = x_0^2 + x_1^2$$

- $\frac{\partial f}{\partial x_0} = 2x_0$
- $\frac{\partial f}{\partial x_1} = 2x_1$



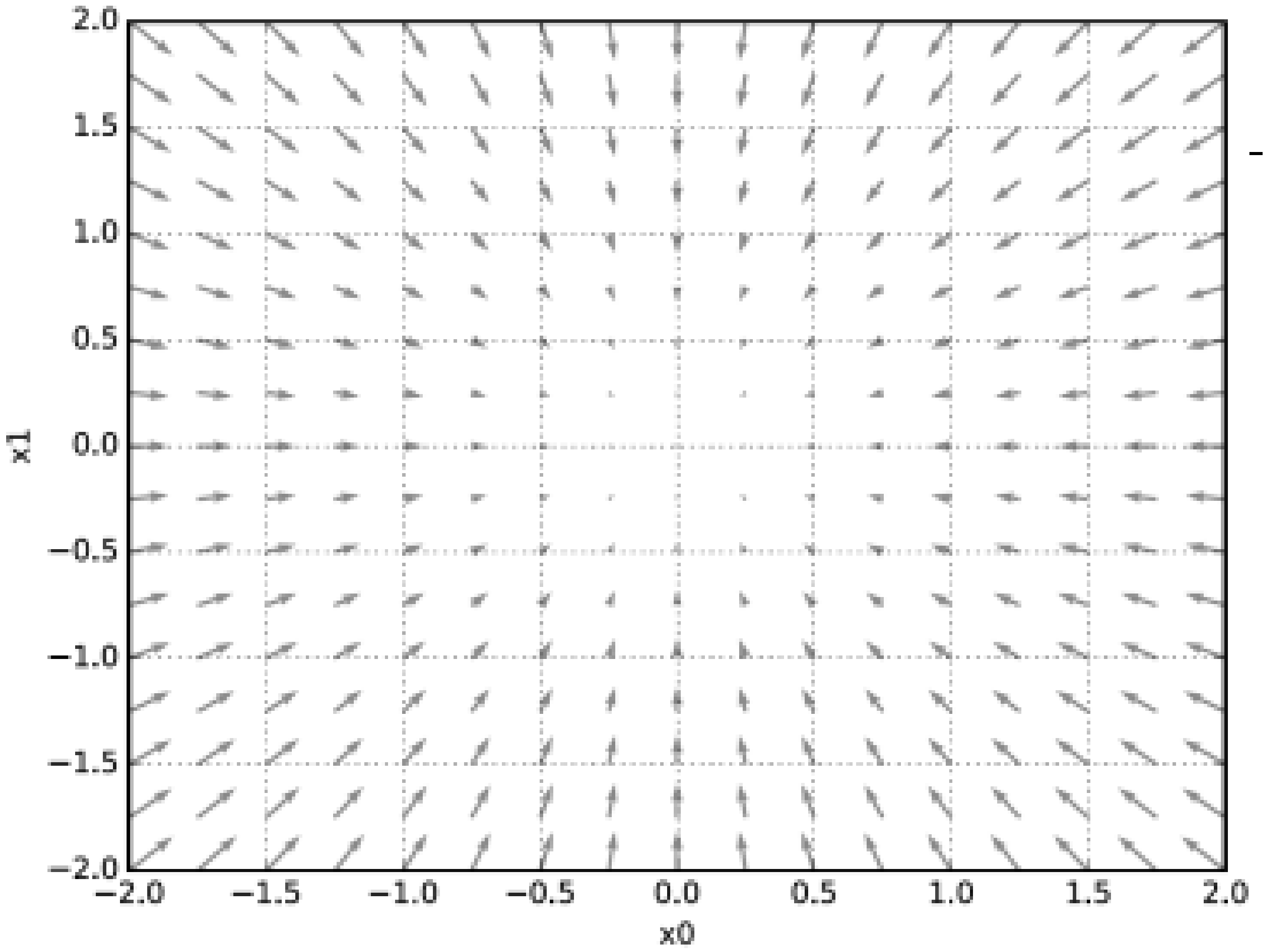
기울기(Gradient)

- 모든 변수의 편미분을 벡터로 정한 것.
- 즉, 모든 변수의 편미분을 동시에 계산하고 싶다면

$$f(x_0,\ x_1) = x_0^2 + x_1^2$$ 에서

$$\left(\frac{\partial f}{\partial x_0},\quad \frac{\partial f}{\partial x_1}\right)$$ 처럼 양쪽의 편미분을 묶어서 계산

$f(x_0, x_1) = x_0^2 + x_1^2$ 의 기울기



- 기울기의 특징

- 기울기 그림은 방향을 가진 벡터(화살표)로 그려짐.
- 기울기는 함수의 '가장 낮은 장소(최소값)'를 가리킴.
- '가장 낮은 곳'에서 멀어질수록 화살표의 크기가 커짐.
- **기울기가 가리키는 쪽은 각 장소에서 함수의 출력 값을 가장 크게 줄이는 방향**

경사법(경사 하강법)

- 신경망에서 최적의 매개변수(가중치와 편향)를 학습시에 찾아야 함.
- 여기서 최적이란 손실 함수가 최소값이 될 때의 매개변수 값.
- 그러나, 일반적인 문제의 손실 함수는 매우 복잡 – 매개변수의 공간이 광대하여 어디가 최소값이 되는 곳인지를 알아내기가 쉽지 않음.
- **기울기를 잘 이용해 함수의 최소값(또는 가능한 한 작은 값)을 찾으려는 것**이 경사 하강법.
- 각 지점에서 함수의 값을 낮추는 방안을 제시하는 지표가 기울기.

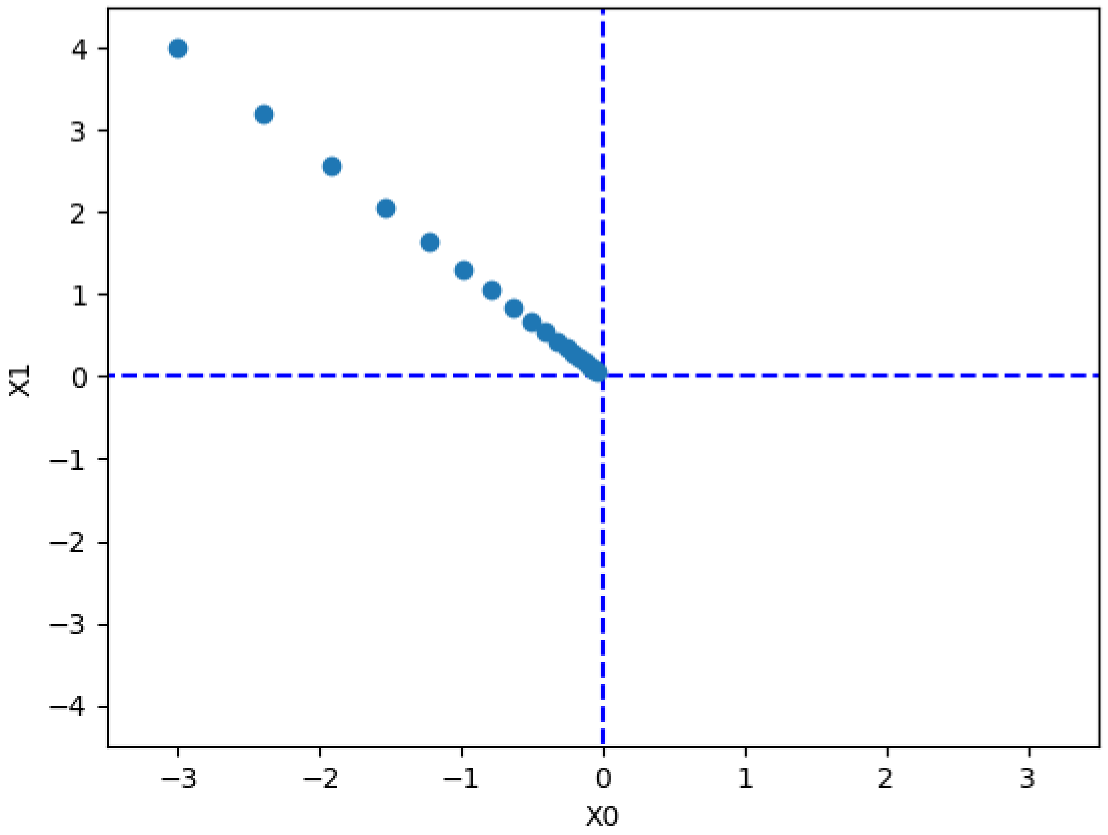
$$x_0 = x_0 - \eta \frac{\partial f}{\partial x_0}$$

$$x_1 = x_1 - \eta \frac{\partial f}{\partial x_1}$$

- η: 학습률(learning rate)
  - 매개변수 값을 얼마나 갱신하느냐를 정하는 것.

경사 하강법

- 경사 하강법에 의한 $f(x_0, x_1) = x_0^2 + x_1^2$ 의 갱신 과정



신경망에서의 기울기

- 가중치 매개변수에 대한 손실 함수의 기울기

- 예) 형상이 2 x 3, 가중치 W, 손실 함수 L인 신경망의 경우

$$\mathbf{W} = \begin{pmatrix} w_{11} & w_{21} & w_{31} \\ w_{12} & w_{22} & w_{32} \end{pmatrix}$$

$$\frac{\partial L}{\partial \mathbf{W}} = \begin{pmatrix} \frac{\partial L}{\partial w_{11}} & \frac{\partial L}{\partial w_{21}} & \frac{\partial L}{\partial w_{31}} \\ \frac{\partial L}{\partial w_{12}} & \frac{\partial L}{\partial w_{22}} & \frac{\partial L}{\partial w_{32}} \end{pmatrix},$$

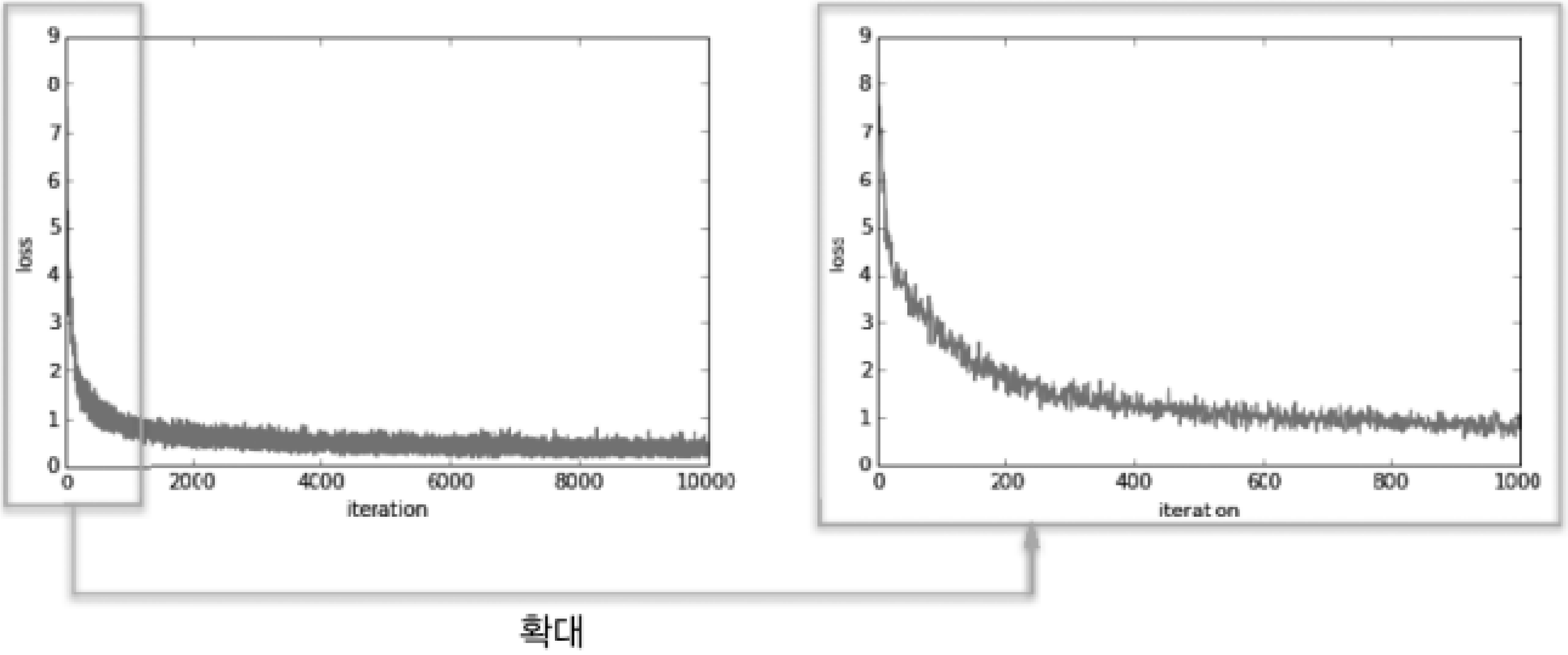
- $\frac{\partial \boldsymbol{L}}{\partial \boldsymbol{W}}$ 의 각 원소는 각각의 원소에 관한 편미분.
- $\frac{\partial \boldsymbol{L}}{\partial \boldsymbol{W}}$ 의 형상이 W와 같다는 점이 중요.

학습 알고리즘 구현하기 – 확률적 경사 하강법(SGD)

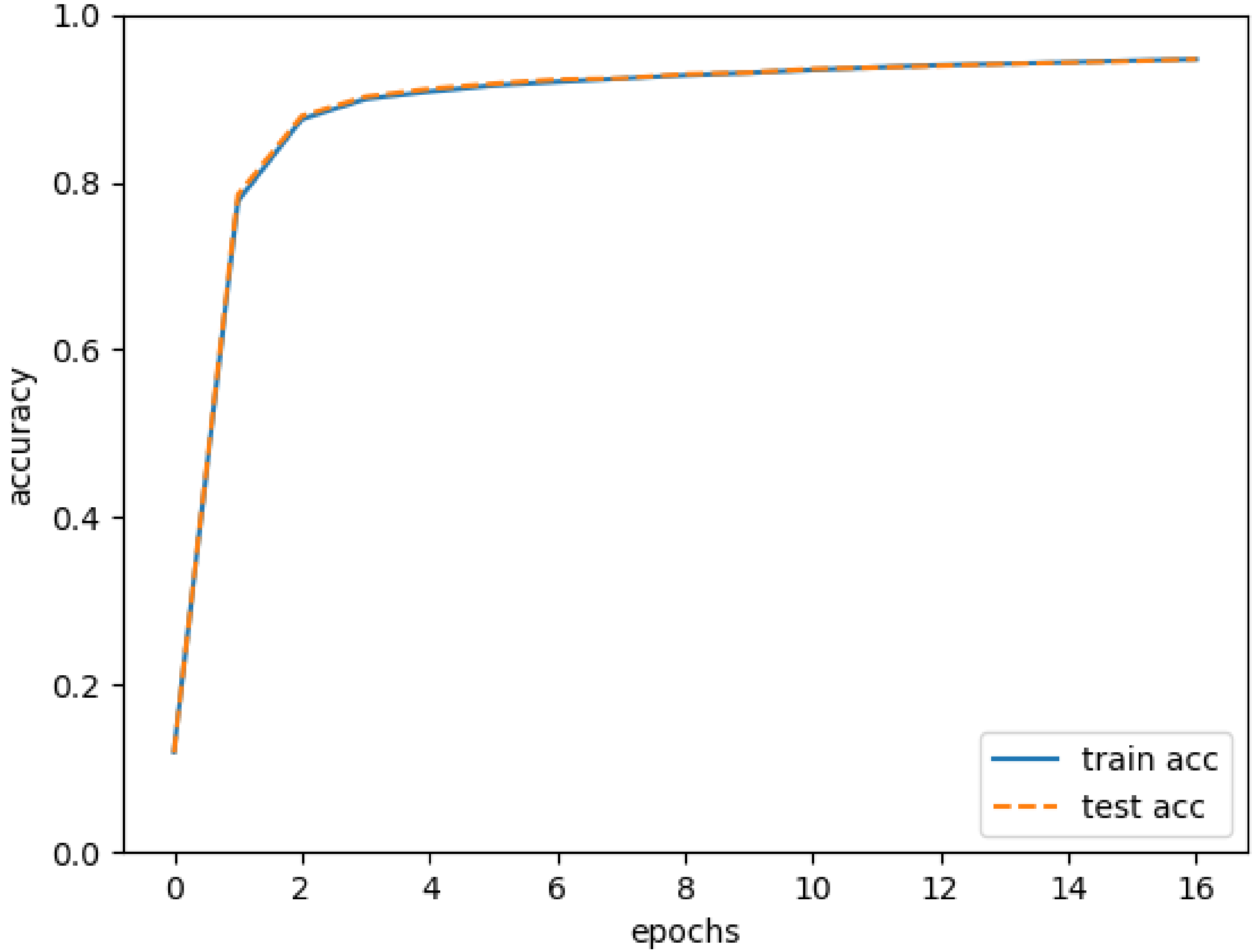
- 확률적 경사 하강법(Stochastic Gradient Descent, SGD)
  - 데이터를 미니배치로 무작위로 선정하여 경사 하강법으로 매개변수를 갱신하는 방법.

- 신경망 학습 절차
  - 전제 – 학습
    - 가중치와 편향을 훈련 데이터에 적응하도록 조정하는 과정.
  - 1단계 – 미니배치
    - 훈련 데이터 중 일부를 무작위로 가져오는 미니배치를 통해 손실 함수 값을 줄이는 것이 목표.
  - 2단계 – 기울기 산출
    - 미니배치의 손실 함수 값을 줄이기 위해 각 가중치 매개변수의 기울기를 구함.
    - 기울기는 손실 함수의 값을 가장 작게 하는 방향을 제시함.
  - 3단계 – 매개변수 갱신
    - 가중치 매개변수를 기울기 방향으로 아주 조금 갱신함.
  - 4단계 – 반복
    - 1 ~ 3단계를 반복함.

손실 함수 값의 추이: 왼쪽은 10,000회, 오른쪽은 1,000회 반복까지의 추이



훈련 데이터와 시험 데이터에 대한 정확도 추이



오차역전파법 (backpropagation)

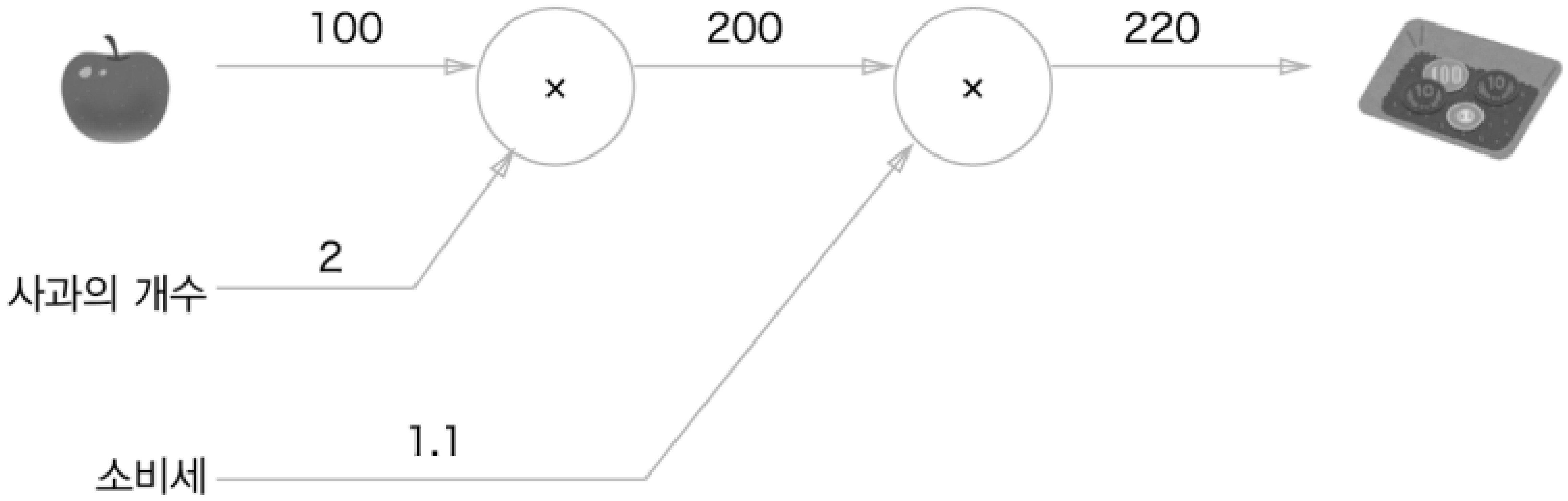
계산 그래프

- 문제1 : 현빈군은 슈퍼에서 1개에 100원인 사과를 2개 샀습니다. 이때 지불 금액을 구하세요.
  단, 소비세가 10% 부과됩니다.

계산 그래프

- 문제1 : 현빈군은 슈퍼에서 1개에 100원인 사과를 2개 샀습니다. 이때 지불 금액을 구하세요.
  단, 소비세가 10% 부과됩니다.

계산 그래프

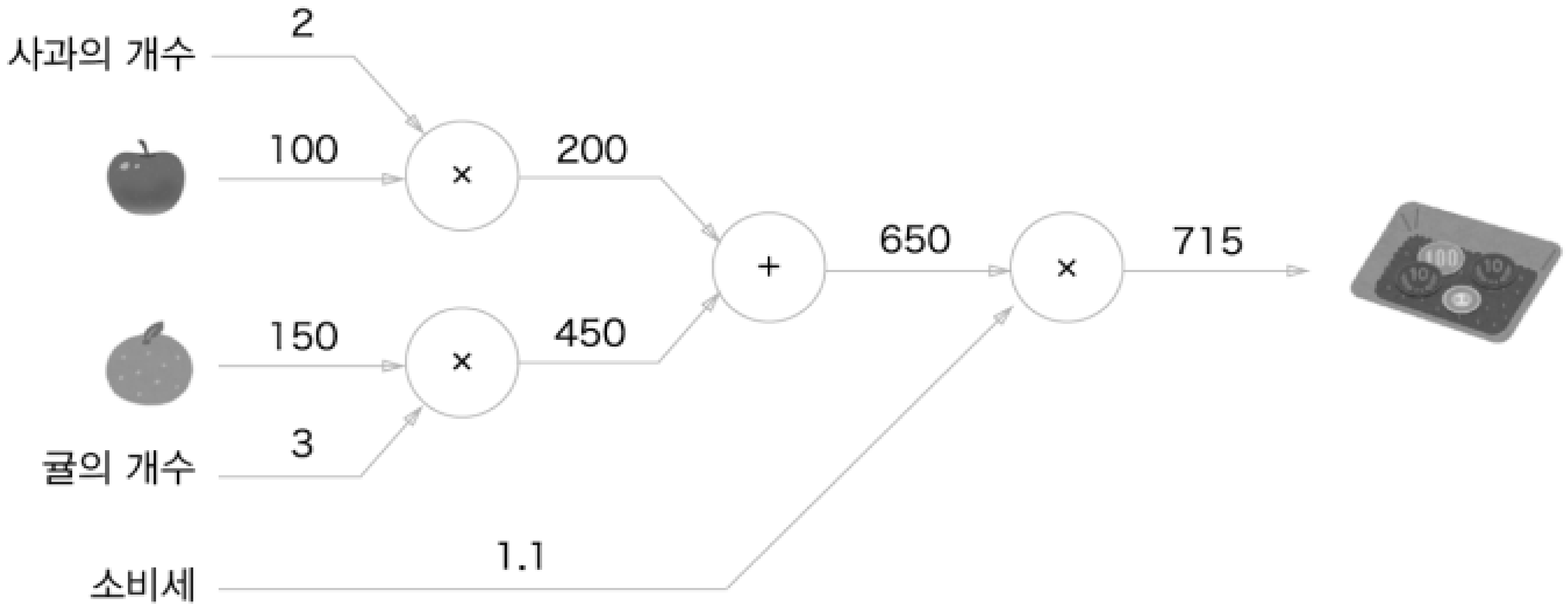


계산 그래프

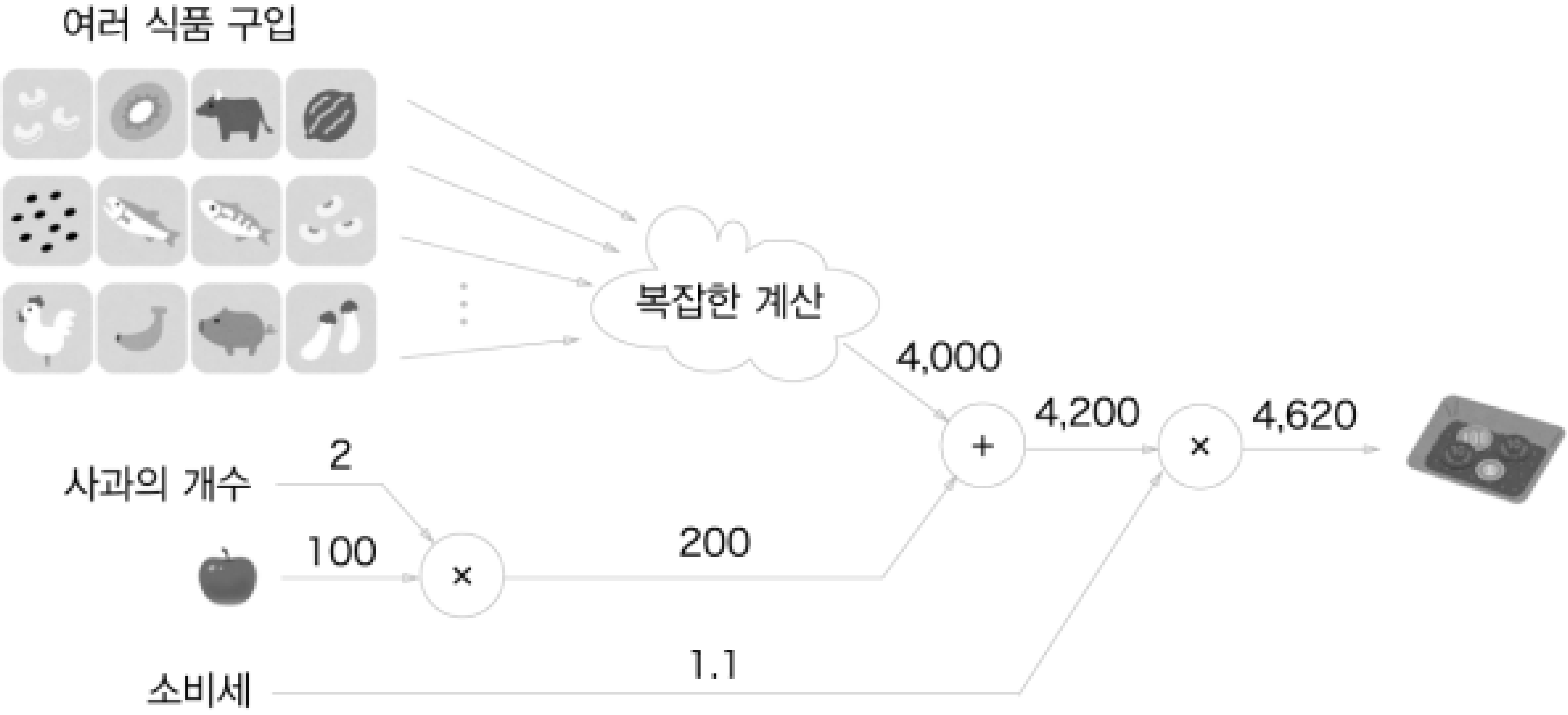
- 문제2 : 현빈군은 슈퍼에서 사과를 2개, 귤을 3개 샀습니다. 사과는 1개에 100원, 귤은 1개 150원 입니다. 소비세가 10%일 때 지불 금액을 구하세요.

계산 그래프

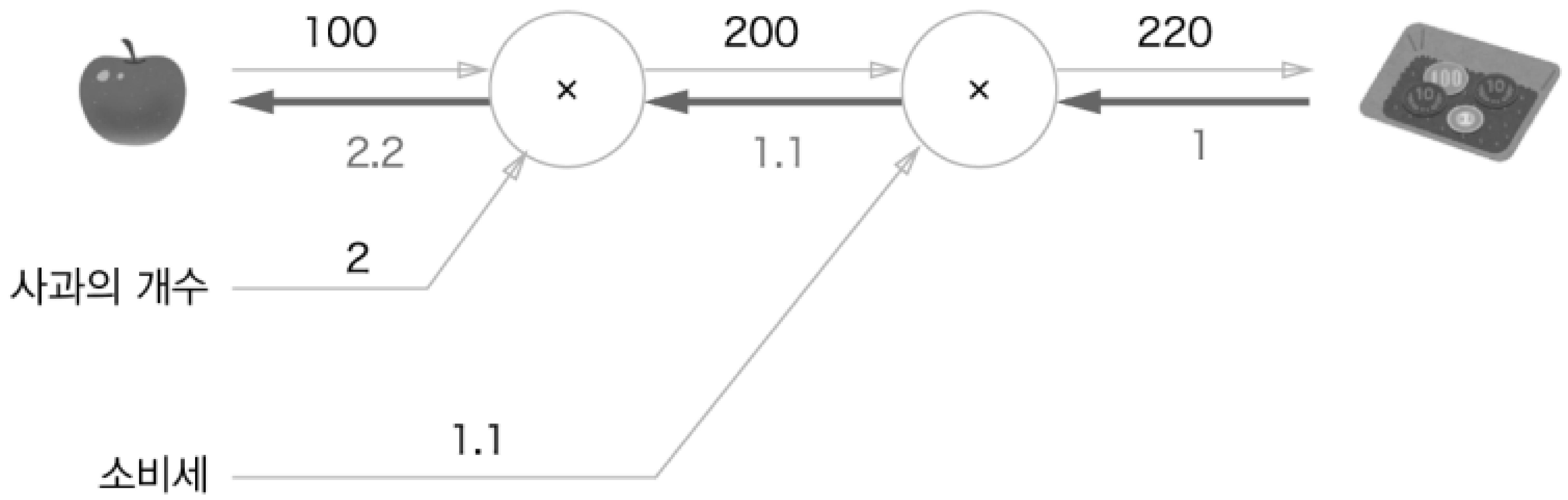
- 문제2 : 현빈군은 슈퍼에서 사과를 2개, 귤을 3개 샀습니다. 사과는 1개에 100원, 귤은 1개 150원 입니다. 소비세가 10%일 때 지불 금액을 구하세요.



국소적 계산

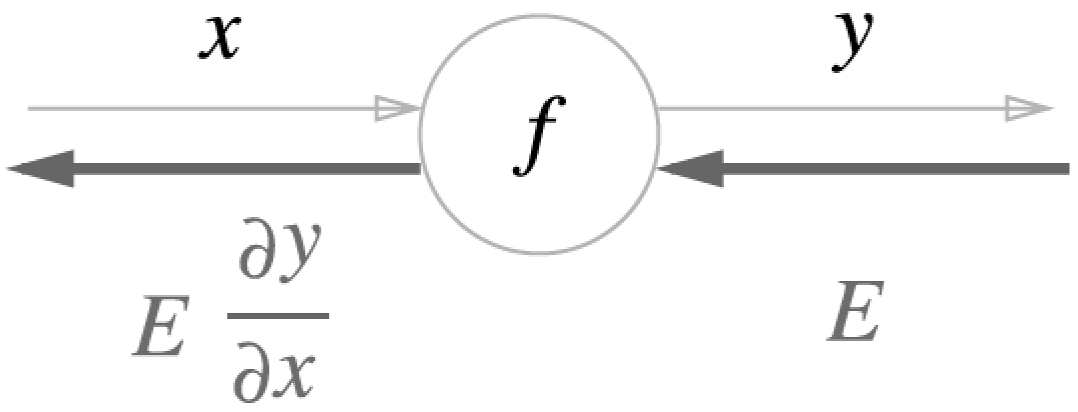


역전파에 의한 미분 값의 전달



계산 그래프의 역전파

$$y = f(x)$$



- 역전파의 계산 절차
  - 신호 E에, 노드의 국소적 미분($\frac{\partial y}{\partial x}$)을 곱한 후 다음 노드로 전달 수행.
  - 국소적 미분이란 : 순전파 때의 y = f(x) 계산의 미분을 구한다는 것이며, 이는 x에 대한 y의 미분 ($\frac{\partial y}{\partial x}$)을 구한다는 뜻.

연쇄법칙이란?

- 합성 함수의 미분은 합성 함수를 구성하는 각 함수의 미분의 곱으로 나타낼 수 있다.
- 합성 함수 : 여러 함수로 구성된 함수.

$$z = (x + y)^2$$

$$\downarrow$$

$$\left\{ \begin{array}{c} z = t^2 \\ t = x + y \end{array} \right\}$$

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

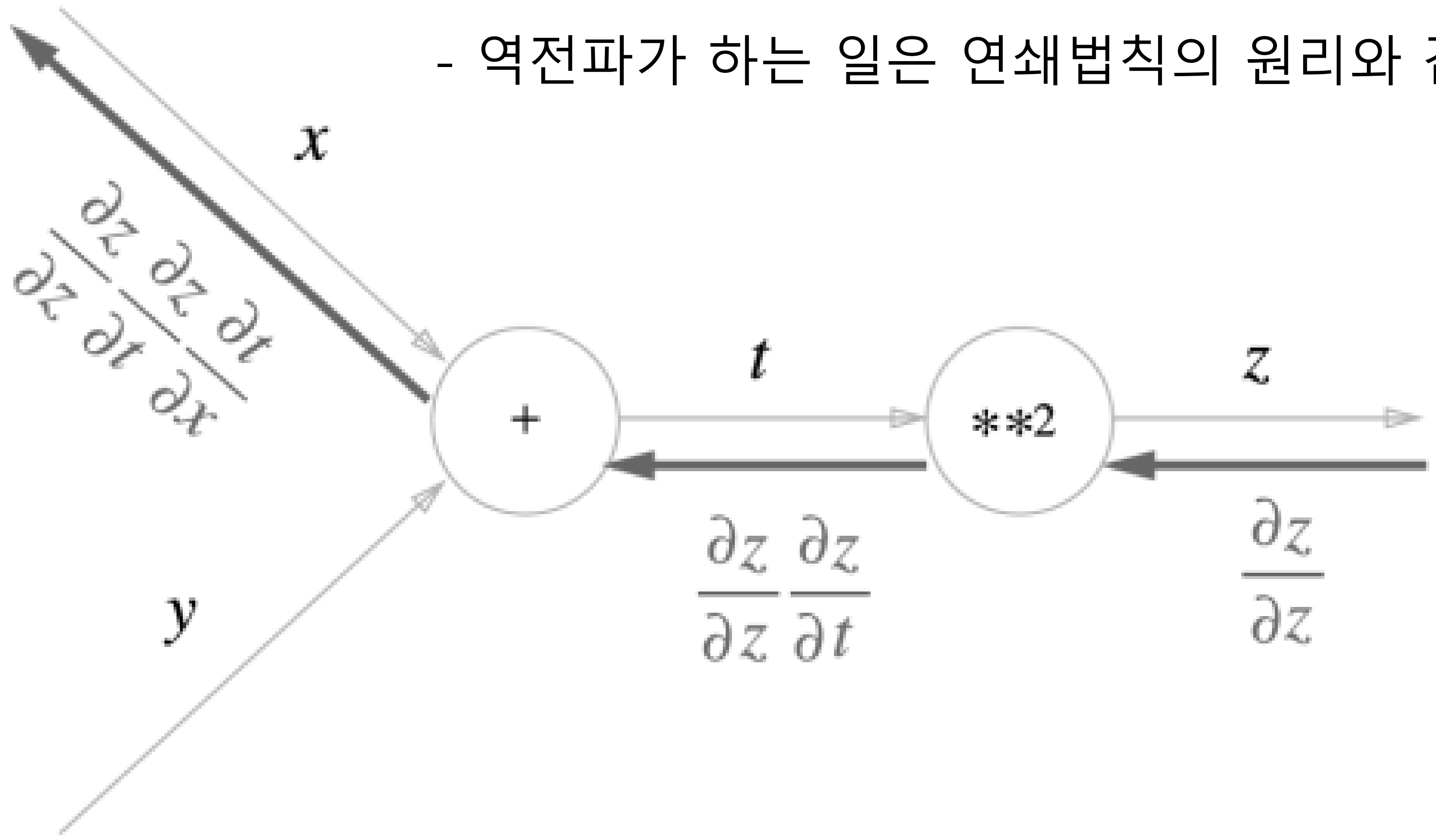
$$\frac{\partial z}{\partial t} = 2t$$

$$\frac{\partial t}{\partial x} = 1$$

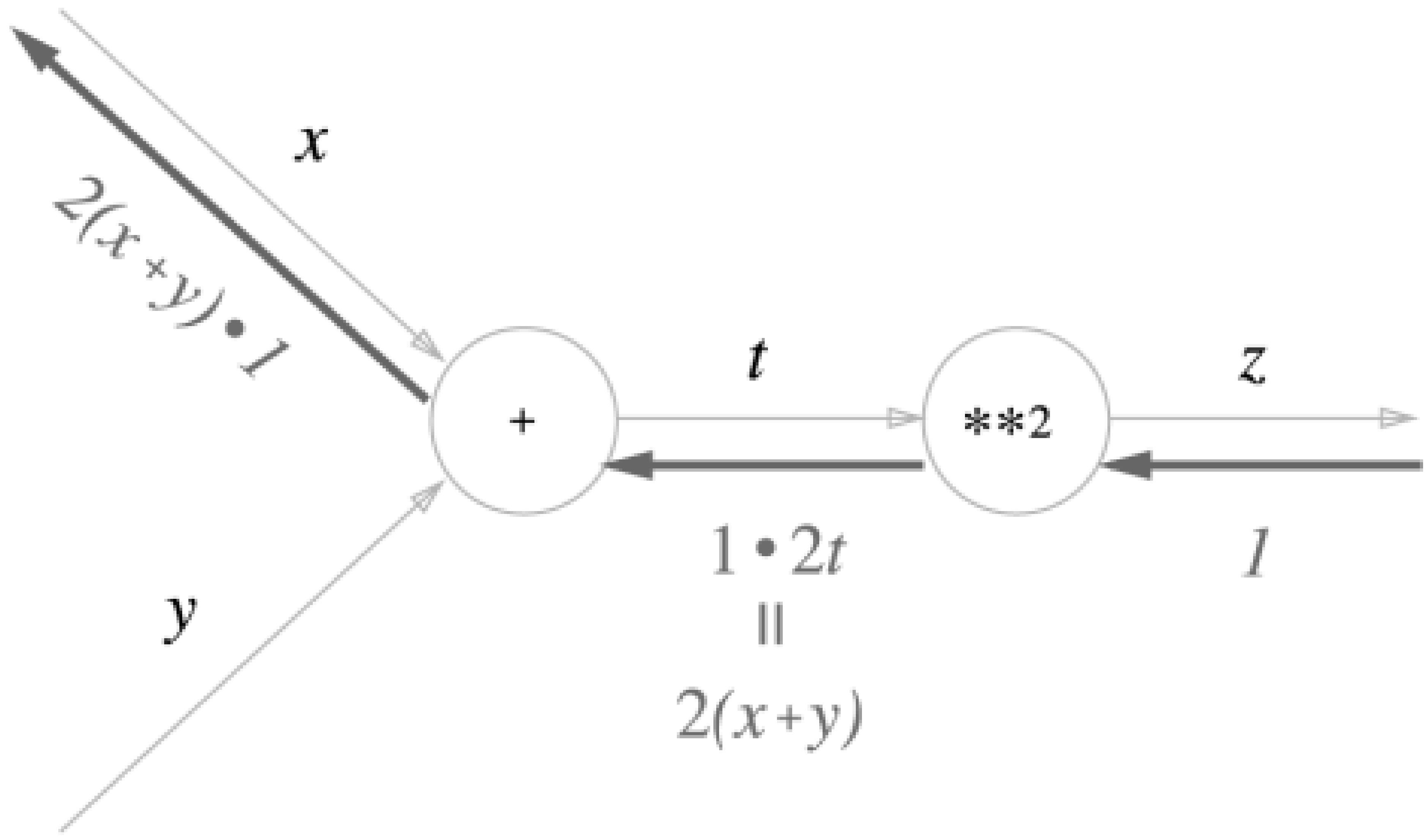
$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial t}\frac{\partial t}{\partial x} = 2t \cdot 1 = 2(x + y)$$

연쇄법칙과 계산 그래프

- 역전파가 하는 일은 연쇄법칙의 원리와 같다.

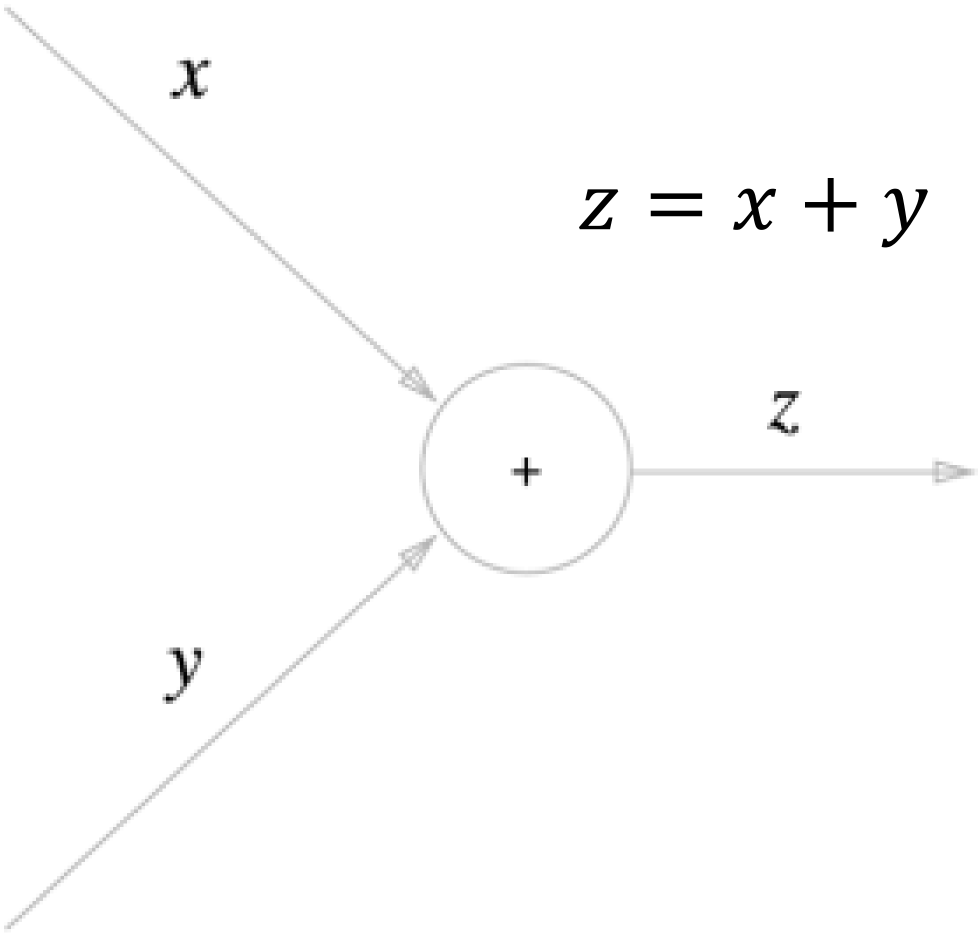


연쇄법칙과 계산 그래프



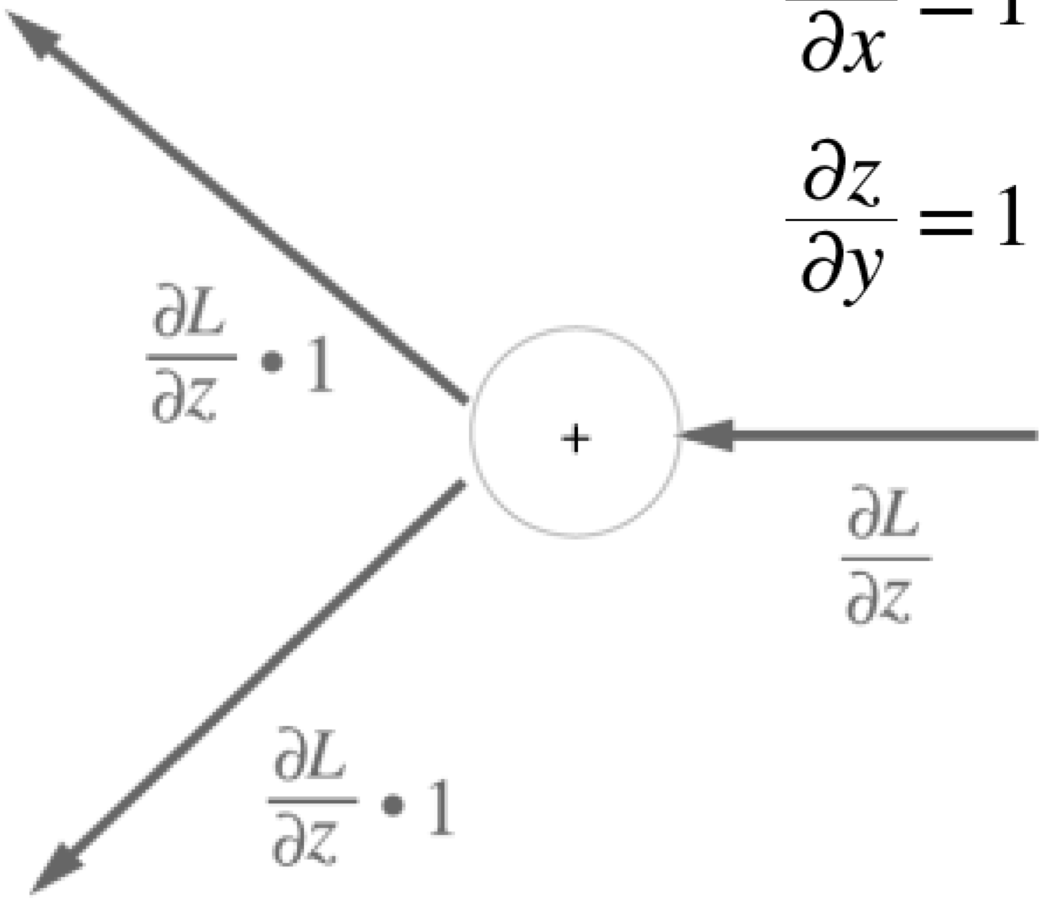
덧셈 노드의 역전파

$$z = x + y$$

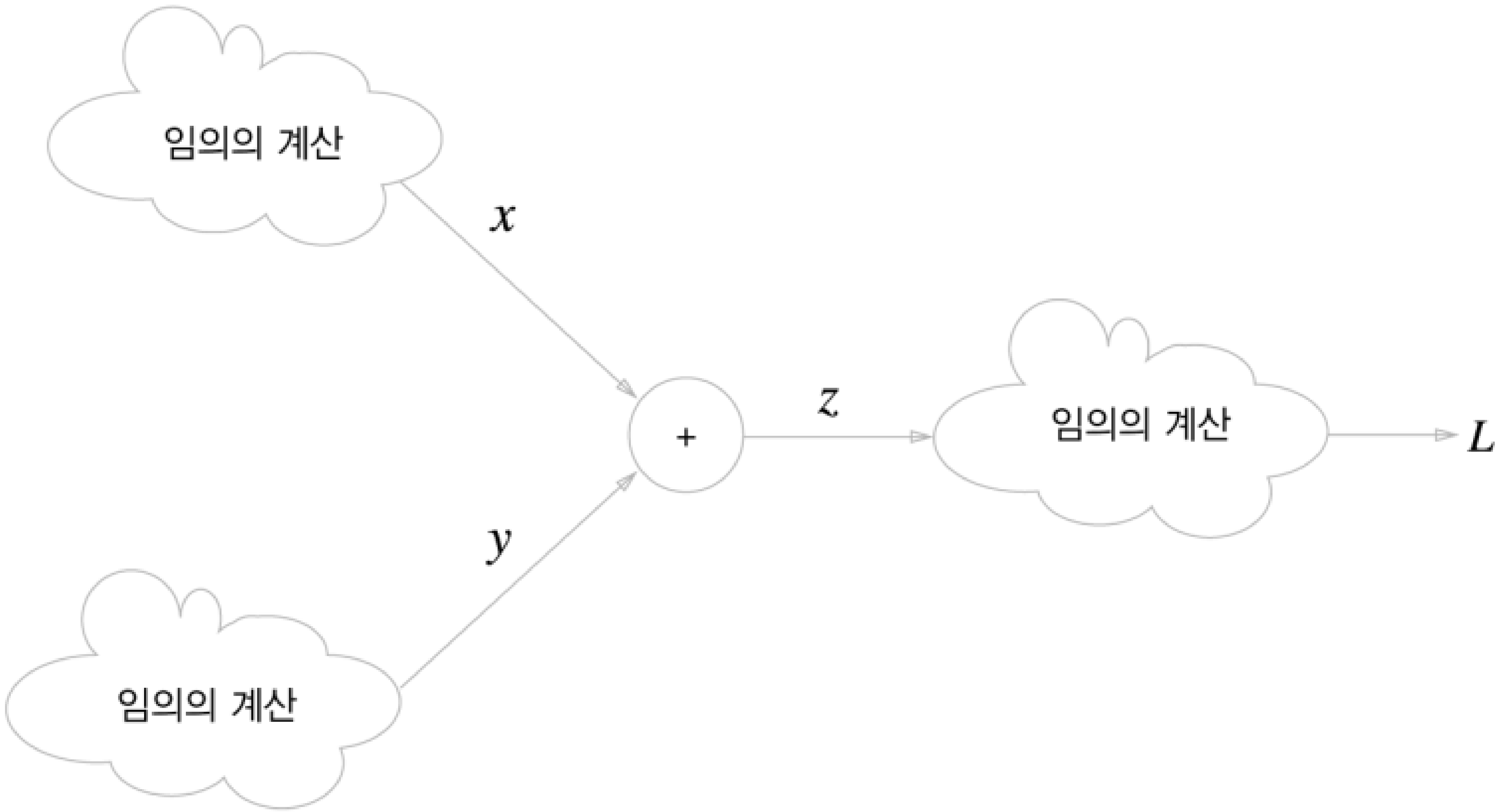


$$\frac{\partial z}{\partial x} = 1$$

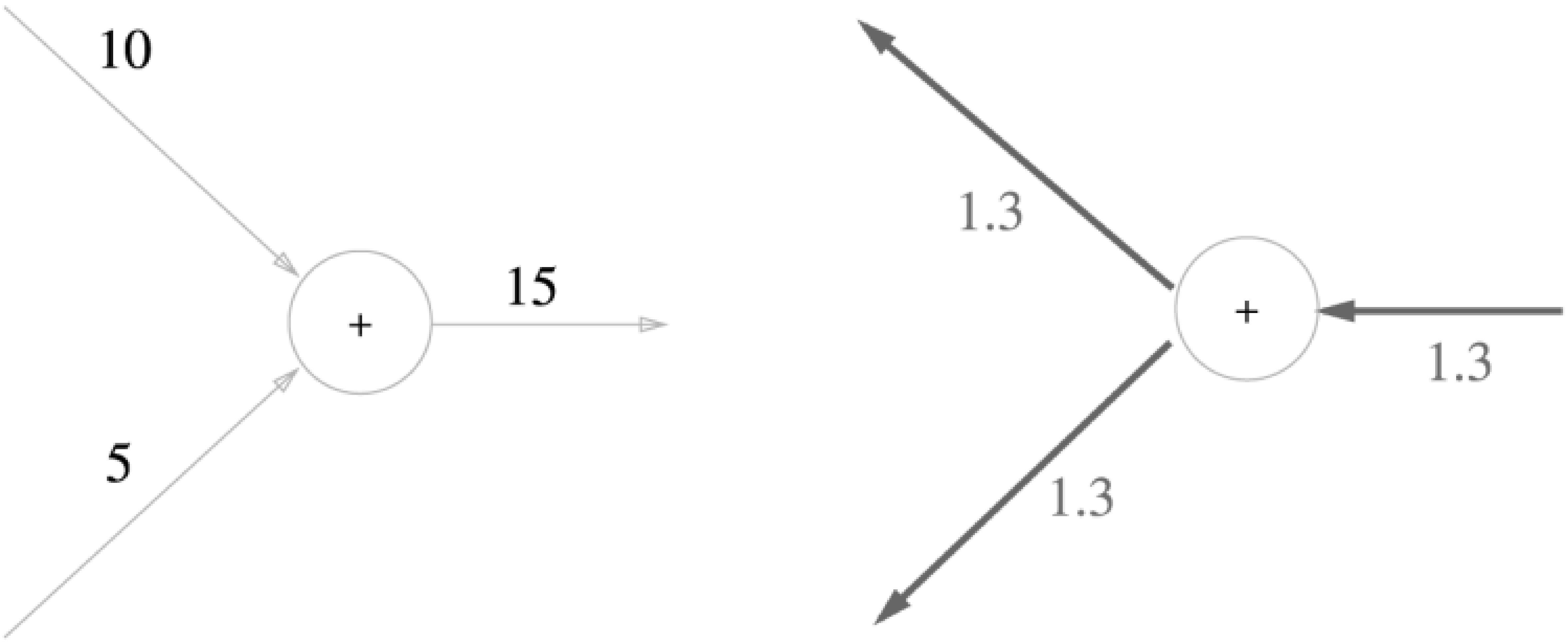
$$\frac{\partial z}{\partial y} = 1$$



덧셈 노드의 역전파

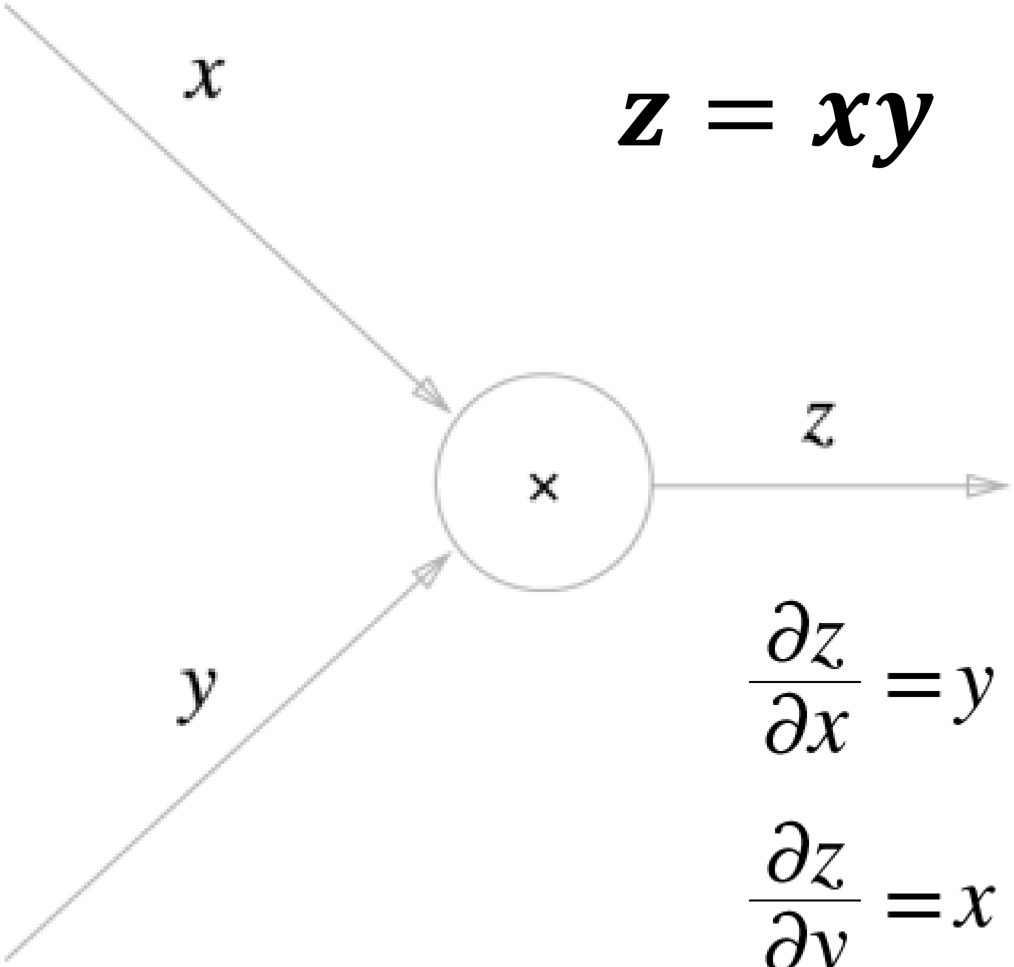


덧셈 노드의 역전파의 예



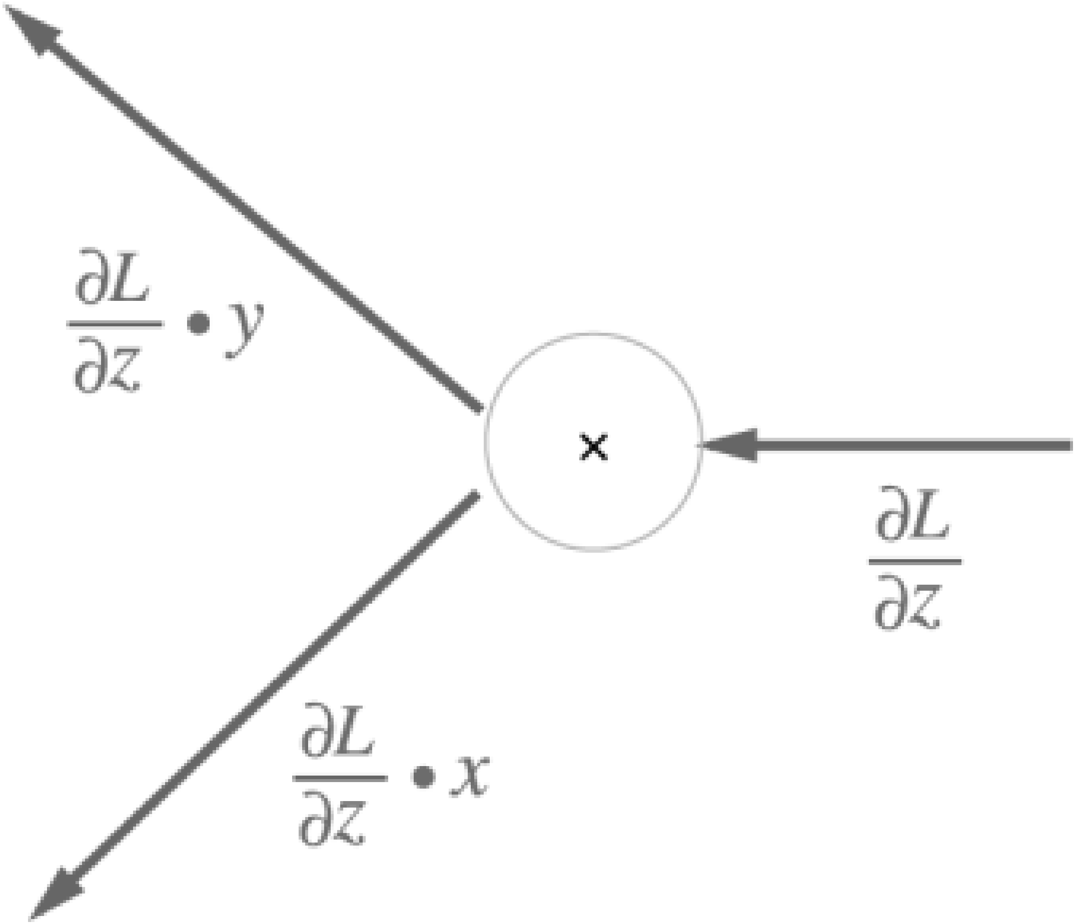
곱셈 노드의 역전파

$$z = xy$$

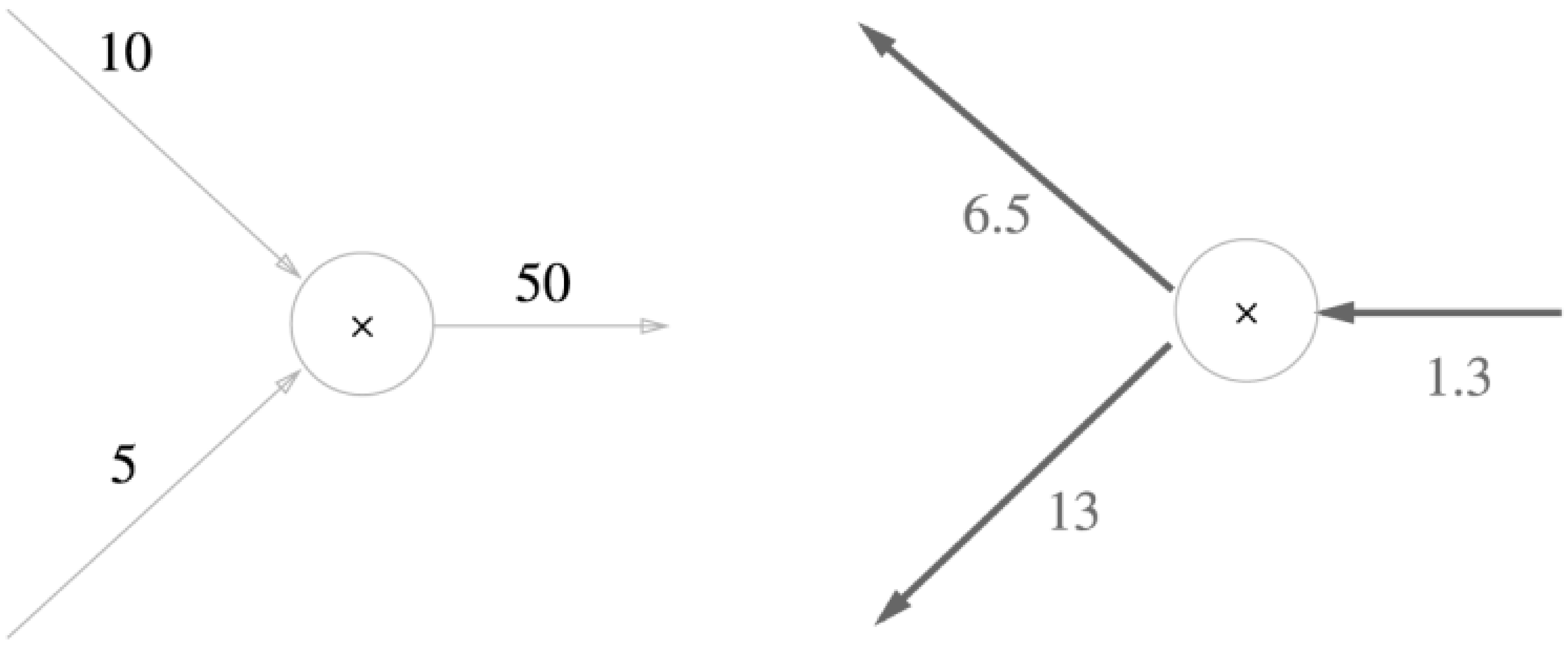


$$\frac{\partial z}{\partial x} = y$$

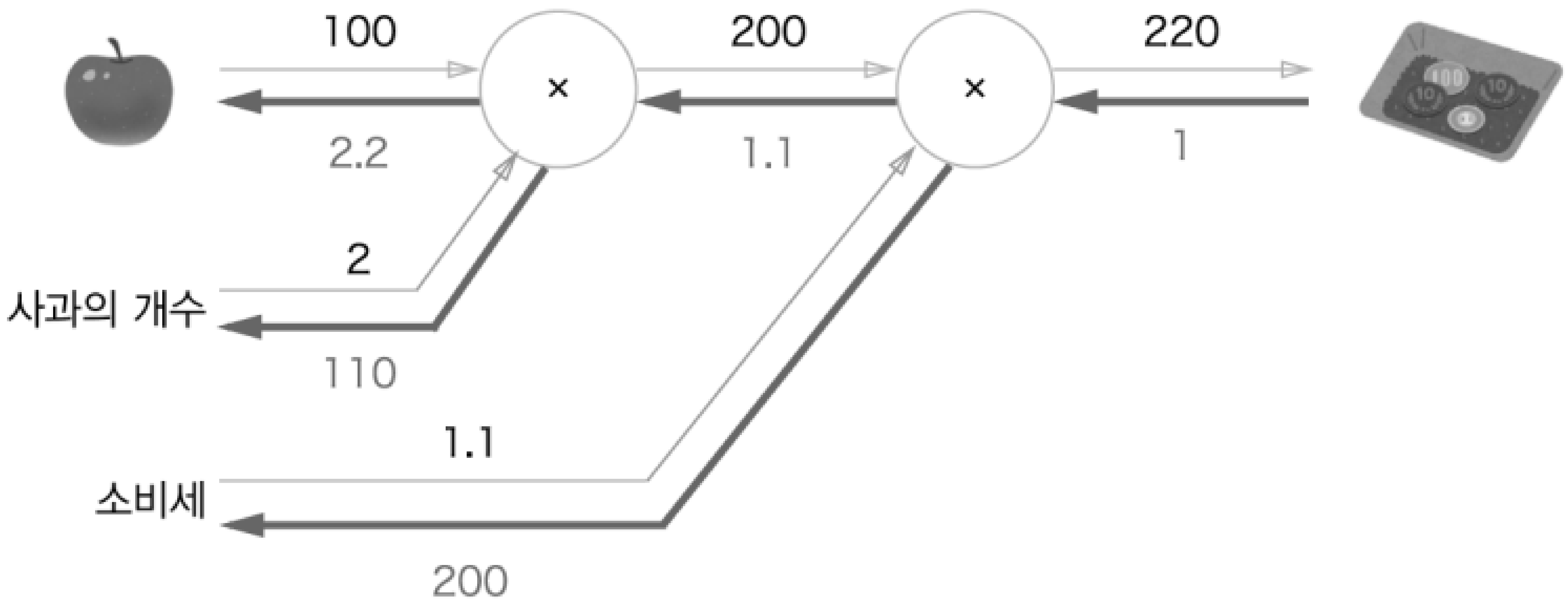
$$\frac{\partial z}{\partial y} = x$$



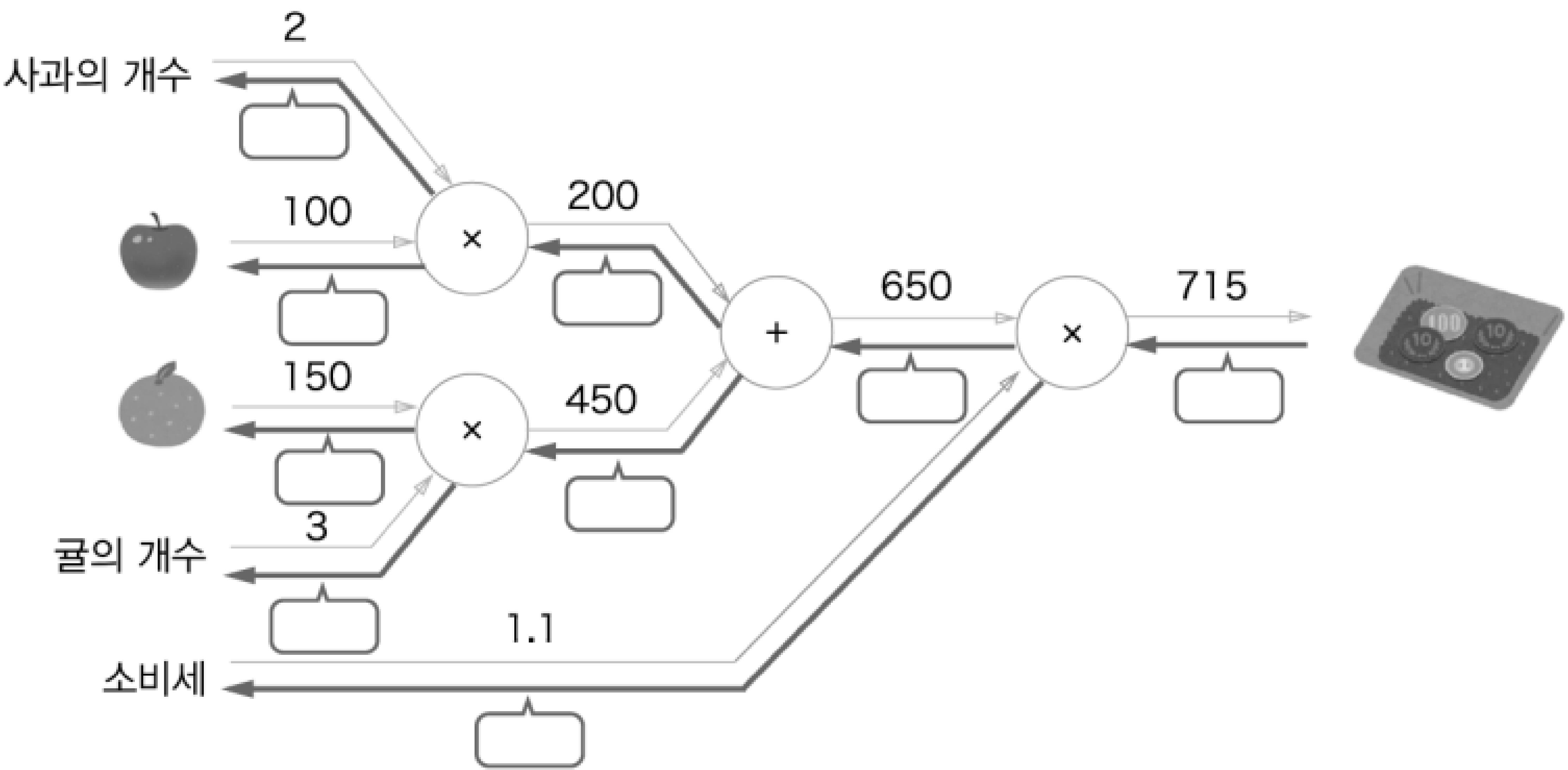
곱셈 노드의 역전파의 예



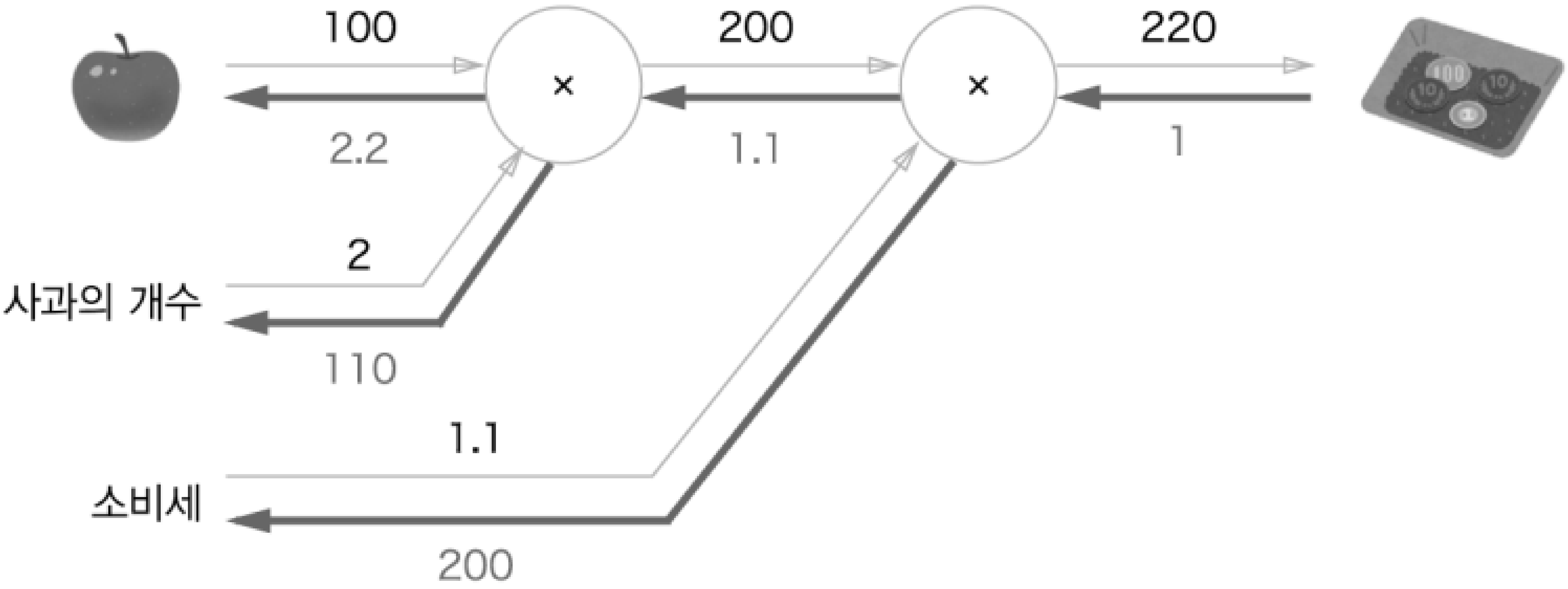
사과 쇼핑 역전파의 예



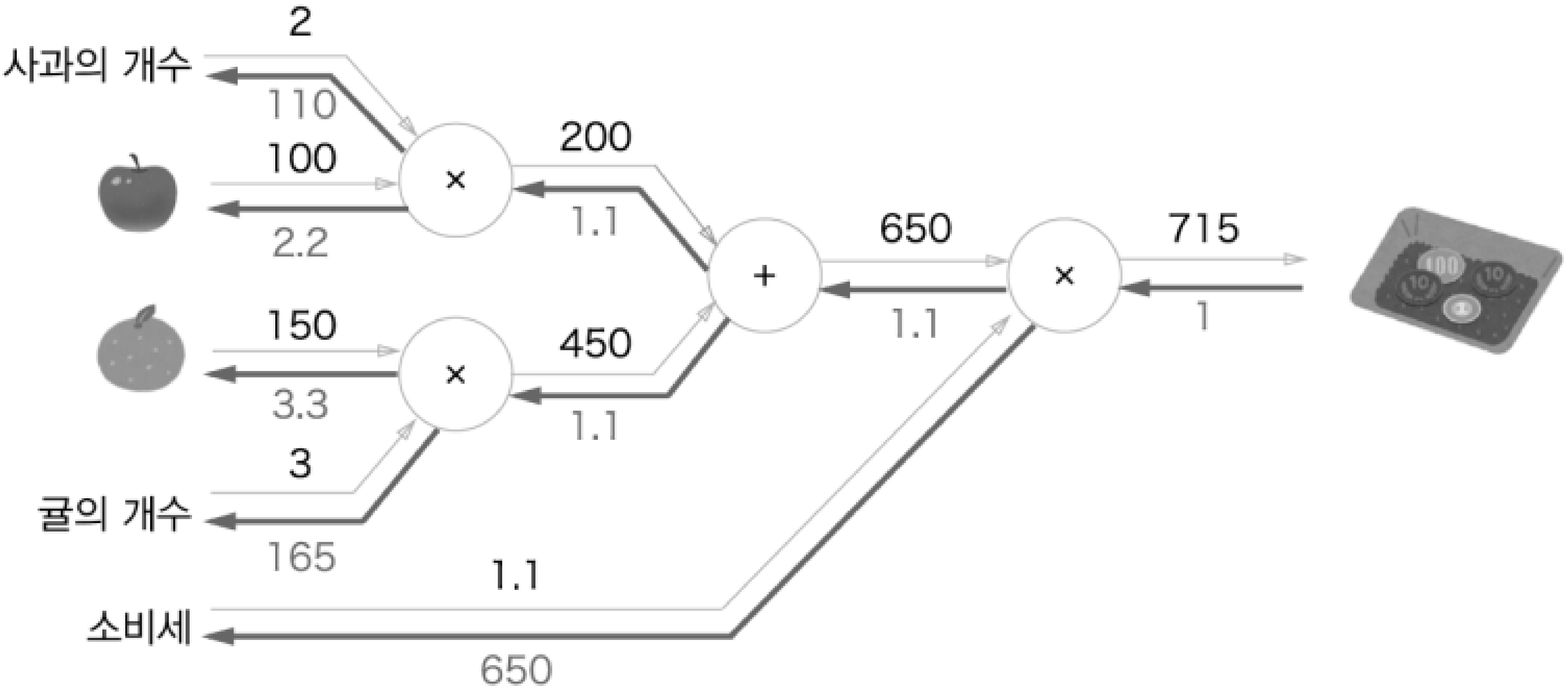
사과와 귤 쇼핑 역전파의 예



곱셈 계층 – 사과 2개 구입



덧셈 계층 – 사과 2개와 귤 3개 구입



활성화 함수 계층 구현하기 – ReLU 계층

- 계산 그래프를 신경망에 적용.

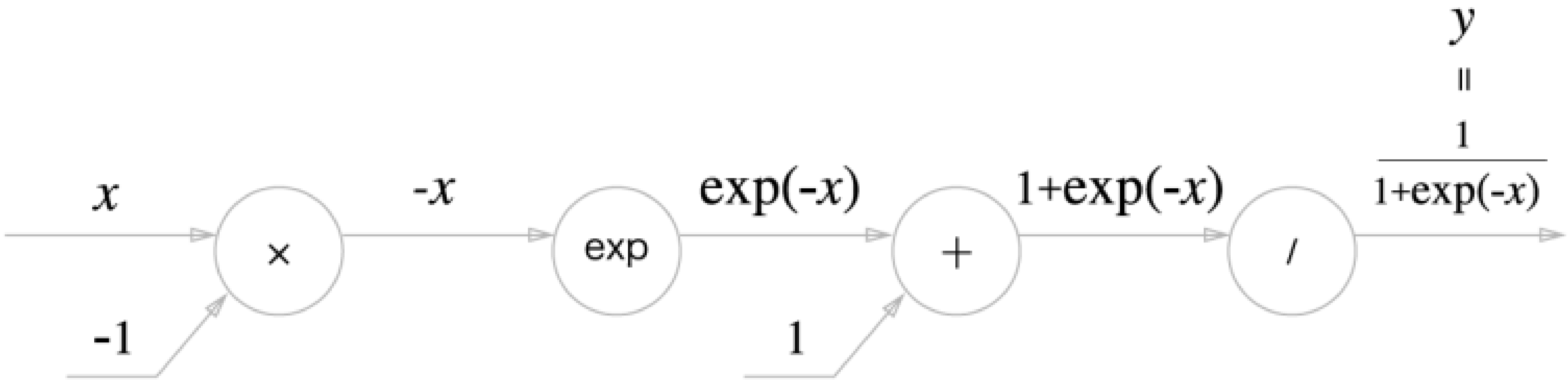
$$y = \begin{cases} x & (x > 0) \\ 0 & (x \le 0) \end{cases}$$

$$\frac{\partial y}{\partial x} = \begin{cases} 1 & (x > 0) \\ 0 & (x \le 0) \end{cases}$$

ReLU 계층의 계산 그래프

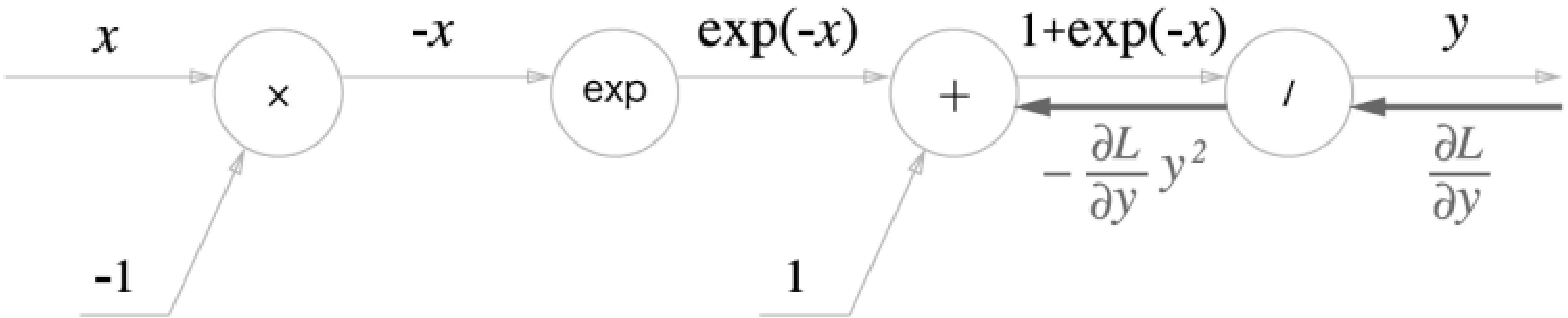
활성화 함수 계층 구현하기 – Sigmoid 계층

$$y = \frac{1}{1 + \exp(-x)}$$ **: 시그모이드 함수**



Sigmoid 계층의 계산 그래프(순전파)

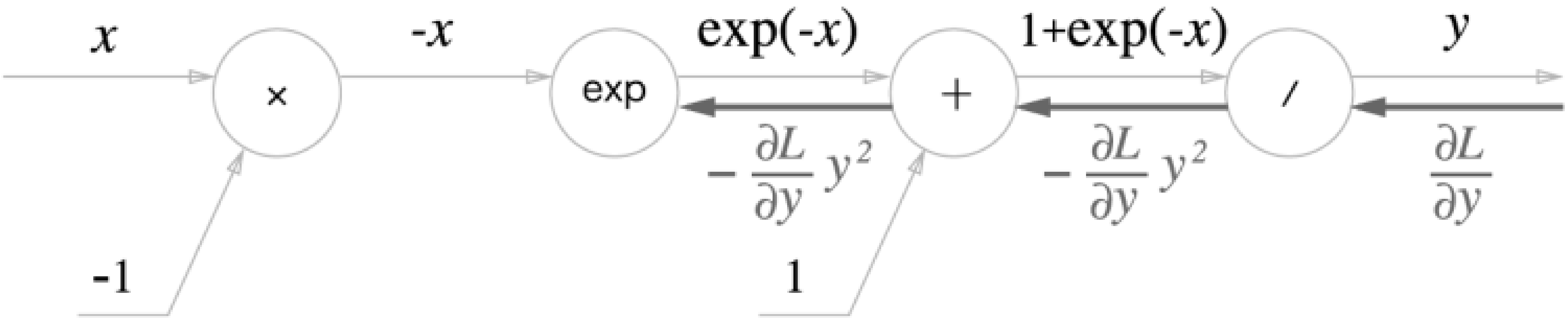
Sigmoid 계층의 계산 그래프(역전파) - 1단계



$$y = \frac{1}{t} \xrightarrow{\text{미분}} \frac{\partial y}{\partial t} = -\frac{1}{t^2}$$

(단, $t = 1 + \exp(-x)$) $$= -y^2$$

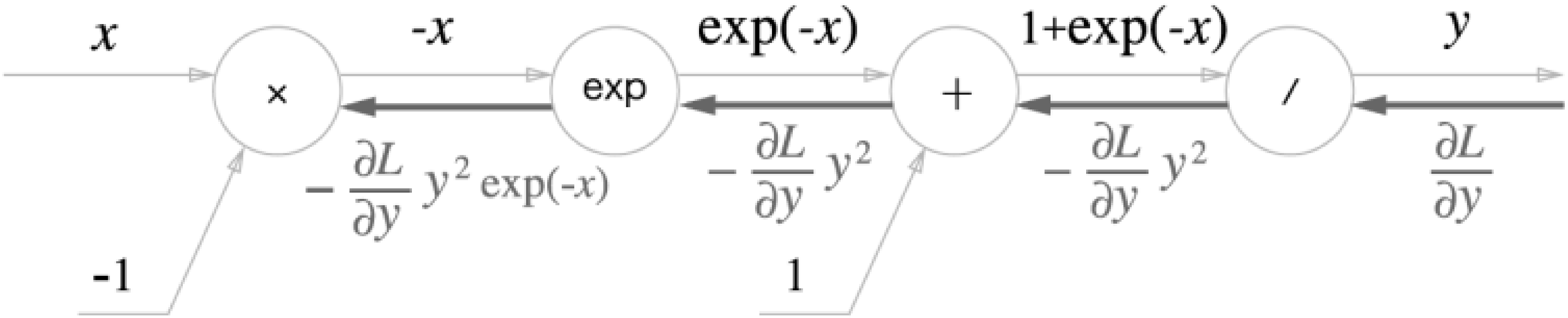
Sigmoid 계층의 계산 그래프(역전파) - 2단계



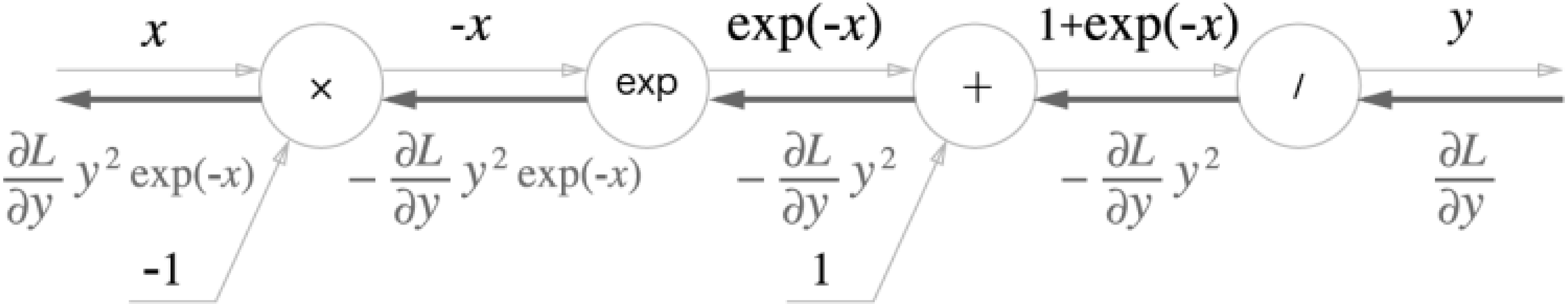
'+' 노드는 상류의 값을 여과 없이 하류로 내보냄.

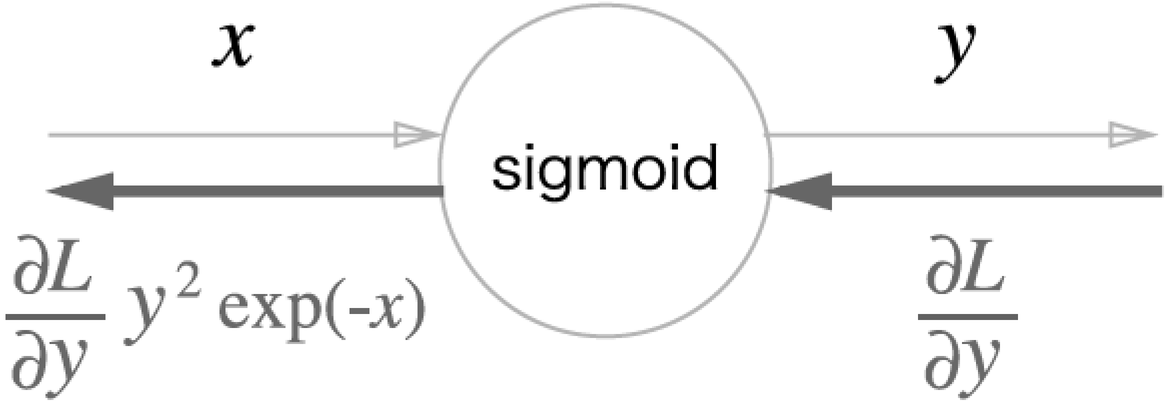
Sigmoid 계층의 계산 그래프(역전파) - 3단계

$$y = \exp(x) \xrightarrow{\text{미분}} \frac{\partial y}{\partial x} = \exp(x)$$



Sigmoid 계층의 계산 그래프(역전파) - 4단계

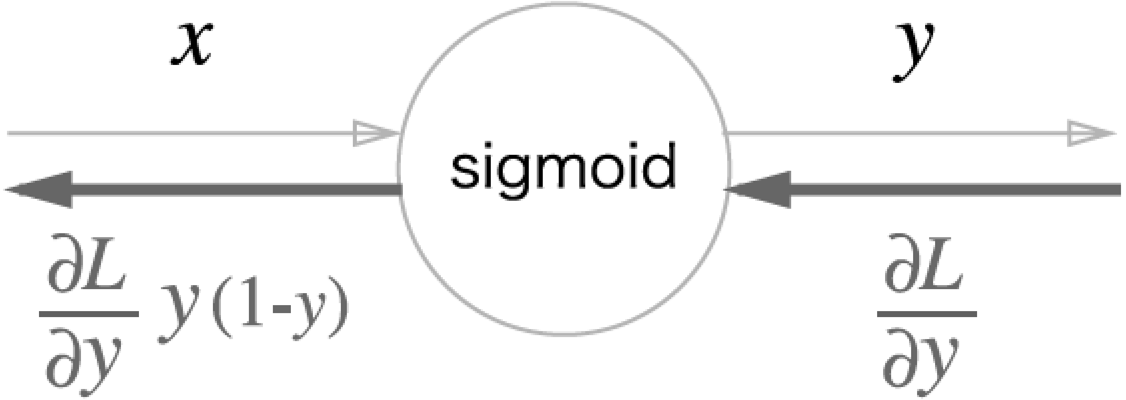




Sigmoid 계층의
계산 그래프(간소화 버전)

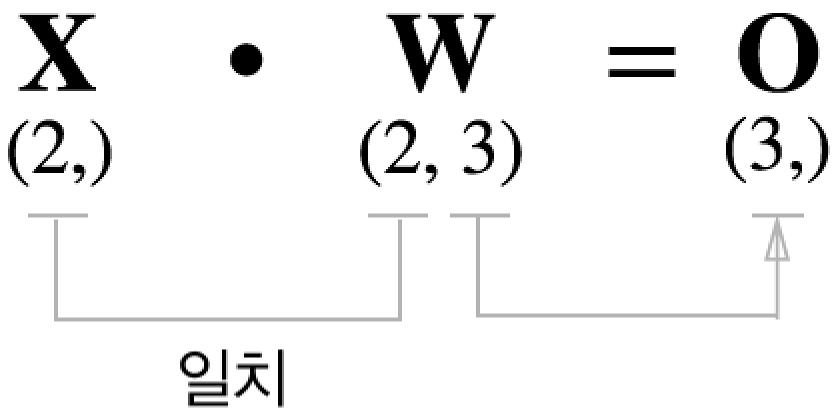
순전파의 출력 y만으로 역전파 계산

$$\begin{aligned}\frac{\partial L}{\partial y} y^2 \exp(-x) &= \frac{\partial L}{\partial y} \frac{1}{(1+\exp(-x))^2} \exp(-x) \\ &= \frac{\partial L}{\partial y} \frac{1}{1+\exp(-x)} \frac{\exp(-x)}{1+\exp(-x)} \\ &= \frac{\partial L}{\partial y} y(1-y)\end{aligned}$$



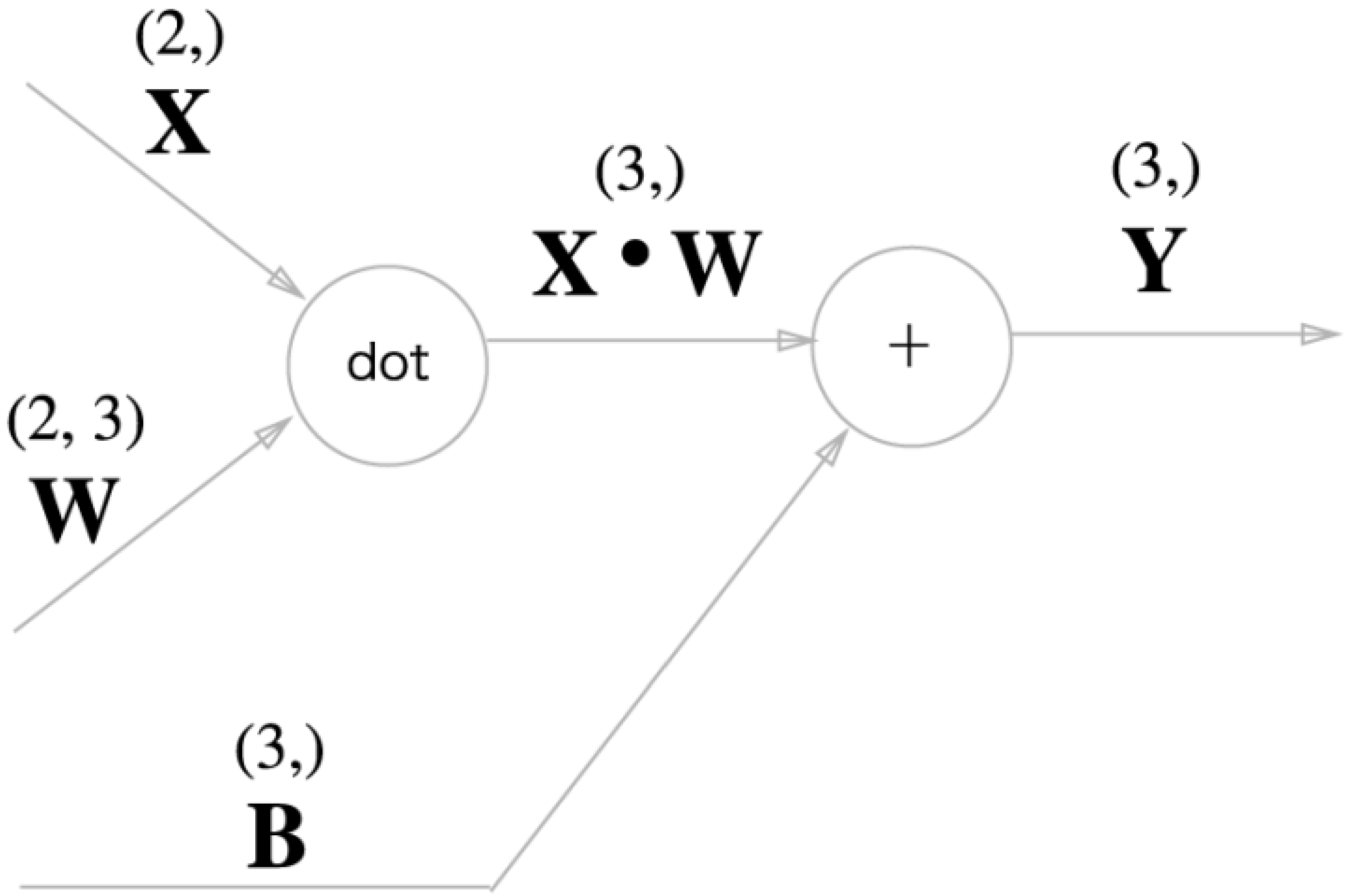
Affine 계층

- 행렬의 내적에서는 대응하는 차원의 원소 수 일치 시켜야 함.



- 어파인 변환(Affine transformation) : 신경망의 순전파 때 수행하는 행렬의 내적을 기하학에서 일컫는 말.
- Affine 계층 : affine 변환을 수행하는 처리.

Affine 계층의 계산 그래프 : 변수가 행렬(다차원 배열)임에 주의



Affine 계층의 역전파

$$\frac{\partial L}{\partial \mathbf{X}} = \frac{\partial L}{\partial \mathbf{Y}} \cdot \mathbf{W}^{\mathrm{T}}$$ (T : 전치행렬)

$$\mathrm{W} = \begin{pmatrix} w_{11} & w_{21} & w_{31} \\ w_{12} & w_{22} & w_{32} \end{pmatrix}$$

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

$$\mathrm{W}^{\mathrm{T}} = \begin{pmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \end{pmatrix}$$

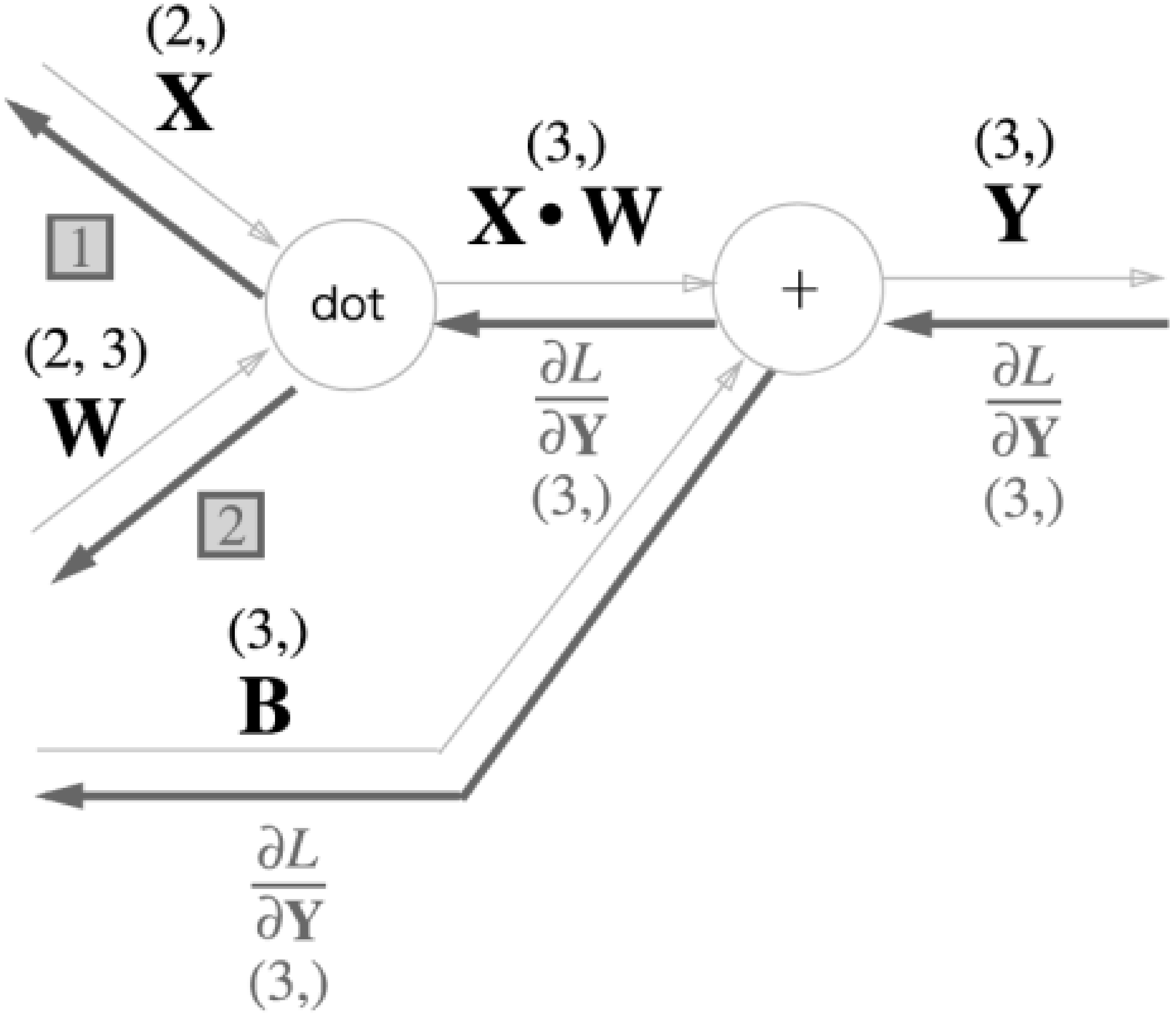
Affine 계층의 역전파

1 $\dfrac{\partial L}{\partial \mathbf{X}} = \dfrac{\partial L}{\partial \mathbf{Y}} \cdot \mathbf{W}^{\mathrm{T}}$

(2,) (3,) (3, 2)

2 $\dfrac{\partial L}{\partial \mathbf{W}} = \mathbf{X}^{\mathrm{T}} \cdot \dfrac{\partial L}{\partial \mathbf{Y}}$

(2, 3) (2, 1) (1, 3)

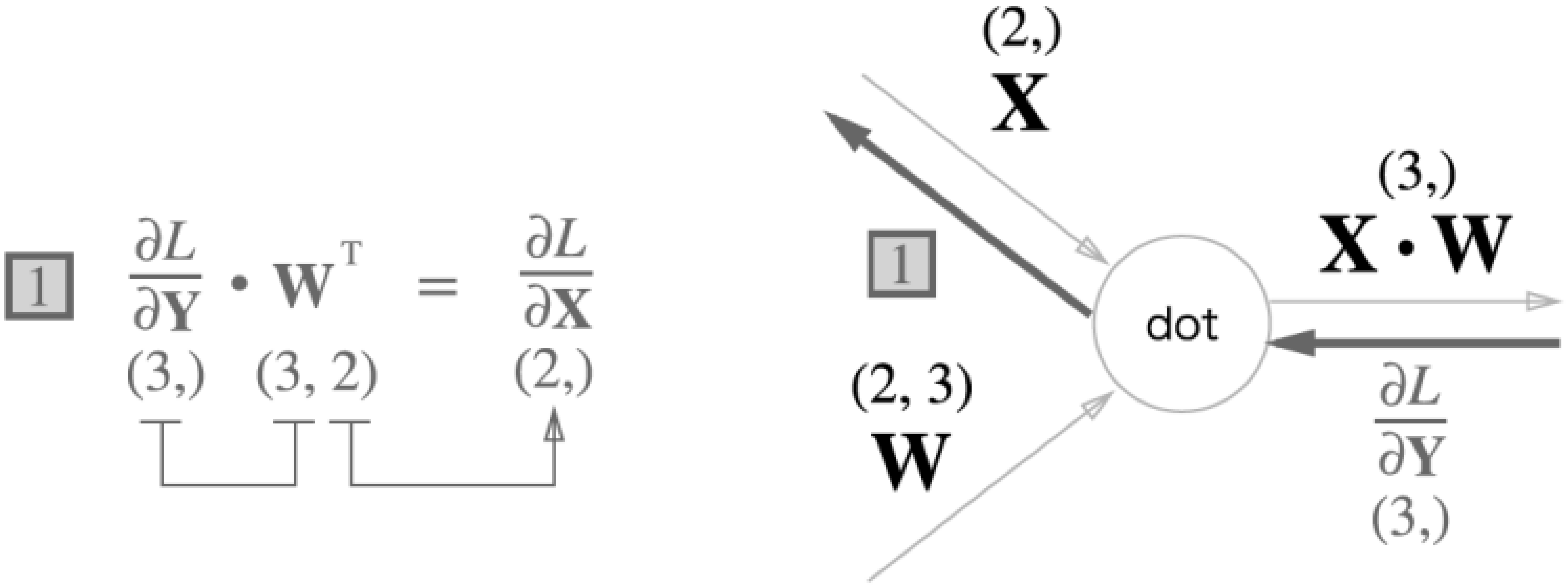


각 변수의 형상 주의

$$\mathbf{X} = (x_0, x_1, \cdots, x_n)$$

$$\frac{\partial L}{\partial \mathbf{X}} = \left(\frac{\partial L}{\partial x_0}, \frac{\partial L}{\partial x_1}, \cdots, \frac{\partial L}{\partial x_n} \right)$$

행렬 내적('dot' 노드)의 역전파



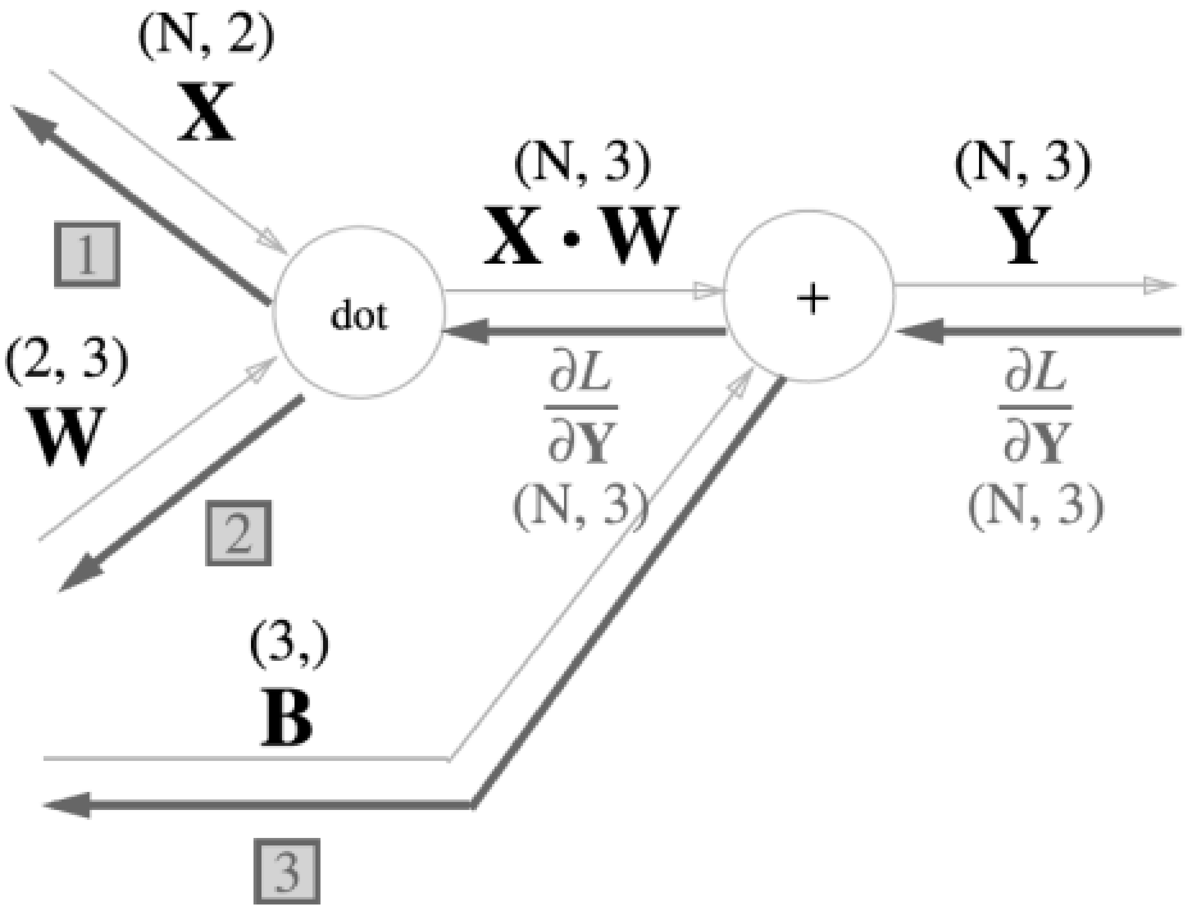
배치용 Affine 계층의 계산 그래프

1 $\dfrac{\partial L}{\partial \mathbf{X}} = \dfrac{\partial L}{\partial \mathbf{Y}} \cdot \mathbf{W}^{\mathrm{T}}$

(N, 2) (N, 3) (3, 2)

2 $\dfrac{\partial L}{\partial \mathbf{W}} = \mathbf{X}^{\mathrm{T}} \cdot \dfrac{\partial L}{\partial \mathbf{Y}}$

(2, 3) (2, N) (N, 3)

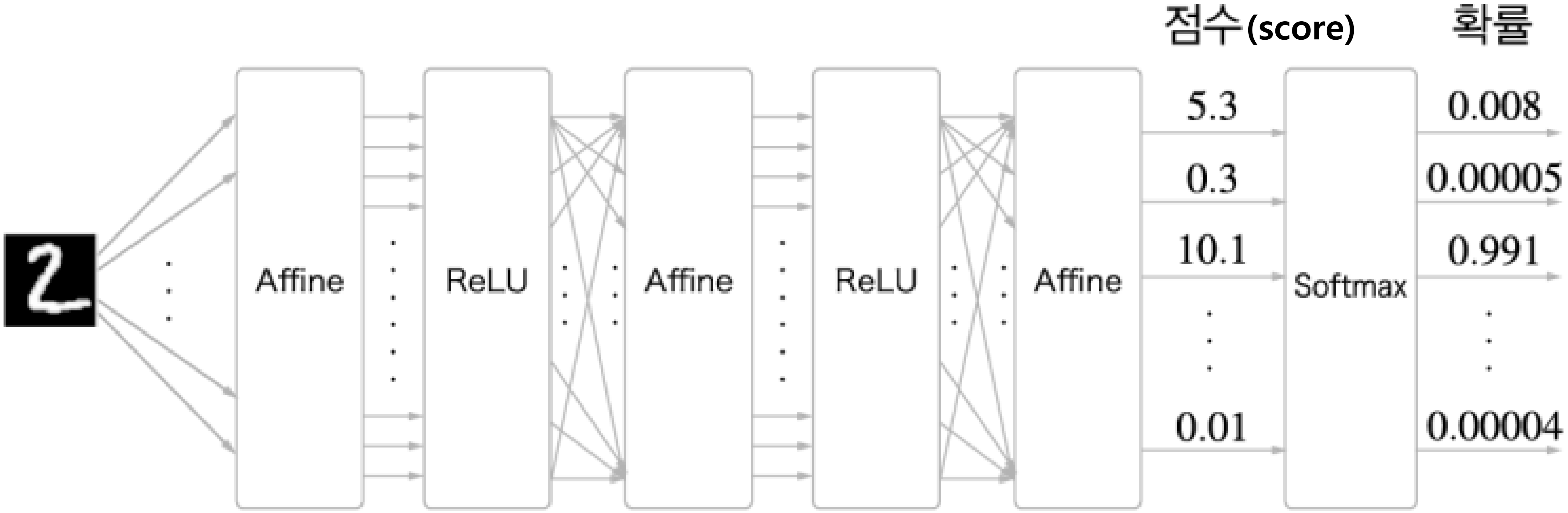


3 $\dfrac{\partial L}{\partial \mathbf{B}} = \dfrac{\partial L}{\partial \mathbf{Y}}$ 의 첫 번째 축(0축, 열방향)의 합

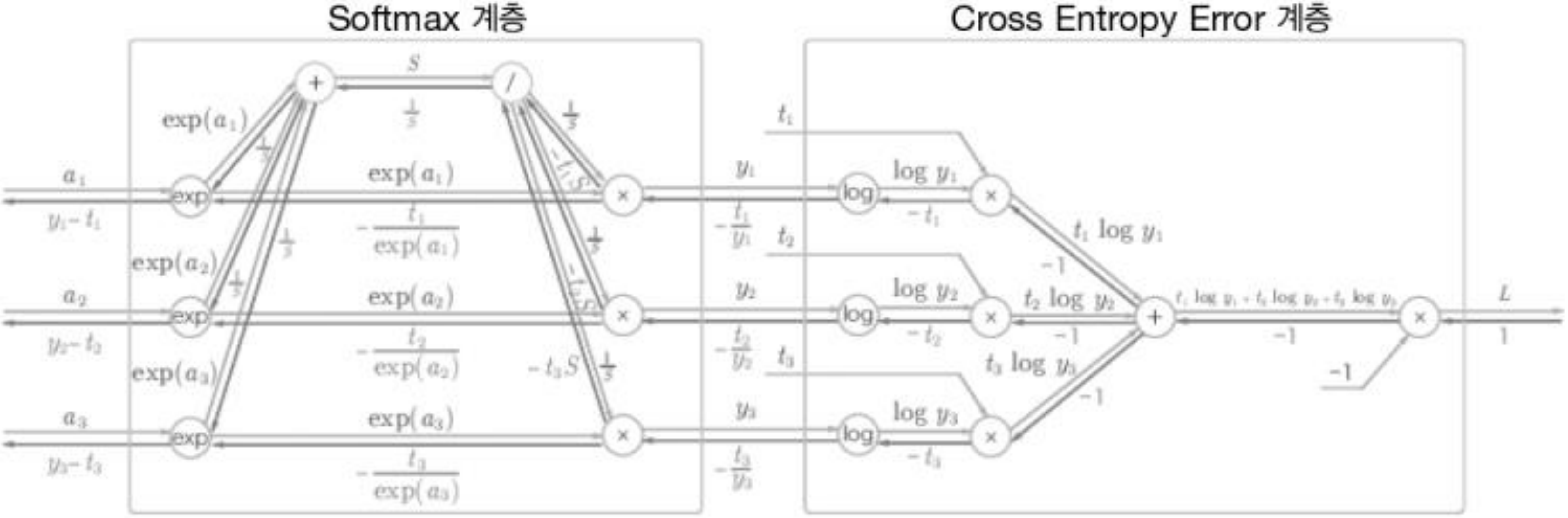
(3) (N, 3)

손글씨 숫자 인식에서의 softmax 계층의 출력

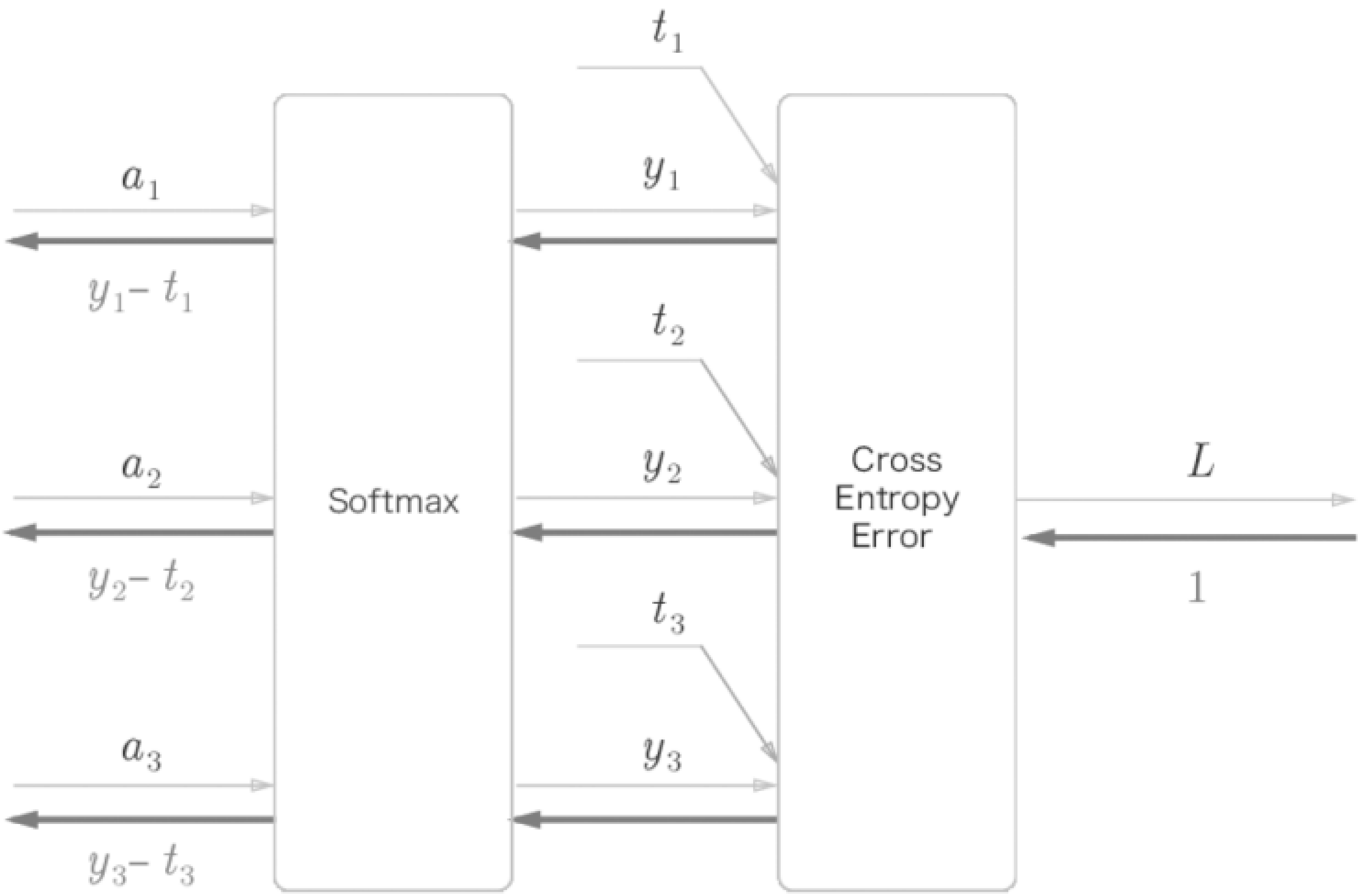
- 소프트맥스 함수 : 입력 값을 정규화하여 출력(출력의 합 : 1)
    - 추론할 때는 일반적으로 소프트맥스 계층을 사용하지 않음.



Softmax-with-Loss 계층의 계산 그래프



간소화한 Softmax-with-Loss 계층의 계산 그래프



학습 관련 기술들

매개변수 갱신

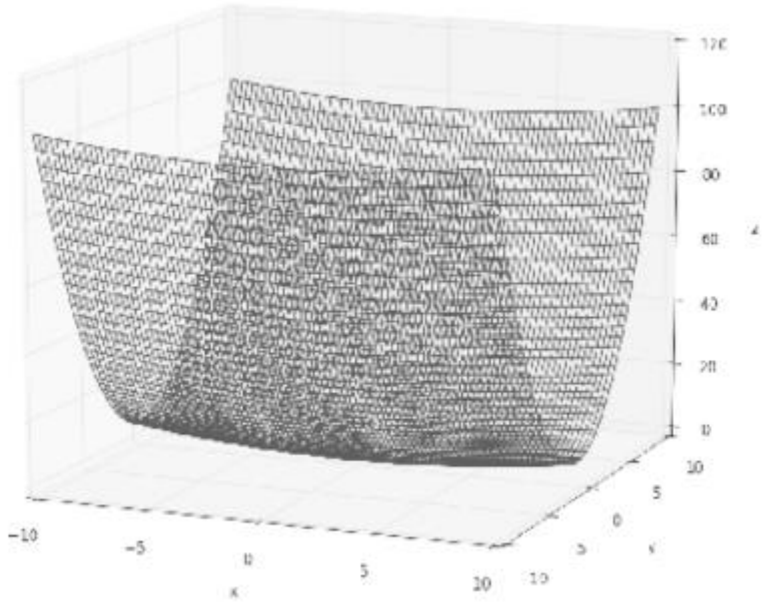
- 신경망 학습의 목적 : 손실 함수의 값을 가능한 한 낮추는 매개변수를 찾는 것.
- 확률적 경사 하강법(SGD : Stochastic Gradient descent)
  - 최적의 매개변수 값을 찾는 단서로 매개변수의 기울기(미분)를 이용했음.

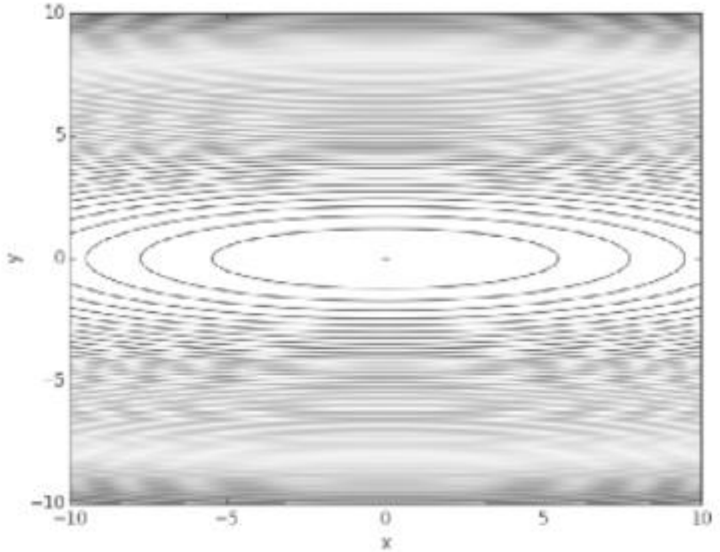
$$\mathbf{W} \leftarrow \mathbf{W} - \eta \frac{\partial L}{\partial \mathbf{W}}$$

- $\mathbf{W}$: 갱신할 가중치 매개변수
- $\frac{\partial L}{\partial \mathbf{W}}$: W에 대한 손실 함수의 기울기
- η : 학습률(0.01 혹은 0.001등)

SGD 단점

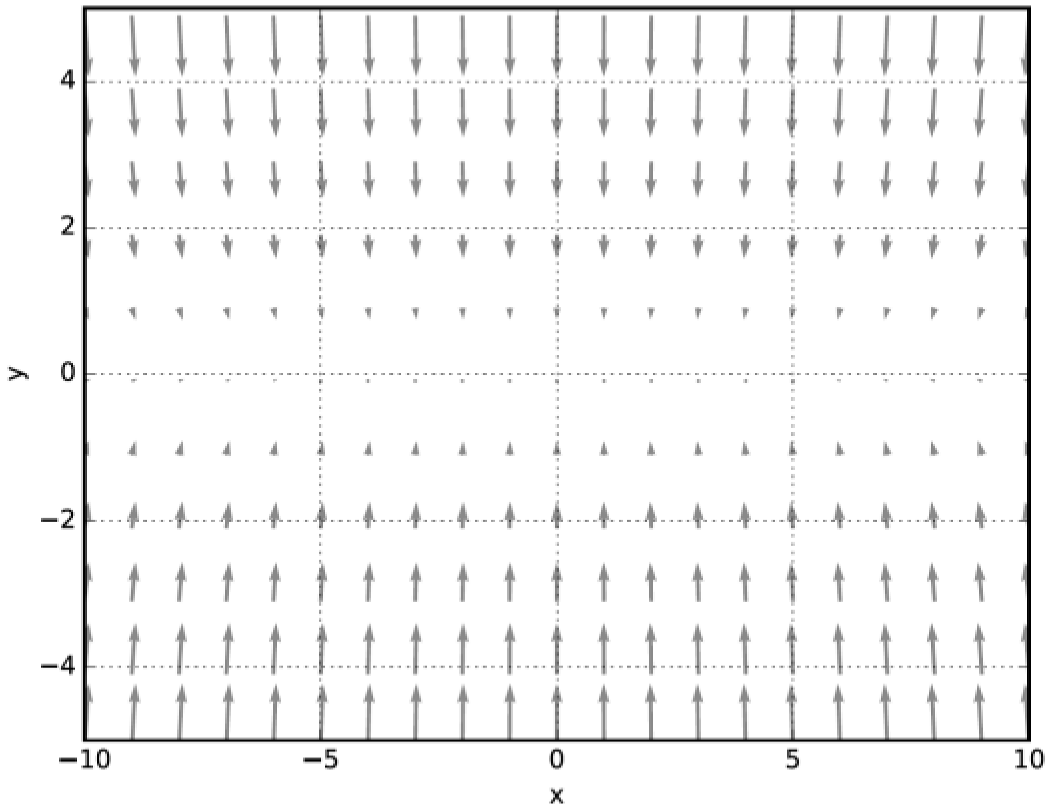
$$f(x,y) = \frac{1}{20}x^2 + y^2$$





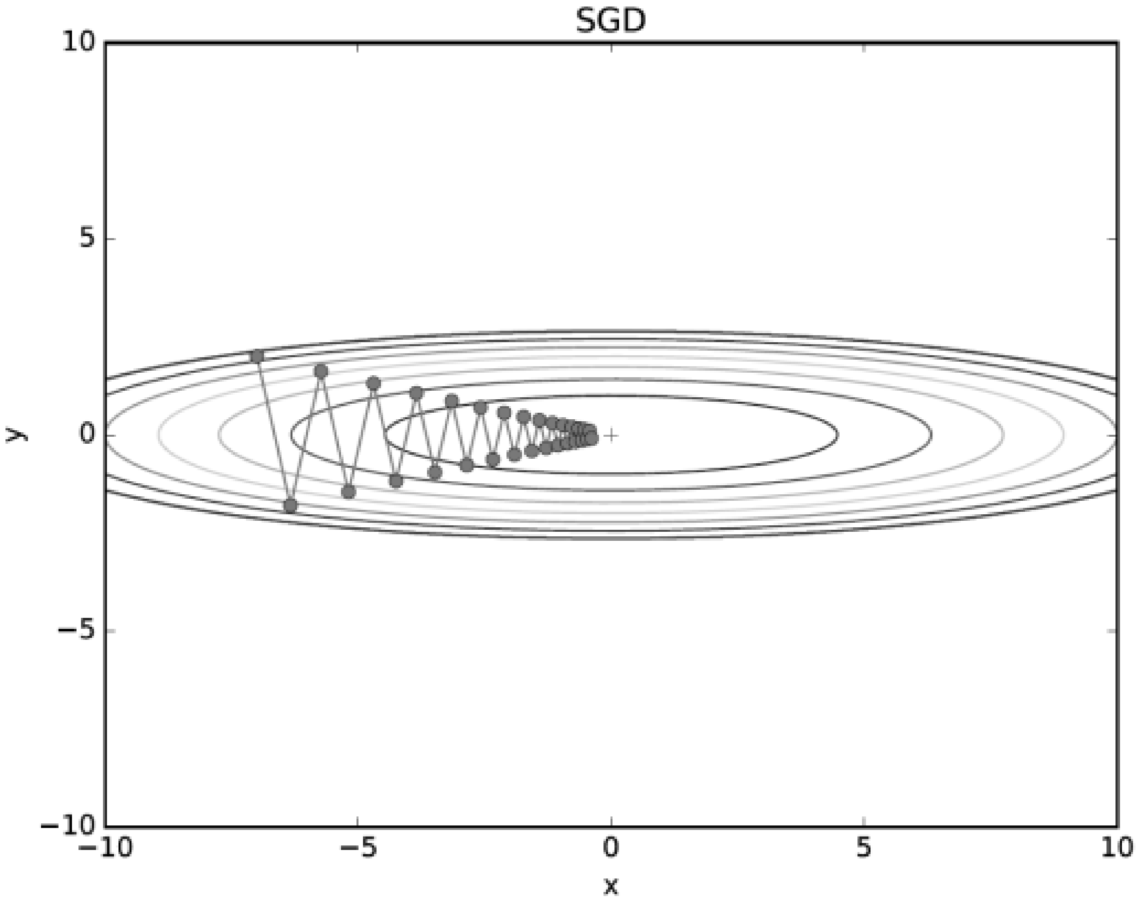
$f(x,y) = \frac{1}{20}x^2 + y^2$ 의 그래프(왼쪽)와 등고선(오른쪽)

SGD 단점



$f(x,y) = \frac{1}{20}x^2 + y^2$ 의 기울기

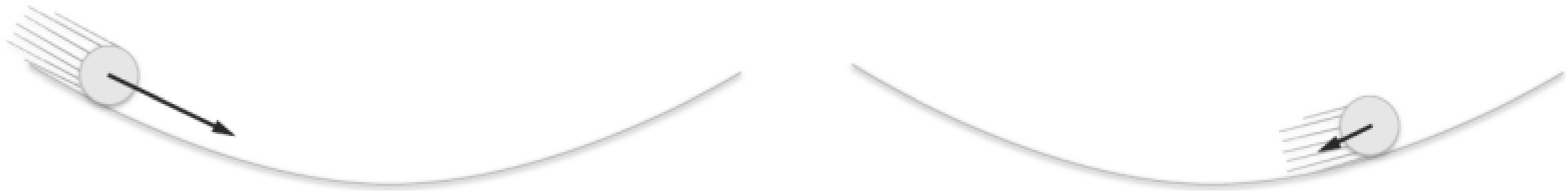
SGD 단점



- SGD에 의한 최적화 갱신 경로 : 최소값인 (0, 0)까지 지그재그로 이동하니 비효율적
- 비등방성(anisotropy) 함수(방향에 따라 성질, 즉 여기에서는 기울기가 달라지는 함수)에서는 탐색 경로가 비효율적.

SGD 단점의 개선책 - 모멘텀

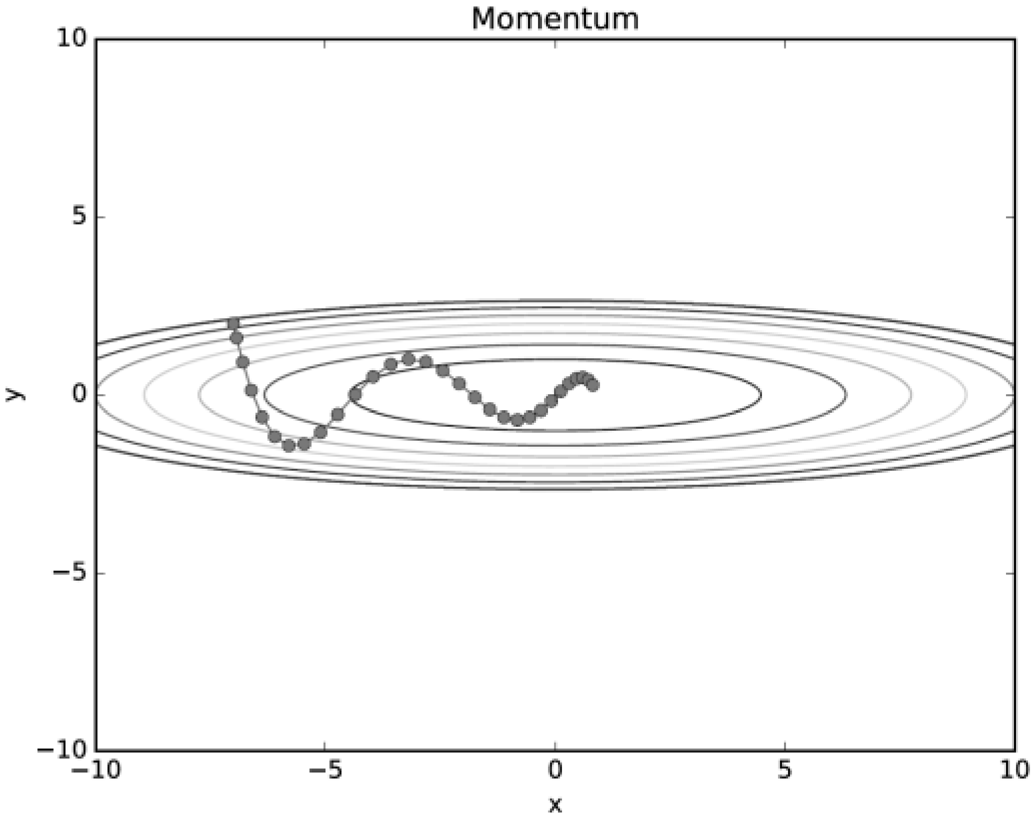
- 모멘텀(Momentum)
  - $\mathbf{v} \leftarrow \alpha\mathbf{v} - \eta \frac{\partial L}{\partial \mathbf{W}}$ V : 속도(Velocity) - 기울기 방향으로 힘을 받아 물체가 가속된다는 물리 법칙
  - $\mathbf{W} \leftarrow \mathbf{W} + \mathbf{v}$



- 모멘텀의 이미지 : 공이 그릇의 곡면(기울기)을 따라 구르듯 움직인다.

- 가장 가까운 로컬 미니멈에 머물지 않고, 더 나은 로컬 미니멈으로 모델을 최적화 할 수 있다.

SGD 단점의 개선책 - 모멘텀



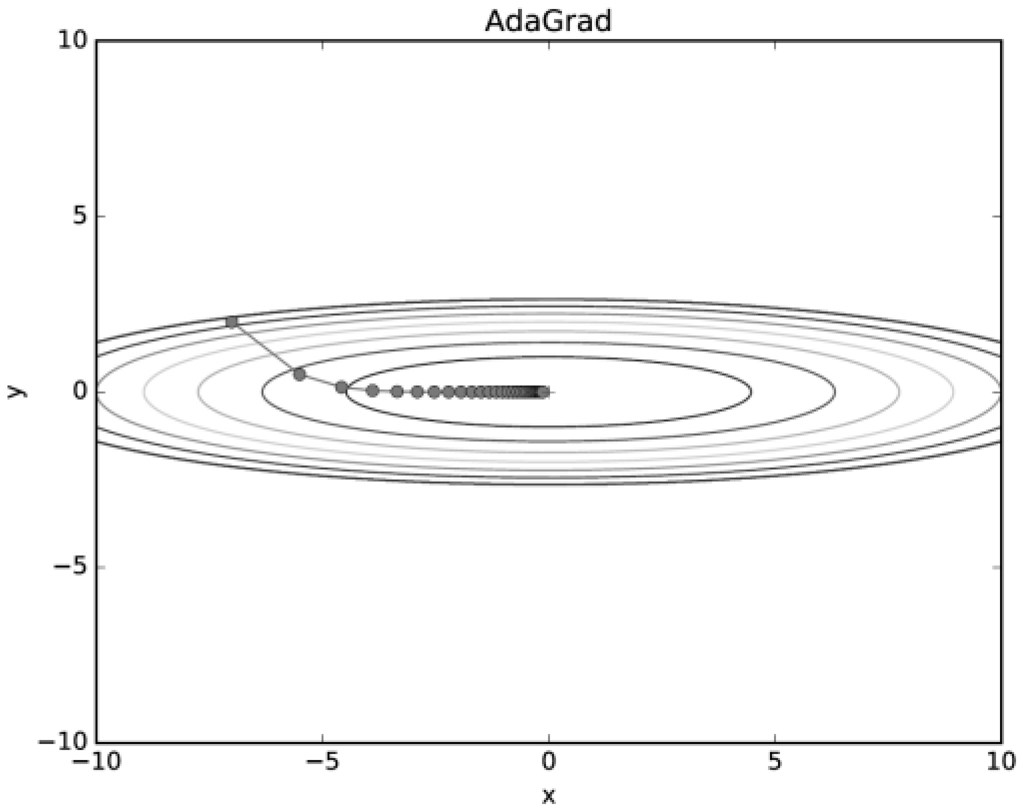
- 모멘텀에 의한 최적화 갱신 경로

AdaGrad

- 신경망 학습에서는 학습률(η) 값이 중요.
- 학습률 값이 너무 작으면 학습 시간이 너무 길어지고, 반대로 너무 크면 발산하여 학습이 제대로 이뤄지지 않음.
- 학습률 감소(learning rate decay) :
  - 학습률을 정하는 효과적 기술.
  - 처음에는 크게 학습하다가 학습률을 점차 줄여가면서 조금씩 작게 학습시키는 방법.
  - 학습률을 서서히 낮추는 가장 간단한 방법은 매개변수 전체의 학습률 값을 일괄적으로 낮추는 것.
- AdaGrad : 각각의 매개변수에 맞춤형 값을 만들어 주는 방식($\odot$: 행렬의 원소별 곱셈).

$$\mathbf{h} \leftarrow \mathbf{h} + \frac{\partial L}{\partial \mathbf{W}} \odot \frac{\partial L}{\partial \mathbf{W}} \quad , \qquad \mathbf{W} \leftarrow \mathbf{W} - \eta \frac{1}{\sqrt{\mathbf{h}}} \frac{\partial L}{\partial \mathbf{W}}$$

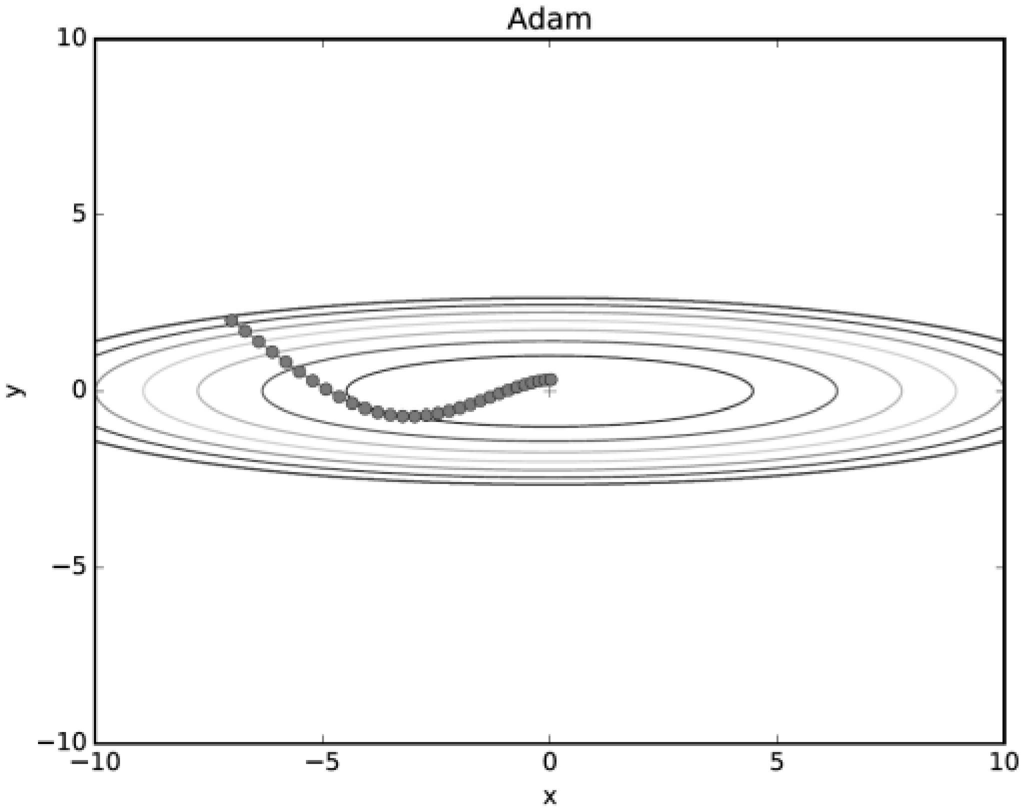
AdaGrad



- 최소값을 향해 효율적으로 이동.
- y축 방향은 기울기가 커서 처음에는 크게 움직이지만, 그 큰 움직임에 비례해 갱신 정도도 큰 폭으로 작아지도록 조정.
- 따라서 y축 방향으로 갱신 강도가 빠르게 약해지고, 지그재그 움직임이 줄어듬.

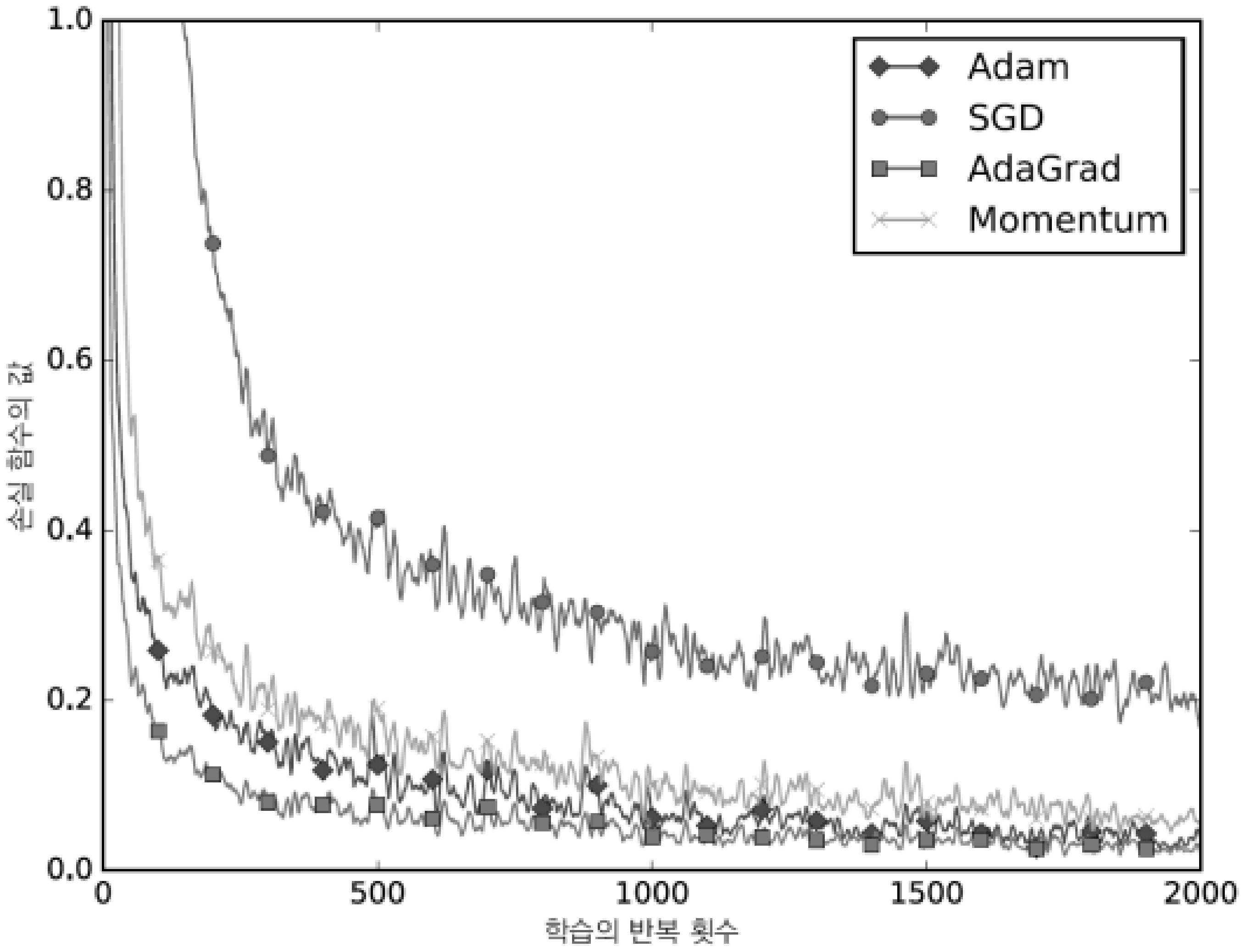
Adam

- 모멘텀과 AdaGrad를 융합한 듯한 방법으로 2015년에 제안된 새로운 방법.



- Adam에 의한 최적화 갱신 경로

MNIST 데이터셋으로 본 갱신 방법 비교

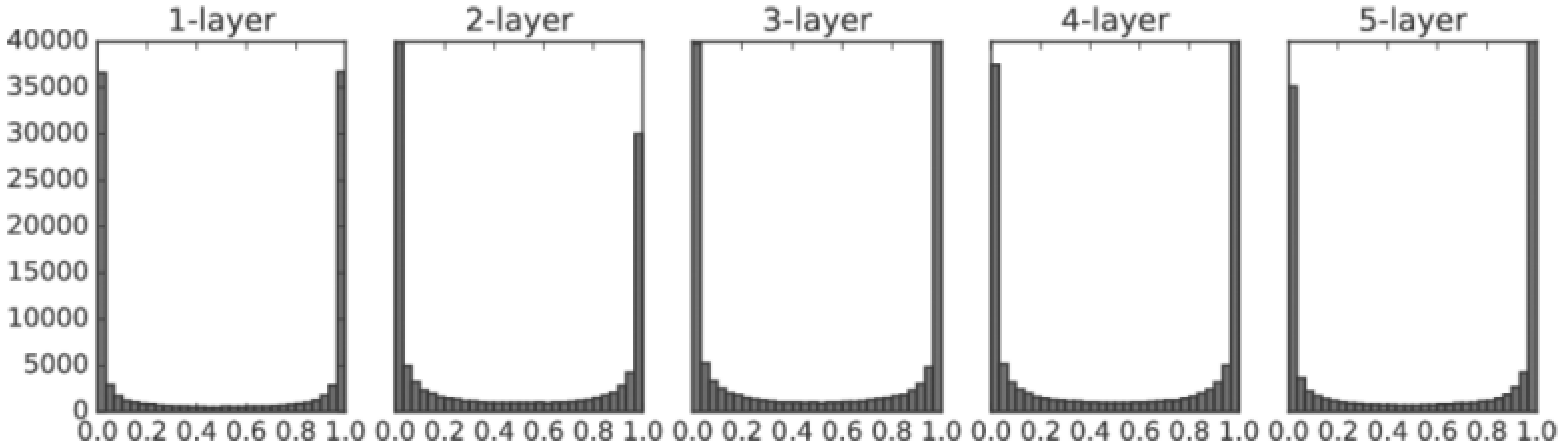


가중치 초기값

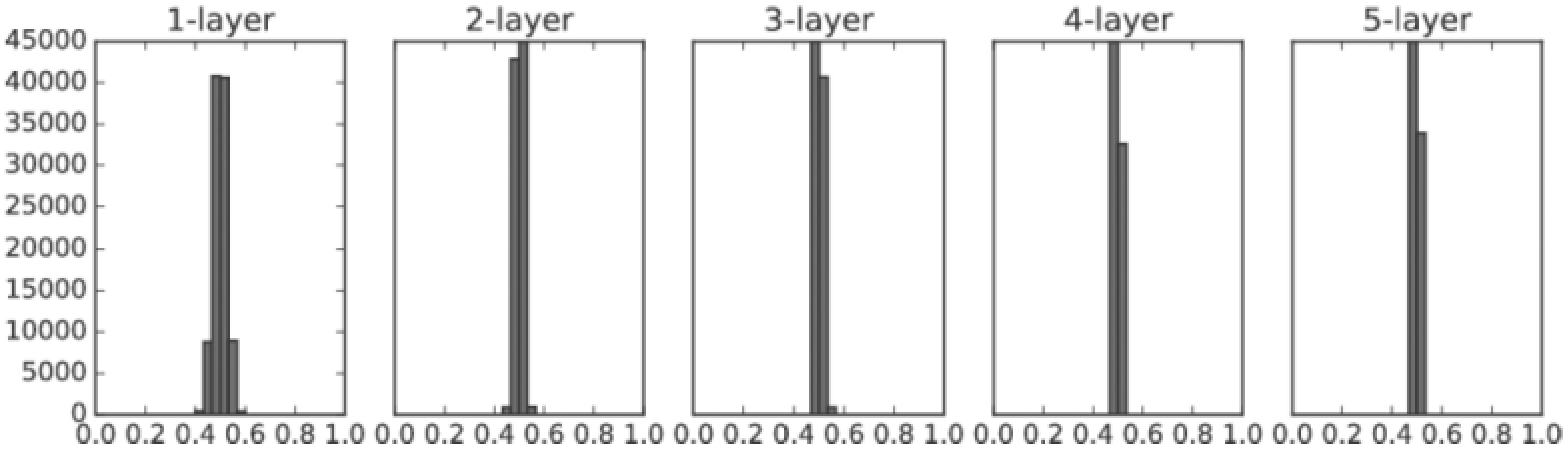
- 가중치 감소(weight decay) 기법
  - 오버피팅을 억제해 범용 성능을 높이는 테크닉.
  - 가중치 매개변수의 값이 작아지도록 학습하는 방법.
  - 즉, 가중치 값을 작게 하여 오버피팅이 일어나지 않게 하는 것.

- 지금까지의 가중치 초기값은 0.01 * np.random.randn(10, 100)처럼 정규분포에서 생성되는 값을 0.01배 한 작은 값(표준편차가 0.01인 정규분포)을 사용.

은닉층의 활성화값 분포

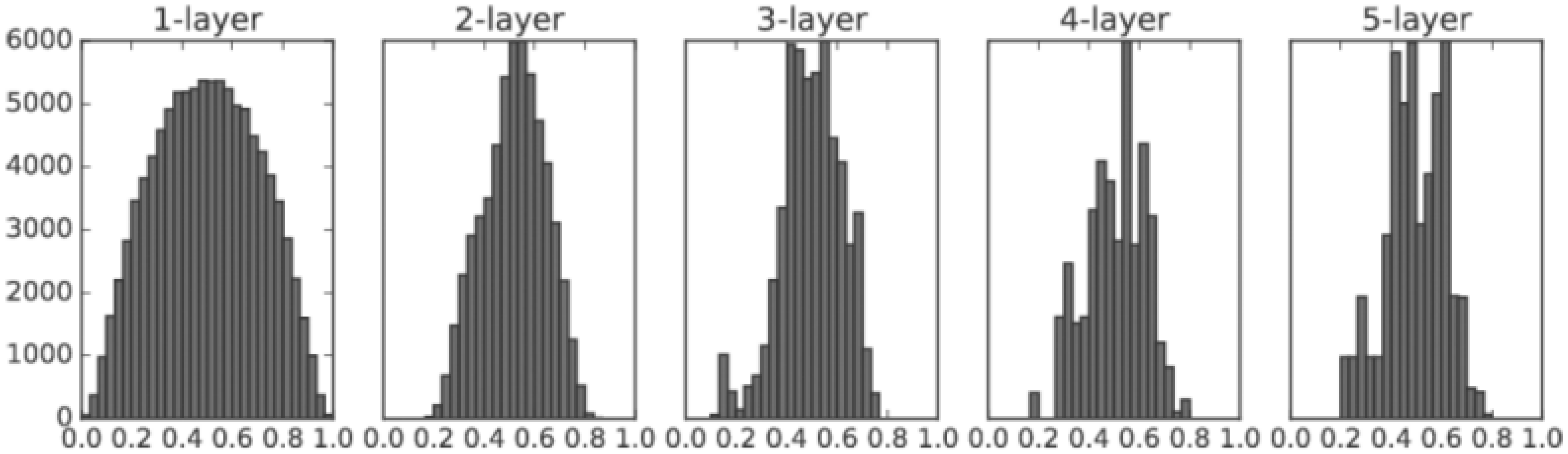


- 가중치를 표준편차가 1인 정규분포로 초기화할 때의 각 층의 활성화값 분포



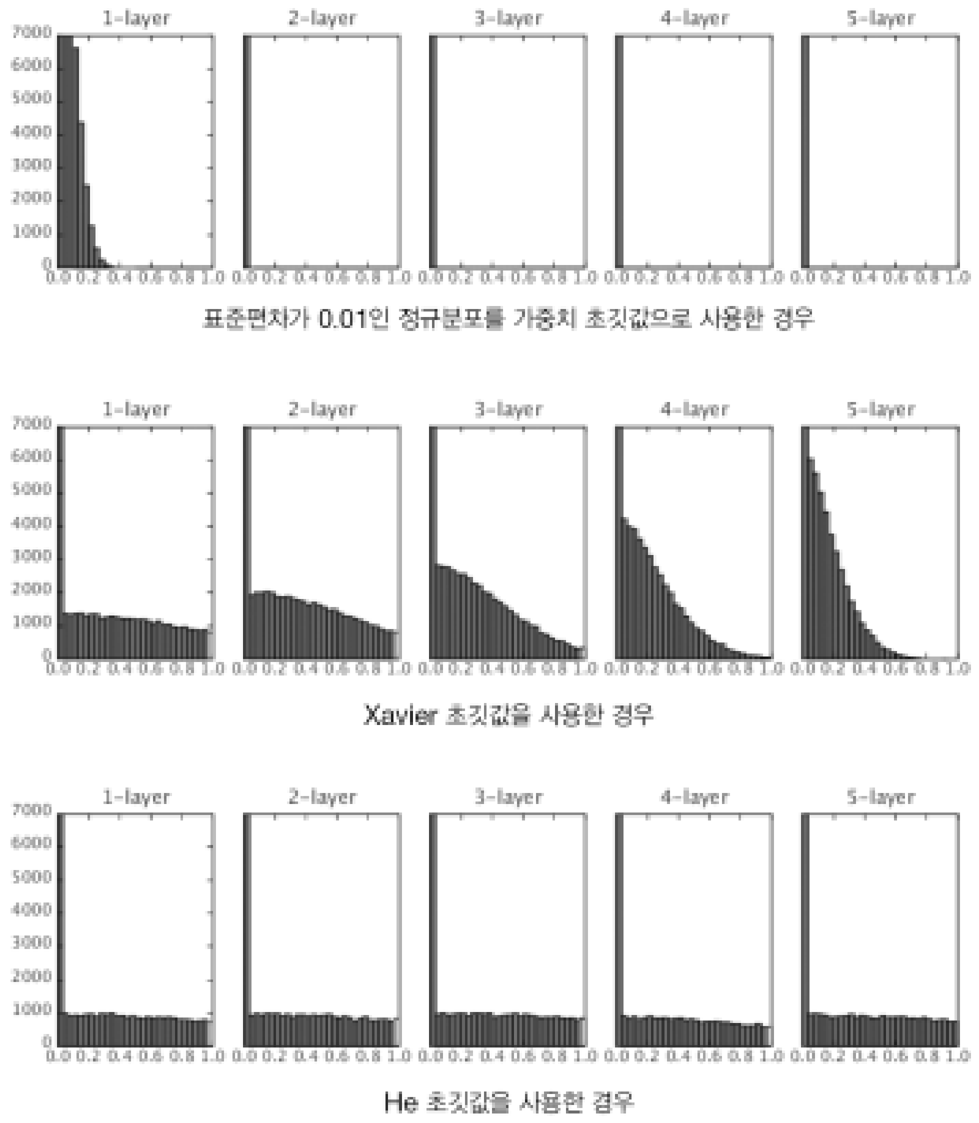
- 가중치를 표준편차가 0.01인 정규분포로 초기화할 때의 각 층의 활성화값 분포

은닉층의 활성화값 분포



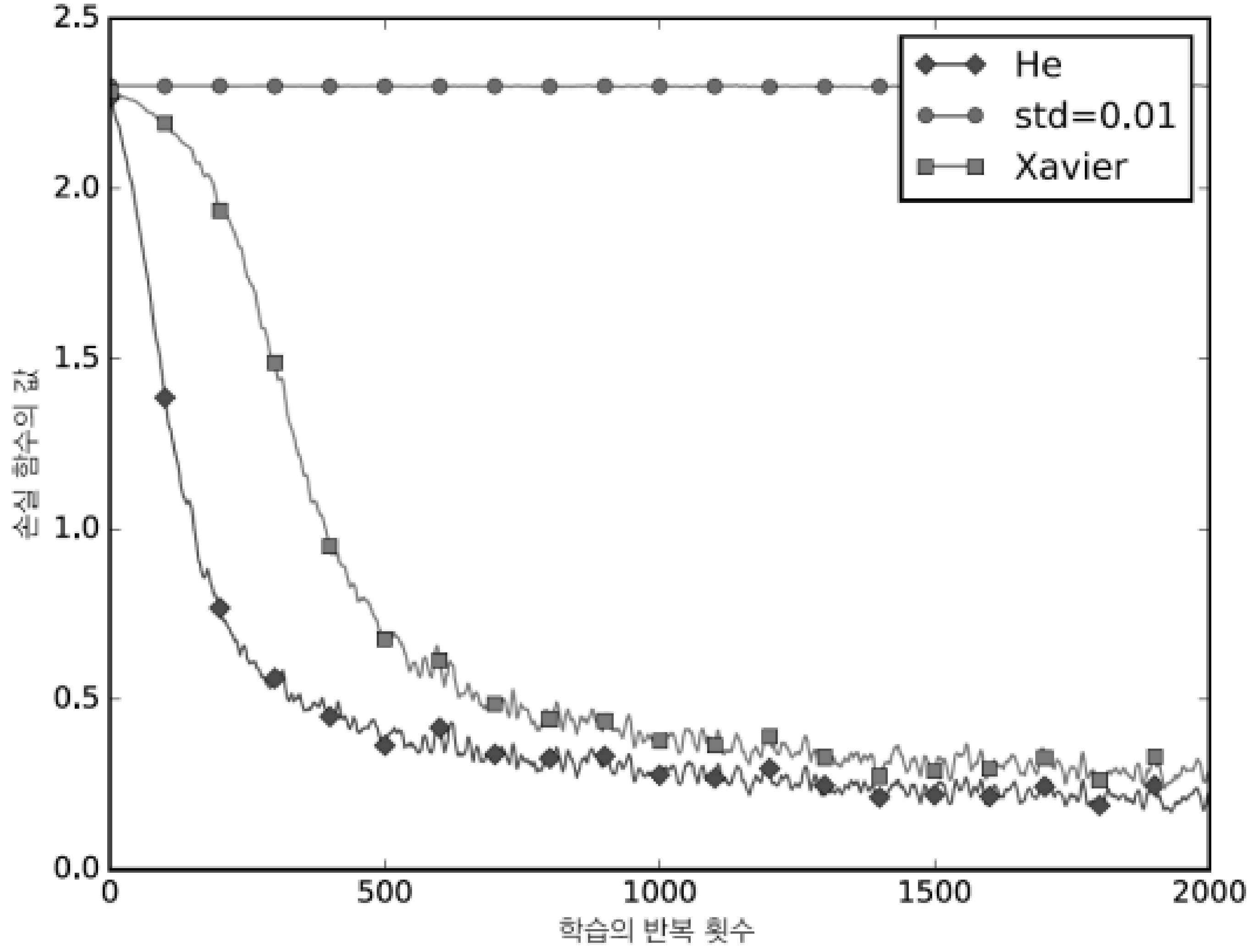
- 가중치의 초기값으로 Xavier 초기값을 이용할 때의 각 층의 활성화값 분포.
- 각 층의 활성화 값들을 광범위하게 분포시킬 목적으로 가중치의 적절한 분포를 찾고자 함.
- 앞 계층의 노드가 n개라면 표준편차가 1 / sqrt(n)인 분포를 사용하면 된다는 결론.
- 활성화 함수로 sigmoid를 사용할 때 Xavier 초기값 적용.
- 활성화 함수로 ReLU를 사용할 때 He 초기값(sqrt(2/n)) 적용.

ReLU를 사용할 때의 가중치 초기값



- 활성화 함수로 ReLU를 사용한 경우의 가중치 초기값에 따른 활성화값 분포변화.
- He 초기값 : 앞 계층의 노드가 n개라면 표준편차가 루트 2 / n인 분포를 사용하면 된다는 결론.

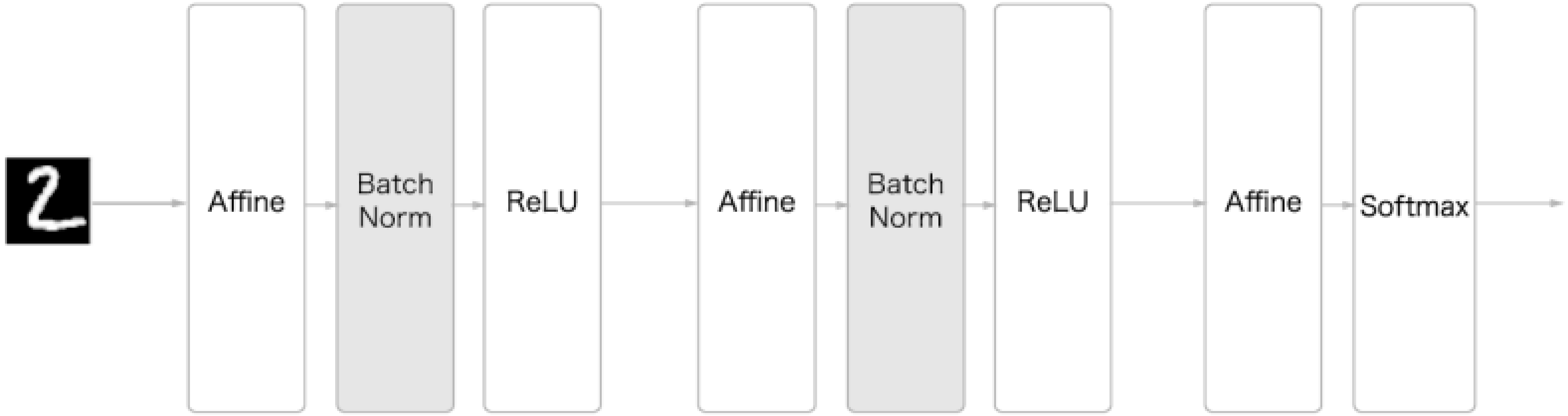
MNIST 데이터셋으로 본 가중치 초기값 비교



- 층별 뉴런 수 : 100개
- 5층 신경망
- 활성화 함수 : ReLU

배치 정규화

- 배치 정규화를 사용한 신경망의 예



$$\mu_B \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i$$

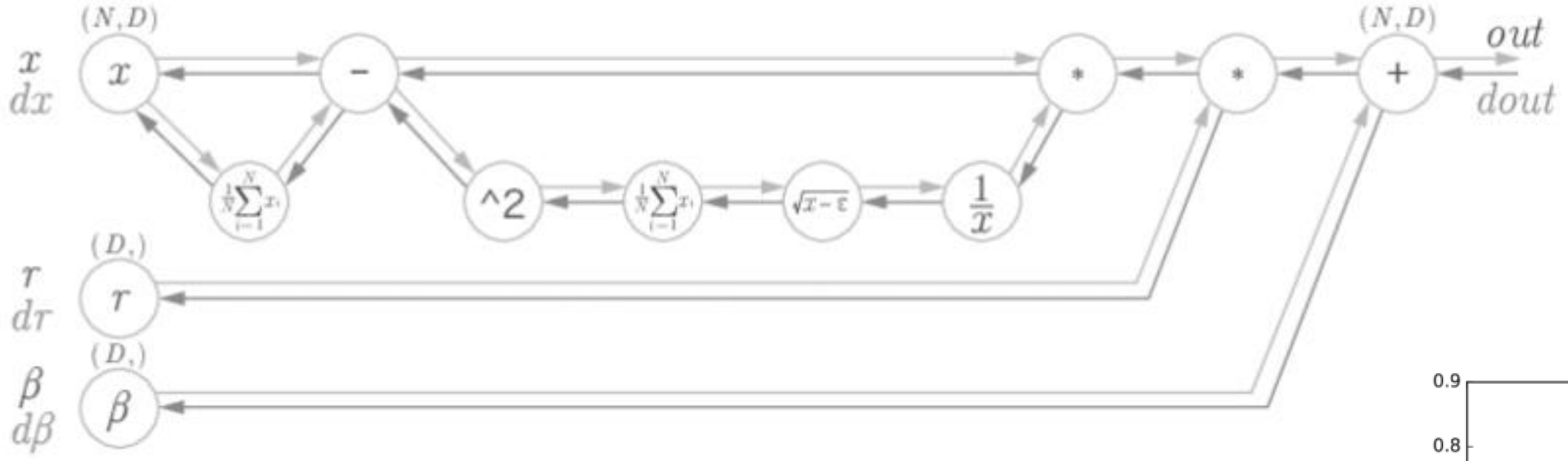
$$\sigma_B^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_B)^2$$

$$\hat{x}_i \leftarrow \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}}$$

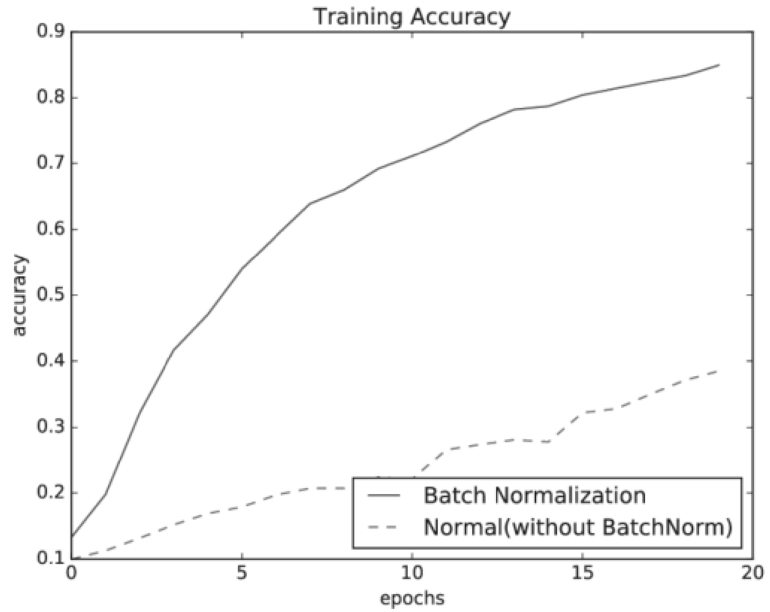
- 배치 정규화는 데이터 분포가 평균이 0, 분산이 1이 되도록 정규화함.

배치 정규화

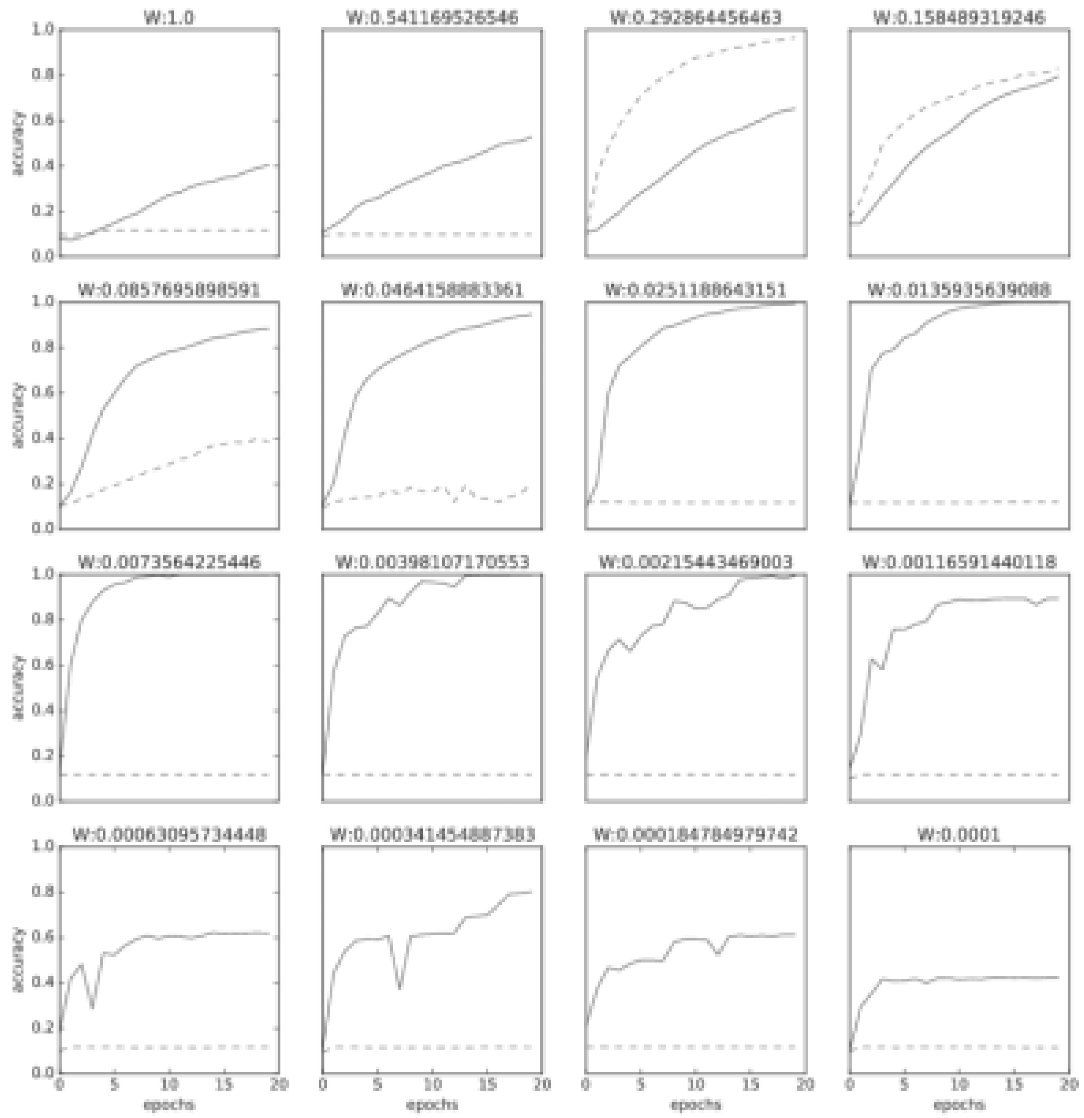
- 배치 정규화 계산 그래프



- 배치 정규화의 효과 : 배치 정규화가 학습 속도를 높인다.



배치 정규화



- 가중치 초기값의 표준편차를 다양하게 바꿔가며 학습경과를 관찰한 그래프.
- 실선이 배치 정규화를 사용한 경우.
- 점선이 배치 정규화를 사용하지 않은 경우.
- W : 가중치 초기값의 표준편차.

올바른 학습(Learning)을 위해

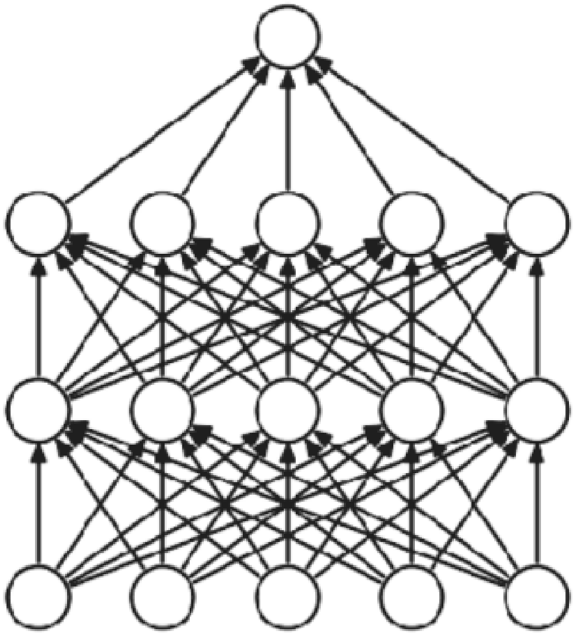
- 오버피팅
  - 신경망이 훈련 데이터에만 지나치게 적응되어 그 외의 데이터에는 제대로 대응하지 못하는 상태.

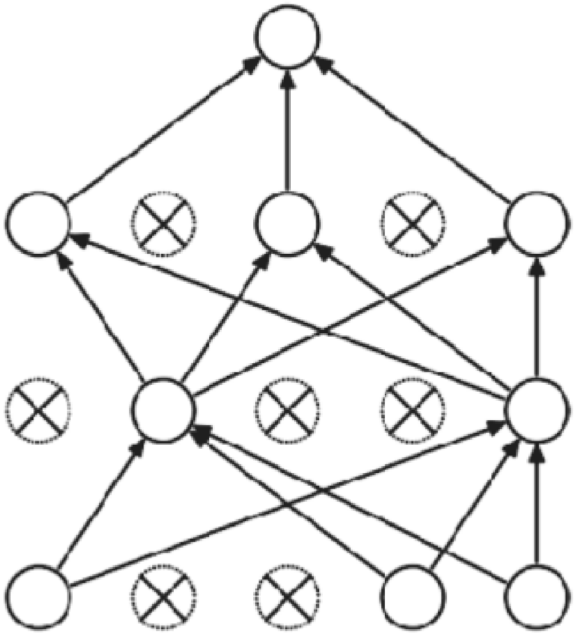
- 오버피팅 발생 주요 원인
  - 매개변수가 많고 표현력이 높은 모델.
  - 훈련 데이터가 적음.

- 오버피팅 억제용 : 가중치 감소(weight decay)
  - 학습 과정에서 큰 가중치에 대해서는 그에 상응하는 큰 페널티를 부과하여 오버피팅 억제하는 방법.
  - 원래 오버피팅은 가중치 매개변수의 값이 커서 발생하는 경우가 많기 때문.

올바른 학습(Learning)을 위해

- 드롭아웃
  - 신경망 모델이 복잡해지면 가중치 감소만으로는 대응하기 어려울 때 이용할 수 있는 기법.
  - 뉴런을 임의로 삭제하면서 학습하는 방법.
  - 훈련때는 데이터를 흘릴 때마다 삭제할 뉴런을 무작위로 선택하고, 시험 때는 모든 뉴런에 신호를 전달하되 각 뉴런의 출력에 훈련 때 삭제한 비율을 곱하여 출력.


(a) 일반 신경망


(b) 드롭아웃을 적용한 신경망

적절한 하이퍼파라미터 값 찾기

- 하이퍼파라미터
  - 각 층의 뉴런 수
  - 배치 크기
  - 매개변수 갱신 시의 학습률
  - 가중치 감소

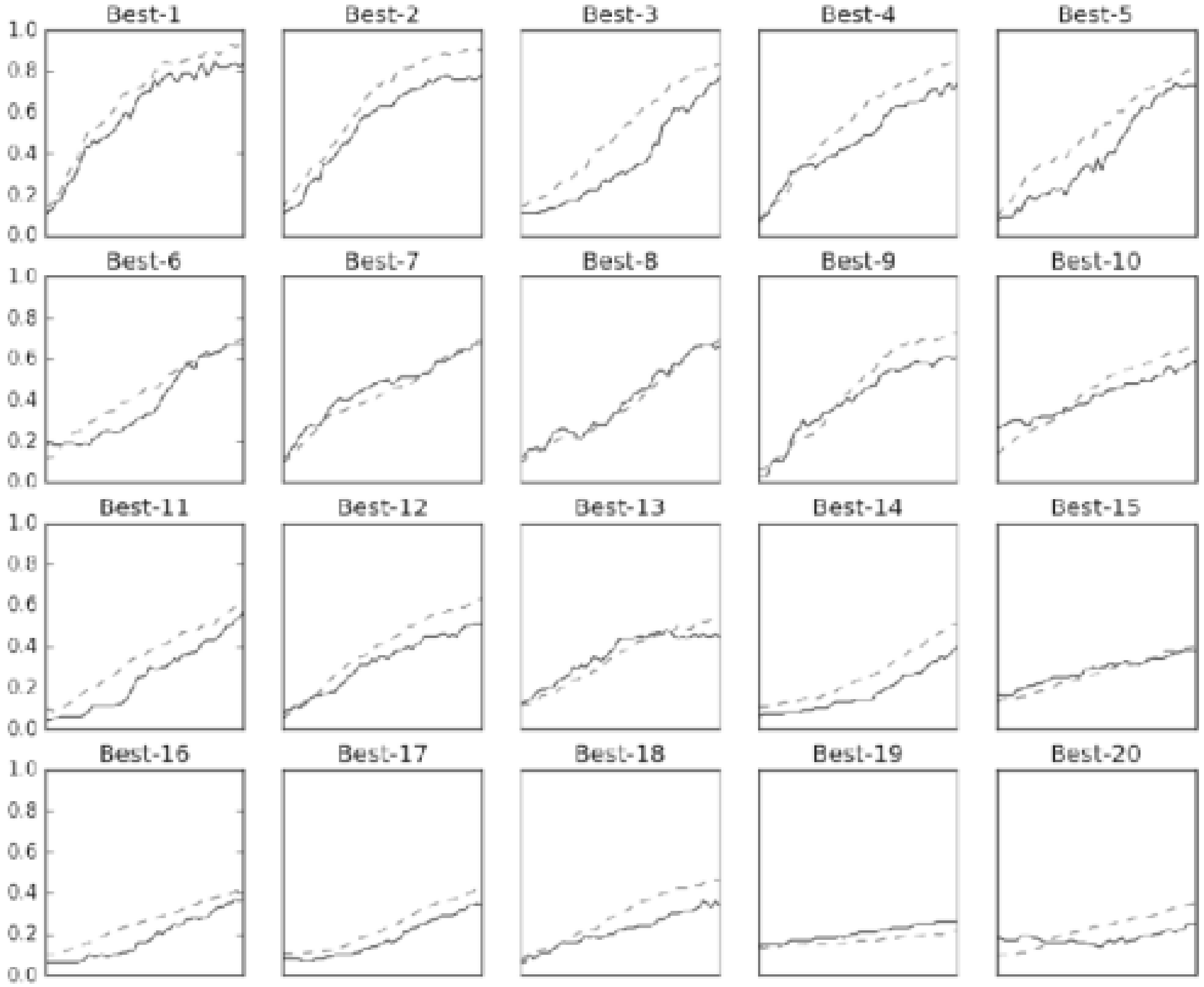
  등의 매개변수.

- 데이터 셋의 구별
  - 훈련 데이터 : 매개변수 학습.
  - 검증 데이터 : 하이퍼파라미터 성능 평가(훈련 데이터 중 20% 정도 분리).
  - 시험 데이터 : 신경망의 범용 성능 평가.

하이퍼파라미터 최적화

- 0단계
  - 하이퍼파라미터 값의 범위를 설정.
- 1단계
  - 설정된 범위에서 하이퍼파라미터의 값을 무작위로 추출.
- 2단계
  - 1단계에서 샘플링한 하이퍼파라미터 값을 사용하여 학습하고, 검증 데이터로 정확도를 평가 (단, 에폭은 작게 설정).
- 3단계
  - 1단계와 2단계를 특정 횟수(100회 등) 반복하여, 그 정확도의 결과를 보고 하이퍼파라미터의 범위를 좁힘.

하이퍼파라미터 최적화 구현

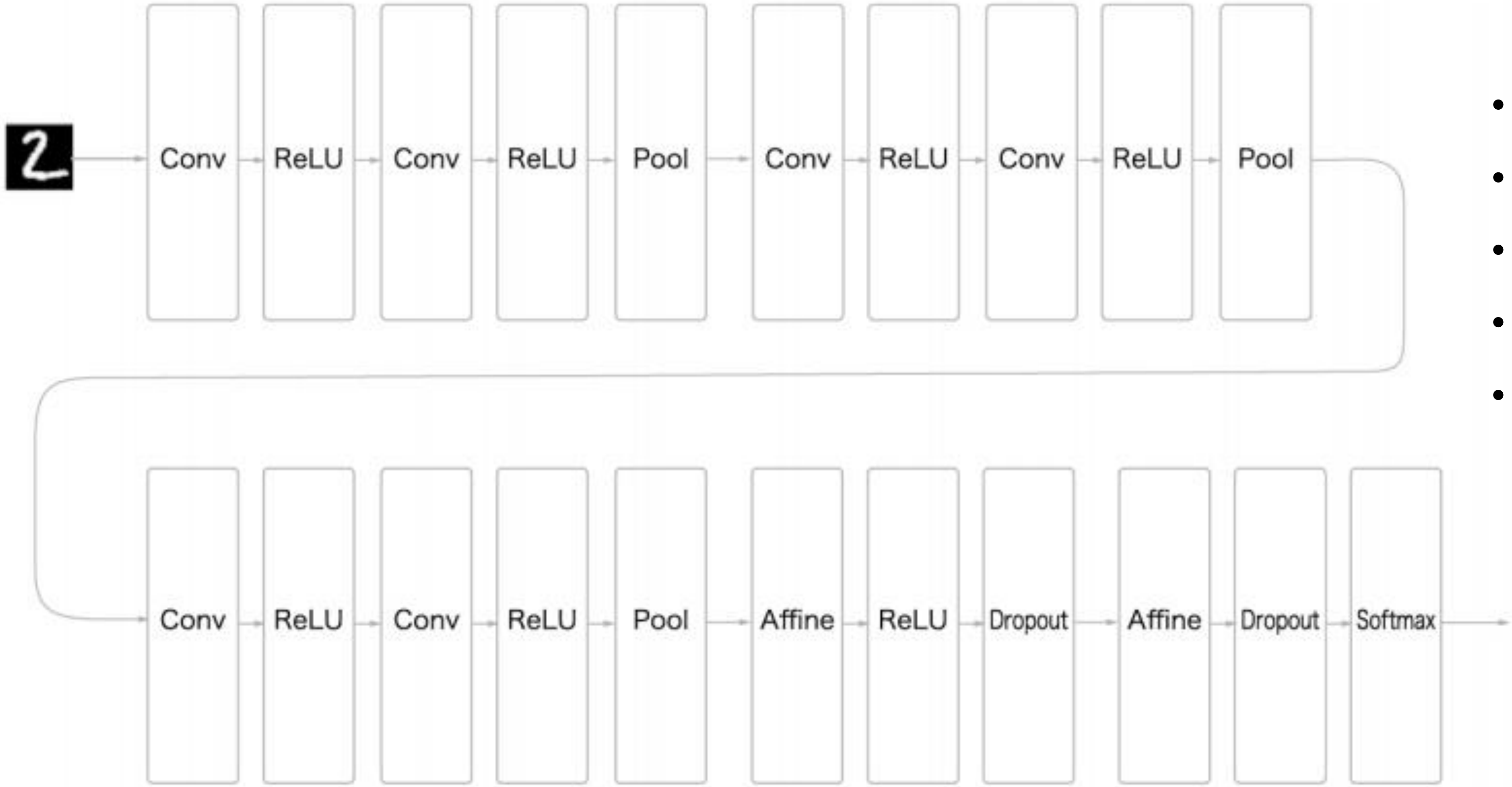


- 실선은 검증 데이터에 대한 정확도.
- 점선은 훈련 데이터에 대한 정확도.

딥러닝
(Deep Learning)

딥러닝

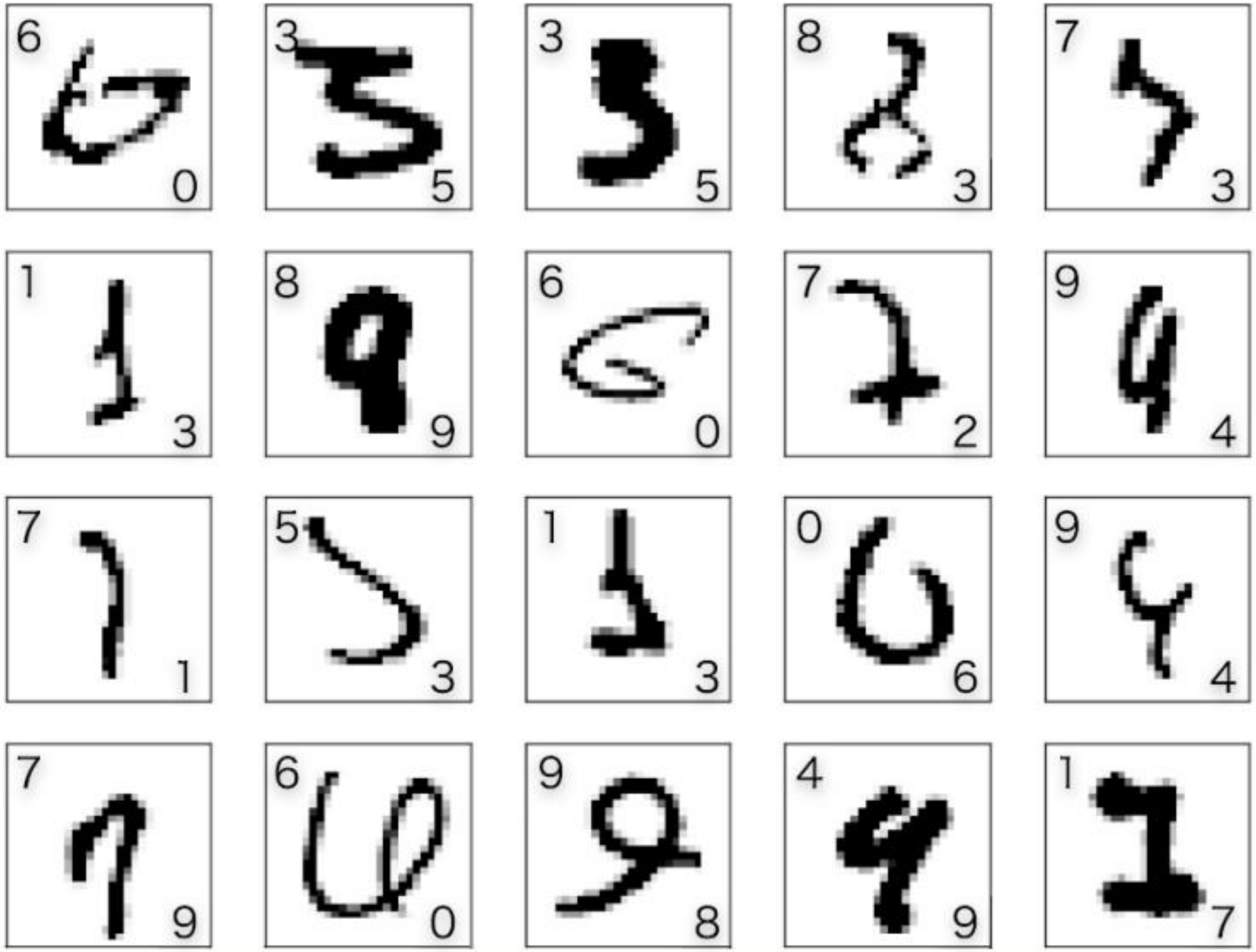
- 층을 깊게 한 심층 신경망
- 심층 신경망은 지금까지 설명한 신경망을 바탕으로 뒷단에 층을 추가하기만 하면 만들 수 있음
- 그 동안 배운 기술을 집약하고 심층 신경망을 만들어 MNIST 데이터 셋의 손글씨 숫자 인식률을 높일 수 있음



- 3x3 의 작은 필터를 사용한 합성곱 계층
- 활성화 함수는 ReLU
- 완전연결 계층 뒤에 드롭아웃 계층 사용
- Adam 을 사용해 최적화
- 가중치 초기값은 'He'의 초기값

딥러닝

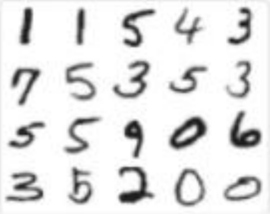
- 모델이 인식하지 못한 이미지들



MNIST 데이터셋에 대한 각 기법의 순위 (2016 년 12 월 시점)

MNIST

who is the best in MNIST ?



MNIST 50 results collected

Units: error %

> Classify handwriten digits. Some additional results are available on the original dataset page.

Result	Method	Venue	Details
0.21%	Regularization of Neural Networks using DropConnect	ICML 2013	
0.23%	Multi-column Deep Neural Networks for Image Classification	CVPR 2012	
0.23%	APAC: Augmented PAttern Classification with Neural Networks	arXiv 2015	
0.24%	Batch-normalized Maxout Network in Network	arXiv 2015	Details
0.29%	Generalizing Pooling Functions in Convolutional Neural Networks: Mixed, Gated, and Tree	AISTATS 2016	Details
0.31%	Recurrent Convolutional Neural Network for Object Recognition	CVPR 2015	
0.31%	On the Importance of Normalisation Layers in Deep Learning with Piecewise Linear Activation Units	arXiv 2015	
0.32%	Fractional Max-Pooling	arXiv 2015	Details

정확도를 높일 수 있는 기술

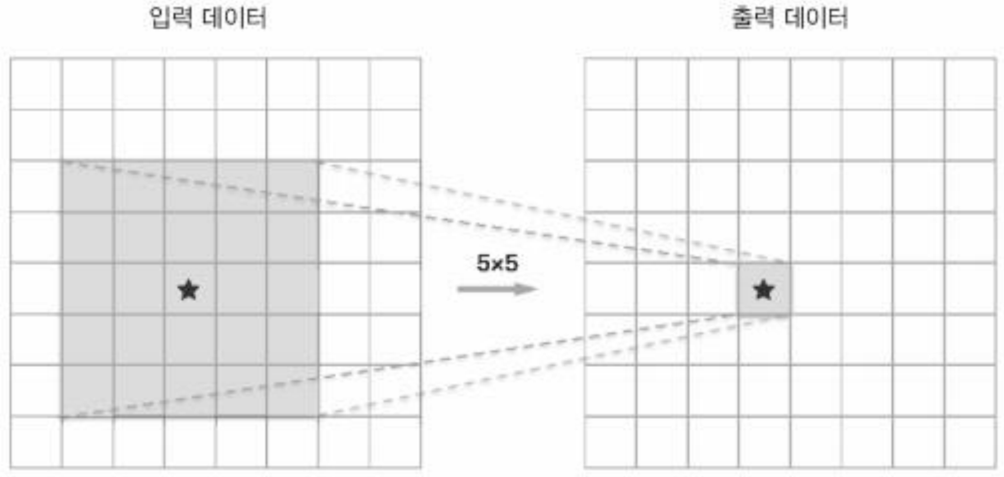
- 앙상블 학습
- 학습률 감소
- 데이터 확장

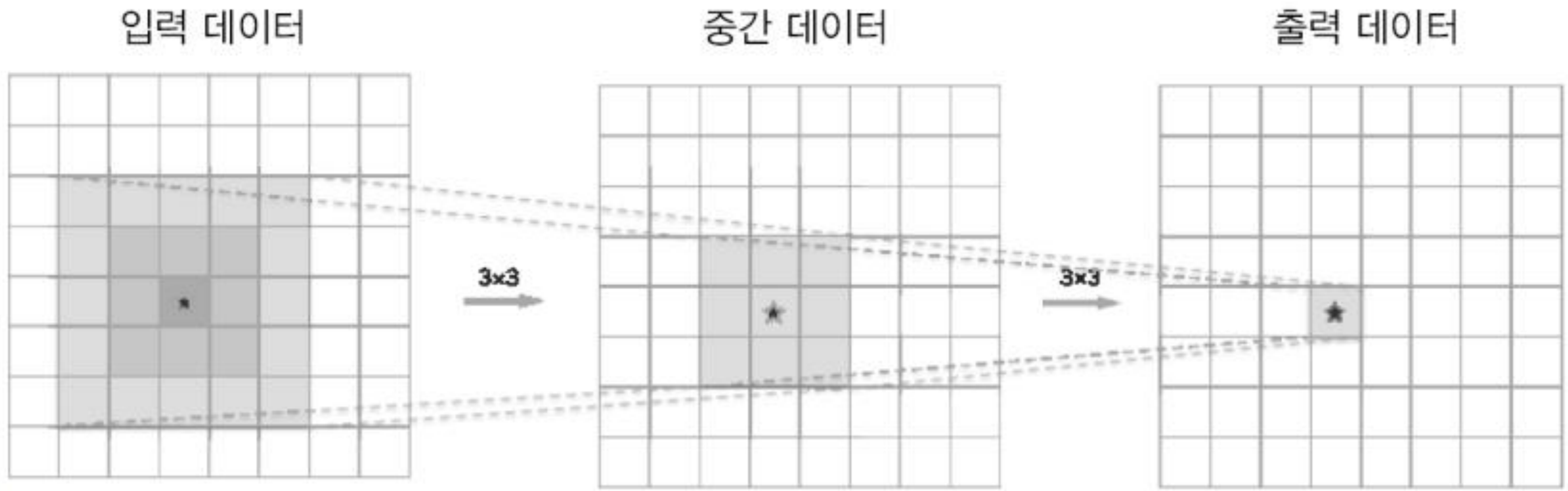
층을 깊게 하는 이유

- ‘층을 깊게 하는 것’이 왜 중요한가에 대한 이론적인 근거는 아직 많이 부족한 것이 사실
- ILSVRC 로 대표되는 대규모 이미지 인식 대회
    - 상위를 차지한 기법이 딥러닝 기반의 신경망을 더 깊게 만드는 방향으로 가고 있음

층을 깊게 할 때의 이점

- 신경망의 매개변수 수가 줄어듦





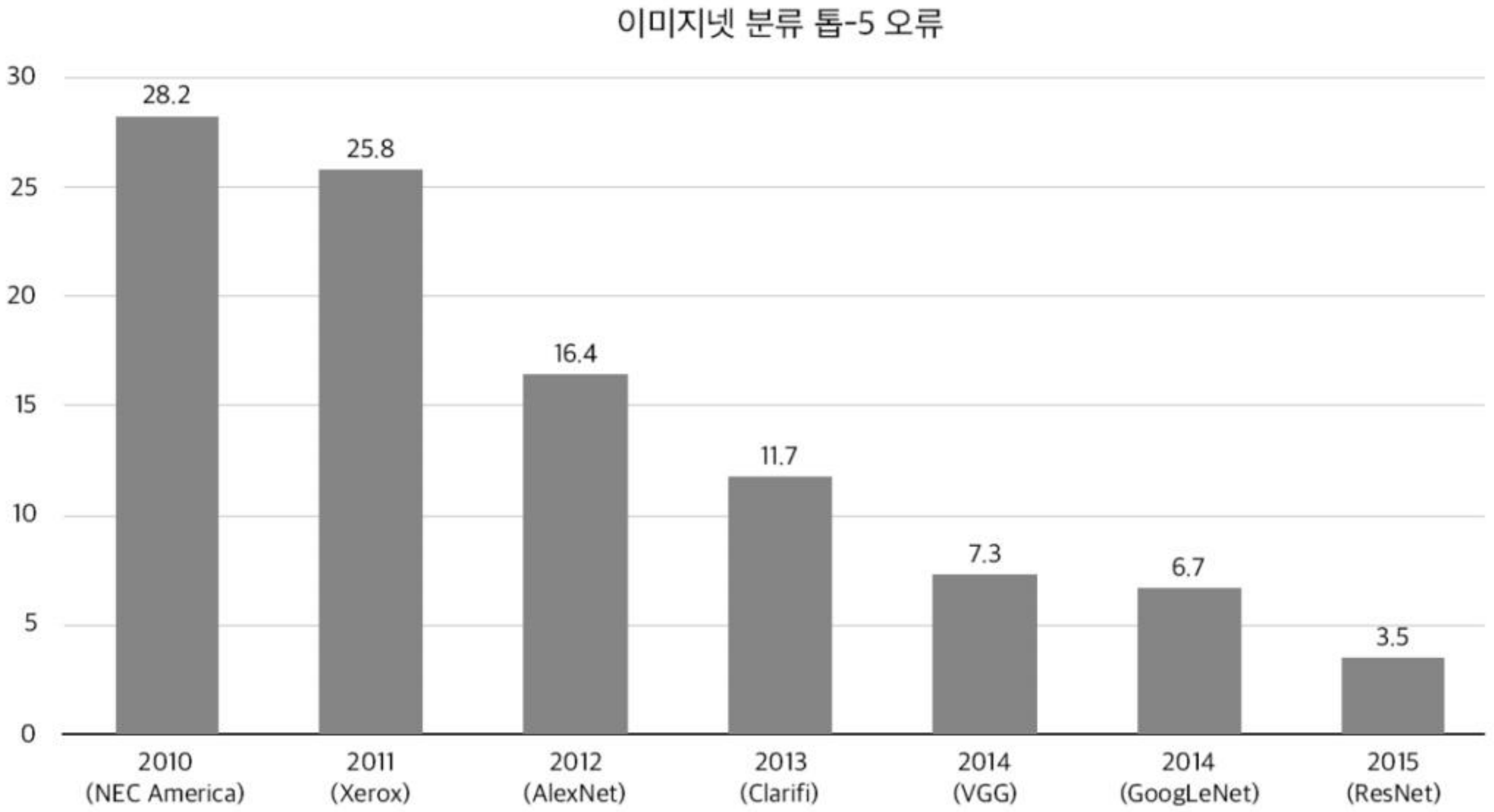
층을 깊게 할 때의 이점

- 학습의 효율성도 층을 깊게 하는 것이 이점 .
- 정보를 계층적으로 전달할 수 있다 .
- 단 , 최근 일어나고 있는 층의 심화는 층이 깊어도 제대로 학습할 수 있도록 해주는 새로운 기술과 환경 (빅데이터와 컴퓨터 연산 능력 등) 이 뒷받침되어 나타난 현상임 .

딥러닝의 초기 역사

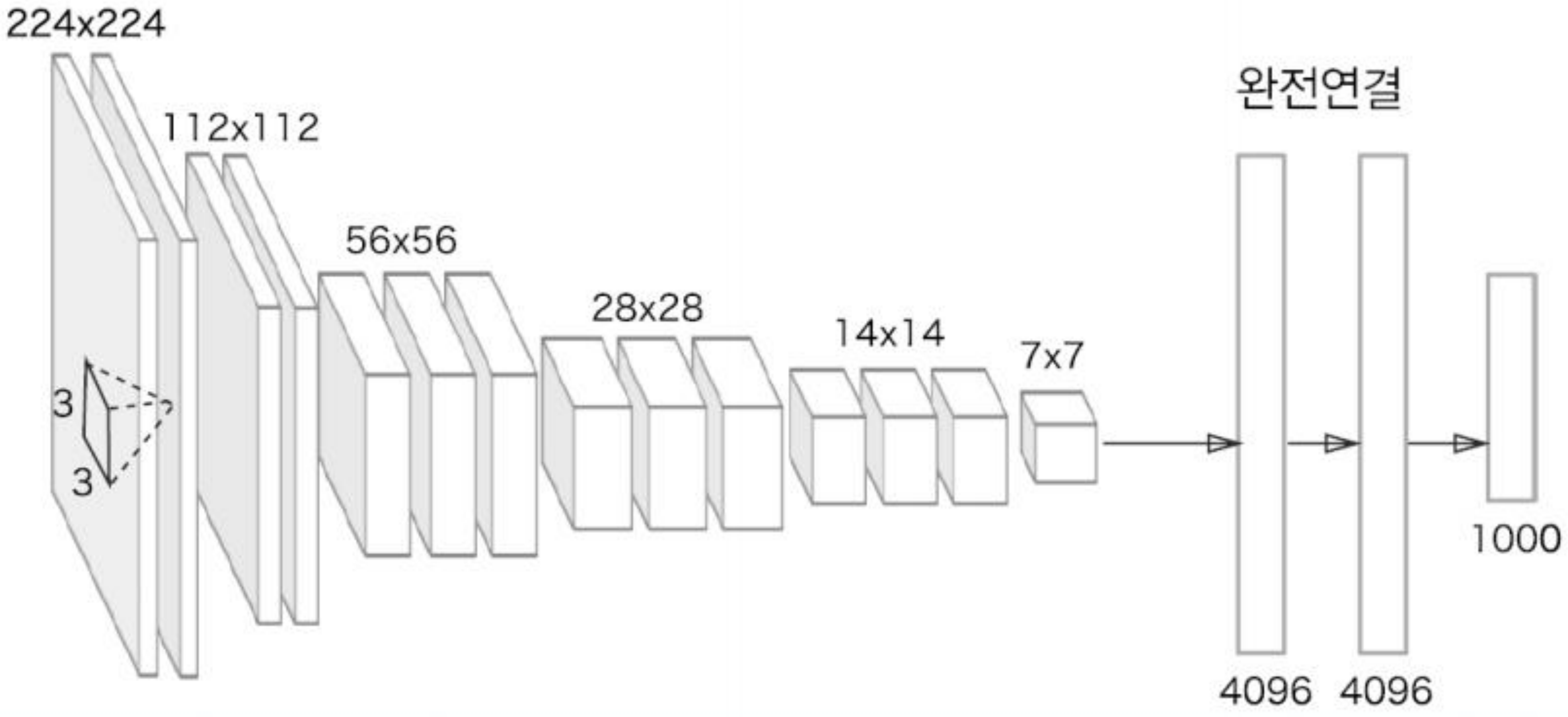
- 2012 년 ILSVRC(ImageNet Large Scale Visual Recongnition Challenge) 대회
  - 이미지 인식 기술을 겨루는 장
  - 딥러닝에 기초한 기법 , 일명 AlexNet 이 압도적 성적으로 우승하면서 그동안의 이미지 인식에 대한 접근법을 뿌리부터 뒤흔듦 .
- ImageNet : 100 만 장이 넘는 이미지를 담고 있는 데이터셋 .

ILSVRC 최우수 팀의 성적 추이



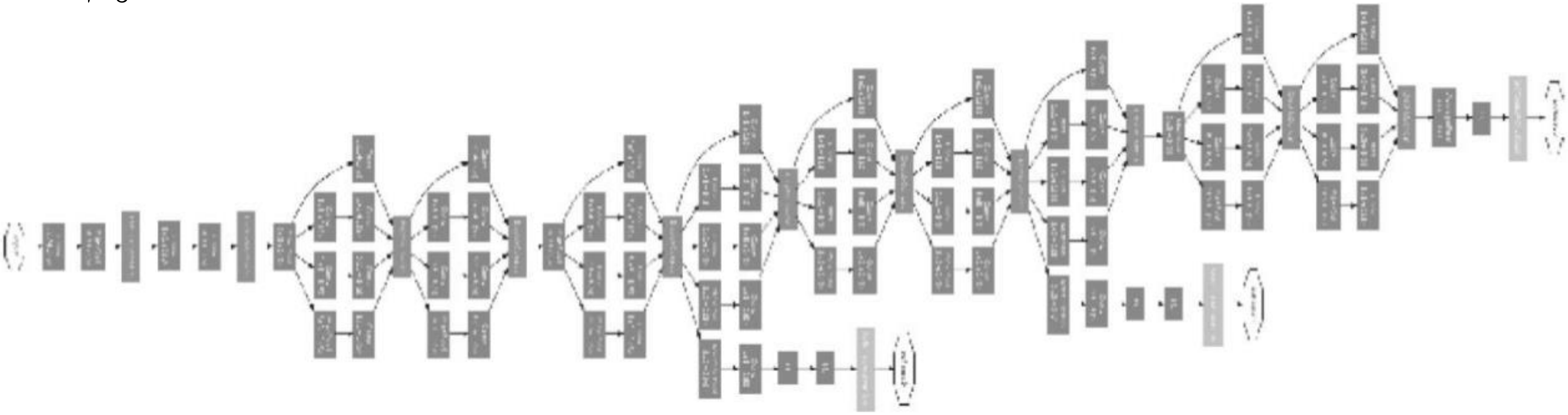
VGG

- 합성곱 계층과 풀링 계층으로 구성되는 기본적인 CNN
- 다만 , 그림과 같이 비중있는 층 (합성곱 계층 , 완전연결 계층) 을 모두 16 층 (혹은 19 층) 으로 심화한 게 특징
- 3x3 의 작은 필터를 사용한 합성곱 계층을 연속으로 거친다 .
- 합성곱 계층을 2~4 회 연속으로 풀링 계층을 두어 크기를 절반으로 줄이는 처리를 반복 .
- 마지막으로 완전연결 계층을 통과시켜 결과를 출력 .

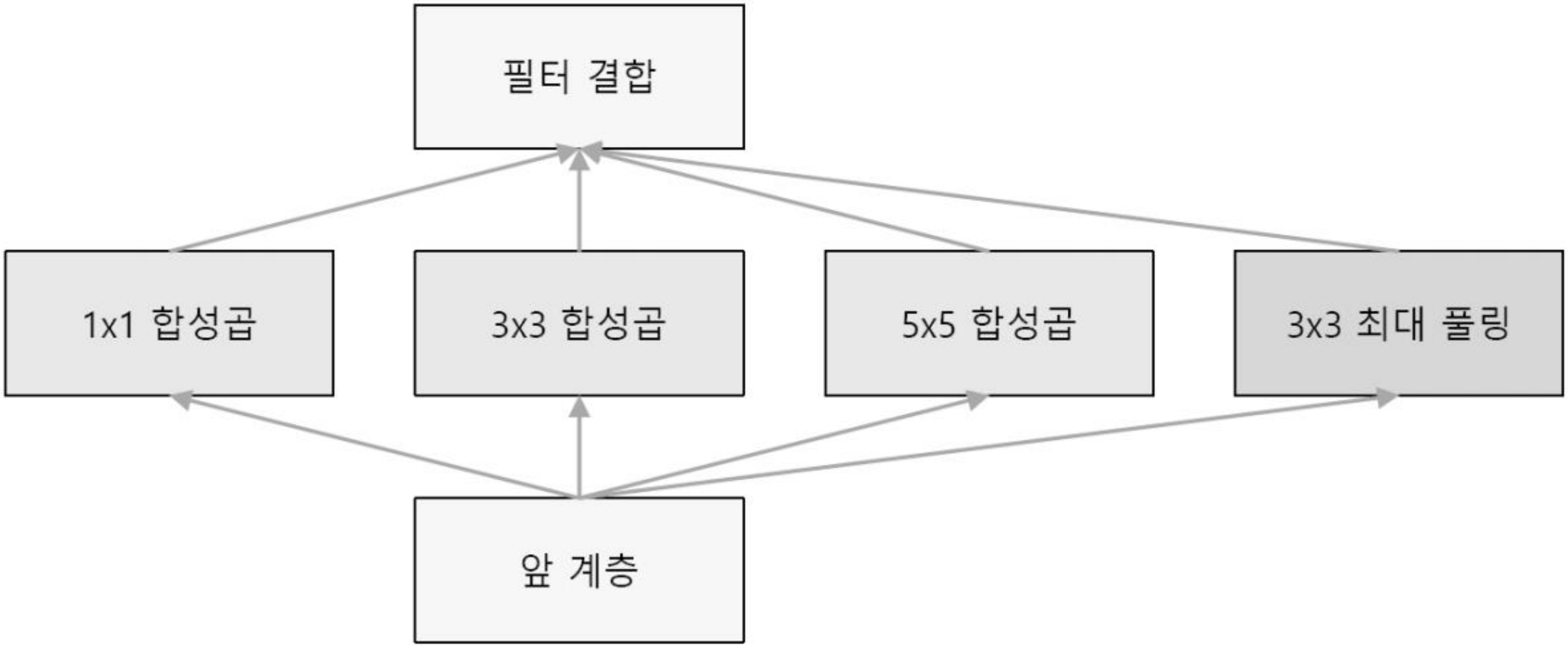


GoogLeNet

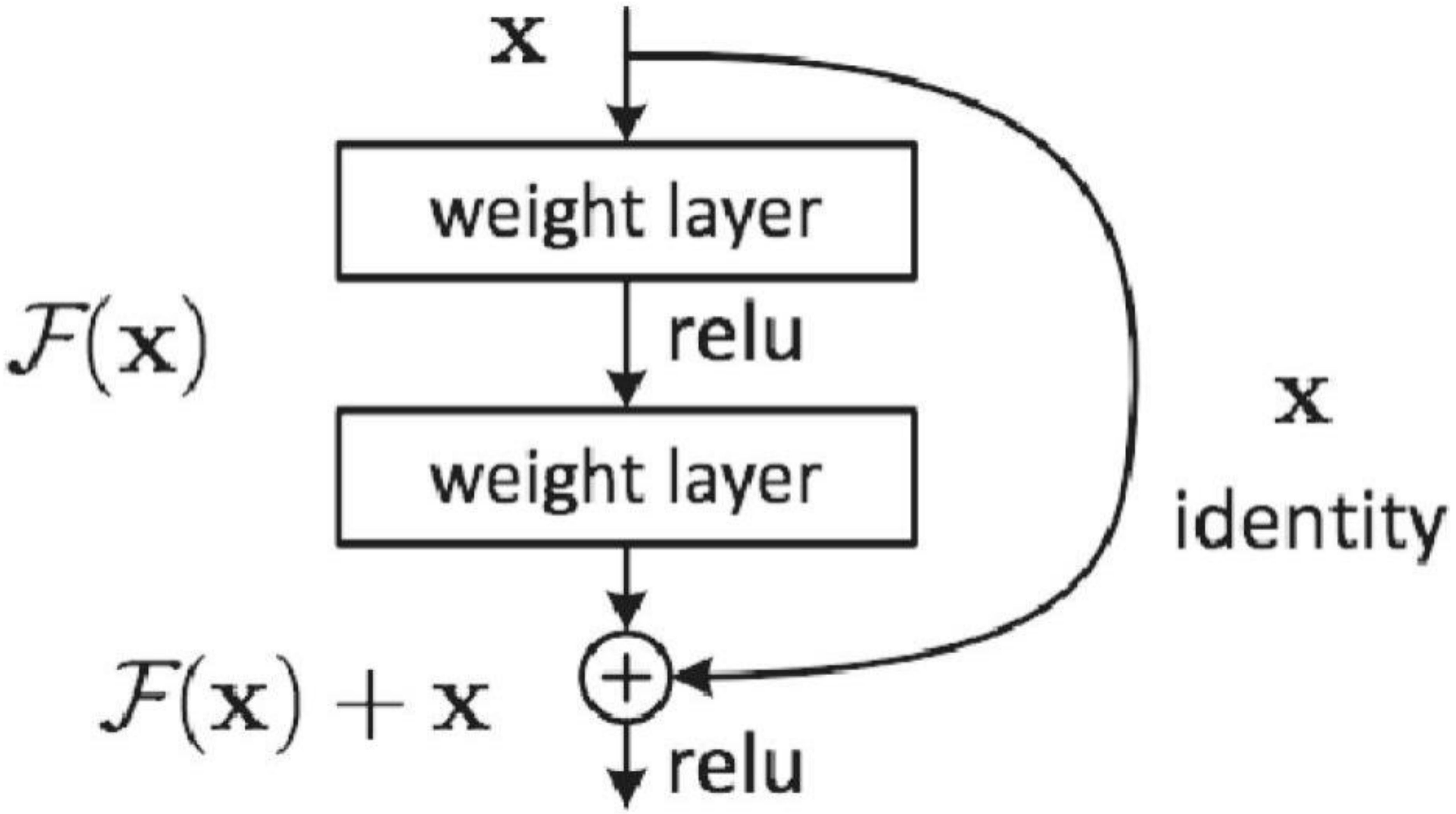
- 구성



GoogLeNet 의 인셉션 구조



ResNet(Residual Network)



ResNet(Residual Network)

- 마이크로소프트의 팀이 개발한 네트워크 .
- 스킵 연결 (skip connection) 을 도입
  - 층을 깊게 하는 것이 성능 향상에 중요하다는 것 알고 있으나 , 지나치게 깊으면 학습이 잘 되지 않고 , 오히려 성능이 떨어지는 경우도 많음 .
  - 이 문제를 해결하기 위해 스킵 연결을 도입해 층의 깊이에 비례해 성능을 향상시킬 수 있게 함 .
  - 입력 데이터를 합성곱 계층을 건너뛰어 출력에 바로 더하는 구조 .
  - 스킵 연결은 층이 깊어져도 학습을 효율적으로 할 수 있도록 해주는데 , 이는 역전파 때 스킵 연결이 신호 감쇠를 막아주기 때문 .