

# 밑바닥부터 시작하는 데이터 과학

O'REILLY®



O'REILLY  
ProgrammingInsight

조엘 그루스 지음  
백은정·김한결·화성주 옮김

## 6 회차

- 퍼셉트론
- 신경망 기초
- 신경망 학습
- 오차역전파
- 학습 관련 기술들





# 1. 퍼셉트론

---

## 퍼셉트론 (Perceptron)

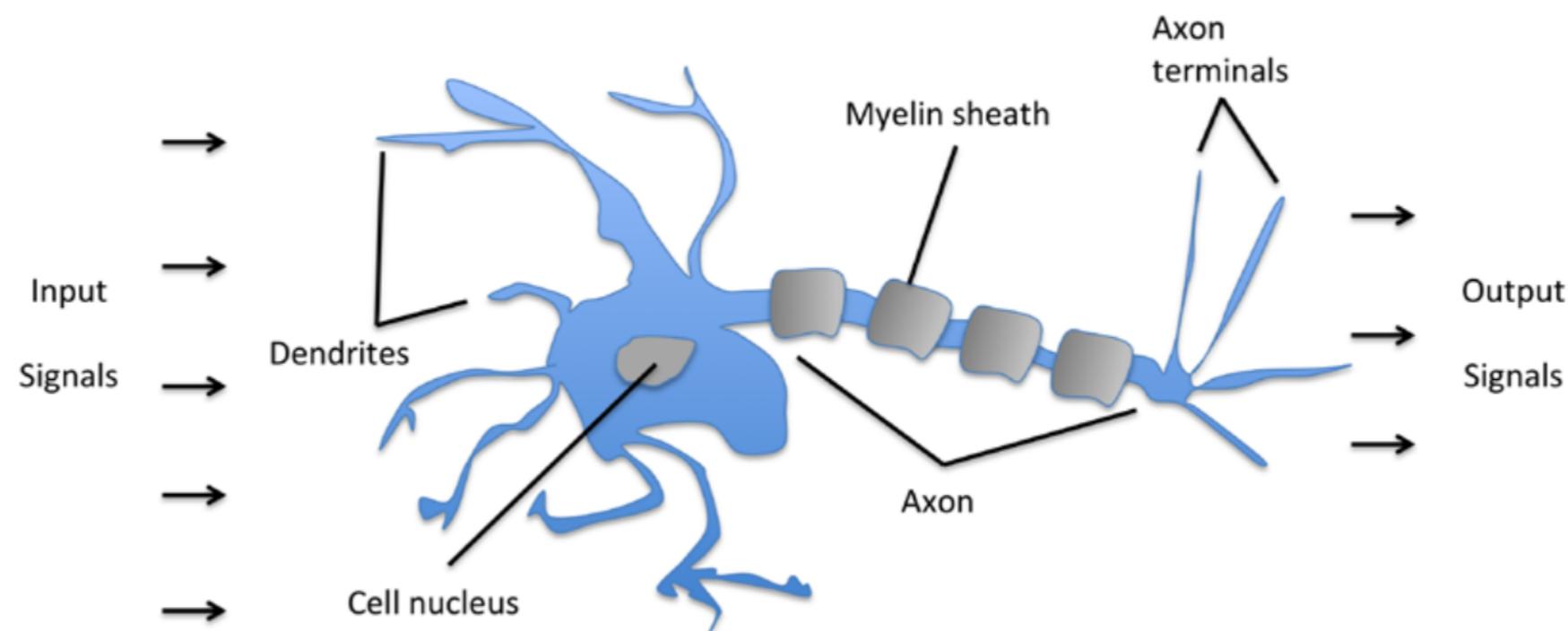


# 1. 퍼셉트론

## 퍼셉트론

### 퍼셉트론 (Perceptron)

- 동물의 신경(neuron) 구조에서 아이디어를 얻은 단순한 형태의 판별 함수 기반 예측 모형(discriminant function based prediction model) 중 하나이다.





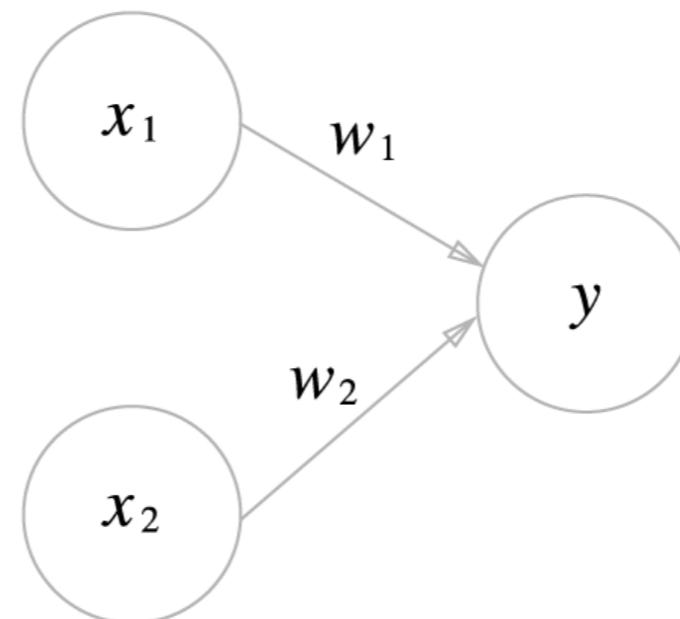
# 1. 퍼셉트론

## 퍼셉트론

### 퍼셉트론 (Perceptron)

- 입력이 2개인 퍼셉트론

- x: 입력
- y: 출력
- w: 가중치



- 뉴런에서 보내온 신호의 총합이 정해진 한계를 넘어설 때만 1을 출력한다.

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



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



# 1. 퍼셉트론

## 퍼셉트론 구현하기 (논리 회로)

### - AND 게이트

- 두 입력이 모두 1일 때만 1을 출력, 그 외에는 0을 출력
- 퍼셉트론을 이용하면 AND 게이트를 쉽게 구현할 수 있다.

$x_1$	$x_2$	$y$
0	0	0
1	0	0
0	1	0
1	1	1

### - NAND 게이트

- Not AND, AND 게이트의 출력을 뒤집은 것.

$x_1$	$x_2$	$y$
0	0	1
1	0	1
0	1	1
1	1	0

### - OR 게이트

- 입력 중 하나 이상이 1이면 1을 출력.

$x_1$	$x_2$	$y$
0	0	0
1	0	1
0	1	1
1	1	1

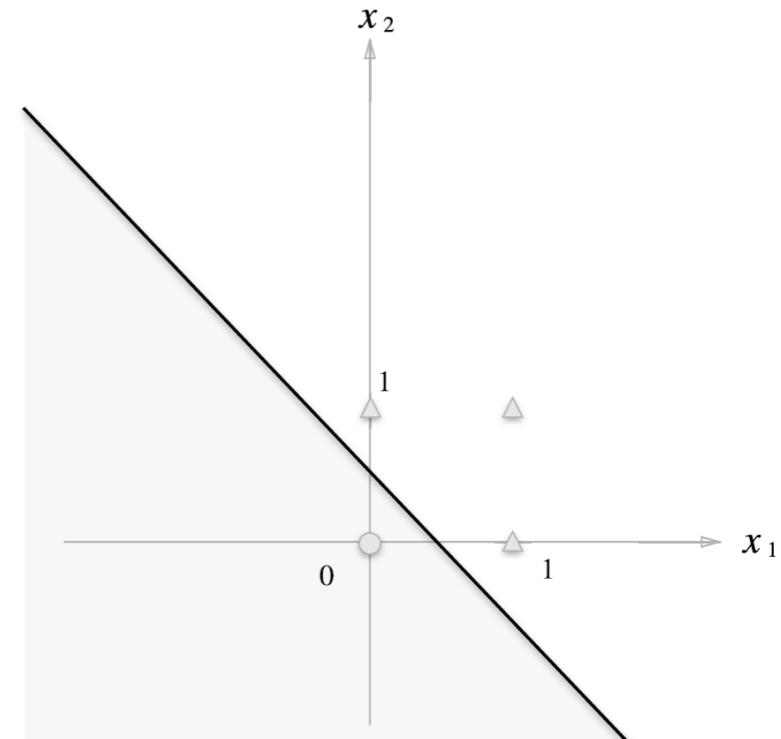


# 1. 퍼셉트론

## 퍼셉트론의 한계

### - OR 게이트 시각화

- 직선으로 원과 삼각형을 잘 나눈다.



### - XOR 게이트

- 베타적 논리합
- 둘 중 한쪽이 1일때만 1을 출력한다.

$x_1$	$x_2$	$y$
0	0	0
1	0	1
0	1	1
1	1	0

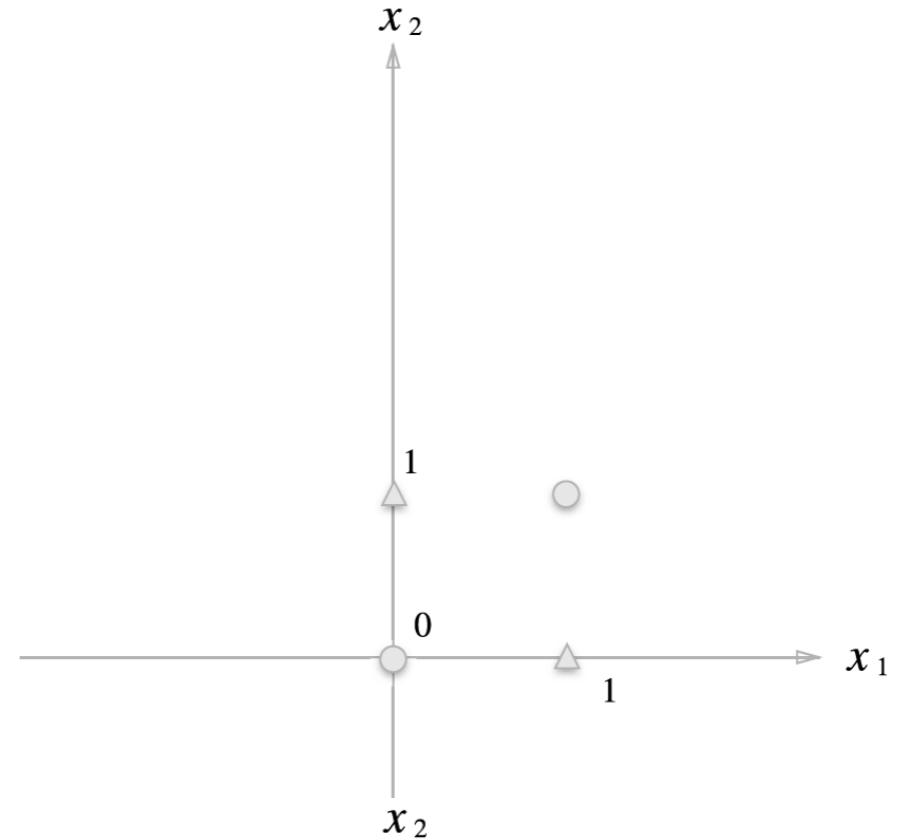


# 1. 퍼셉트론

## 퍼셉트론의 한계

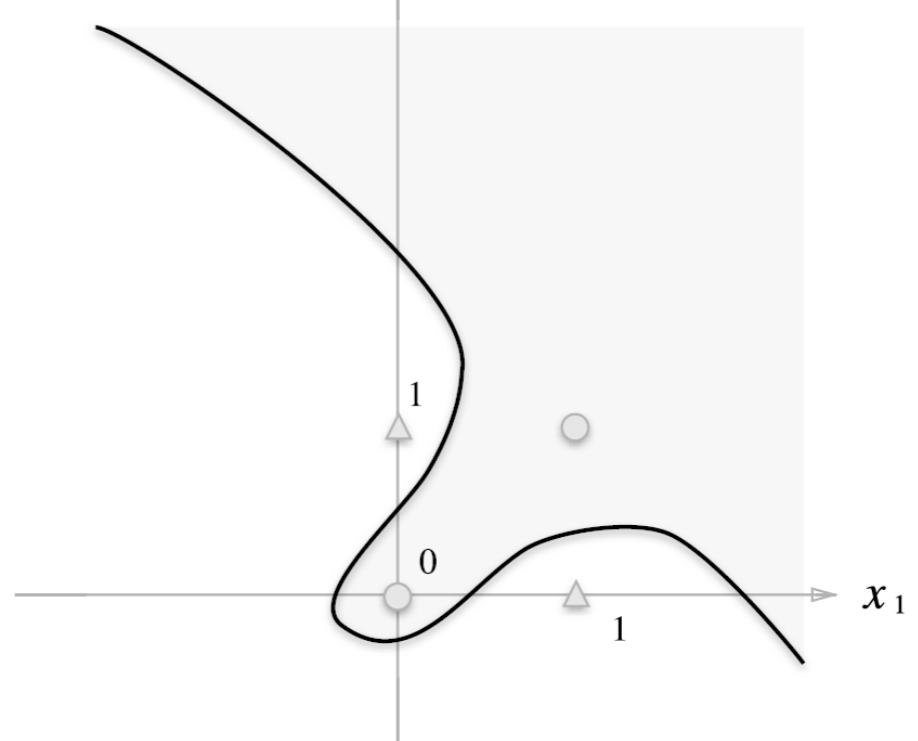
### - XOR 게이트 시각화

- 직선으로 원과 삼각형을 나누기는 불가능하다.



### - 선형과 비선형

- 직선이라는 제약을 없앤다면 XOR 문제도 해결가능하다.



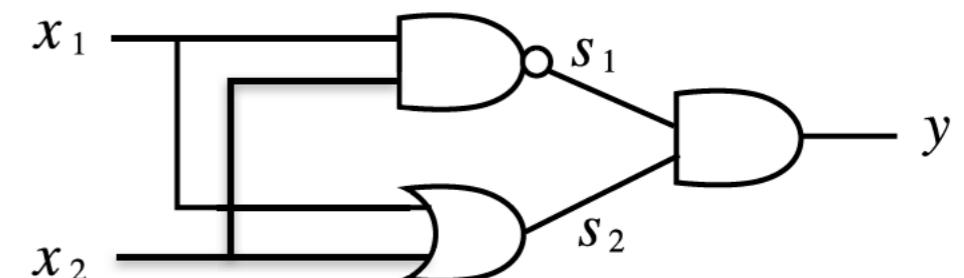
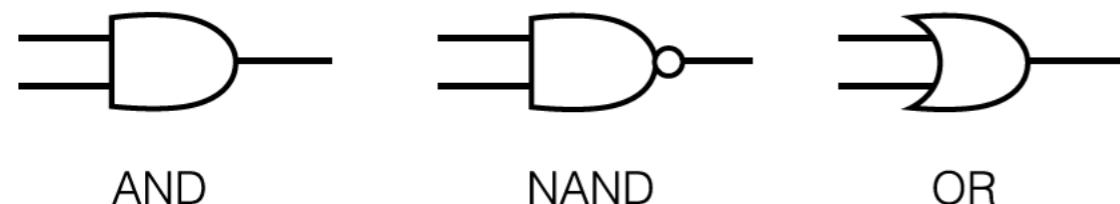


# 1. 퍼셉트론

## 다층 퍼셉트론

### - XOR 게이트 구현

- AND, NAND, OR 게이트 조합으로 XOR게이트 표현



### - XOR 게이트 결과

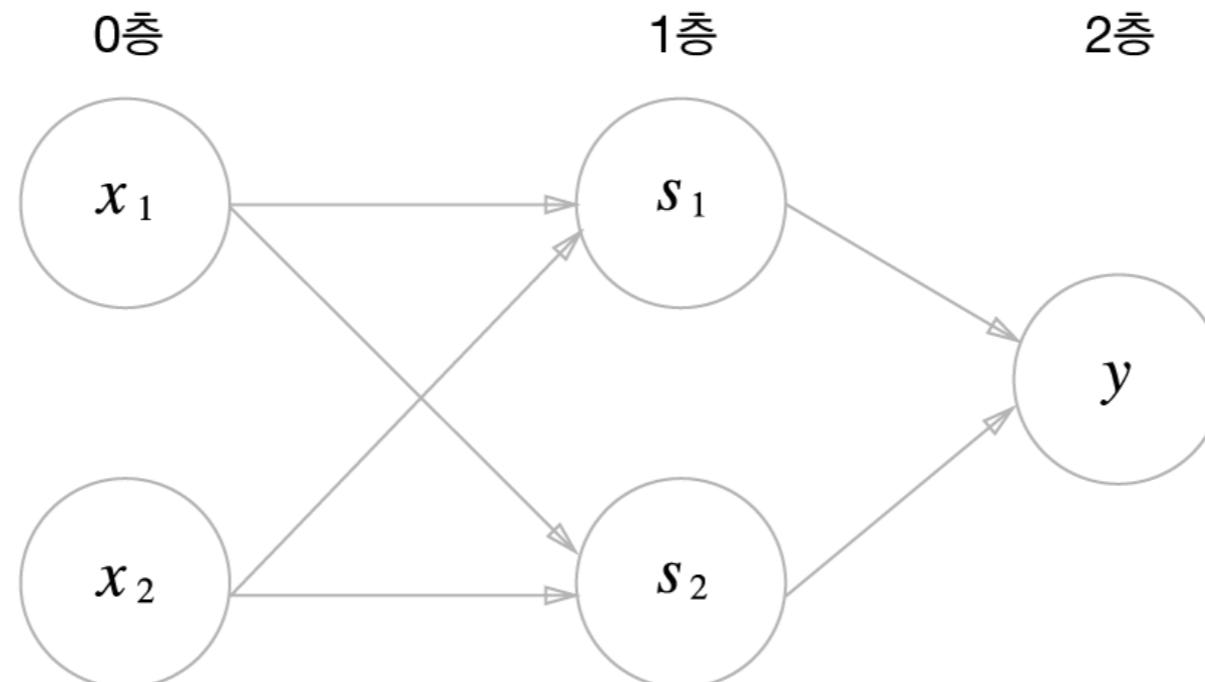
$x_1$	$x_2$	$s_1$	$s_2$	$y$
0	0	1	0	0
1	0	1	1	1
0	1	1	1	1
1	1	0	1	0



# 1. 퍼셉트론

## 다층 퍼셉트론

### - 다층 퍼셉트론 ( Multi Layer Perceptron )



1. 0층의 두 뉴런이 입력 신호를 받아 1층의 뉴런으로 신호를 보낸다.
2. 1층의 뉴런이 2층의 뉴런으로 신호를 보내고, 2층의 뉴런은 이 입력 신호를 바탕으로  $y$ 를 출력한다.

# 단층 퍼셉트론으로 표현하지 못하는 것을 층을 하나 더 늘려 표현할 수 있게 만들었다.  
# 퍼셉트론은 층을 쌓아 ( 더 깊게 하여 ) 다양한 것을 표현할 수 있다.



## 2. 신경망 기초

---

### 신경망 (Neural Network)

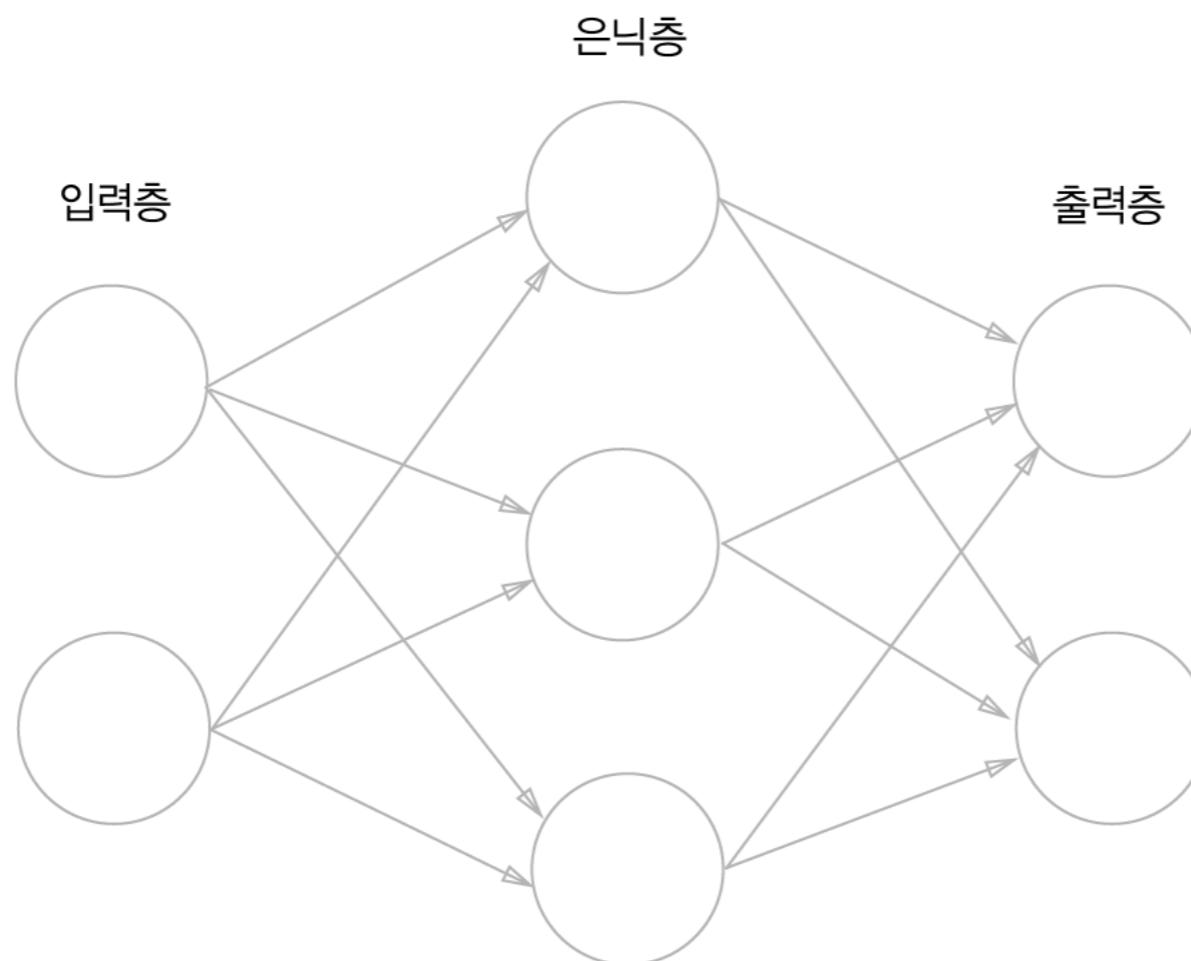


## 2. 신경망

### 신경망 (Neural Network)

#### 신경망의 예

- 다층 퍼셉트론을 신경망이라고 한다.
- 은닉층은 사람 눈에 보이지 않아서, '은닉'이라고 한다.





## 2. 신경망

### 활성화 함수

#### 활성화 함수 (Activation function)

- 입력 신호의 총 합을 출력신호로 변환하는 함수

#### 퍼셉트론의 활성화 함수 예

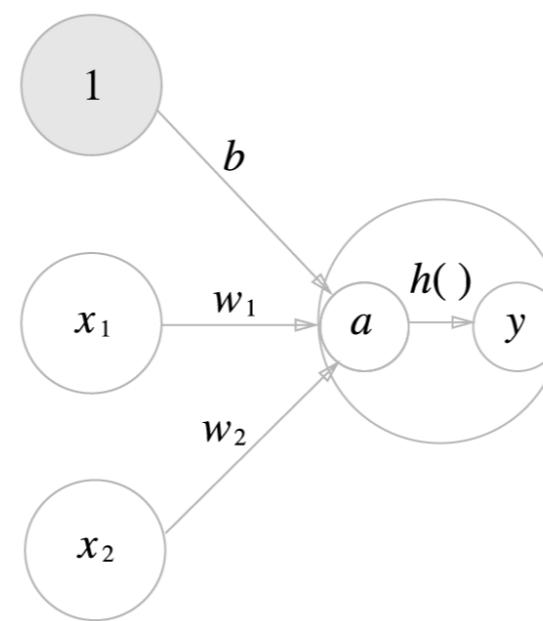
$$y = \begin{cases} 0 & (b + w_1x_1 + w_2x_2 \leq 0) \\ 1 & (b + w_1x_1 + w_2x_2 > 0) \end{cases}$$



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

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

#### 활성화 함수의 처리 과정





## 2. 신경망

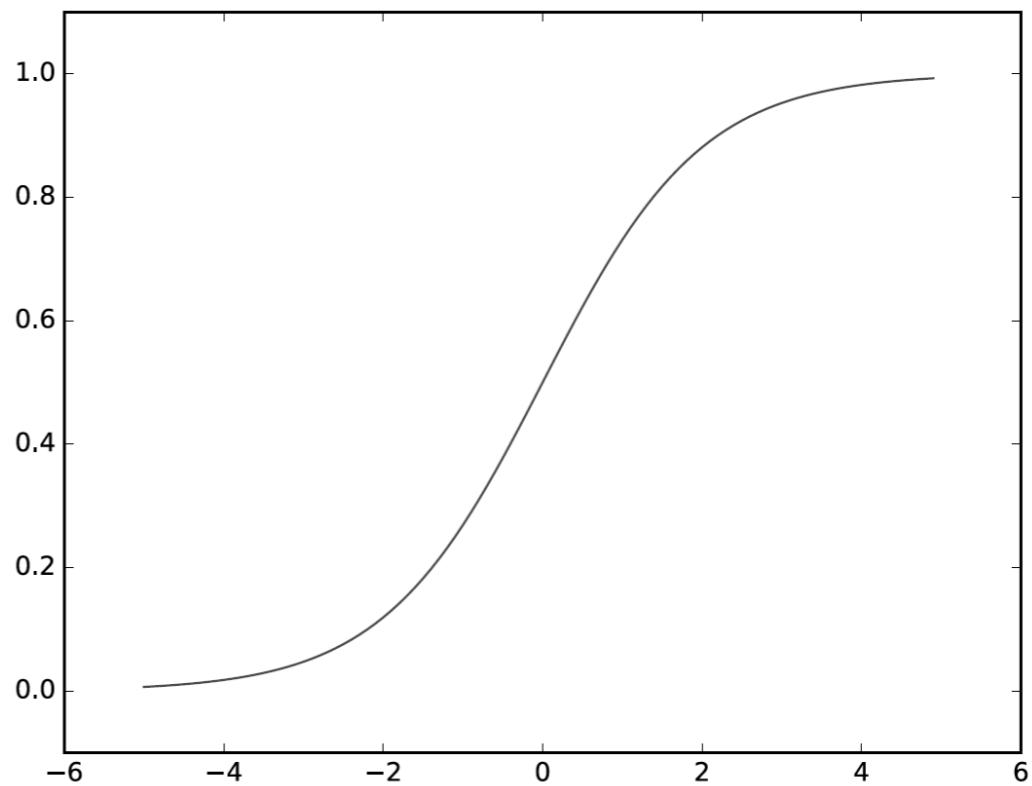
### 활성화 함수의 종류

계단 함수 ( Step function )

시그모이드 함수 ( Sigmoid function )

- 로지스틱 함수

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



# 활성화 함수로 계단 함수를 사용하면 각 뉴런 사이에서의 값이 1, 0 값만 흐르게 되지만, 시그모이드 함수를 사용하면 연속적인 실수가 흐르게 된다.



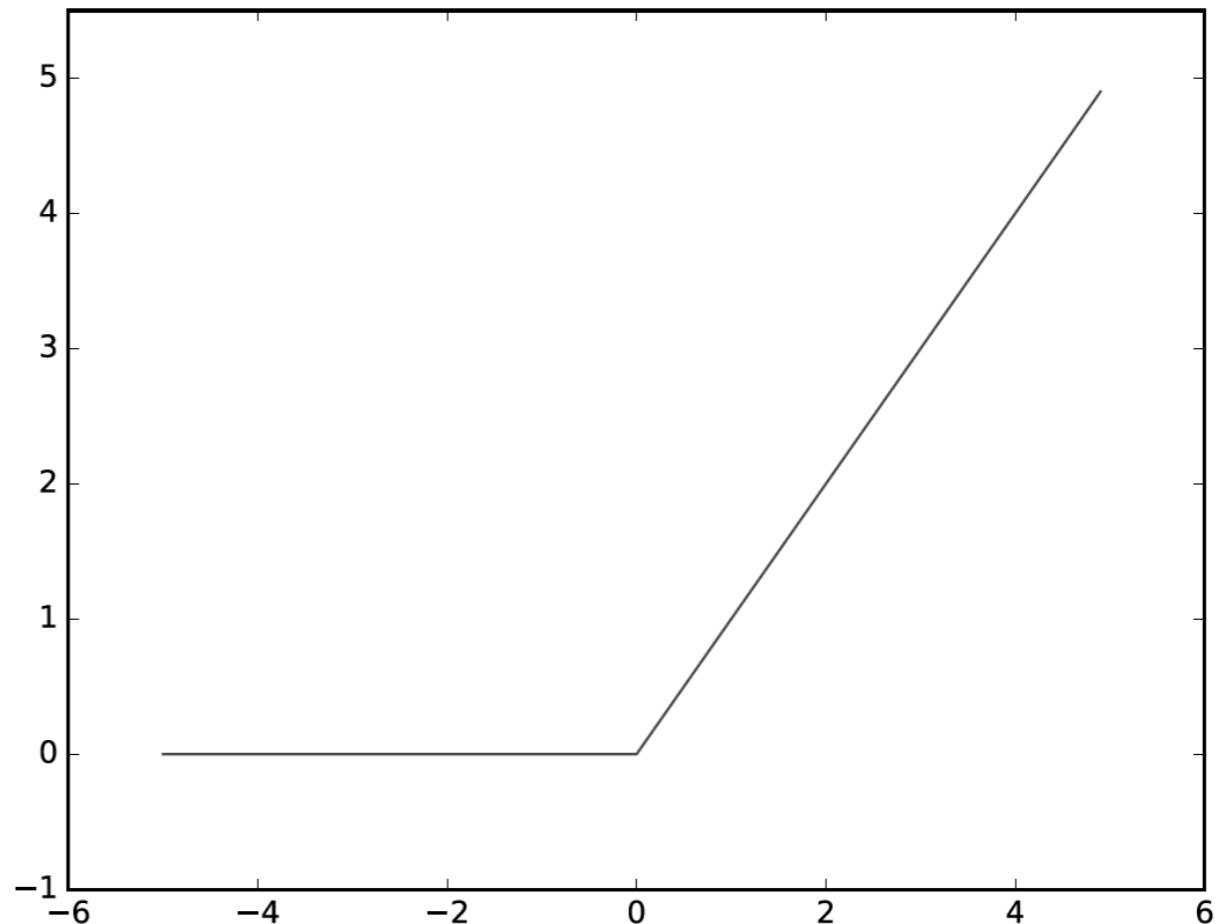
## 2. 신경망

### 활성화 함수의 종류

#### ReLU ( Rectified Linear Unit )

- 입력이 0을 넘으면 그 입력을 그대로 출력하고, 0 이하이면 0을 출력한다.

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





## 2. 신경망

### 활성화 함수의 종류

#### # 신경망에서 비선형 함수를 사용하는 이유

- 선형 함수의 문제는 층을 아주 깊게 해도 은닉층이 없는 네트워크로도 똑같은 기능을 할 수 있다는 데있다.
- $h(x) = cx$ 
  - 3층 네트워크의 경우
  - $y(x) = h(h(h(x)))$
  - $y(x) = c * c * c * x$
  - $y(x) = ax$

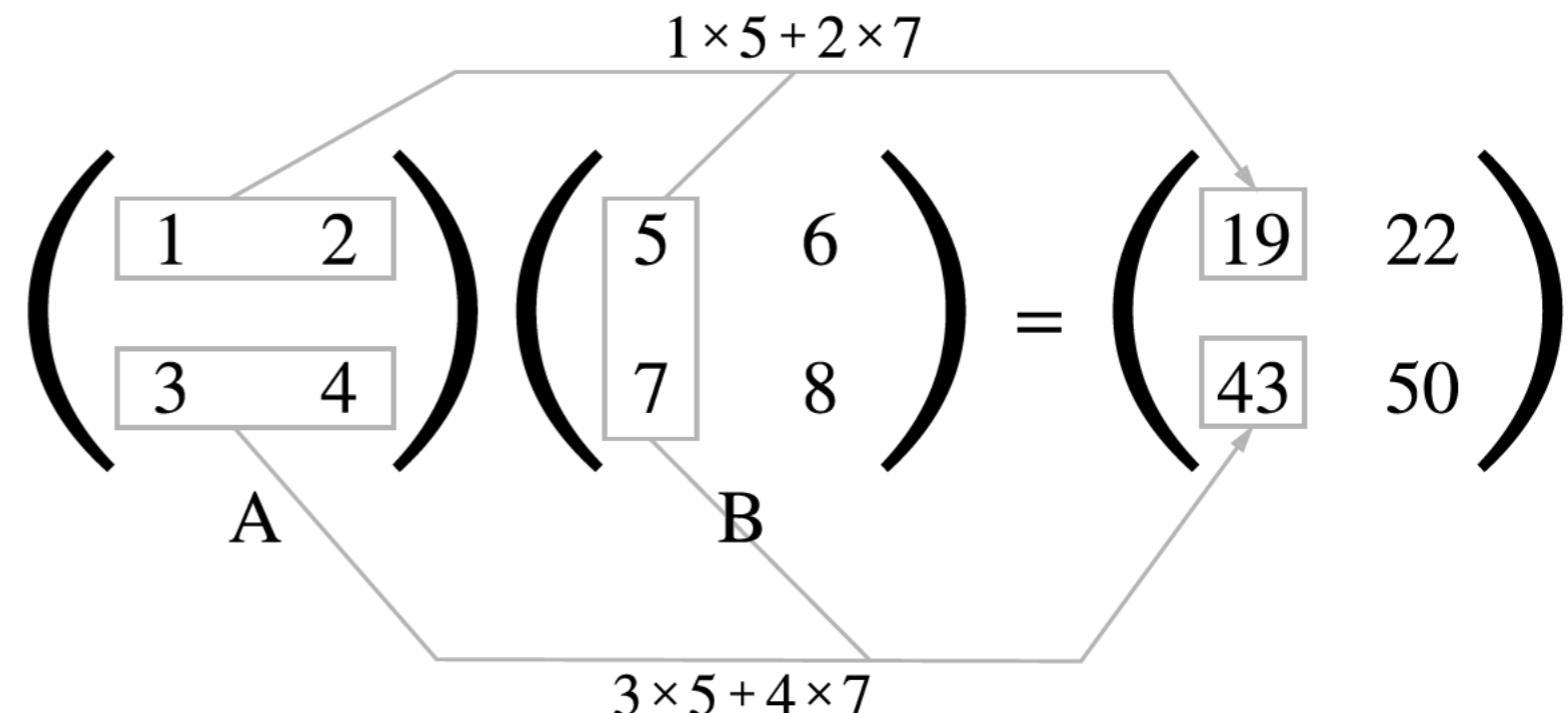
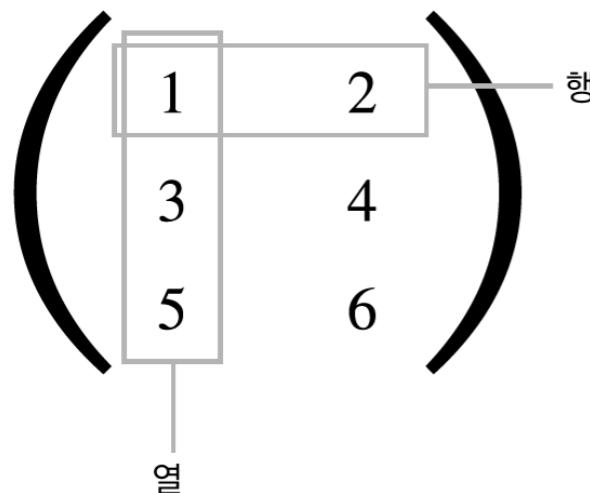
선형 함수를 이용하면 여러 층을 구성하는 이점을 전혀 살릴 수 없으므로, 사용하지 않는다.



## 2. 신경망

### 다차원 배열 계산

#### 행렬 곱



A

B

= C

형상  $3 \times 2$  $2 \times 4$  $3 \times 4$ 

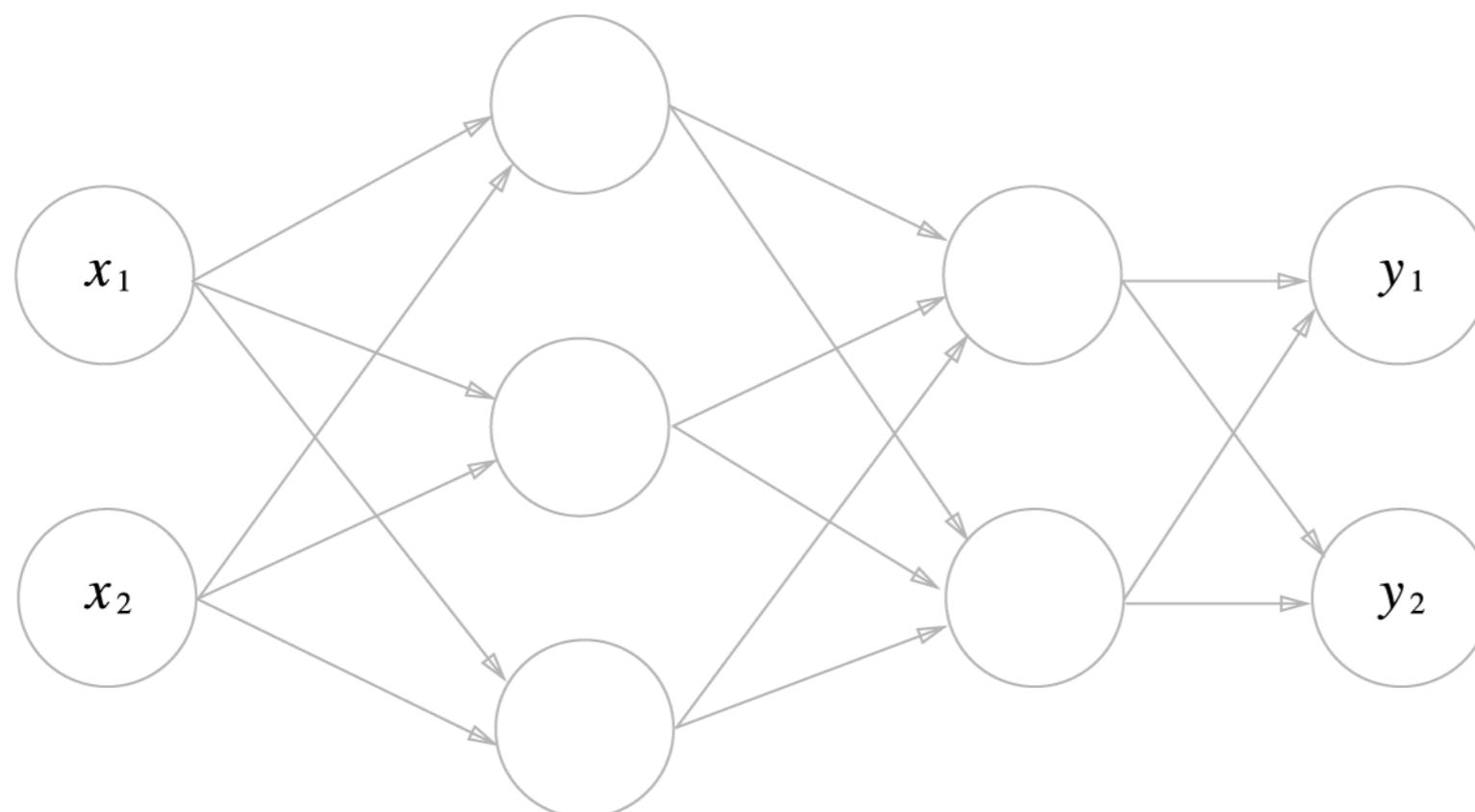


## 2. 신경망

### 3층 신경망 구현하기

#### 3층 신경망

- 입력층 1, 은닉층 2, 출력층 1

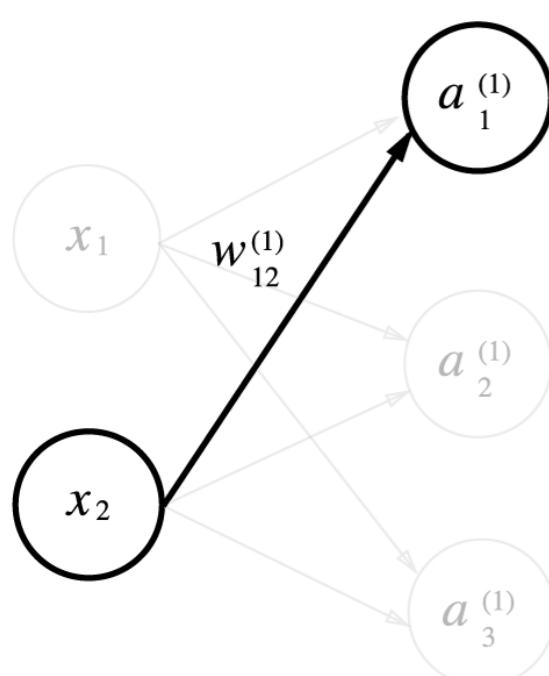




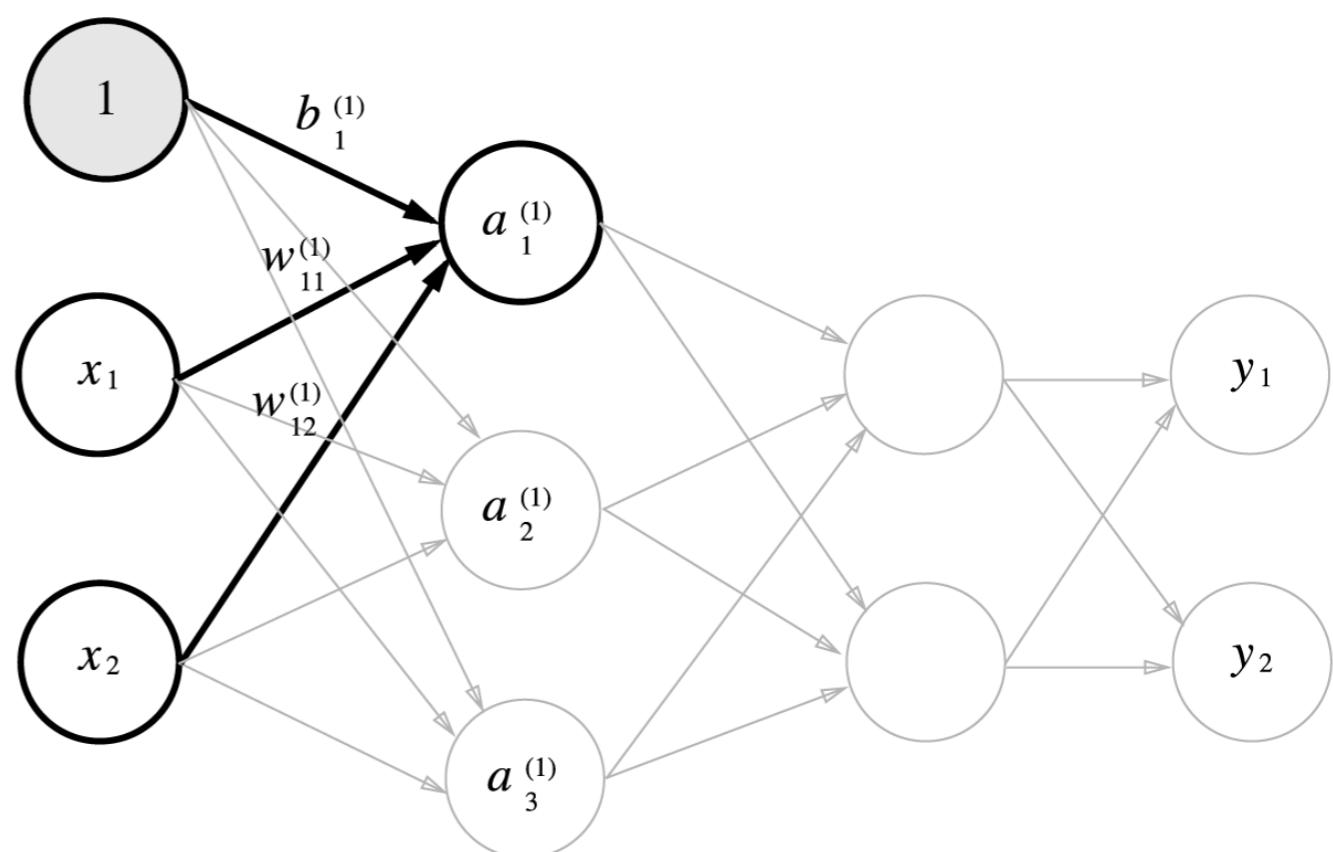
## 2. 신경망

### 3층 신경망 구현하기

#### 각 층의 신호 전달



$W$   
1 (1) 2  
1층의 가중치  
앞 층의 2번째 뉴런  
다음 층의 1번째 뉴런





## 2. 신경망

### 3층 신경망 구현하기

#### 각 층의 신호 전달

- 은닉층 (1층)의 첫 번째 뉴런

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

- 1층 전체를 일반화 시키면,

$$A^{(1)} = (a_1^{(1)}, a_2^{(1)}, a_3^{(1)}), \quad B^{(1)} = (b_1^{(1)}, b_2^{(1)}, b_3^{(1)})$$

$$X = (x_1, x_2)$$

$$W^{(1)} = \begin{pmatrix} w_{11}^{(1)} & w_{21}^{(1)} & w_{31}^{(1)} \\ w_{12}^{(1)} & w_{22}^{(1)} & w_{32}^{(1)} \end{pmatrix}$$

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

- 행렬 곱을 통해 각 층의 신호들을 흐르게 만들 수 있다.

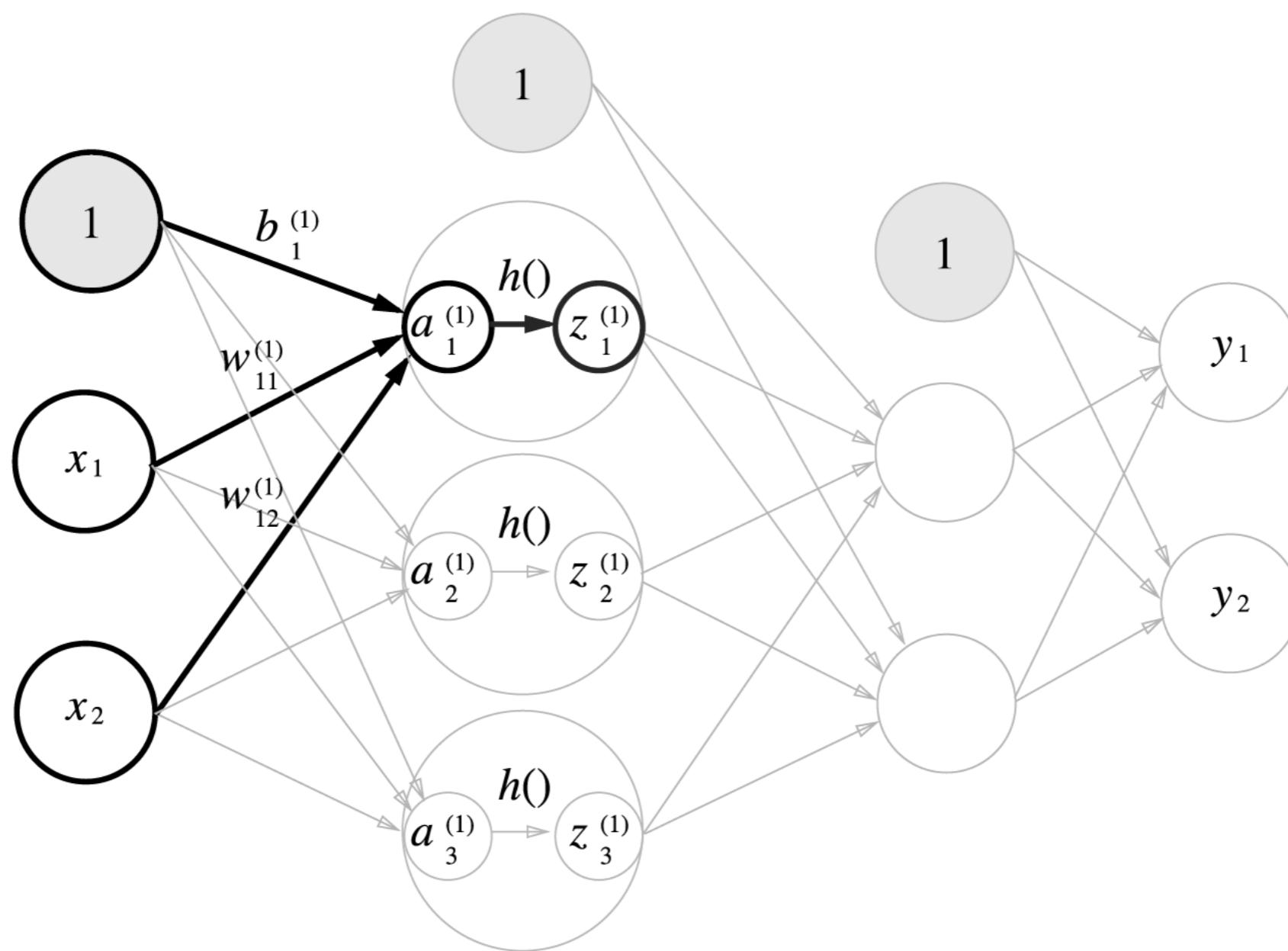


## 2. 신경망

### 3층 신경망 구현하기

#### 각 층의 신호 전달

- 입력층에서 1층으로 전달



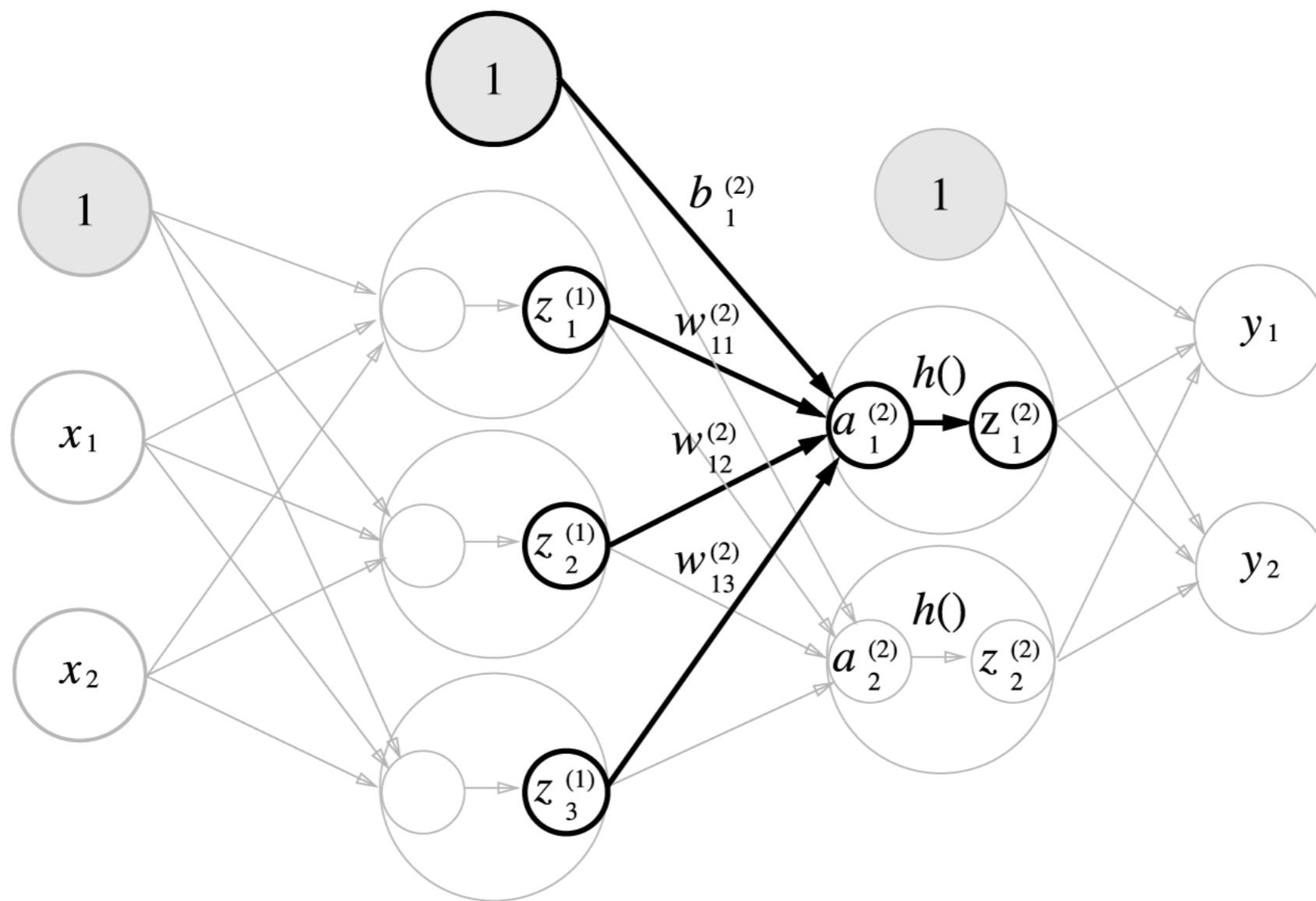


## 2. 신경망

### 3층 신경망 구현하기

#### 각 층의 신호 전달

- 1층에서 2층으로 전달



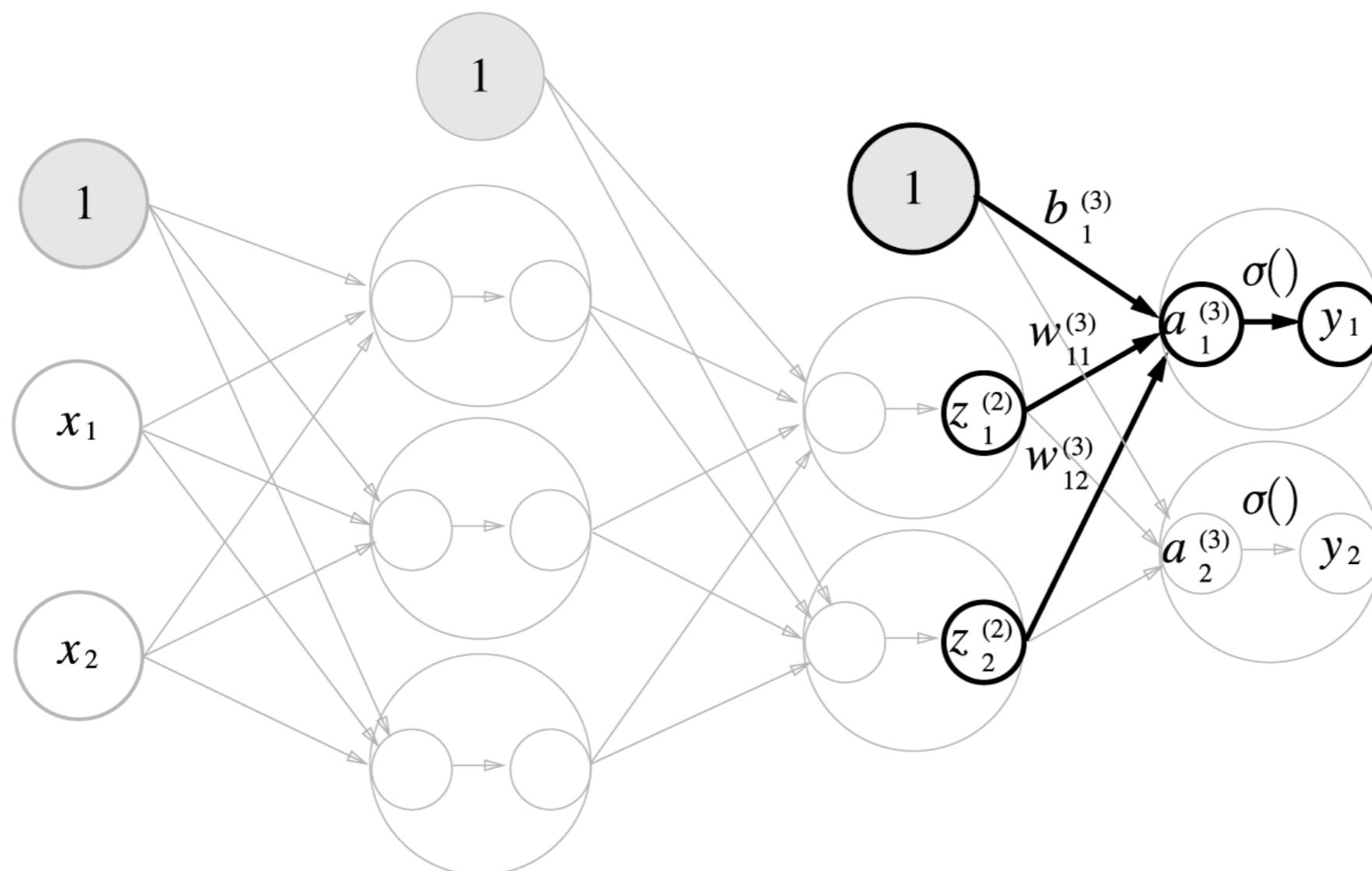


## 2. 신경망

### 3층 신경망 구현하기

#### 각 층의 신호 전달

- 2층에서 출력층으로 전달





## 2. 신경망

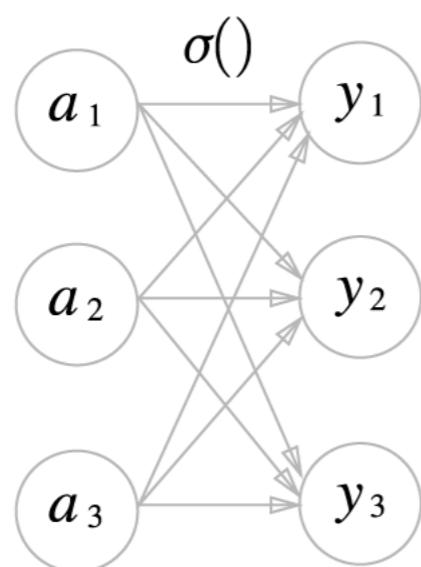
### 3층 신경망 구현하기

#### 출력층 설계

- 소프트 맥스 함수

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

- n: 출력층의 뉴런 수,  $y_k$ : k 번째 출력



- 출력층은 모든 입력 신호로부터 화살표를 받는다.



## 2. 신경망

### 3층 신경망 구현하기

#### 소프트맥스 함수의 특징

- 소프트맥스 출력의 총합은 1이다.
  - 이 성질 덕분에 소프트맥스 함수의 출력을 **확률**로 해석할 수 있다.
    - ex. 74% 확률로 2번째 클래스, 25% 확률로 1번째 클래스, 1% 확률로 0번째 클래스

#### 소프트맥스 함수 구현시 주의점

- 지수 함수를 사용하므로 엄청나게 큰 값이 나올 가능성이 높다.
  - 큰 값이 나오면 오버플로 (overflow) 문제가 발생함.
  - 입력값을 작은 값으로 만들어서 큰 값이 나오지 못하도록 한다.

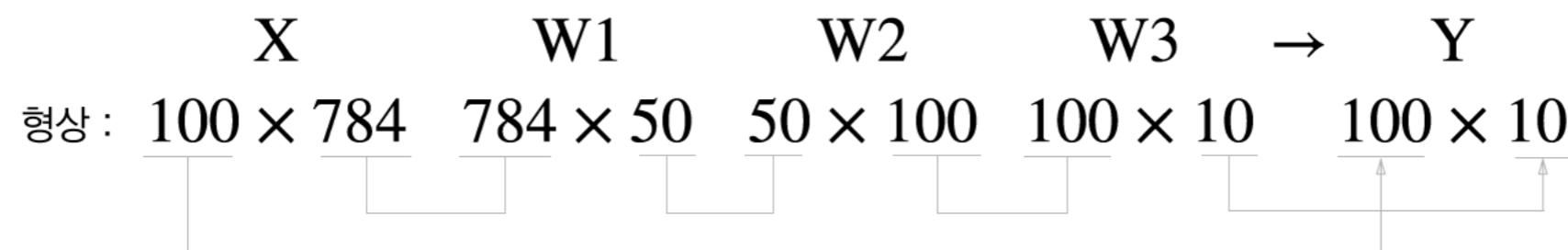
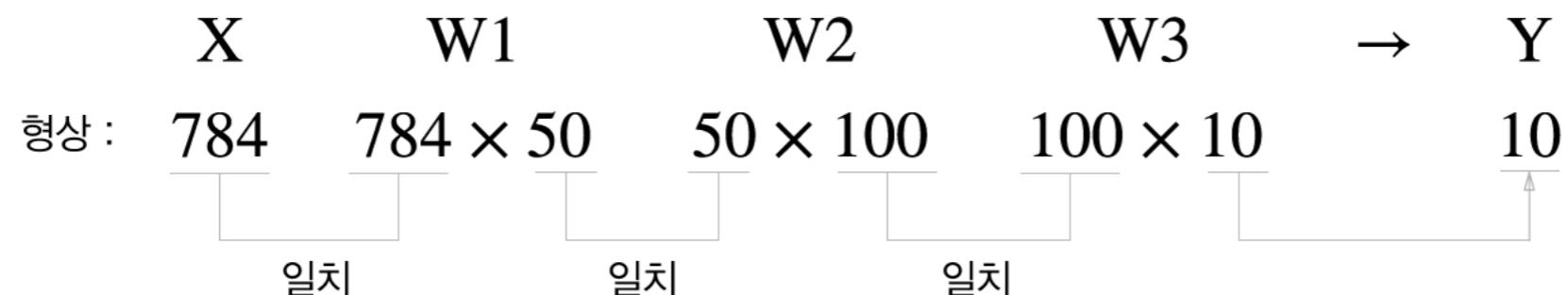
$$\begin{aligned} y_k &= \frac{\exp(a_k)}{\sum_{i=1}^n \exp(a_i)} = \frac{C \exp(a_k)}{C \sum_{i=1}^n \exp(a_i)} \\ &= \frac{\exp(a_k + \log C)}{\sum_{i=1}^n \exp(a_i + \log C)} \\ &= \frac{\exp(a_k + C')}{\sum_{i=1}^n \exp(a_i + C')} \end{aligned}$$

## 2. 신경망

## 3층 신경망 구현하기

## 배치 처리 (batch)

- 한줄씩 처리하는 것 보다 데이터를 읽는 시간이 줄어들어 빨라진다.
  - 배치사이즈 (batch size) 만큼 결과를 한번에 출력한다.





### 3. 신경망 학습

---

신경망 학습



### 3. 신경망 학습

#### 데이터 주도 학습

##### | 데이터 주도 학습

###### - 기존 문제 해결 방식

- 사람의 경험과 직관을 단서로 시행착오를 거듭하며 문제 해결

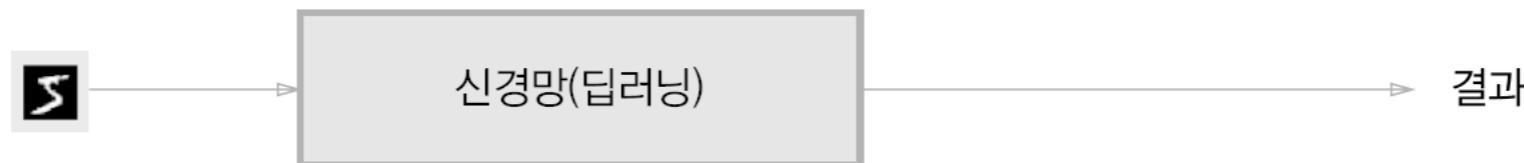
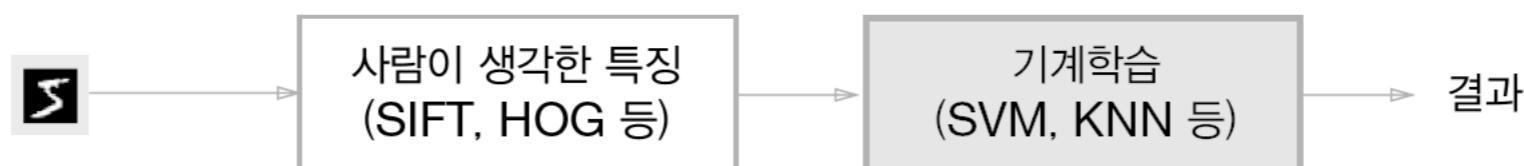
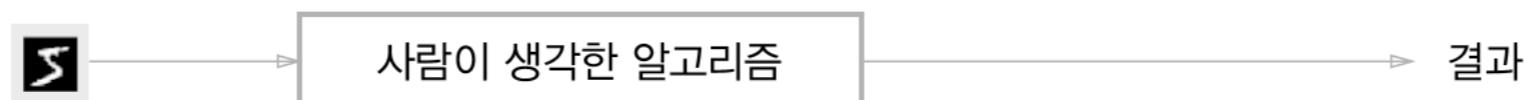
###### - 기계학습

- 사람의 개입을 최소화하고 수집한 데이터로부터 패턴을 찾으려 시도

###### - 딥러닝

- 사람의 개입을 (거의) 없애고 데이터에만 의존

- 이러한 특징 때문에 end-to-end machine learning이라고도 한다.





### 3. 신경망 학습

#### 손실함수 (Loss function)

##### 평균 제곱 오차 (Mean squared error, MSE)

$$E = \frac{1}{2} \sum_k (\underline{y}_k - \underline{t}_k)^2$$

##### 교차 엔트로피 오차 (cross entropy error, CEE)

- 정답인 값의 출력값  $y$ 가 1에 가까울수록 0에 가까워진다.

$$E = - \sum_k \underline{t}_k \log \underline{y}_k$$

# 분류 문제에서는 교차엔트로피오차를 사용하는 것이 더 좋다.



### 3. 신경망 학습

#### 미니배치 학습

##### 미니배치 학습

- 기계학습 알고리즘들은 훈련 데이터에 대한 손실 함수의 값을 구하고, 그 값을 최대한 줄여주는  $w$  (매개변수)를 구하게 된다.
- 보통 손실 함수의 값은 아래와 같이 전체 데이터를 다 사용해서 계산하게 되는데,

$$E = -\frac{1}{N} \sum_n \sum_k t_{nk} \log y_{nk}$$

- 데이터가 굉장히 많을 경우에 한 번의 손실함수 값을 구하는데 모든 데이터를 하게 된다.
- 손실함수 값을 구하기 위해 전체 데이터 중 **무작위로 추출된 일부 데이터**만 사용해도 모든 데이터를 사용한 손실함수 값과 큰 편차가 생기지 않는다. ( 중심극한정리 ! )
- 따라서, 일부 데이터만 무작위로 선택해 학습하고 그 손실함수 값을 최소화 시키기 위한  $w$ 를 구하는 과정을 거쳐도 문제가 없다.
- 이러한 학습 방법을 **미니배치 학습**이라고 한다.



### 3. 신경망 학습

#### 최적화

##### 수치 미분

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

##### 편미분

#### 기울기 (gradient)

- 모든 변수의 편미분 벡터

#### 경사 하강법 (gradient descent)

- 기울어진 방향으로 일정 거리만큼 이동

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

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



# 3. 신경망 학습

## 학습 알고리즘 구현

### 알고리즘 전체 과정

- 신경망에는 적응 가능한 가중치와 편향이 있고, 이 가중치와 편향을 훈련 데이터에 적응하도록 조정하는 과정을 ‘학습’이라고 한다.

#### 1 단계 - 미니배치

- 훈련 데이터 중 일부를 무작위로 가져온다. 이렇게 선별한 데이터를 미니배치라고 하며, 그 미니배치의 손실 함수 값을 줄이는 것이 목표다.

#### 2 단계 - 기울기 산출

- 미니배치의 손실 함수 값을 줄이기 위해 각 가중치 매개변수의 기울기를 구한다. 기울기는 손실 함수의 값을 가장 작게하는 방향을 제시한다.

#### 3 단계 - 가중치 갱신

- 가중치를 기울기 방향으로 아주 조금 갱신한다.

#### 4 단계 - 반복

- 1 ~ 3단계를 반복한다.



## 4. 오차역전파

---

오차역전파 (backpropagation)

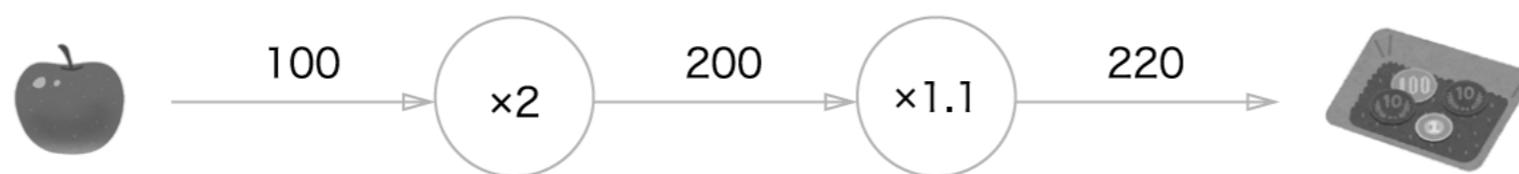


## 4. 오차역전파

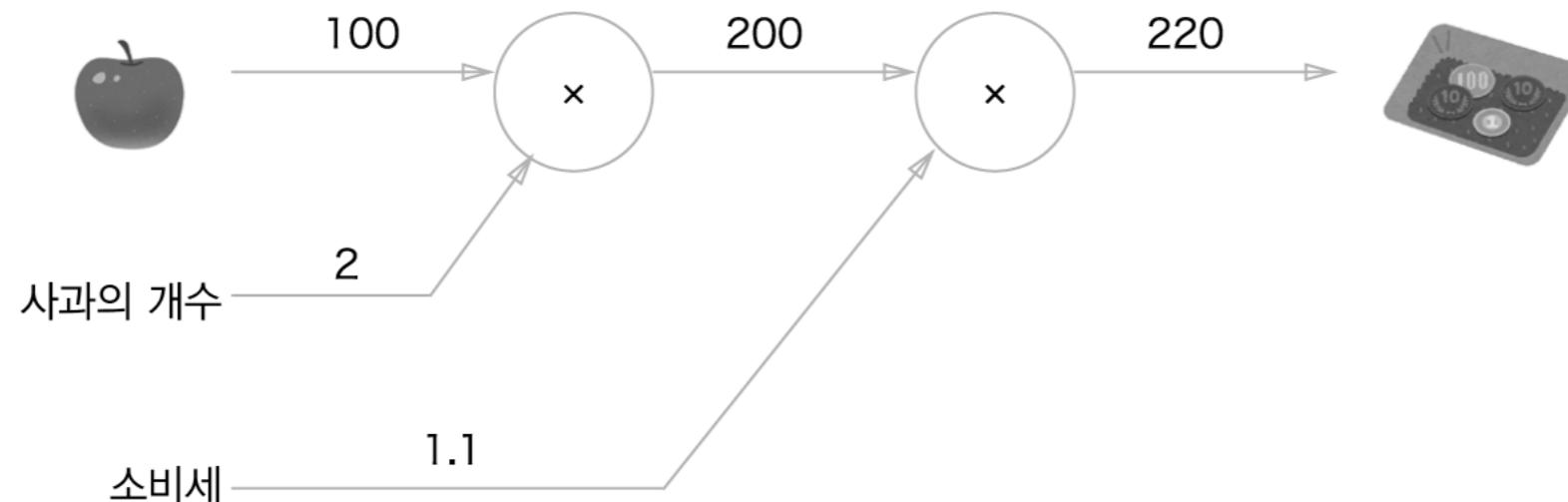
### 오차역전파 (backpropagation)

#### 계산 그래프

- 오차역전파를 잘 이해하기 위해 계산그래프 방식으로 접근해보자
  - (1) 사과 1개 100원, 소비세 10%, 사과 1개 구매시, 지불 금액은?



- (2) 사과 1개 100원, 소비세 10%, 사과 2개 구매시, 지불 금액은?



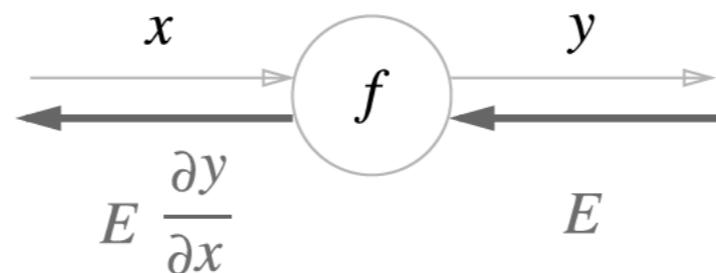


## 4. 오차역전파

### 오차역전파 (backpropagation)

#### 계산 그래프의 역전파

- $y = f(x)$  의 역전파



#### 연쇄법칙

- 합성함수의 미분법

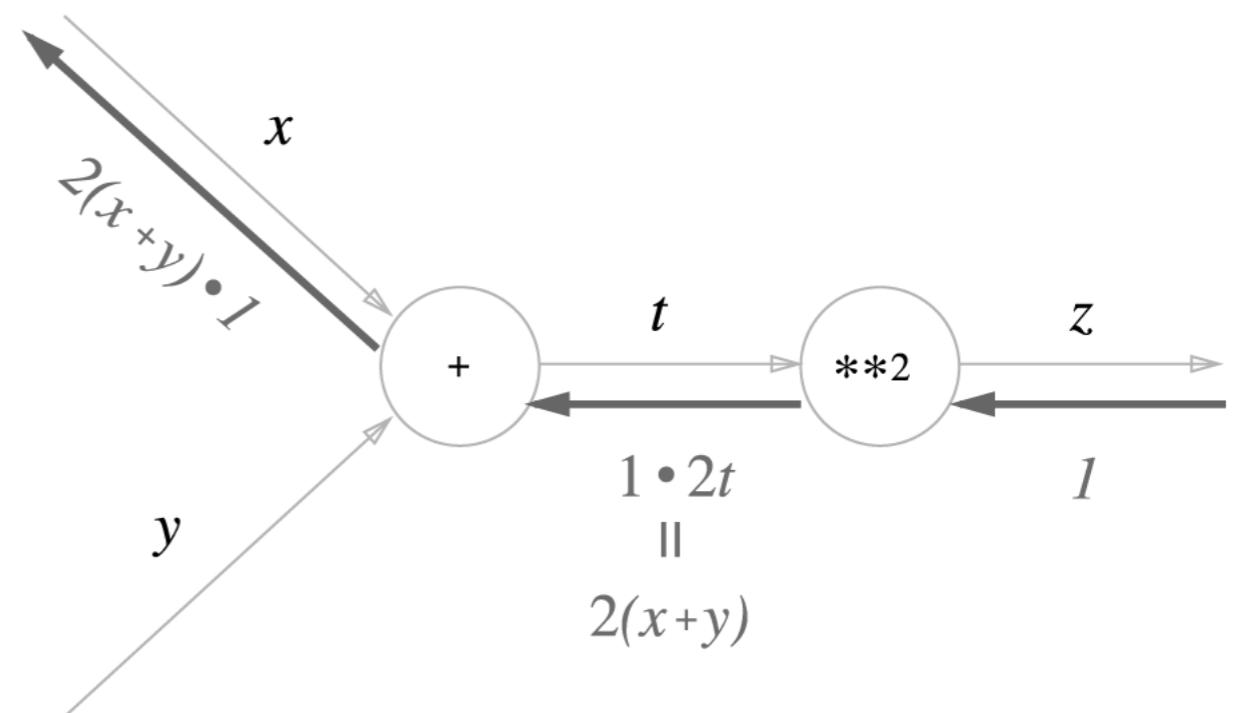
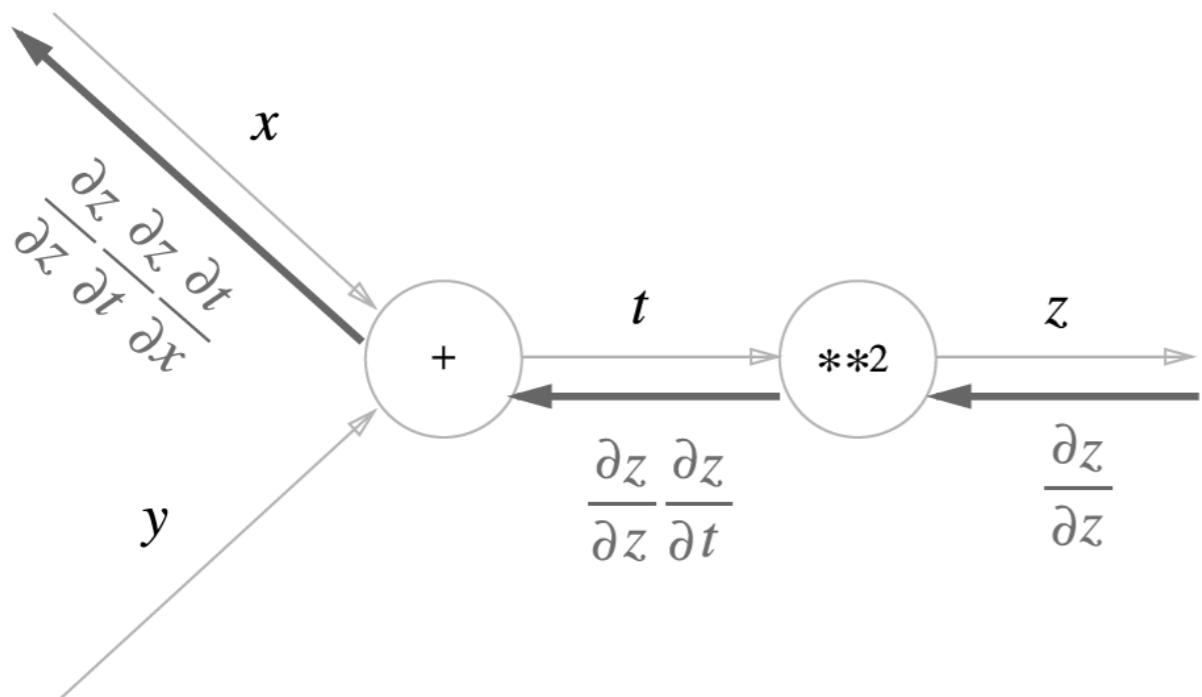
$$\begin{aligned} z &= t^2 \\ t &= x + y \end{aligned} \quad \rightarrow \quad \frac{\partial z}{\partial x} = \frac{\partial z}{\partial t} \frac{\partial t}{\partial x} \quad \rightarrow \quad \begin{aligned} \frac{\partial z}{\partial t} &= 2t \\ \frac{\partial t}{\partial x} &= 1 \end{aligned} \quad \rightarrow \quad \frac{\partial z}{\partial x} = \frac{\partial z}{\partial t} \frac{\partial t}{\partial x} = 2t \cdot 1 = 2(x + y)$$



# 4. 오차역전파

## 오차역전파 (backpropagation)

### 연쇄법칙과 계산 그래프



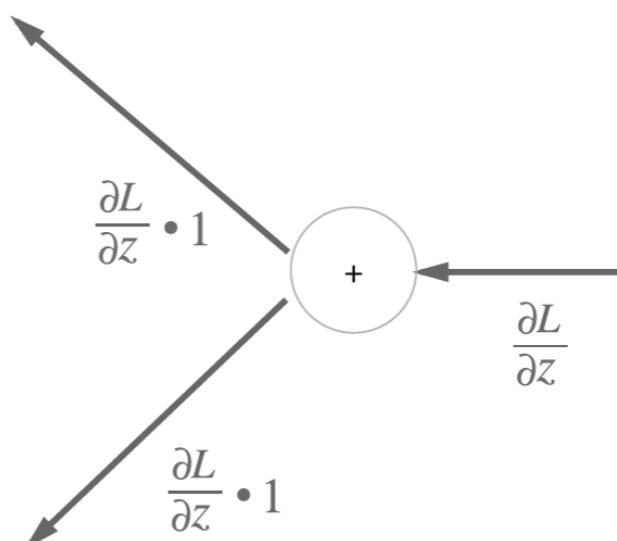
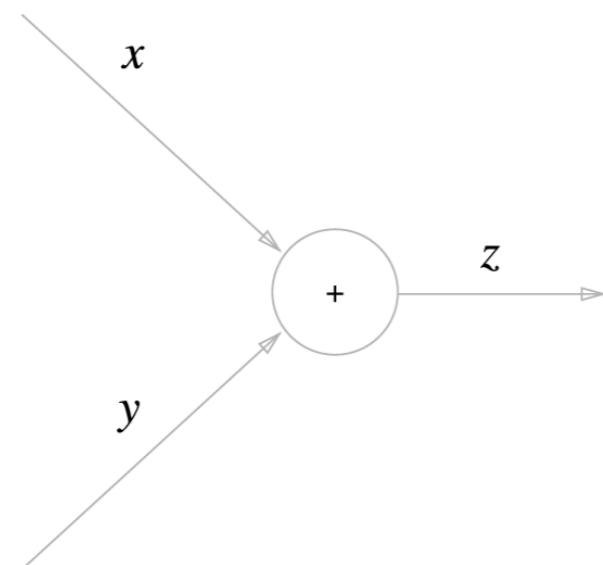


## 4. 오차역전파

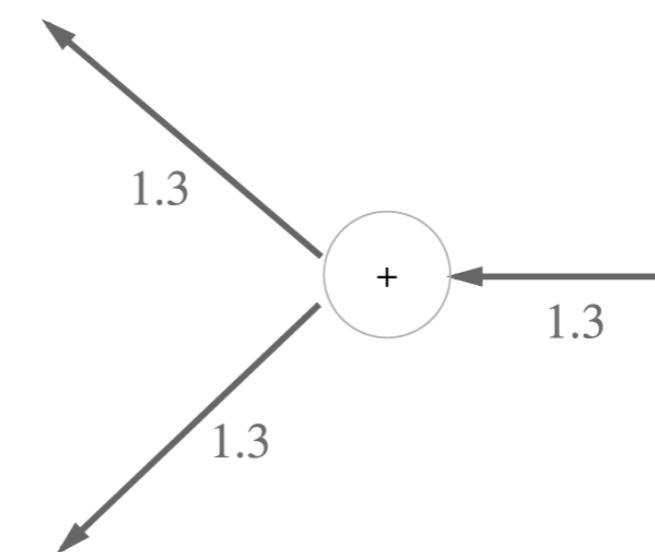
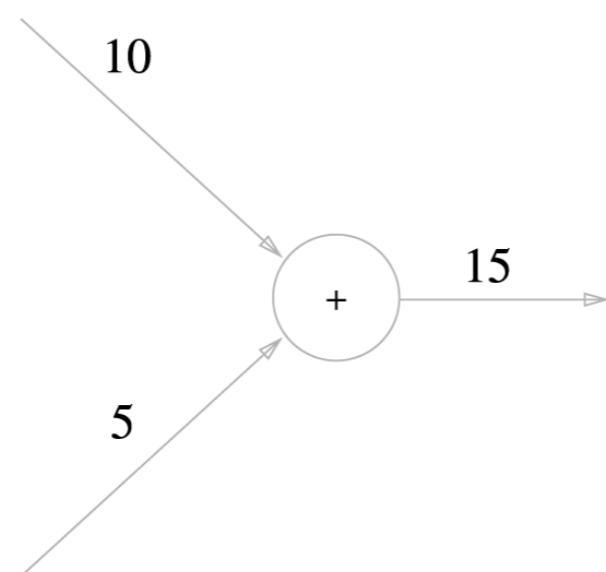
### 오차역전파 (backpropagation)

#### 역전파

- 덧셈 노드



- 예시



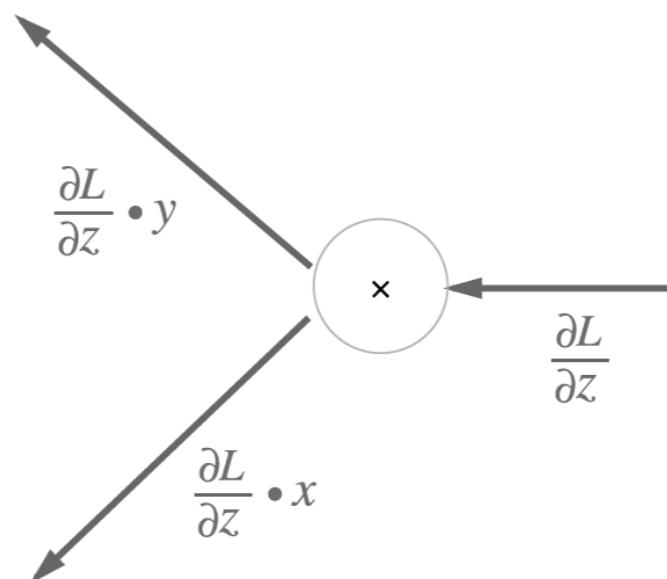
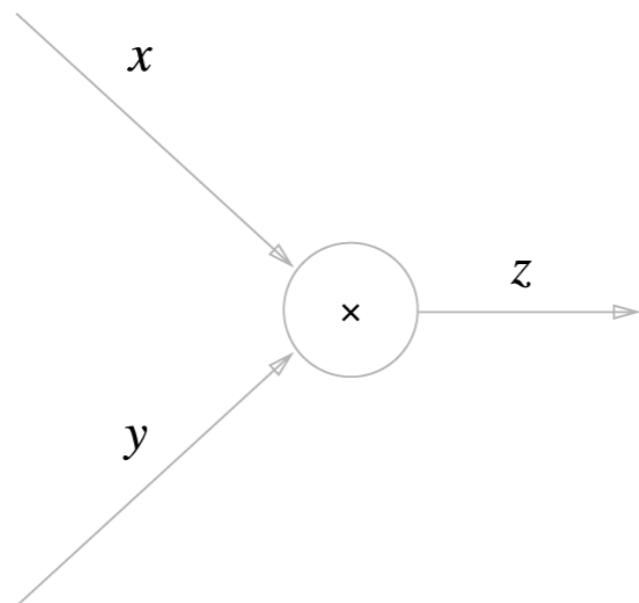


## 4. 오차역전파

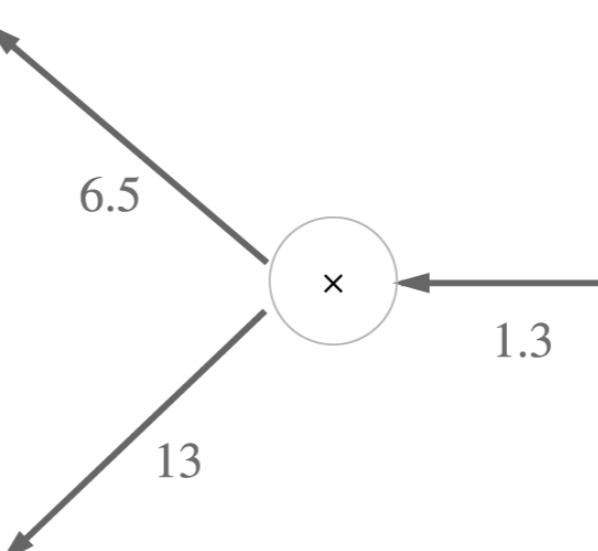
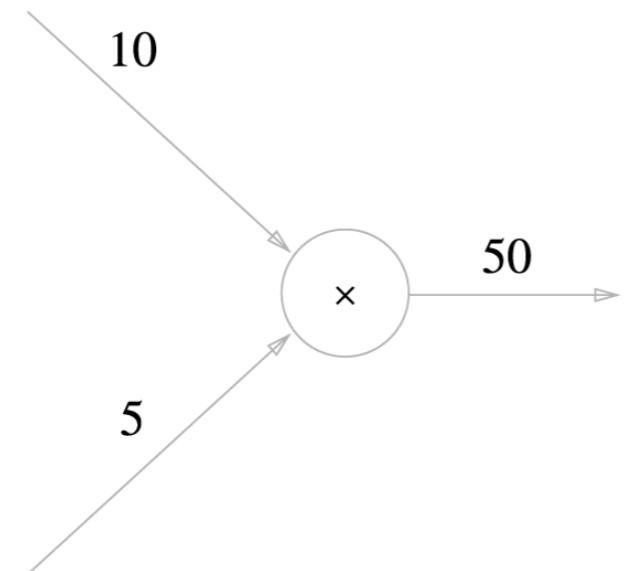
### 오차역전파 (backpropagation)

#### 역전파

- 곱셈 노드



- 예시



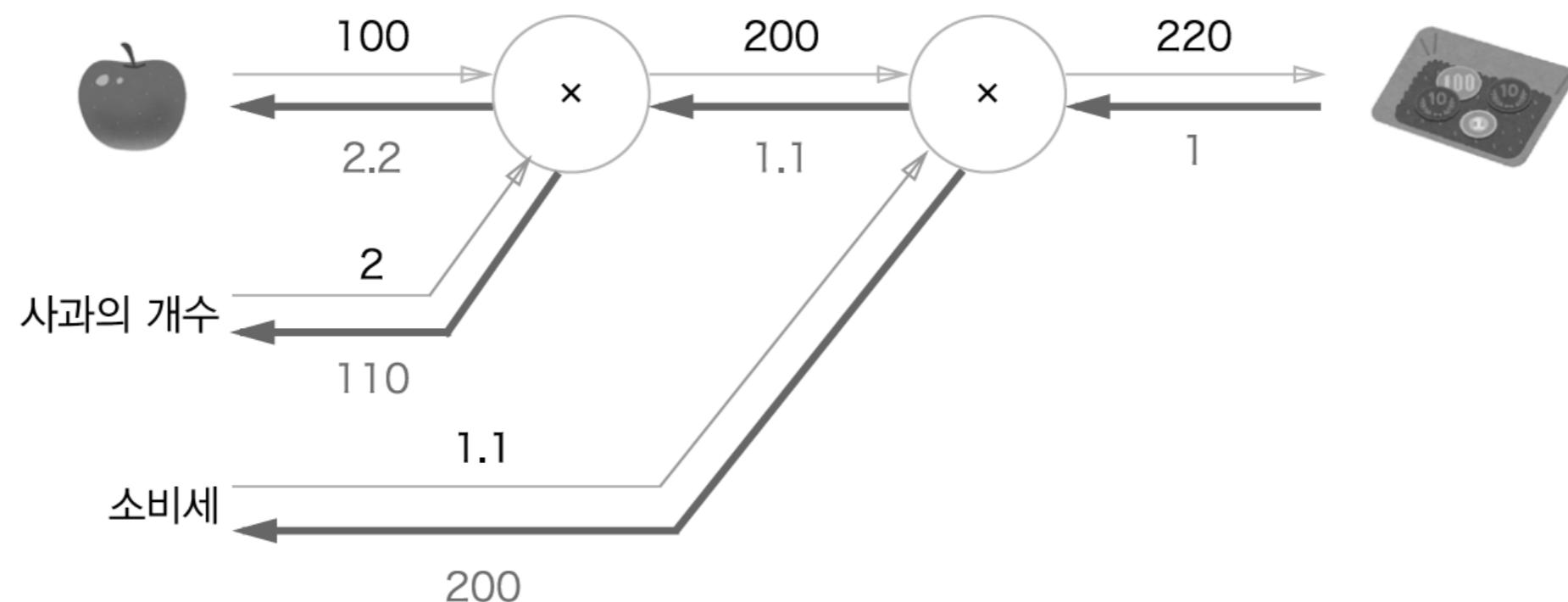


## 4. 오차역전파

### 오차역전파 (backpropagation)

#### 역전파

- 사과 쇼핑 예시



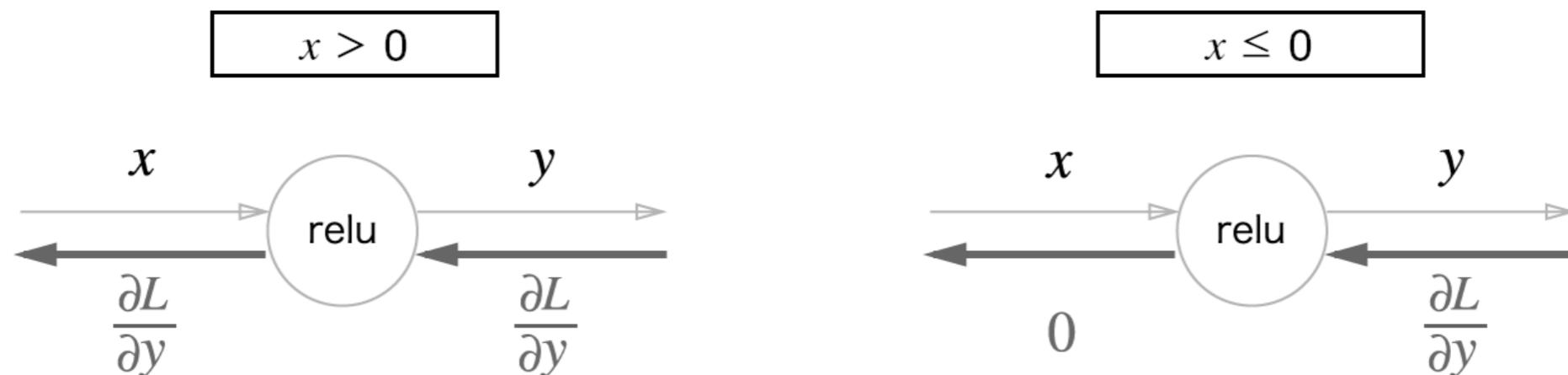


# 4. 오차역전파

## 오차역전파 (backpropagation)

### 활성화 함수 계층의 역전파

- ReLU 계층
  - ReLU의 역전파는 간단하게 구현할 수 있다.





## 4. 오차역전파

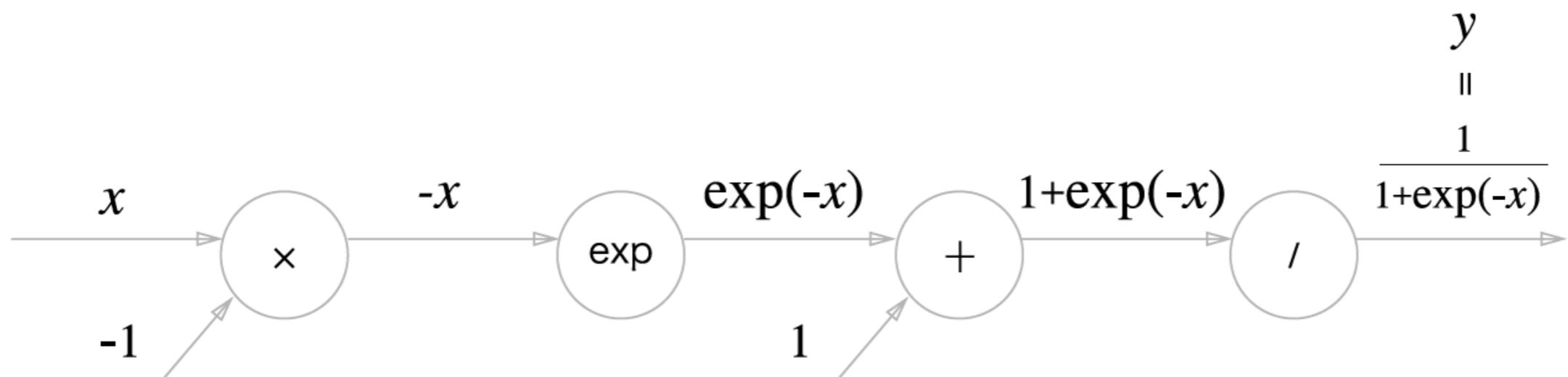
### 오차역전파 (backpropagation)

#### 활성화 함수 계층의 역전파

- Sigmoid 계층

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

- Sigmoid 계층의 계산 그래프





## 4. 오차역전파

### 오차역전파 (backpropagation)

#### 활성화 함수 계층의 역전파

- Sigmoid 계층

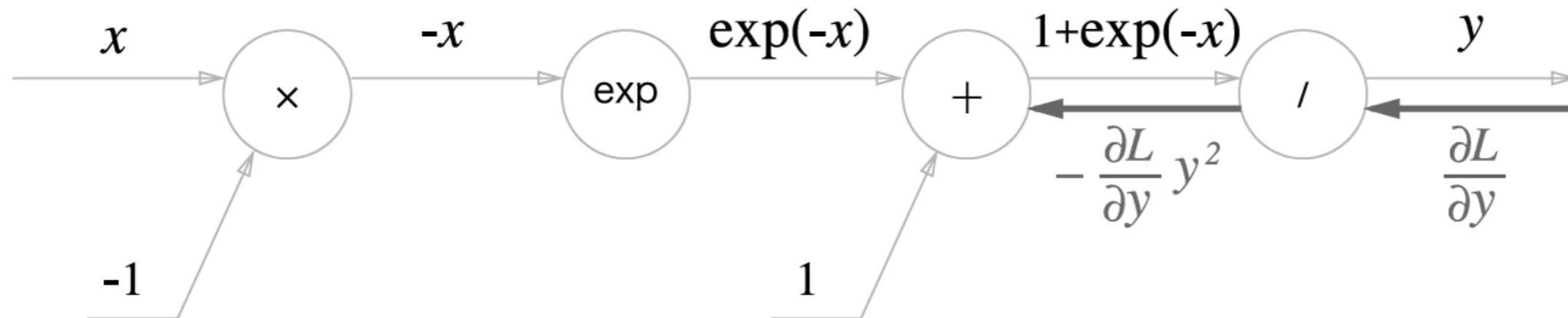
$$y = \frac{1}{1 + \exp(-x)}$$

- 1 단계

- / 노드, 즉  $y = \frac{1}{x}$  를 미분한다.

$$\frac{\partial y}{\partial x} = -\frac{1}{x^2}$$

$$= -y^2$$





## 4. 오차역전파

### 오차역전파 (backpropagation)

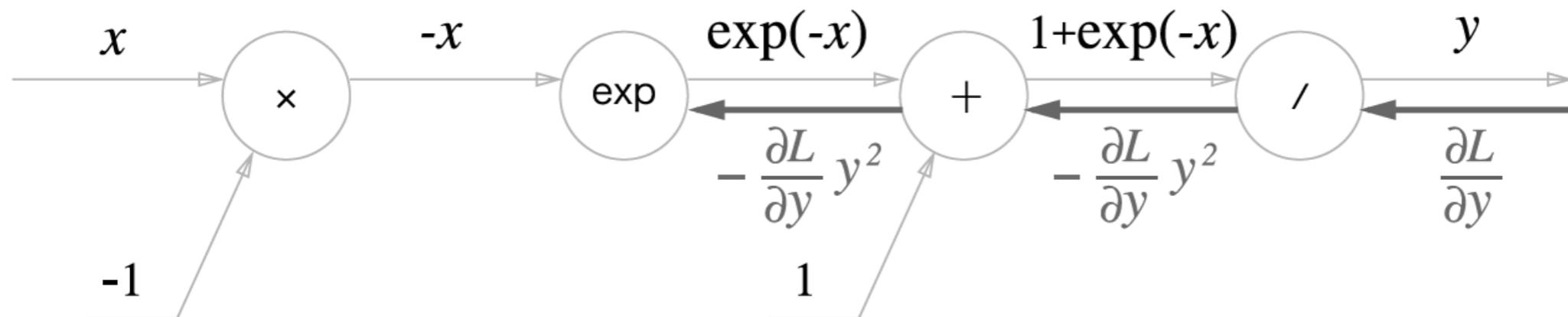
#### 활성화 함수 계층의 역전파

- Sigmoid 계층

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

- 2 단계

- + 노드는 훌러보낸다.





# 4. 오차역전파

## 오차역전파 (backpropagation)

### 활성화 함수 계층의 역전파

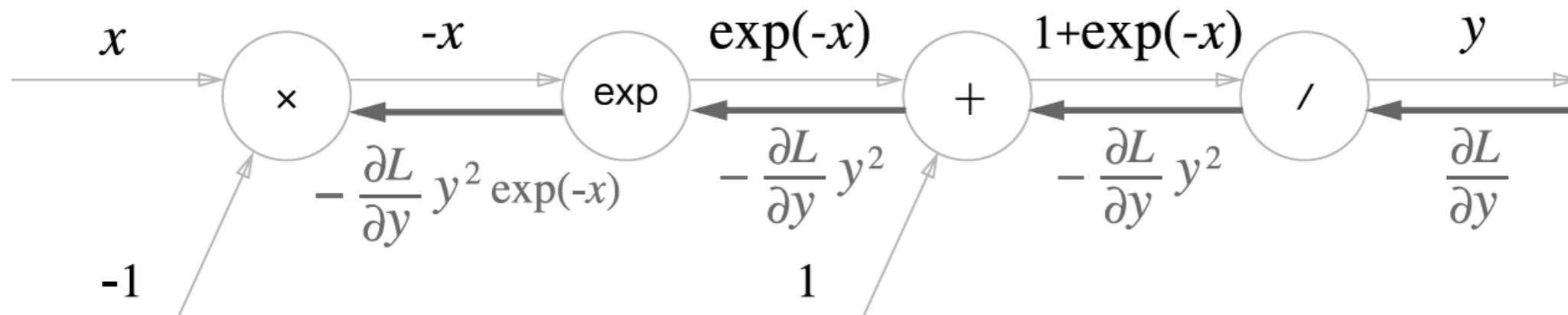
- Sigmoid 계층

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

- 3 단계

- exp 노드는  $y = \exp(x)$  연산을 수행하며, 그 미분은 아래와 같다.

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





# 4. 오차역전파

## 오차역전파 (backpropagation)

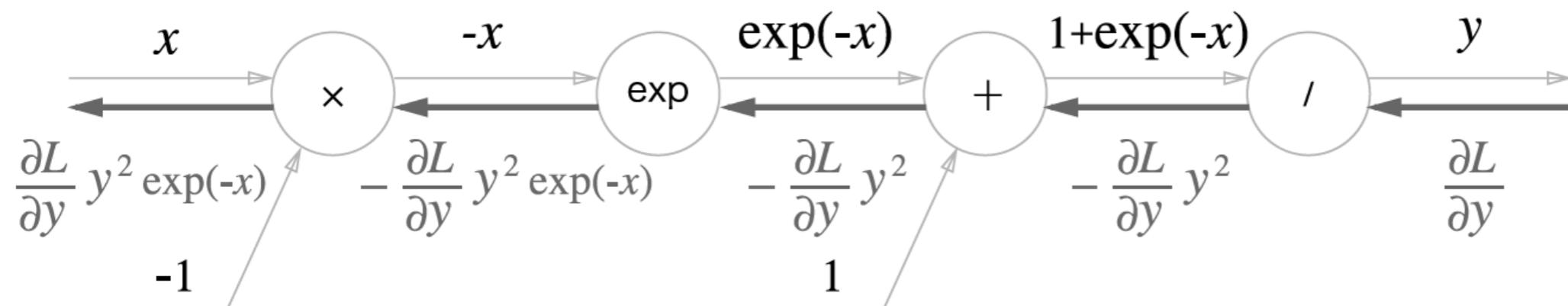
### 활성화 함수 계층의 역전파

- Sigmoid 계층

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

- 4 단계

- X 노드는 순전파 때의 값을 서로 바꿔 곱한다.



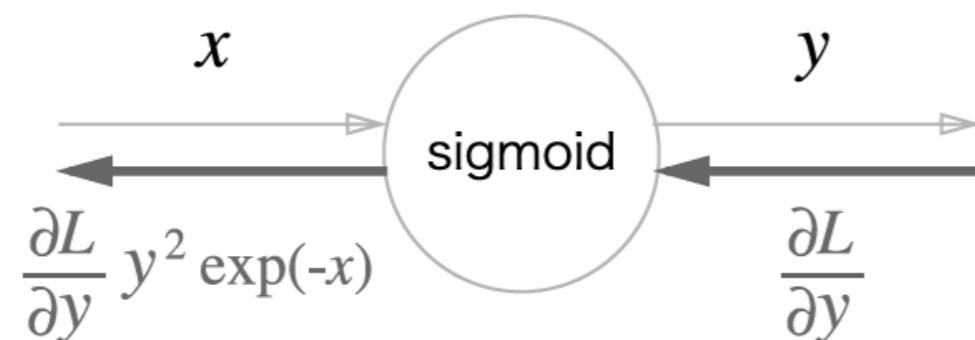


## 4. 오차역전파

### 오차역전파 (backpropagation)

#### 활성화 함수 계층의 역전파

- Sigmoid 계층
  - 모든 과정을 간략하게 나타내면 아래와 같다.



- $\frac{\partial L}{\partial y} y^2 \exp(-x)$  를 정리하면 다음과 같다.

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

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

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



## 4. 오차역전파

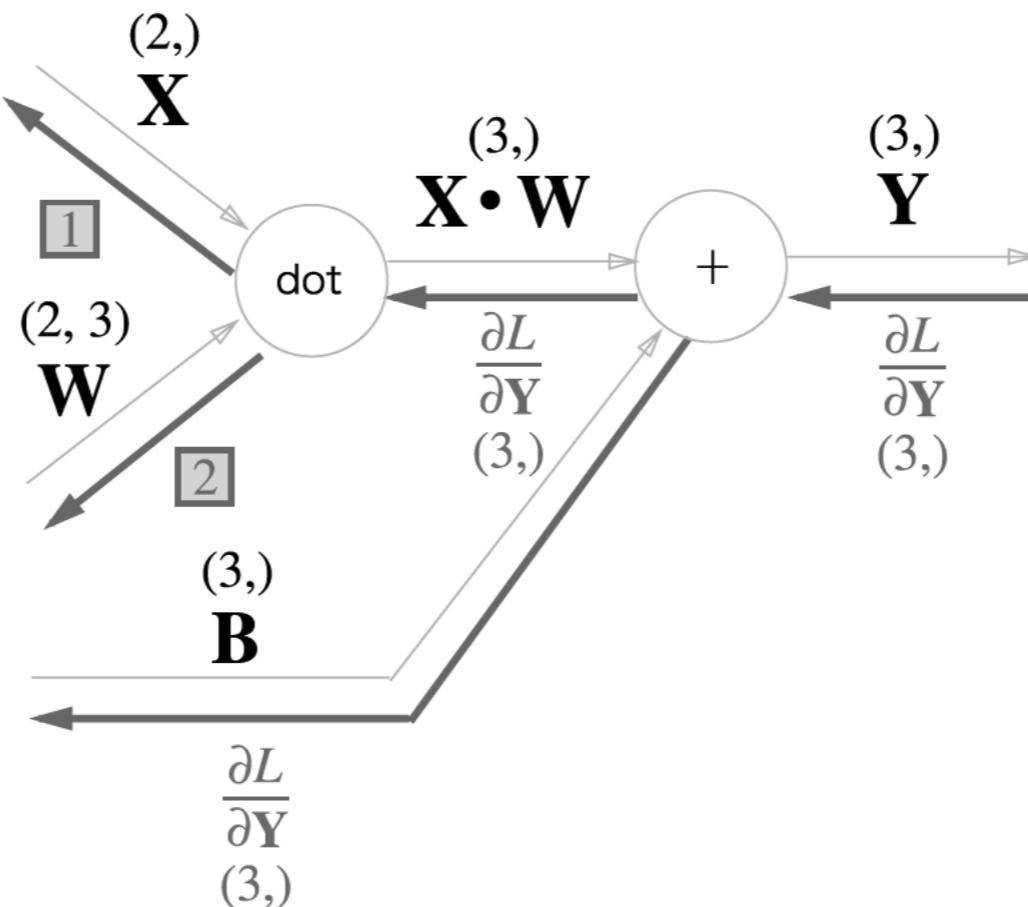
### 오차역전파 (backpropagation)

#### Affine 계층의 역전파

- 기하학에서는 행렬의 내적을 affine transformation이라고 한다.

$$\boxed{1} \quad \frac{\partial L}{\partial \mathbf{X}} = \frac{\partial L}{\partial \mathbf{Y}} \mathbf{W}^T \quad (2,) \quad (3,) \quad (3, 2)$$

$$\boxed{2} \quad \frac{\partial L}{\partial \mathbf{W}} = \mathbf{X}^T \frac{\partial L}{\partial \mathbf{Y}} \quad (2, 3) \quad (2, 1) \quad (1, 3)$$

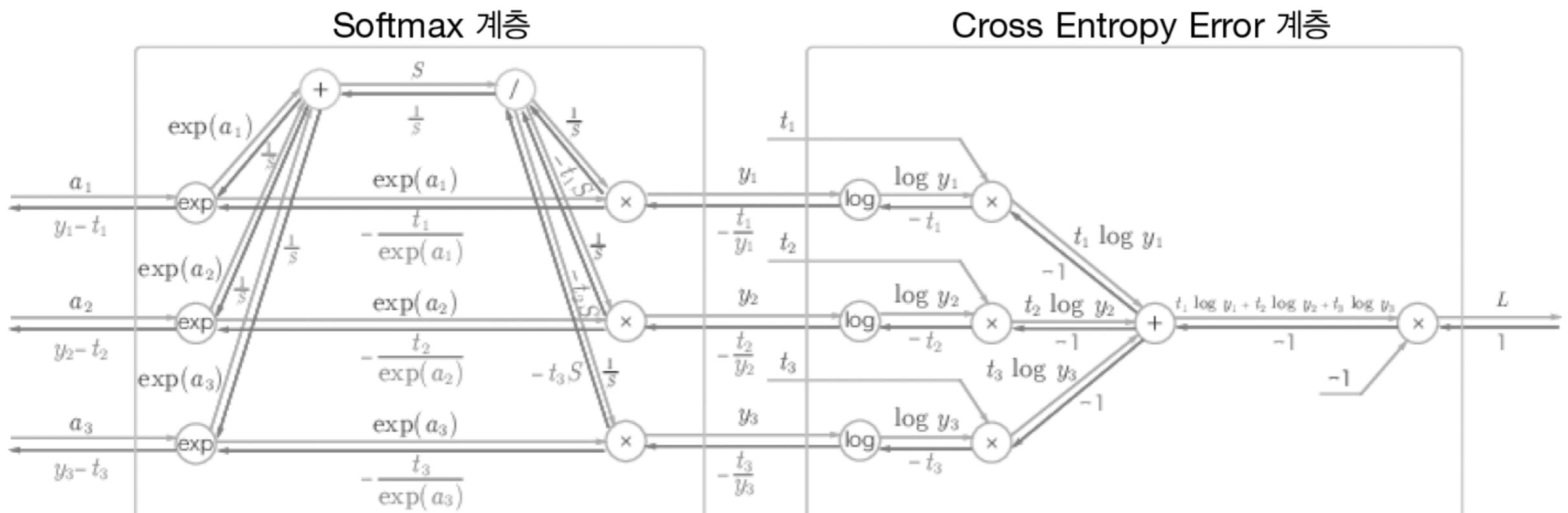




# 4. 오차역전파

## 오차역전파 (backpropagation)

### softmax-with-loss 계층의 역전파

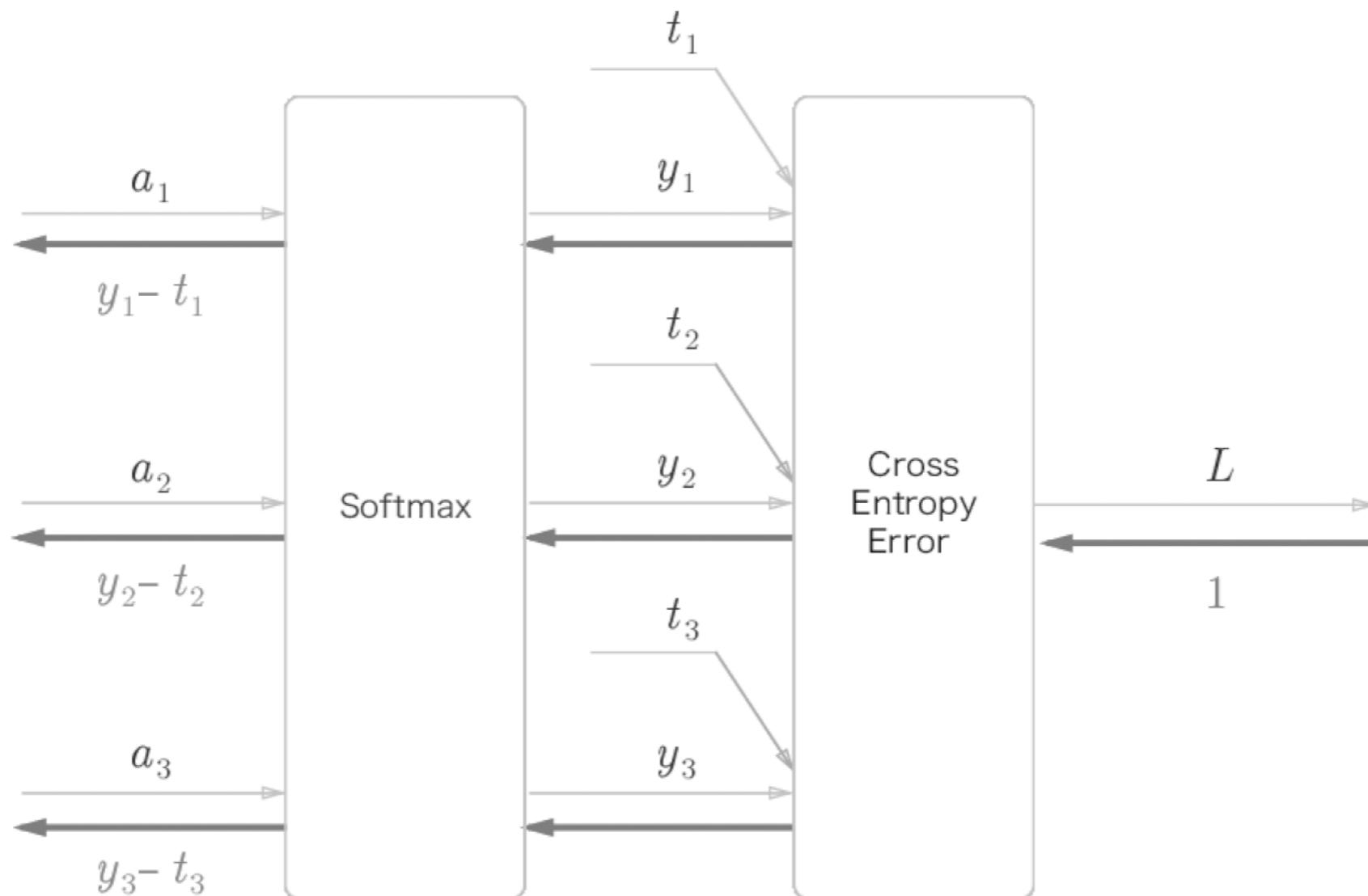




## 4. 오차역전파

### 오차역전파 (backpropagation)

softmax-with-loss 계층의 역전파





# 4. 오차역전파

---

## 오차역전파 (backpropagation)

| 오차 역전파를 이용한 학습 알고리즘 구현



## 5. 학습 관련 기술들

---

최적화, 배치정규화, 드랍아웃



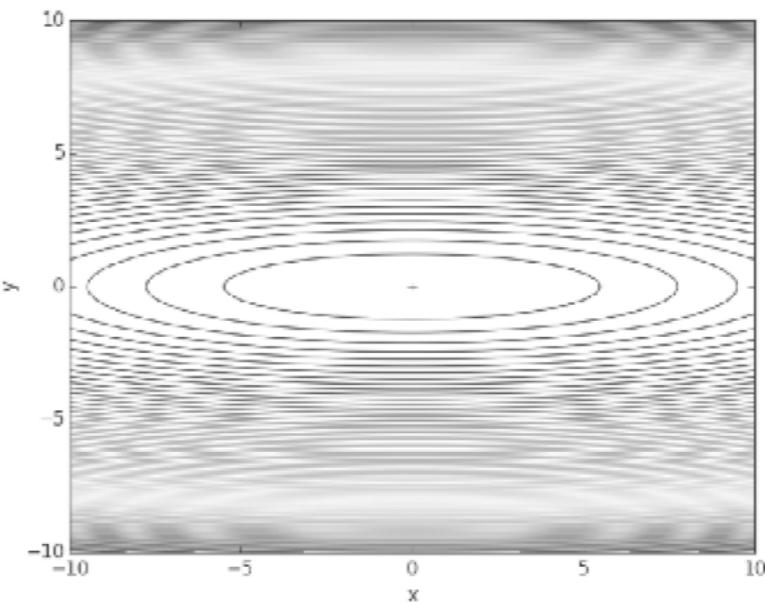
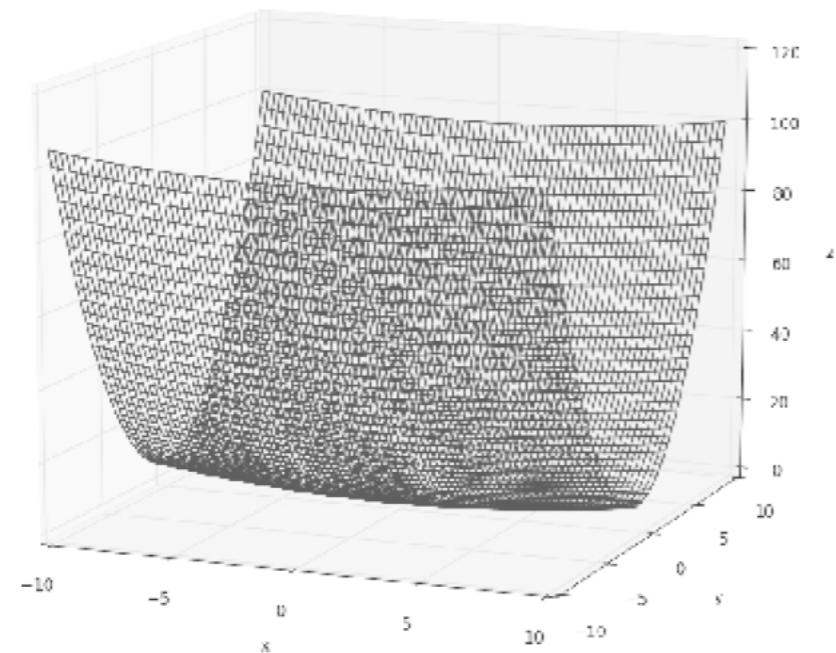
# 5. 학습 관련 기술들

## 최적화

### 확률적 경사 하강법 (SGD)

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

- 단점
  - 문제에 따라 비효율적으로 최적점을 찾아간다.

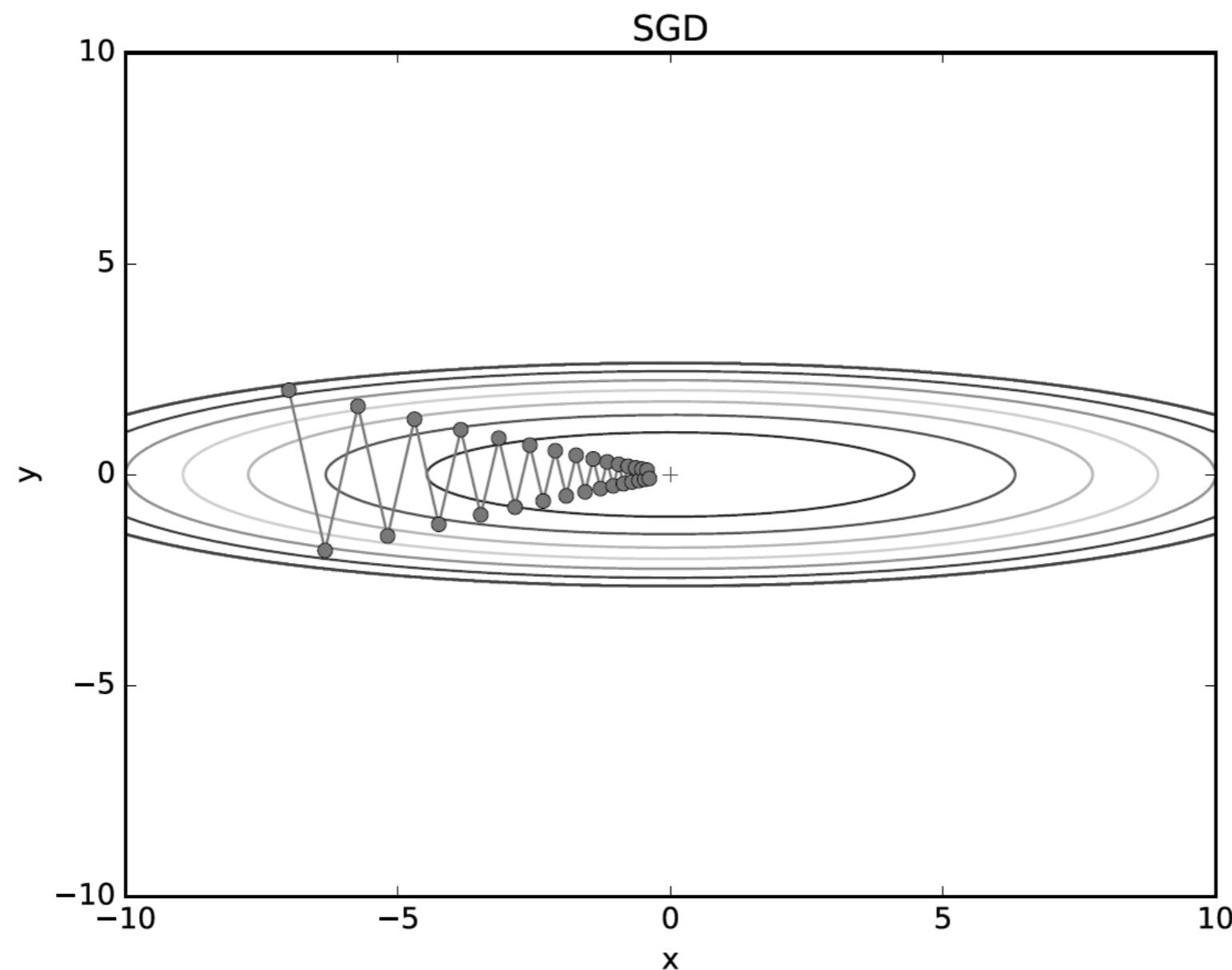




# 5. 학습 관련 기술들

## 최적화

### 확률적 경사 하강법 (SGD)





# 5. 학습 관련 기술들

## 최적화

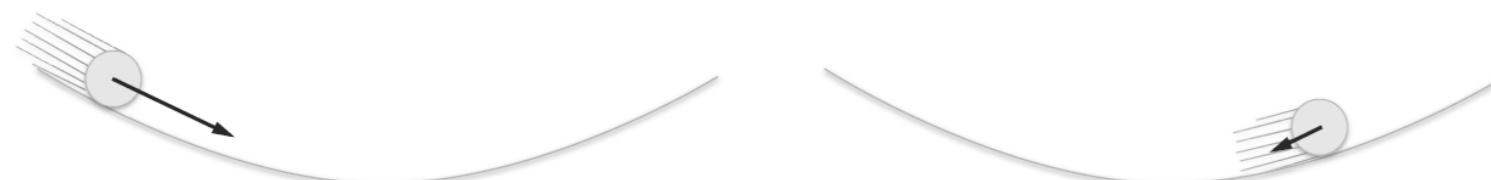
### 모멘텀 (Monmentum)

- 기울기의 움직임을 업데이트한다.

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

$$\mathbf{W} \leftarrow \mathbf{W} + \mathbf{v}$$

- $\mathbf{v}$ 는 이전의 기울기 정보를 계속 가지고 있기 때문에 한쪽 방향으로 계속 움직이면 그 방향으로 가속하는 효과를 내고, 갑자기 다른 방향으로 기울기가 업데이트 된다면, 다른 방향으로 한번에 움직이지 못하도록, 마치 지면 마찰같은 효과를 낸다.



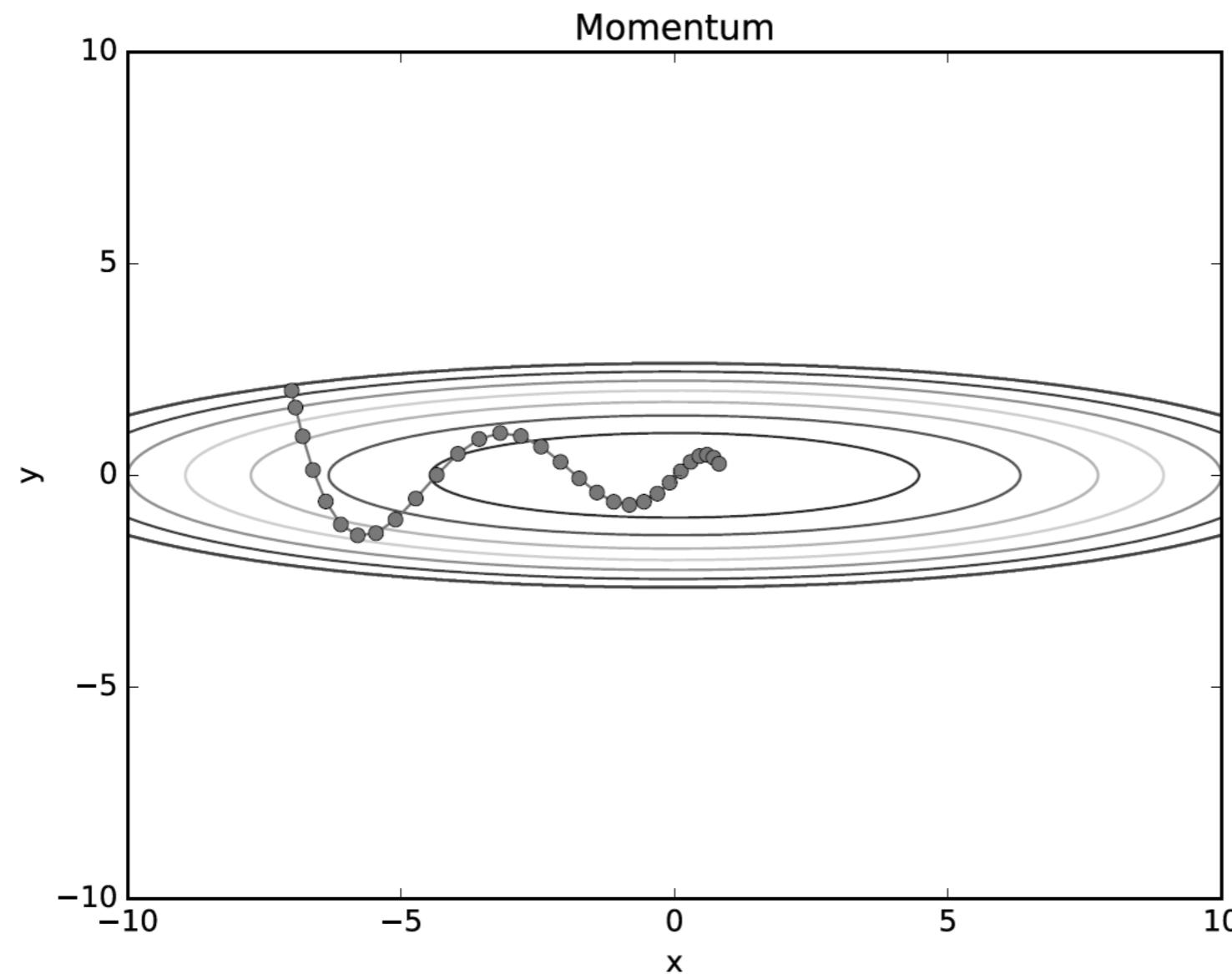


# 5. 학습 관련 기술들

## 최적화

### 모멘텀 (Monmentum)

- 기울기의 움직임을 업데이트한다.





# 5. 학습 관련 기술들

## 최적화

### AdaGrad

- 학습률을 변수마다 적응적으로 조정한다.

$$\mathbf{h} \leftarrow \mathbf{h} + \frac{\partial L}{\partial \mathbf{W}} \odot \frac{\partial L}{\partial \mathbf{W}}$$

$$\mathbf{W} \leftarrow \mathbf{W} - \eta \frac{1}{\sqrt{\mathbf{h}}} \frac{\partial L}{\partial \mathbf{W}}$$

- 매개변수 (w)의 원소 중에서 많이 움직인 (크게 갱신된) 원소는 학습률이 낮아진다.

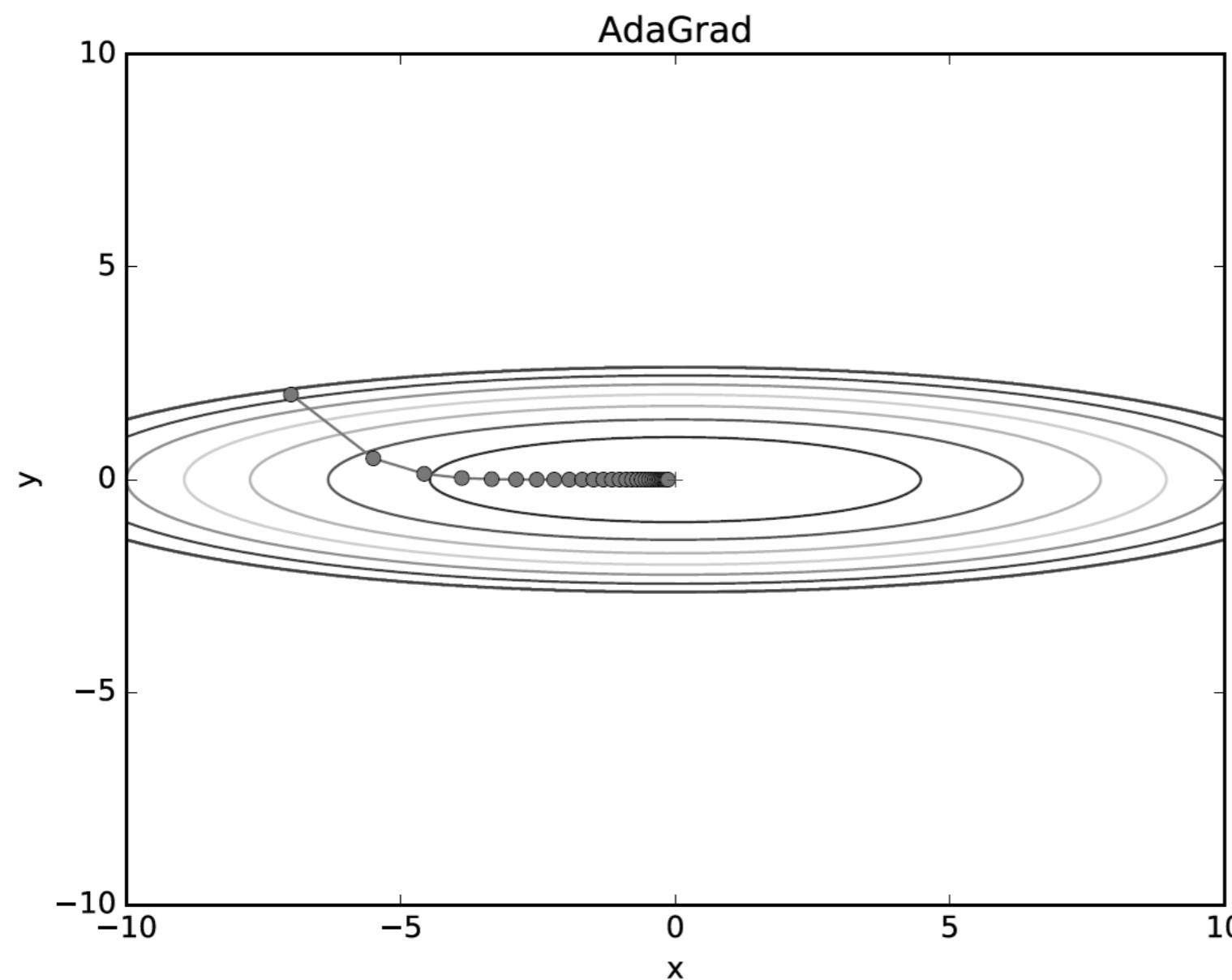


# 5. 학습 관련 기술들

## 최적화

### AdaGrad

- 학습률을 변수마다 적응적으로 조정한다.



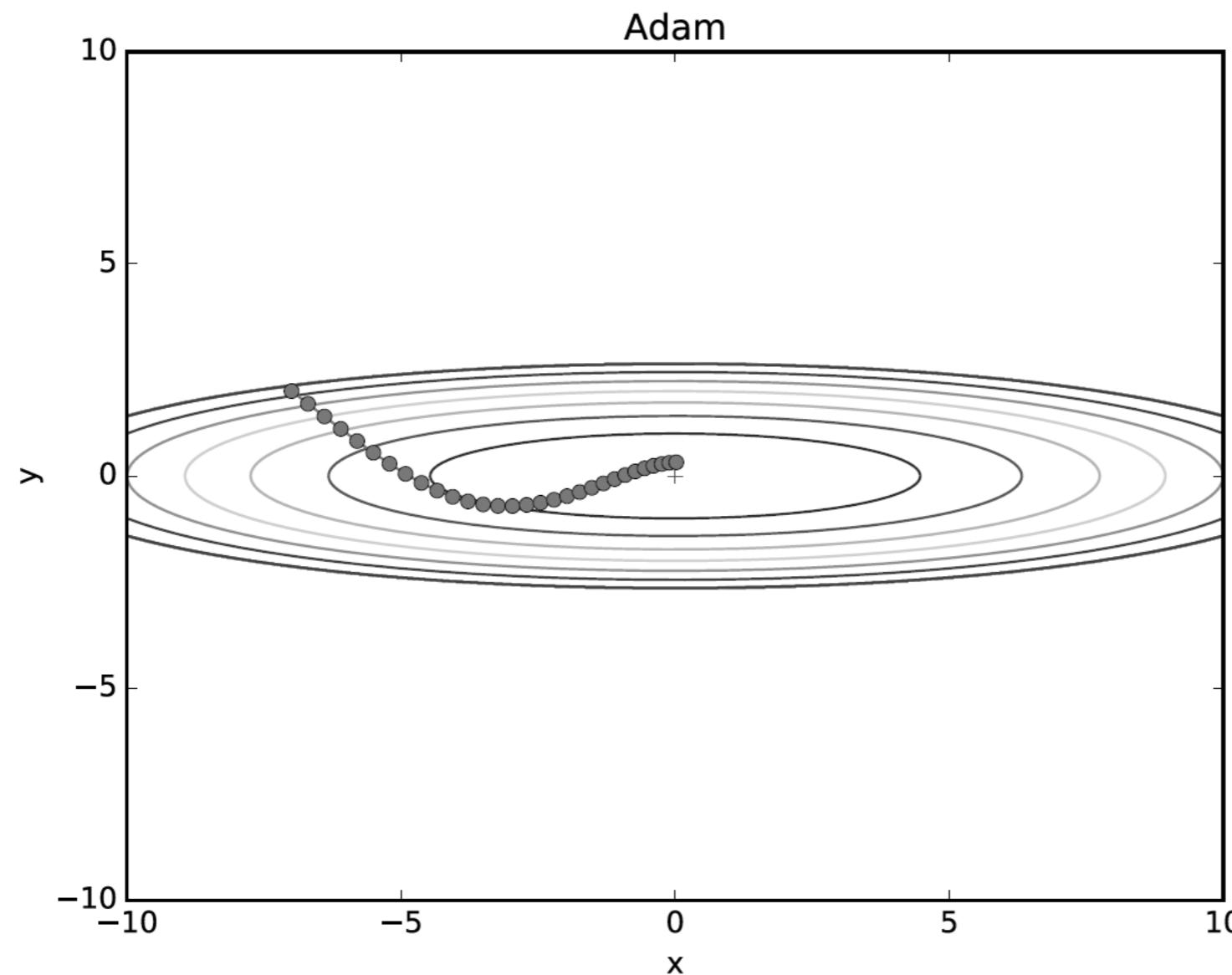


# 5. 학습 관련 기술들

## 최적화

### Adam

- 모멘텀과 AdaGrad를 융합한 것이다.



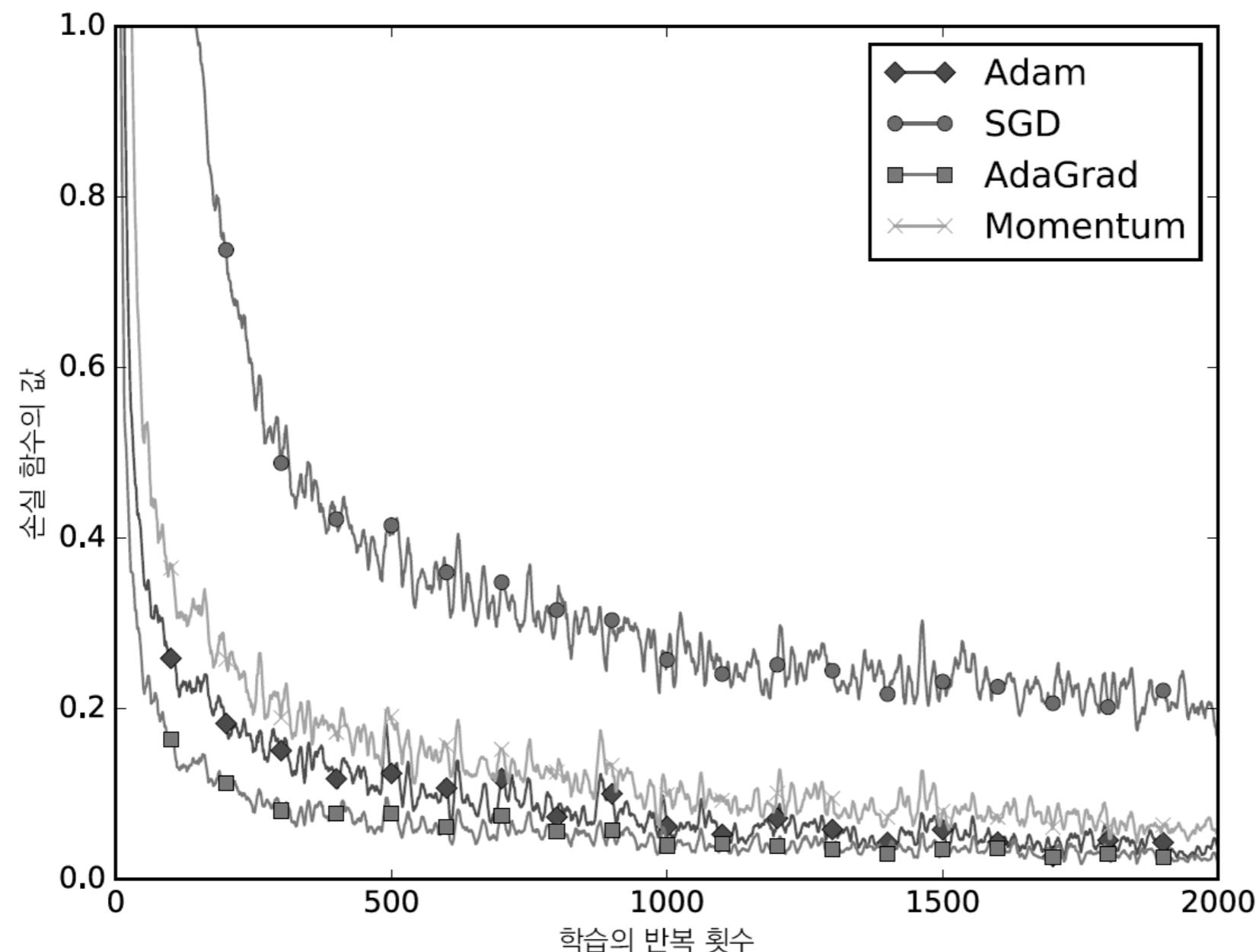
- 대부분의 문제에서 잘 작동한다.



# 5. 학습 관련 기술들

## 최적화

### 최적화 기법 비교



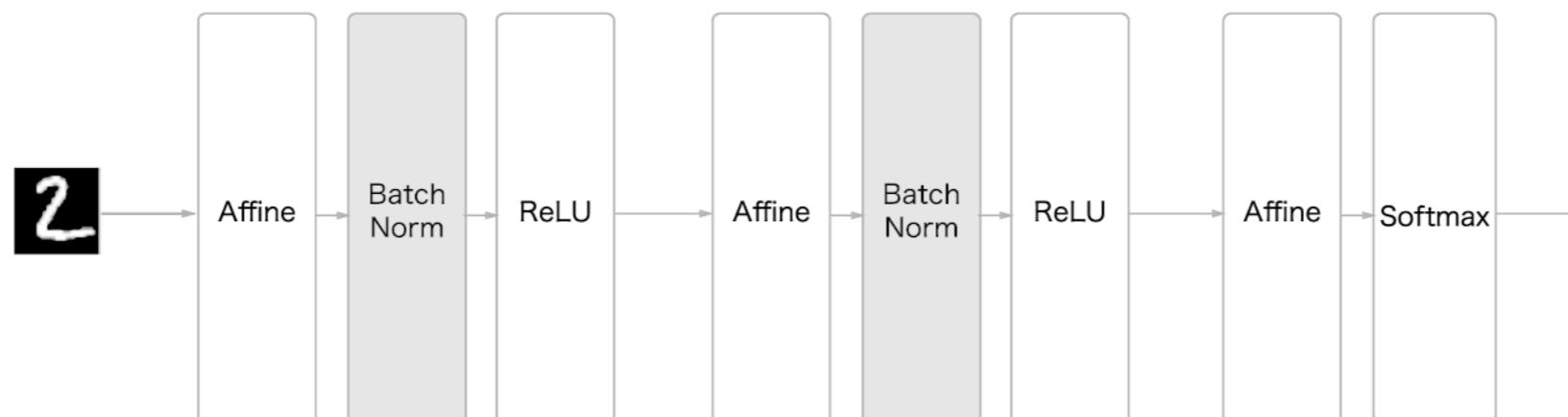


# 5. 학습 관련 기술들

## 배치 정규화

### 배치 정규화 (Batch Normalization)

- 신경망 학습에서는 초기값 설정이 중요한데, 어떤 초기값을 설정하느냐에 따라 활성화값의 분포가 잘 분포할 수도, 한 쪽으로 쏠릴 수도 있다.
  - 적절한 초기값을 설정하면 이런 문제를 회피할 수 있지만, 배치 정규화를 이용하면 초기값 선택의 문제와 더불어, **학습 속도 개선**, **오버피팅 방지**의 이점도 얻을 수 있다.



- 미니배치 단위로 데이터 분포의 평균이 0, 분산이 1이 되도록 정규화한다.

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

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

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

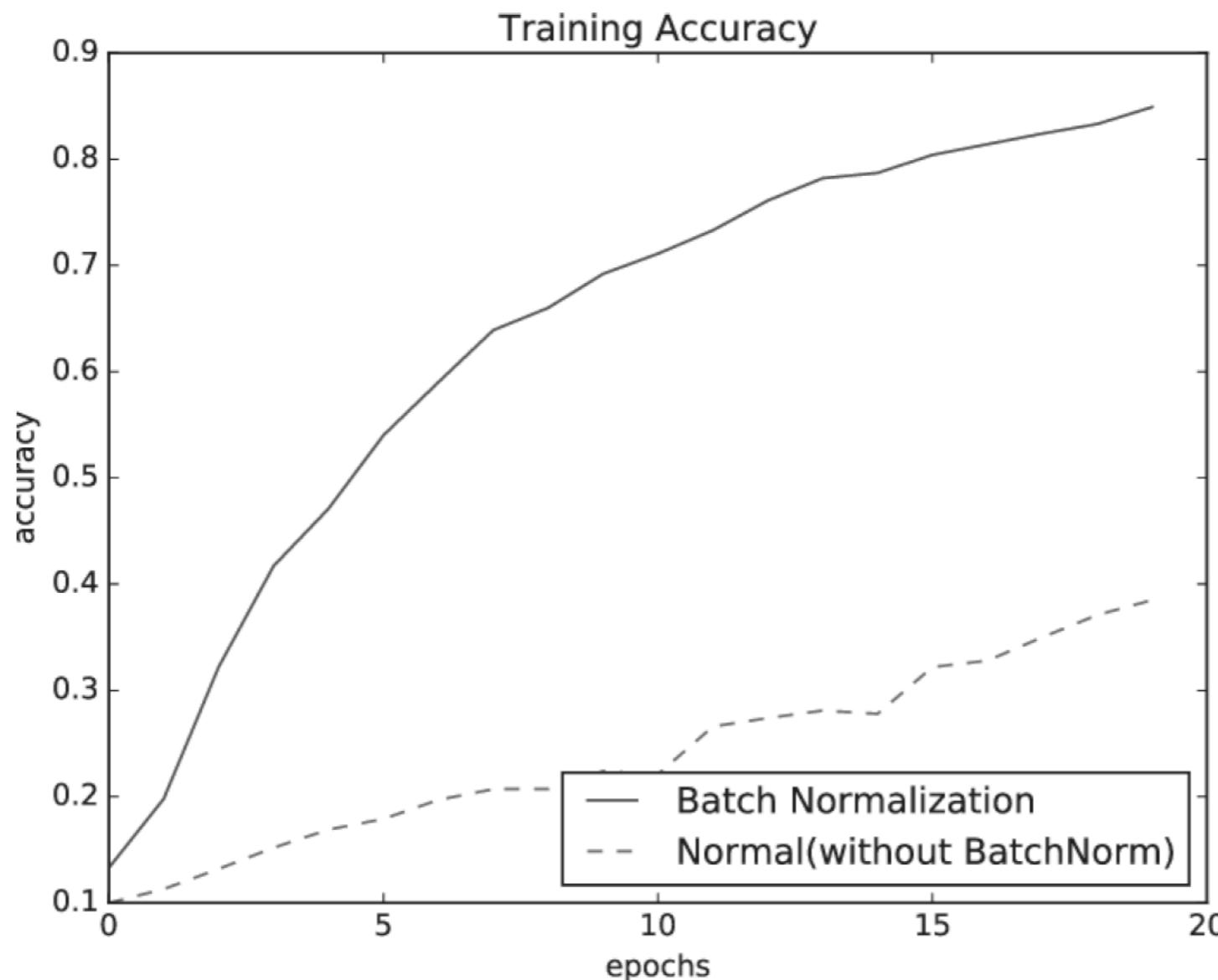


# 5. 학습 관련 기술들

## 배치 정규화

### 배치 정규화 (Batch Normalization)

- 배치 정규화 효과



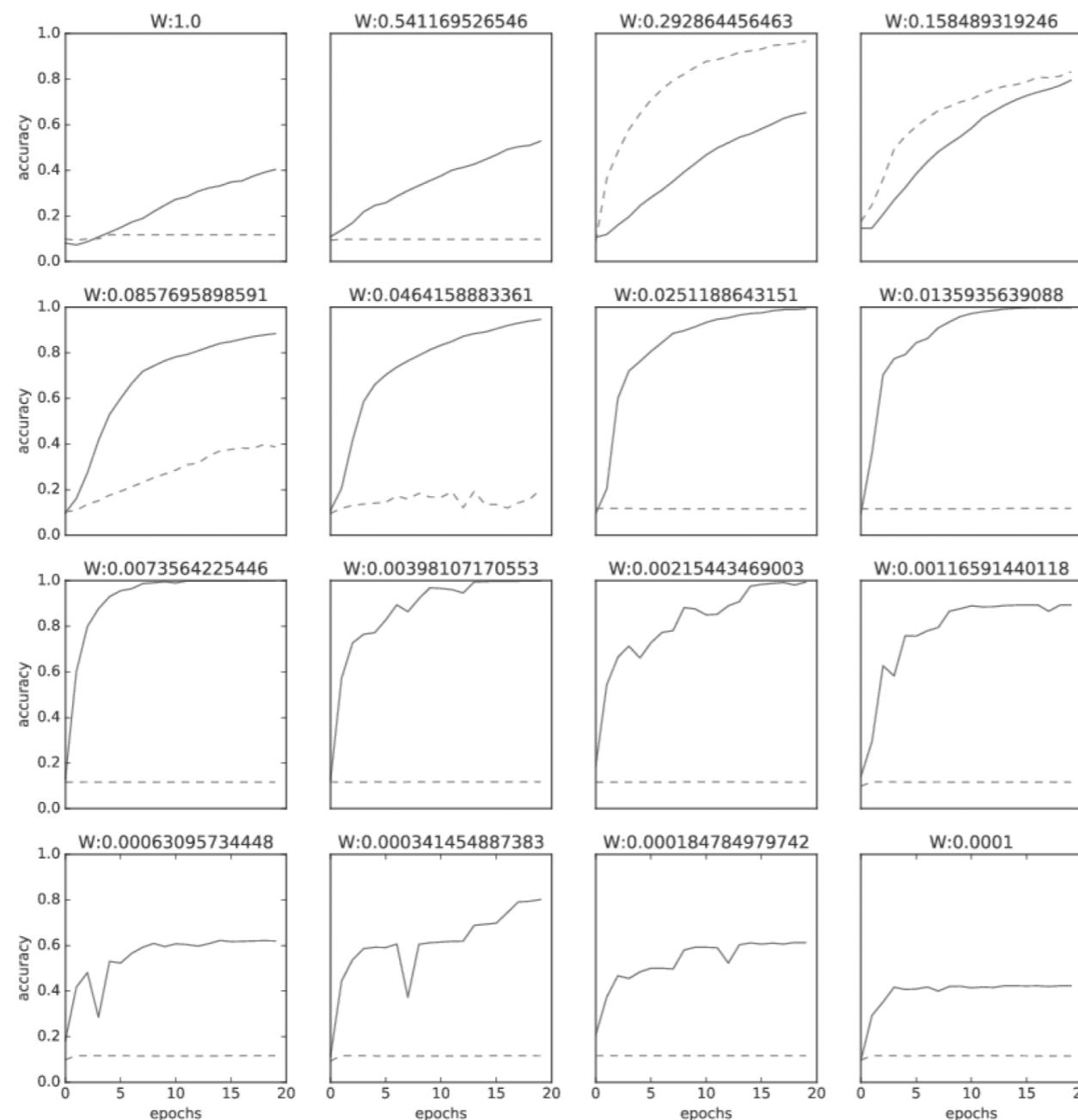


# 5. 학습 관련 기술들

## 배치 정규화

### 배치 정규화 (Batch Normalization)

- 가중치 초기값에 따른 정확도 및 학습 속도





# 5. 학습 관련 기술들

## 드랍 아웃 (dropout)

### 오버피팅 문제

- 매개변수가 많고 표현력이 높은 모델,
- 훈련데이터가 적은 경우에 오버피팅이 많이 발생한다.

### 정규화 (regularization)

- 오버피팅 문제를 해결하기 위해 기계학습에서 주로 사용했던 방법
  - 정규화항을 손실함수에 더해줘서 가중치를 감소 (weight decay)시키는 방식
  - 하지만, 이 방법도 신경망 모델이 복잡해지면서 크게 효과가 없어졌다.

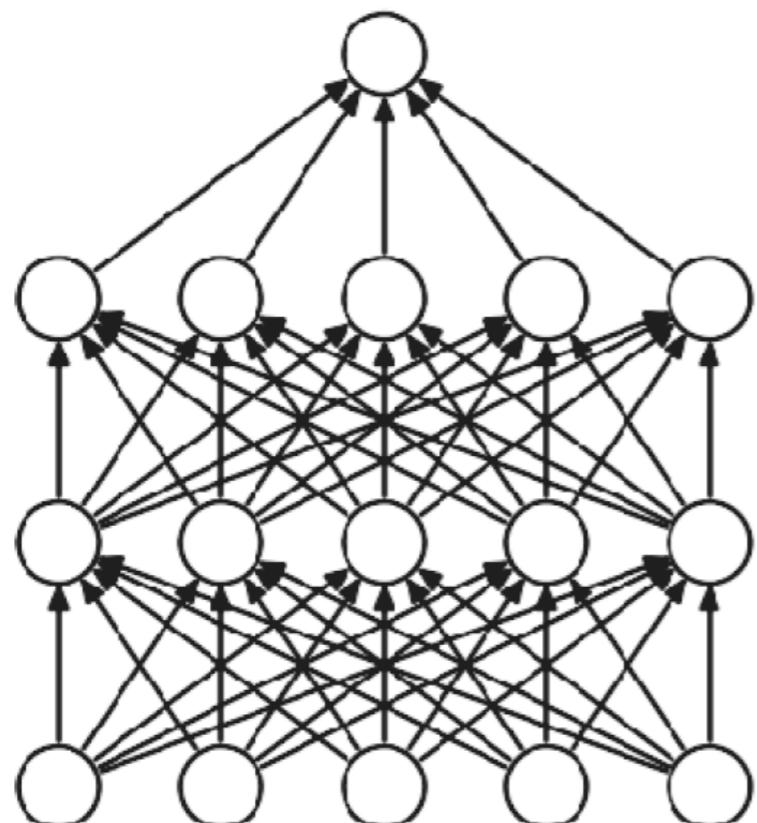


# 5. 학습 관련 기술들

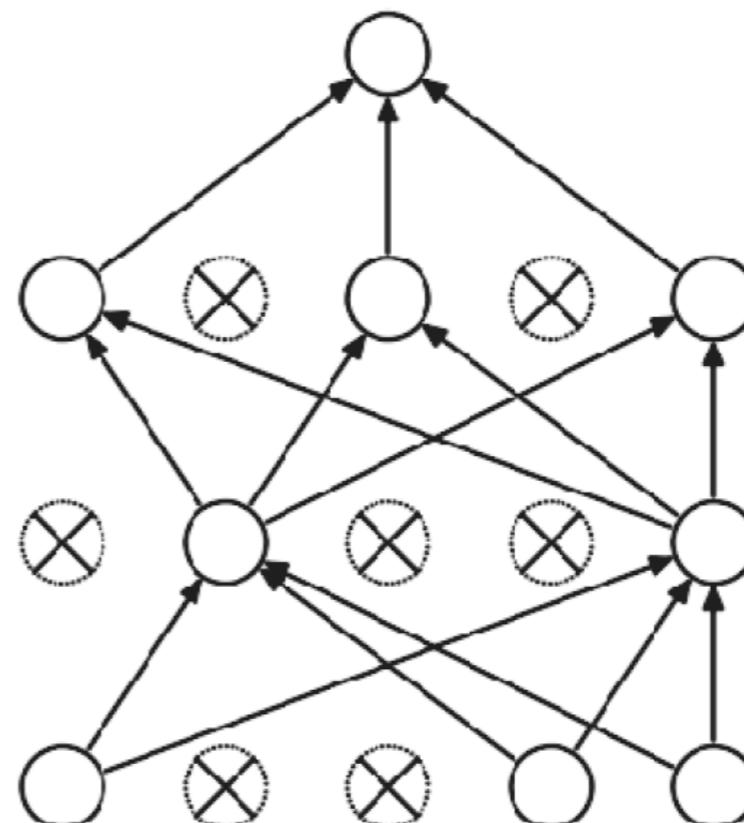
## 드랍 아웃 (dropout)

### 드랍아웃 (dropout)

- 은닉층의 뉴런을 무작위로 골라 삭제하는 방법



(a) 일반 신경망



(b) 드롭아웃을 적용한 신경망

- 드랍아웃이 학습 때 뉴런을 무작위로 삭제하는 것은 매번 다른 모델을 만드는 것과 비슷한 효과는 낸다
- 따라서, 드랍아웃은 **양상블**과 밀접한 관련이 있다.



다음에  
또!  
같이!  
만나요!