

Beginning of DeepLearning

PinkLAB Edu

20 November, 2025

Table of Contents

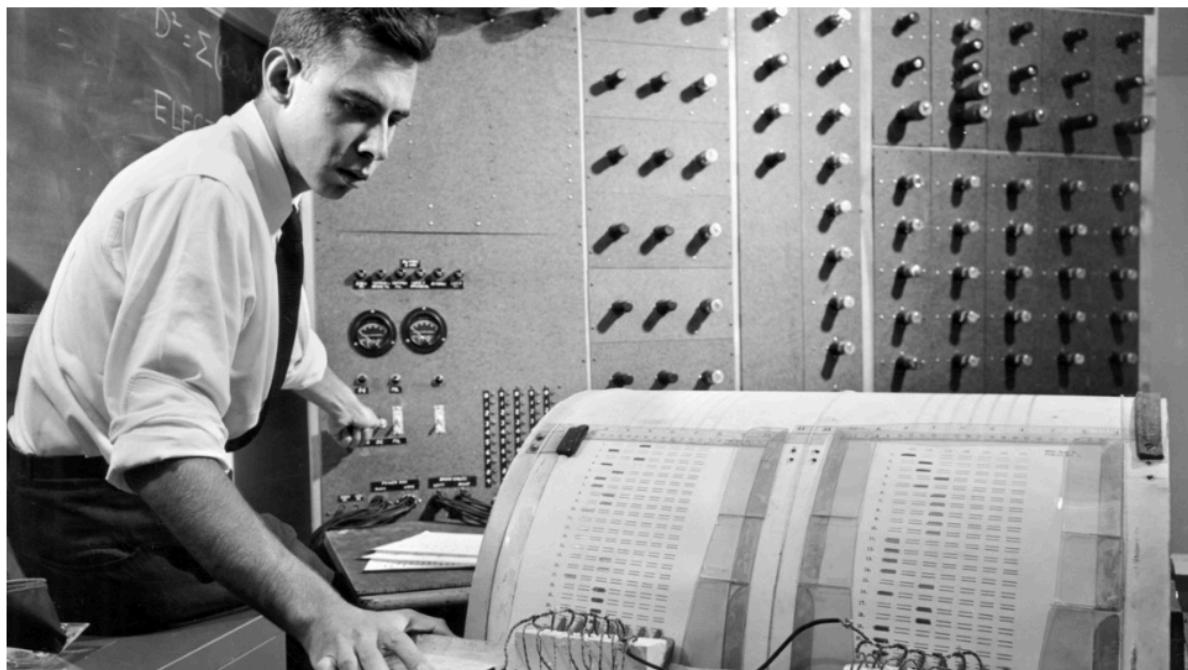
1	퍼셉트론 이야기	5
1.1	누군가 저에게 딥러닝에서 가장 큰 사건이 뭐냐고 묻는다면? - 프랭크 로젠틀릿	5
1.2	1958년 퍼셉트론의 탄생	5
1.3	퍼셉트론은 미 해군 연구소에서 학습하는 과정을 시연하는데 성공	6
1.4	퍼셉트론과 그 후 아달라인은 학습이라는 것을 성공했다	6
1.5	동시대의 두 천재	7
1.6	민스키의 기호주의	7
1.7	연결주의 등장	8
1.8	기호주의와 연결주의	9
1.9	로젠틀릿의 최후	10
1.10	AI 원터 - 그 후	10
2	퍼센트론을 만나러~	11
2.1	입력이 2개인 퍼셉트론	11
2.2	수식으로는	11
2.3	AND 게이트	11
2.4	OR 게이트	12
2.5	NAND 게이트	12
2.6	AND 게이트를 퍼셉트론의 추론기만 사용하도록~	12
2.7	결과도 잘 나옴	13
2.8	θ 를 $-b$ 로 바꾸고 수식 정리	13
2.9	가중치와 편향으로 표현한 함수	13
2.10	결과는 뭐 당연히~	13
2.11	NAND	14
2.12	뭐 당연한 결과를~	14
2.13	OR	14
2.14	역시 당연한 결과를~	14
3	퍼셉트론의 한계	15
3.1	유명한 XOR	15
3.2	XOR 문제	15

3.3 해결 방법은 직선이 아니라 곡선.....	16
3.4 이렇게.....	16
3.5 코드는?	16
3.6 결과는??.....	17
3.7 다층의 퍼셉트론이 필요하다.....	17
4 신경망.....	18
4.1 편향을 고려하면 이렇게 표현할 수도 있다	18
4.2 아무튼 우리는 이런 수식을 가지고 표현하고 있는 중	18
4.3 활성화 함수의 등장 - 여기서는 h	18
4.4 그림으로 표현	19
4.5 만약 0과 1의 출력을 내고 싶다면 활성화 함수는 시그모이드	19
4.6 시그모이드 함수 만들기	19
4.7 그려서 확인해보기	20
4.8 활성화함수는 비선형이 주로 사용됨.....	20
4.9 또 다른 활성화함수 ReLU	20
4.10 생긴건~.....	21
4.11 신경망은 행렬의 곱 연산이 많이 사용된다	22
4.12 이렇게 행렬을 정의하고	22
4.13 연산하면 그만~	22
4.14 3층 신경망	23
4.15 흔히 사용하는 가중치의 표기법	23
4.16 입력층에서 그 다음단계로 진행하는 첫 과정	24
4.17 하나의 노드에 대한 수식	24
4.18 은닉층 (1층)에 대한 수식	24
4.19 코드? 쉽지	25
4.20 그런데 사실 하나 더 있다.. 바로 활성화 함수	25
4.21 여기까지 코드로 하면	26
4.22 이제 일층에서 이층으로 가자	26
4.23 이제 그림보다 코드가 더 쉬울껄	27
4.24 이층에서 삼층 - 출력층으로 가자.....	27
4.25 코드로는 요렇게~	28
4.26 약간 멋진 모습으로 forward 혹은 inference를 구현해 본다면	28

4.27 모델의 가중치들	28
4.28 모델의 forward 연산	28
4.29 추론~	29
5 출력층의 종류	30
5.1 모델의 목표에 따라	30
5.2 Softmax 함수	30
5.3 Softmax를 코드로 하면	30
5.4 누가 제일 큰지 알려준다	30
5.5 여기서 잠깐~ exp의 overflow 문제	31
5.6 당연히 softmax도 에러가 난다	31
5.7 극복할 방법은 수학의 마법	31
5.8 코드로는 이렇게	32
5.9 에러가 안난다	32
6 MNIST에 적용	33
6.1 MNIST 다운 받기	33
6.2 가입이 안되어있으신가요?	34
6.3 MNIST 읽기	35
6.4 데이터를 numpy로	35
6.5 Image를 보여주는 함수	35
6.6 그려주세요	36
6.7 미리 가중치를 읽고 - 다음에 학습하는 법을 배웁니다	36
6.8 모델을 구현하고	37
6.9 와우~ Accuracy 92.5%	37
6.10 혹시 지금까지의 연산의 흐름을 알겠나요?	37
6.11 만약 100개씩 묶어서 처리한다면~	38
6.12 이 코드가 유명한 배치사이즈를 설정한 코드	38

1 퍼셉트론 이야기

1.1 누군가 저에게 딥러닝에서 가장 큰 사건이 뭐냐고 묻는다면? - 프랭크 로젠틀랫



1.2 1958년 퍼셉트론의 탄생



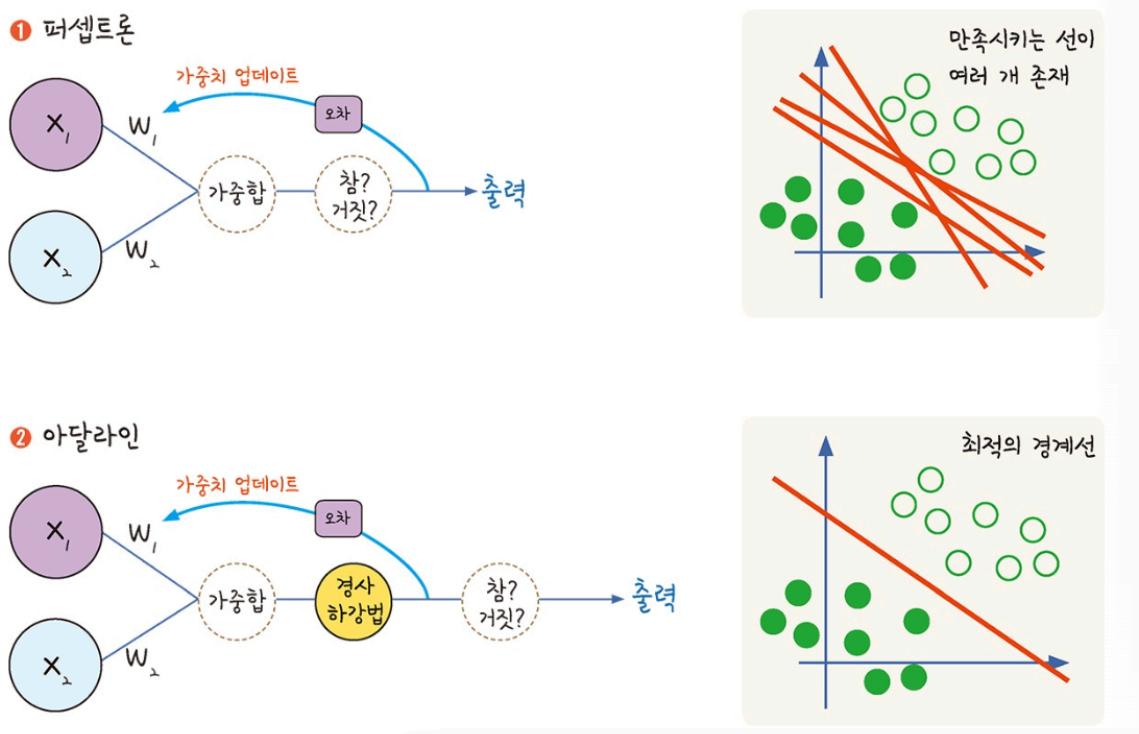
- 로젠틀랫이 1958년 발표한 논문 '퍼셉트론(THE PERCEPTRON: A PROBABILISTIC MODEL FOR INFORMATION STORAGE AND ORGANIZATION IN THE BRAIN)
- "If we are eventually to understand the capability of higher organisms for perceptual recognition, generalization, recall, and thinking, we must first have answers to three fundamental question."*(우리가 궁극적으로 지각적 인식, 일반화, 기억 그리고 사고를 할 수 있는 고등 유기체의 능력을 이해하기 위해서는 먼저 세 가지 근본적인 질문에 대해 답할 수 있어야 한다.**)

- How is information about the physical world sensed, or detected, by the biological system?
(생물 시스템은 실제 물리적 세계에 대한 정보를 어떻게 감지할까?**)
- In what form is information stored, or remembered? (어떤 형태로 저장되거나 기억될까? **)
- How does information contained in storage, or in memory, influence recognition and behavior? (스토리지 또는 메모리에 저장된 정보가 인식과 동작에 어떻게 영향을 미칠까? **)

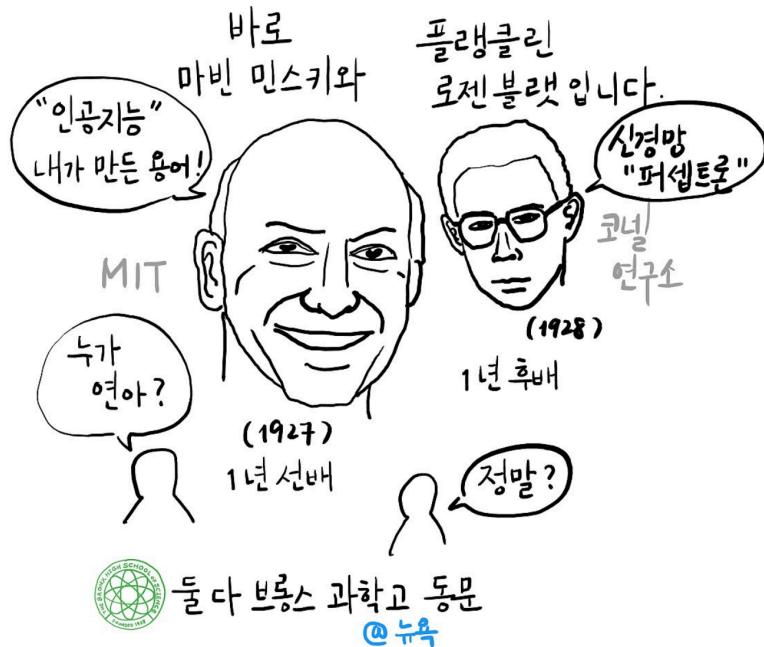
1.3 퍼셉트론은 미 해군 연구소에서 학습하는 과정을 시연하는데 성공

- 뉴런을 모델링한 맥컬록-피츠 모델에서 가중치라는 개념을 도입
- 이 실험은 로젠블랫의 목표인 “스스로 고유한 아이디어를 가질 수 있는, 즉 생각할 수 있는 최초의 기계”라는 가능성을 증명함
- 5톤 크기의 거대한 컴퓨터인 IBM 704에 여러 개의 편지 카드들을 읽어 들였고, 50번의 시행착오 끝에 왼쪽과 오른쪽에 마크 표시한 편지 카드들을 구별하는 방법을 스스로 학습함
- 이 시연은 당시 전 사회적으로 큰 관심을 받아서 뉴욕타임스 등 유력지에도 비중있게 소개됨.
- 당시 기사의 표현을 빌리자면 ‘걷고, 이야기하고, 보고, 글을 쓸 수 있을 뿐만 아니라 자기 복제와 자신의 존재 인식이 가능한**** 인공지능의 시대가 올 것이라는 낙관론이 퍼짐 - 1950년대말, 1960년대초

1.4 퍼셉트론과 그 후 아달라인은 학습이라는 것을 성공했다



1.5 동시대의 두 천재

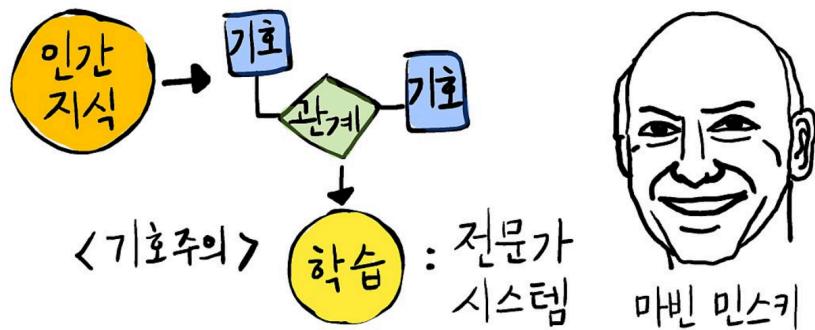


1 출처 :

1

1.6 민스키의 기호주의

두각을 먼저 드러낸 사람은 민스키였습니다.
그는 1956년 다트머스 회의에서 최초로 '인공지능'
용어를 사용하였으며 개념을 확립했습니다.



2 출처 :

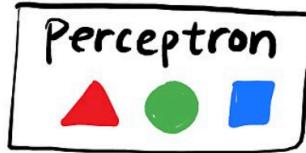
2

1. <https://m.post.naver.com/my.naver?memberNo=50533718>

1.7 연결주의 등장

그러던 어느날

기호주의로 독주하고 있던
민斯基에게 새로운 도전자가
나타납니다. 로젠블랫은
당시 주류인 '기호주의' 대신
'연결주의'를 선택합니다.



컴퓨터도 **신경망**으로
학습시켜 추론하게하자
(퍼셉트론)



민斯基와 동문
1년 후배분?

3 출처 :

3

-
2. <https://m.post.naver.com/my.naver?memberNo=50533718>
 3. <https://m.post.naver.com/my.naver?memberNo=50533718>

1.8 기호주의와 연결주의



- 퍼셉트론의 한계를 비판한 마빈 민스키와 시모어 페퍼트
- 인간의 지식을 기호화해 컴퓨터에 입력하면 사람과 똑같은 출력을 내줄 것 - 기호주의
- 컴퓨터도 인간 뇌의 신경망처럼 학습시키면 근삿값을 출력할 수 있다 - 퍼셉트론
- 민스키는 1969년 '퍼셉트론즈(Perceptrons)'란 책을 발표해 퍼셉트론 개념의 한계를 비판했고, 그 결과 신경망 연구에 대한 관심과 지원은 끊어짐

1.9 로젠틀랫의 최후



아... ↗

그 해 여름 로젠틀랫은
체서피크만에서 요트에 몸을 기울인 채
마지막 모습이 발견되었습니다.

4 출처 :

4

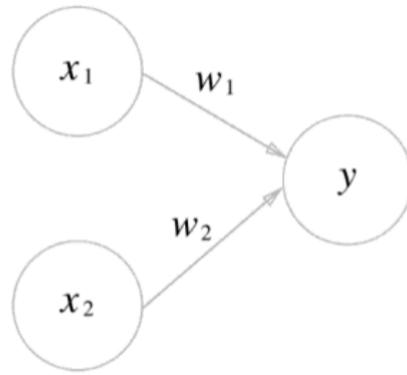
1.10 AI 윈터 - 그 후

- 1986년 제프리 힌튼에 의해 다층 퍼셉트론과 역전파 알고리즘이 재발견
- 1989년 “Backpropagation applied to handwritten zip code recognition, LeCun, 1989”

4. <https://m.post.naver.com/my.naver?memberNo=50533718>

2 퍼셉트론을 만나러~

2.1 입력이 2개인 퍼셉트론



2.2 수식으로는

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

- 여기서 theta는 threshold

2.3 AND 게이트

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

2.4 OR 게이트

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

2.5 NAND 게이트

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

2.6 AND 게이트를 퍼셉트론의 추론기만 사용하도록~

```
def AND(x1, x2):
    w1, w2, theta = 0.5, 0.5, 0.7
    tmp = x1 * w1 + x2 * w2
    if tmp <= theta:
        return 0
    elif tmp > theta:
        return 1
```

✓ 0.0s

Python

2.7 결과도 잘 나옴

```
AND(0, 0), AND(1, 0), AND(0, 1), AND(1, 1)
✓ 0.0s
```

Python

2.8 theta를 -b로 바꾸고 수식 정리

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

2.9 가중치와 편향으로 표현한 함수

```
import numpy as np

def AND(x1, x2):
    x = np.array([x1, x2])
    w = np.array([0.5, 0.5])
    b = -0.7
    tmp = np.sum(w * x) + b
    if tmp <= 0:
        return 0
    else:
        return 1
✓ 0.1s
```

Python

2.10 결과는 뭐 당연히~

```
AND(0, 0), AND(1, 0), AND(0, 1), AND(1, 1)
✓ 0.0s
```

Python

2.11 NAND

```
def NAND(x1, x2):
    x = np.array([x1, x2])
    w = np.array([-0.5, -0.5])
    b = 0.7
    tmp = np.sum(w * x) + b
    if tmp <= 0:
        return 0
    else:
        return 1
```

✓ 0.0s

Python

2.12 뭐 당연한 결과를~

```
NAND(0, 0), NAND(1, 0), NAND(0, 1), NAND(1, 1)
```

✓ 0.0s

Python

(1, 1, 1, 0)

2.13 OR

```
def OR(x1, x2):
    x = np.array([x1, x2])
    w = np.array([0.5, 0.5])
    b = -0.2
    tmp = np.sum(w * x) + b
    if tmp <= 0:
        return 0
    else:
        return 1
```

✓ 0.0s

Python

2.14 역시 당연한 결과를~

```
OR(0, 0), OR(1, 0), OR(0, 1), OR(1, 1)
```

✓ 0.0s

Python

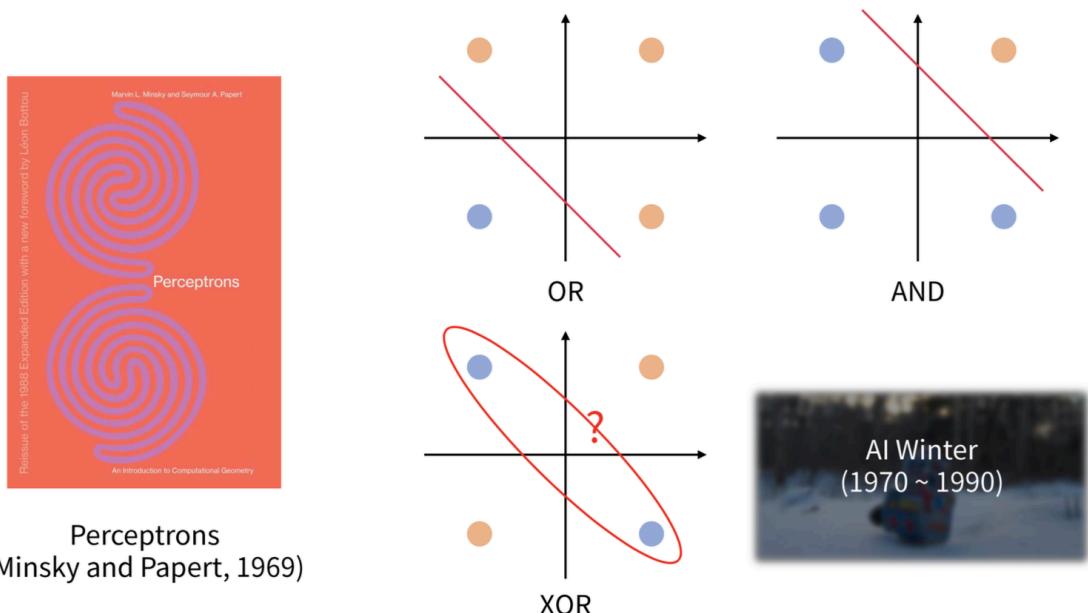
(0, 1, 1, 1)

3 퍼셉트론의 한계

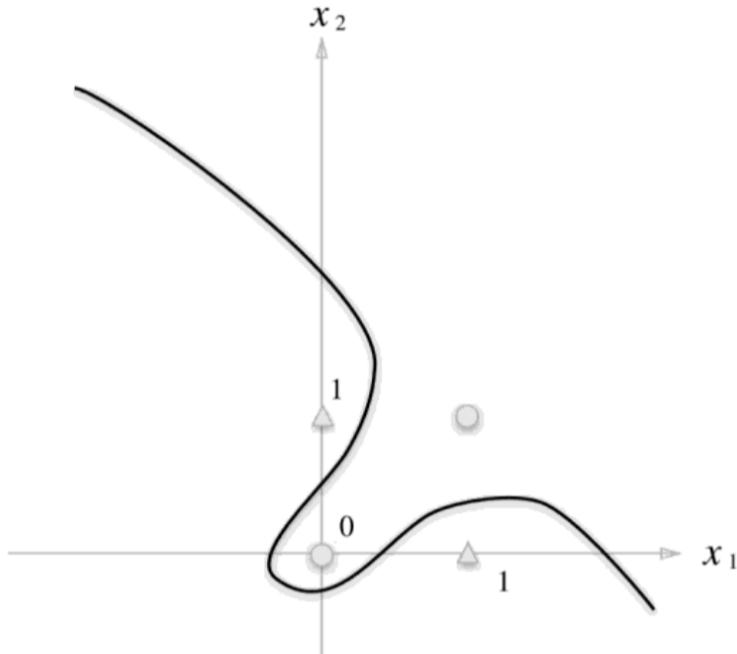
3.1 유명한 XOR

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

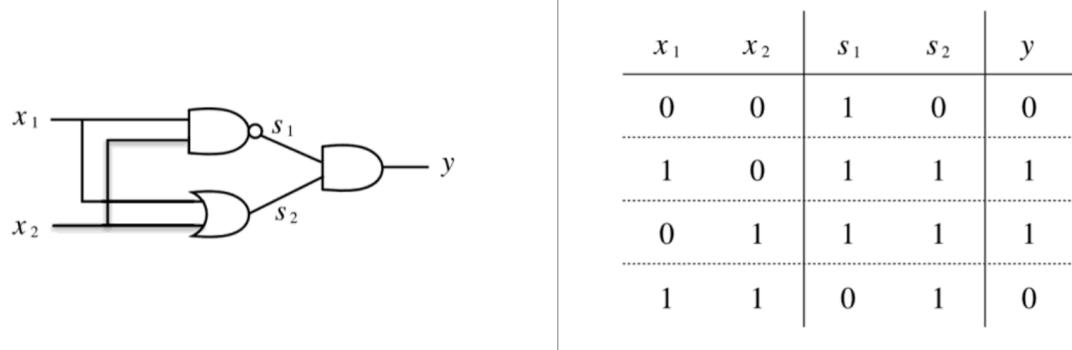
3.2 XOR 문제



3.3 해결 방법은 직선이 아니라 곡선



3.4 이렇게



3.5 코드는?

```

def XOR(x1, x2):
    s1 = NAND(x1, x2)
    s2 = OR(x1, x2)
    y = AND(s1, s2)

    return y
  
```

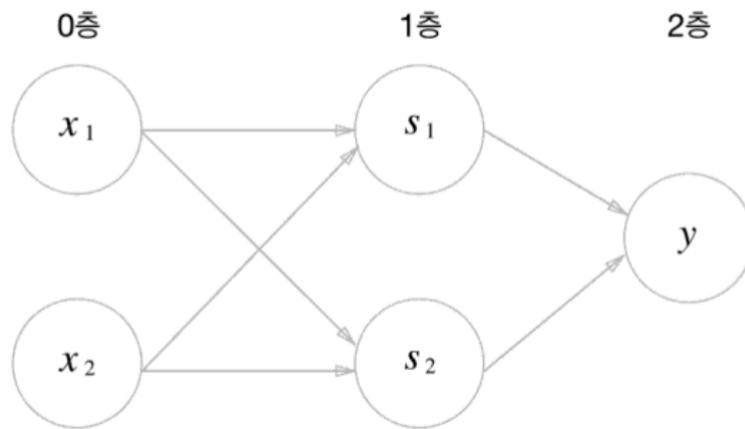
✓ 0.0s

Python

3.6 결과는??

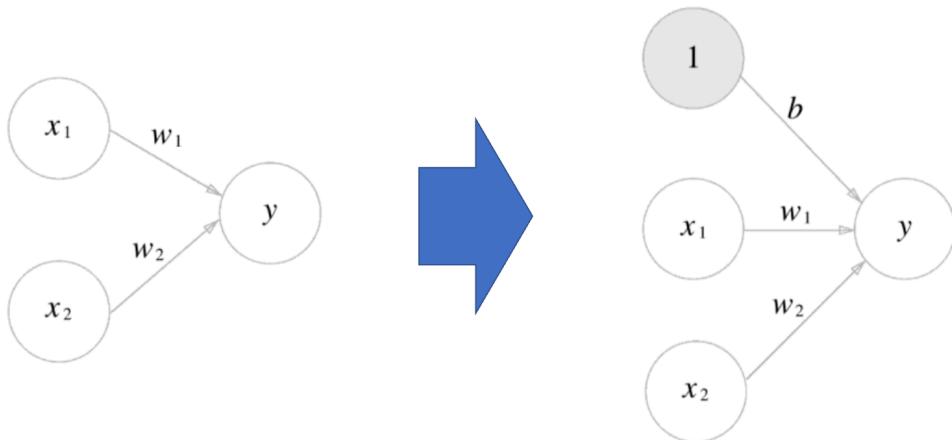
```
XOR(0, 0), XOR(1, 0), XOR(0, 1), XOR(1, 1)
✓ 0.0s
(0, 1, 1, 0) Python
```

3.7 다층의 퍼셉트론이 필요하다



4 신경망

4.1 편향을 고려하면 이렇게 표현할 수도 있다



4.2 아무튼 우리는 이런 수식을 가지고 표현하고 있는 중

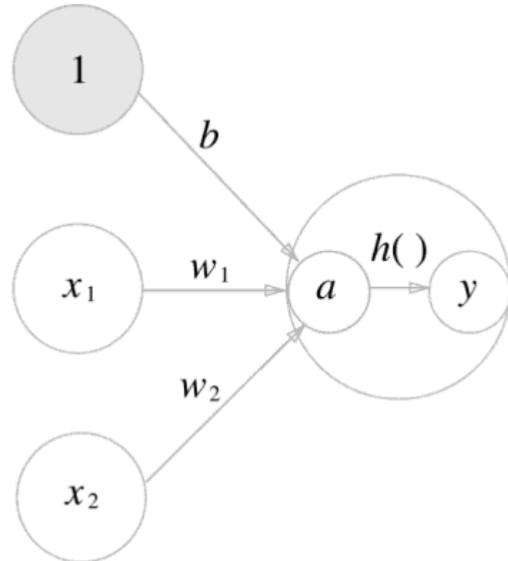
$$y = \begin{cases} 0 & (b + w_1x_1 + w_2x_2 \leq 0) \\ 1 & (b + w_1x_1 + w_2x_2 > 0) \end{cases}$$

4.3 활성화 함수의 등장 - 여기서는 h

$$a = b + w_1x_1 + w_2x_2$$

$$y = h(a)$$

4.4 그림으로 표현



4.5 만약 0과 1의 출력을 내고 싶다면 활성화 함수는 시그모이드

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

4.6 시그모이드 함수 만들기

```
def sigmoid(x):
    return 1 / (1 + np.exp(-x))
```

Python

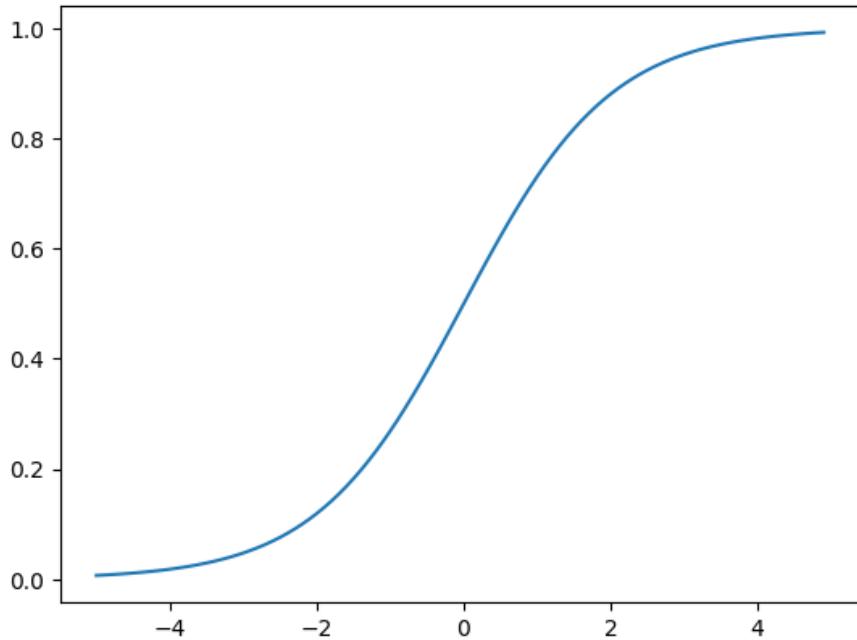
4.7 그려서 확인해보기

```
import matplotlib.pyplot as plt

x = np.arange(-5.0, 5.0, 0.1)
y = sigmoid(x)
plt.plot(x, y)
plt.show()
```

✓ 0.3s

Python



4.8 활성화함수는 비선형이 주로 사용됨

- 함수의 합성에서 선형함수를 사용하면 합성의 의미가 없음

4.9 또 다른 활성화함수 ReLU

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

4.10 생간건~

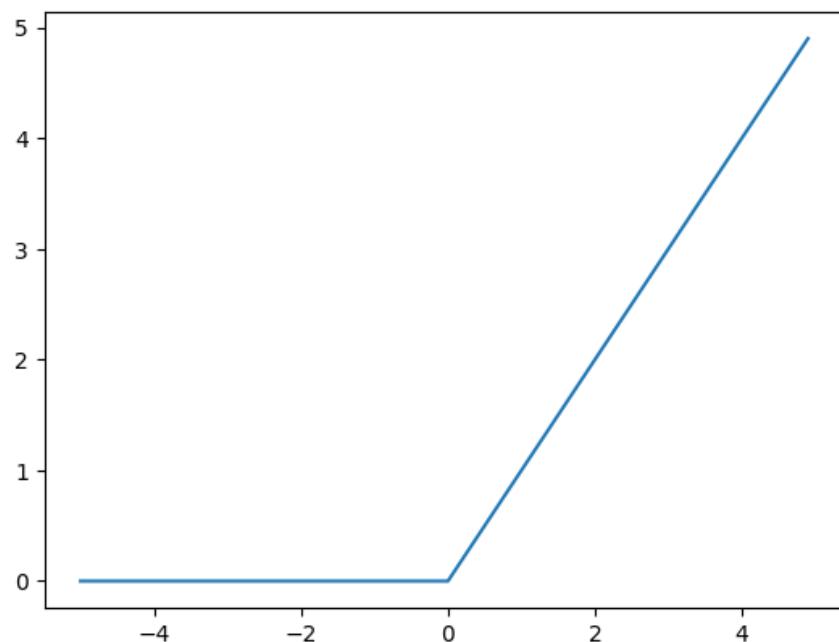
```
def relu(x):
    return np.maximum(0, x)
✓ 0.0s
```

Python

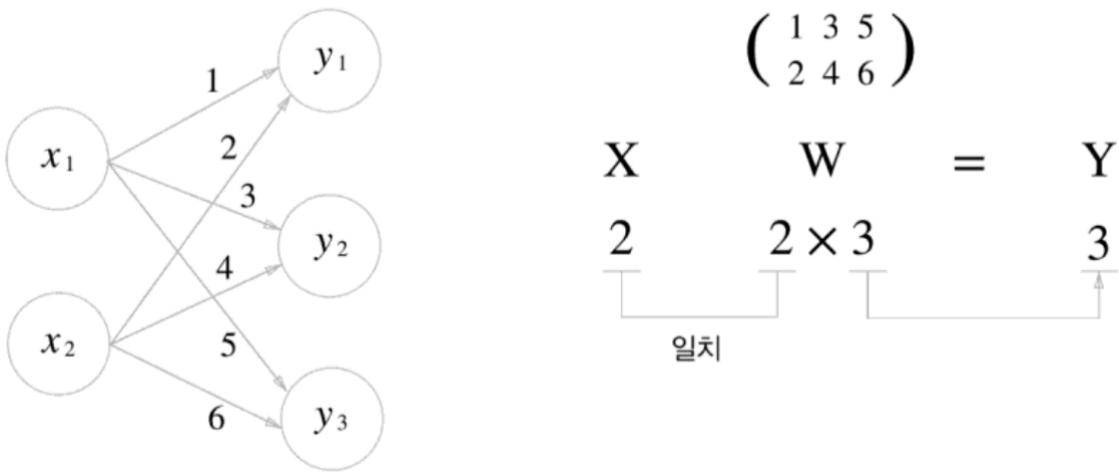
```
import matplotlib.pyplot as plt

x = np.arange(-5.0, 5.0, 0.1)
y = relu(x)
plt.plot(x, y)
plt.show()
✓ 0.0s
```

Python



4.11 신경망은 행렬의 곱 연산이 많이 사용된다



4.12 이렇게 행렬을 정의하고

```
X = np.array([1, 2])
W = np.array([[1, 5, 7], [2, 4, 6]])

X.shape, W.shape
✓ 0.0s
```

Python

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

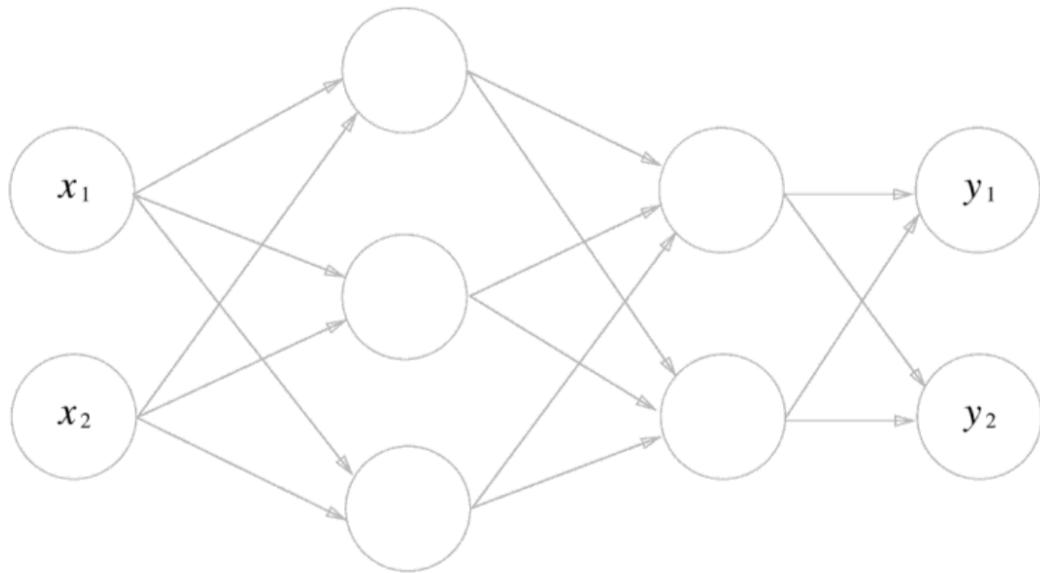
4.13 연산하면 그만~

```
np.dot(X, W)
✓ 0.0s
```

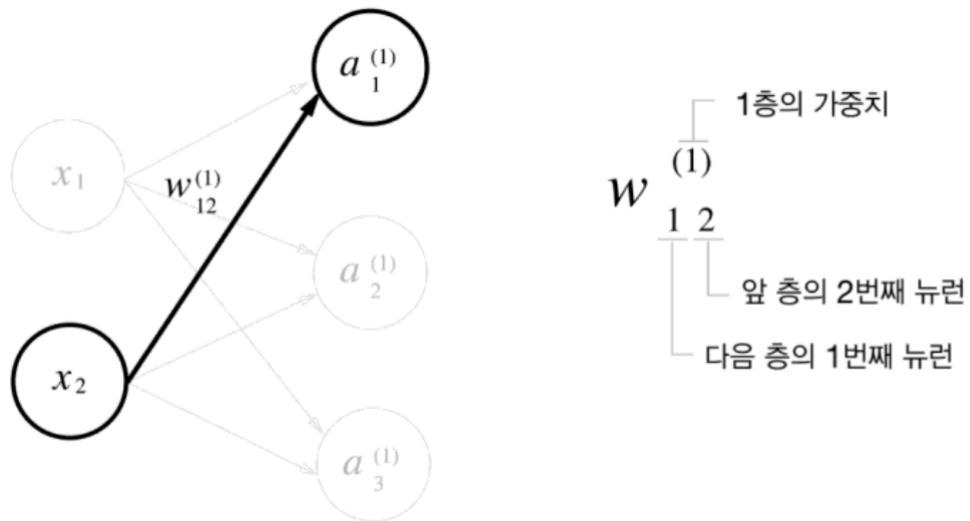
Python

array([5, 13, 19])

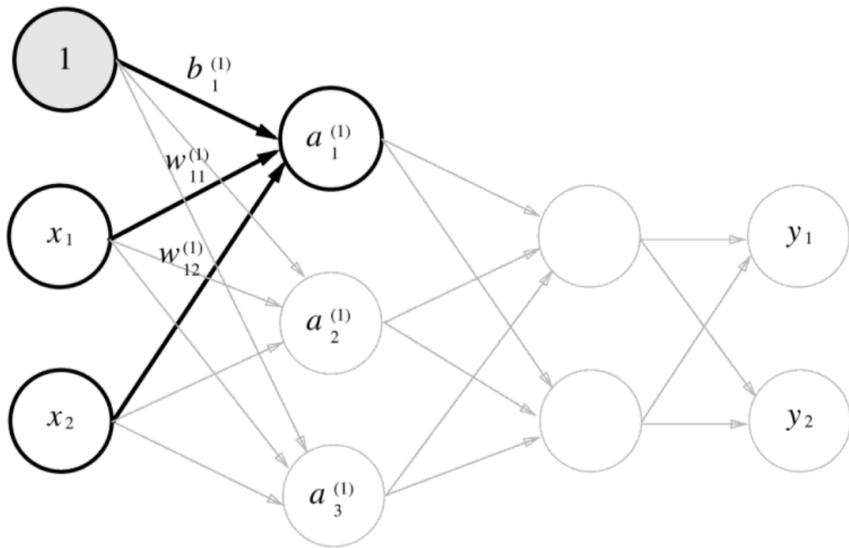
4.14 3층 신경망



4.15 흔히 사용하는 가중치의 표기법



4.16 입력층에서 그 다음단계로 진행하는 첫 과정



4.17 하나의 노드에 대한 수식

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

4.18 은닉층 (1층)에 대한 수식

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

$$\mathbf{A}^{(1)} = (a_1^{(1)} \ a_2^{(1)} \ a_3^{(1)}), \quad \mathbf{X} = (x_1 \ x_2), \quad \mathbf{B}^{(1)} = (b_1^{(1)} \ b_2^{(1)} \ b_3^{(1)})$$

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

4.19 코드? 쉽지

```
X = np.array([1.0, 0.5])
W1 = np.array([[0.1, 0.3, 0.5], [0.2, 0.4, 0.6]])
B1 = np.array([0.1, 0.2, 0.3])

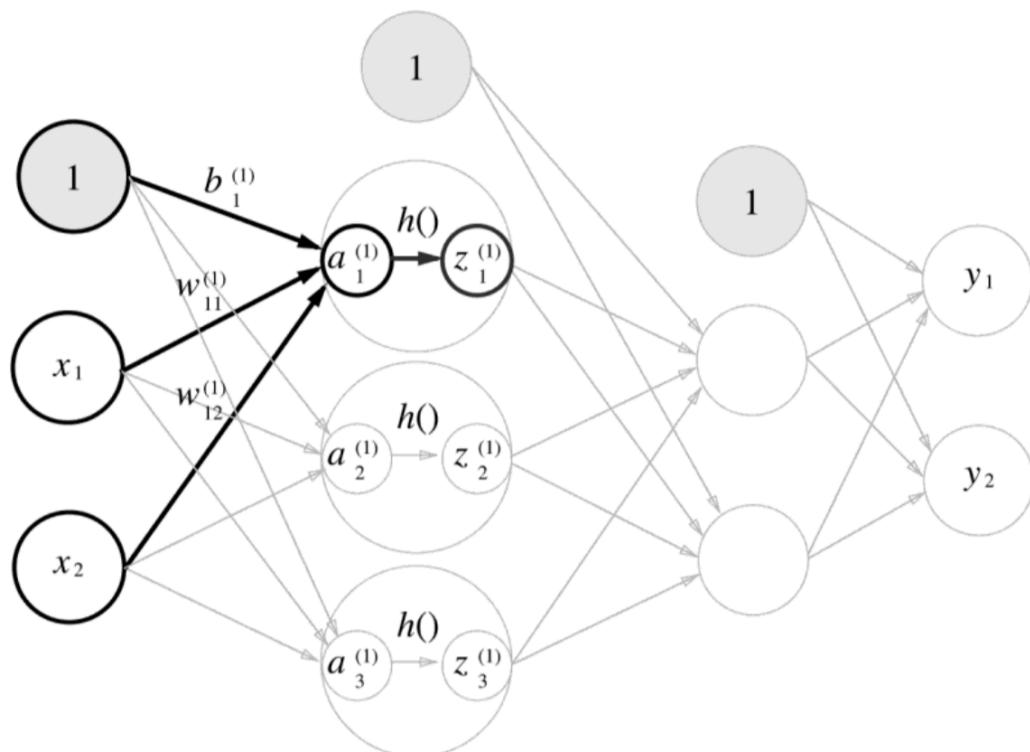
print(W1.shape)
print(X.shape)
print(B1.shape)

A1 = np.dot(X, W1) + B1
✓ 0.0s
```

Python

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

4.20 그런데 사실 하나 더 있다.. 바로 활성화 함수



4.21 여기까지 코드로 하면

```
Z1 = sigmoid(A1)

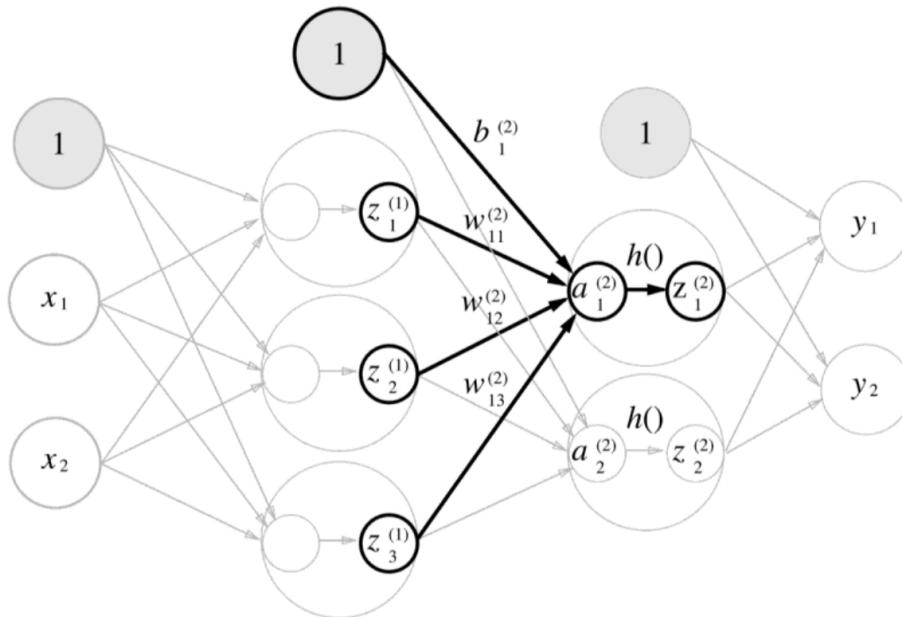
print(A1)
print(Z1)

✓ 0.0s
```

Python

```
[0.3 0.7 1.1]
[0.57444252 0.66818777 0.75026011]
```

4.22 이제 일층에서 이층으로 가자



4.23 이제 그림보다 코드가 더 쉬울껄

```

W2 = np.array([[0.1, 0.4], [0.2, 0.5], [0.3, 0.6]])
B2 = np.array([0.1, 0.2])

print(Z1.shape)
print(W2.shape)
print(B2.shape)

A2 = np.dot(Z1, W2) + B2
Z2 = sigmoid(A2)

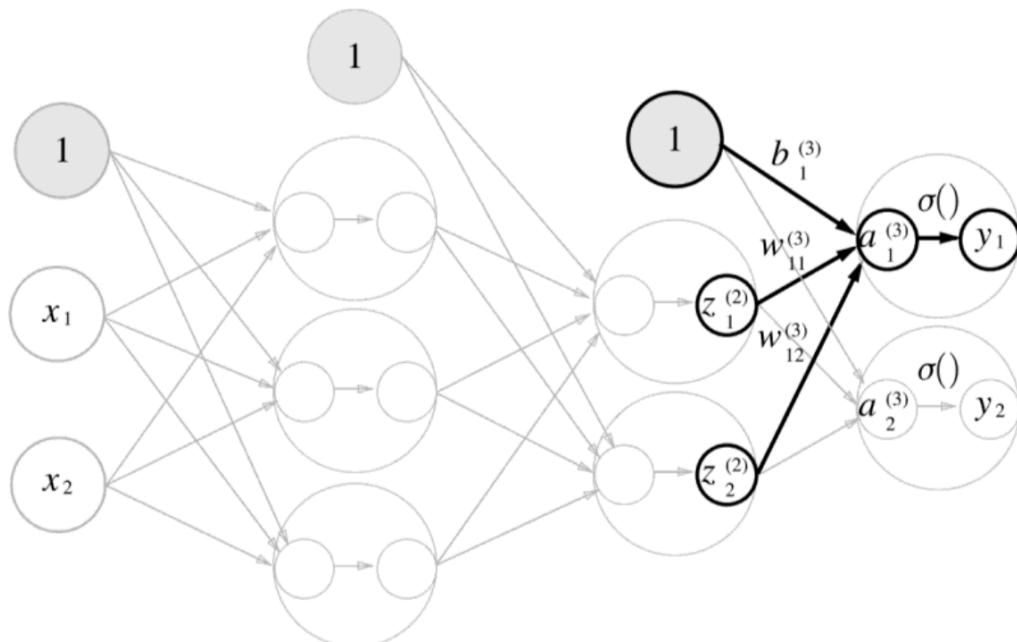
```

✓ 0.0s

Python

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

4.24 이층에서 삼층 - 출력층으로 가자



4.25 코드로는 요렇게~

```
def identity_function(x):
    return x

W3 = np.array([[0.1, 0.3], [0.2, 0.4]])
B3 = np.array([0.1, 0.2])

A3 = np.dot(Z2, W3) + B3
Y = identity_function(A3)

✓ 0.0s
```

Python

4.26 약간 멋진 모습으로 forward 혹은 inference를 구현해 본다면

4.27 모델의 가중치들

```
def init_network():
    network = {}
    network['W1'] = np.array([[0.1, 0.3, 0.5], [0.2, 0.4, 0.6]])
    network['b1'] = np.array([0.1, 0.2, 0.3])
    network['W2'] = np.array([[0.1, 0.4], [0.2, 0.5], [0.3, 0.6]])
    network['b2'] = np.array([0.1, 0.2])
    network['W3'] = np.array([[0.1, 0.3], [0.2, 0.4]])
    network['b3'] = np.array([0.1, 0.2])

    return network

✓ 0.0s
```

Python

4.28 모델의 forward 연산

```
def forward(network, x):
    W1, W2, W3 = network["W1"], network["W2"], network["W3"]
    b1, b2, b3 = network["b1"], network["b2"], network["b3"]

    a1 = np.dot(X, W1) + b1
    z1 = sigmoid(a1)

    a2 = np.dot(z1, W2) + b2
    z2 = sigmoid(a2)

    a3 = np.dot(z2, W3) + b3
    y = identity_function(a3)

    return y

✓ 0.0s
```

Python

4.29 추론~

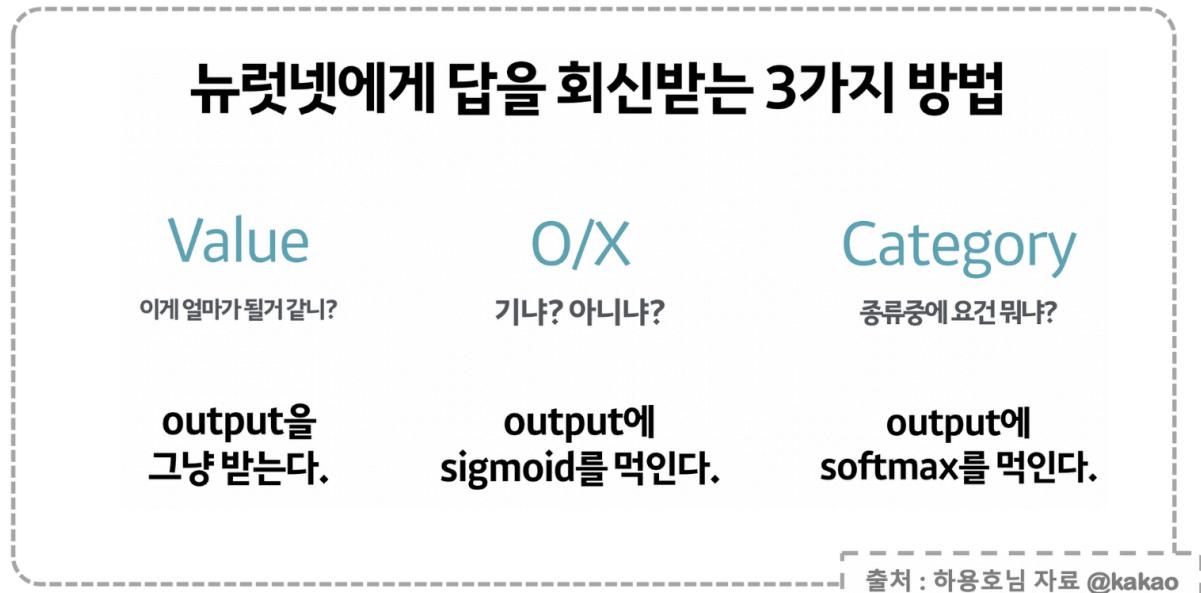
```
network = init_network()
x = np.array([1.0, 0.5])
y = forward(network, x)
y
✓ 0.0s
```

Python

```
array([0.31682708, 0.69627909])
```

5 출력층의 종류

5.1 모델의 목표에 따라



5.2 Softmax 함수

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

5.3 Softmax를 코드로 하면

```
def softmax(a):
    return np.exp(a) / np.sum(np.exp(a))
```

Python

5.4 누가 제일 큰지 알려준다

```
softmax([1, 2, 3])
array([0.09003057, 0.24472847, 0.66524096])
```

Python

5.5 여기서 잠깐~ exp의 overflow 문제

```
np.exp(10), np.exp(100), np.exp(1000)
✓ 0.0s
Python
/tmp/ipykernel_68186/1145674931.py:1: RuntimeWarning: overflow encountered in exp
np.exp(10), np.exp(100), np.exp(1000)

(np.float64(22026.465794806718),
 np.float64(2.6881171418161356e+43),
 np.float64(inf))
```

5.6 당연히 softmax도 에러가 난다

```
softmax([10, 100, 1000])
✓ 0.0s
Python
/tmp/ipykernel_68186/675541793.py:2: RuntimeWarning: overflow encountered in exp
    return np.exp(a) / np.sum(np.exp(a))
/tmp/ipykernel_68186/675541793.py:2: RuntimeWarning: invalid value encountered in
divide
    return np.exp(a) / np.sum(np.exp(a))

array([ 0.,  0., nan])
```

5.7 극복할 방법은 수학의 마법

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

5.8 코드로는 이렇게

```
def softmax(a):
    c = np.max(a)
    exp_a = np.exp(a - c)
    return exp_a / np.sum(exp_a)
```

✓ 0.0s

Python

5.9 에러가 안난다

```
softmax([10, 100, 1000])
```

✓ 0.0s

Python

```
array([0., 0., 1.])
```

6 MNIST에 적용

6.1 MNIST 다운 받기

- <https://www.kaggle.com/datasets/oddrationale/mnist-in-csv?resource=download>

The screenshot shows the Kaggle dataset page for 'MNIST in CSV'. At the top, there's a user profile icon, the name 'DARIEL DATO-ON', and a note 'UPDATED 7 YEARS AGO'. To the right are buttons for 'New Notebook' and 'Download', with 'Download' highlighted by a red box. Below this is a section titled 'MNIST in CSV' with the subtext 'The MNIST dataset provided in a easy-to-use CSV format'. Underneath are links for 'Data Card', 'Code (707)', 'Discussion (2)', and 'Suggestions (0)'. A large 'About Dataset' section follows, containing a heading 'The MNIST dataset provided in a easy-to-use CSV' with a red box around it. Below this, text explains that the original dataset is in a difficult format and was converted by Joseph Redmon. It also states that the dataset consists of two files: 'mnist_train.csv' and 'mnist_test.csv'. To the right of this text is a code block showing how to download the dataset using the Kaggle API. At the bottom right of the page are 'Tags' for 'Computer Science', 'Image', and 'Beginner'. A large red box highlights the 'Download dataset as zip (16 MB)' button.

MNIST in CSV

The MNIST dataset provided in a easy-to-use CSV format

Data Card Code (707) Discussion (2) Suggestions (0)

About Dataset

The MNIST dataset provided in a easy-to-use CSV

The [original dataset](#) is in a format that is difficult for beginners to use. That's why [Joseph Redmon](#) to provide the [MNIST dataset in a CSV format](#).

The dataset consists of two files:

1. `mnist_train.csv`
2. `mnist_test.csv`

The `mnist_train.csv` file contains the 60,000 training examples and labels. The `mnist_test.csv` contains 10,000 test examples and labels. Each row consists of 785 values: the first value is the label (a number from 0 to 9) and the remaining 784 values are the pixel values (a number from 0 to 255).

Tags

Computer Science Image Beginner

DOWNLOAD VIA

kagglehub

New to Kaggle API? Here's how to [set up your API keys](#).

```
import kagglehub

# Download latest version
path = kagglehub.dataset_download("oddrationale/mnist-in-csv")

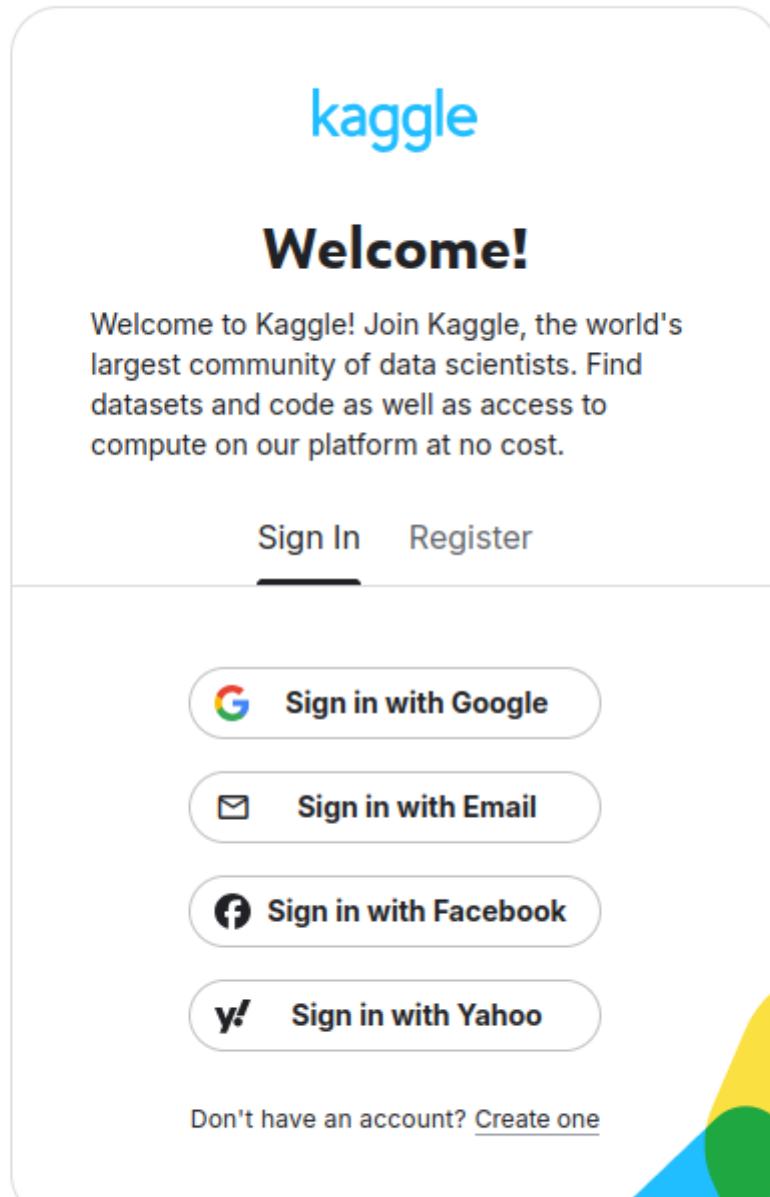
print("Path to dataset files:", path)
```

Download dataset as zip (16 MB)

Export metadata as Croissant

- 데이터셋을 다운 받아줍니다.

6.2 가입이 안되어있으신가요?



- 머신러닝 / 딥러닝을 해보기에 재미있는 다양한 데이터가 있으니 가입해보시는건 어떨까요?

6.3 MNIST 읽기

```
import pandas as pd

df_train = pd.read_csv("./data/mnist_train.csv")
df_test = pd.read_csv("./data/mnist_test.csv")

df_train.shape, df_test.shape
✓ 1.9s
```

Python

6.4 데이터를 numpy로

```
X_train = np.array(df_train.iloc[:, 1:])
y_train = np.array(df_train["label"])

X_test = np.array(df_test.iloc[:, 1:])
y_test = np.array(df_test["label"])
✓ 0.0s
```

Python

6.5 Image를 보여주는 함수

```
import matplotlib.pyplot as plt

def img_show(img):
    plt.imshow(img, cmap="gray")
    plt.axis("off")
    plt.show()
✓ 0.0s
```

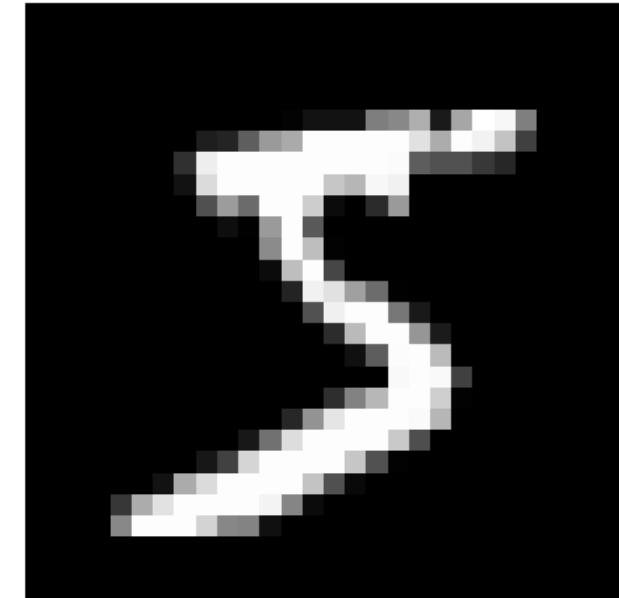
Python

6.6 그려주세요

```
img = X_train[0]
label = y_train[0]

img = img.reshape(28, 28)

img_show(img)
```



Python

6.7 미리 가중치를 읽고 - 다음에 학습하는 법을 배웁니다

- https://github.com/oreilly-japan/deep-learning-from-scratch/blob/01dd7a0ad931a034d0dfb6dd8284af5706ad01e4/ch03/sample_weight.pkl

```
import pickle

def init_network():
    with open("./data/sample_weight.pkl", "rb") as f:
        network = pickle.load(f)
    return network
```

✓ 0.0s

Python

6.8 모델을 구현하고

```
def predict(network, x):
    W1, W2, W3 = network["W1"], network["W2"], network["W3"]
    b1, b2, b3 = network["b1"], network["b2"], network["b3"]

    a1 = np.dot(x, W1) + b1
    z1 = sigmoid(a1)

    a2 = np.dot(z1, W2) + b2
    z2 = sigmoid(a2)

    a3 = np.dot(z2, W3) + b3
    y = softmax(a3)

    return y
```

✓ 0.0s

Python

6.9 와우~ Accuracy 92.5%

```
network = init_network()
accuracy_cnt = 0

for i in range(len(X_train)):
    y_pred = predict(network, X_train[i])
    p = np.argmax(y_pred)
    if p == y_train[i]:
        accuracy_cnt += 1

print("Accuracy : " + str(float(accuracy_cnt) / len(X_train)))
```

✓ 2.1s

Python

[/tmp/ipykernel_73329/1253679003.py:2](#): RuntimeWarning: overflow encountered in exp

$$\text{return } 1 \wedge (1 + \exp(-x))$$

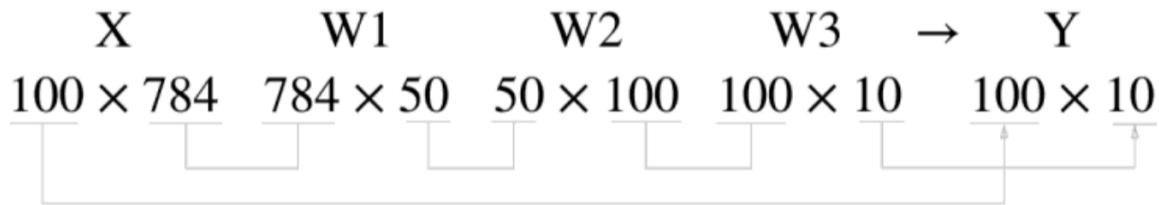
Accuracy : 0.9251833333333334

6.10 혹시 지금까지의 연산의 흐름을 알겠나요?



- 딱 코드와 맞진 않은 부분(Y)이 있지만 개념적으로는~

6.11 만약 100개씩 묶어서 처리한다면~



6.12 이 코드가 유명한 배치사이즈를 설정한 코드

```
batch_size = 100
accuracy_cnt = 0

for i in range(0, len(X_train), batch_size):
    X_batch = X_train[i : i + batch_size]
    y_batch = predict(network, X_batch)
    p = np.argmax(y_batch, axis=1)
    accuracy_cnt += np.sum(p == y_train[i : i + batch_size])

print("Accuracy : " + str(float(accuracy_cnt) / len(X_train)))
```

✓ 0.4s Python
`/tmp/ipykernel_73329/1253679003.py:2: RuntimeWarning: overflow encountered in exp`
`return 1 / (1 + np.exp(-x))`
`Accuracy : 0.9251833333333334`