

10 & 11기 정규세션

ToBig's 10기 신혼철

Deep Learning

Neural Network - 1

contents

Unit 01 | 딥러닝이란?

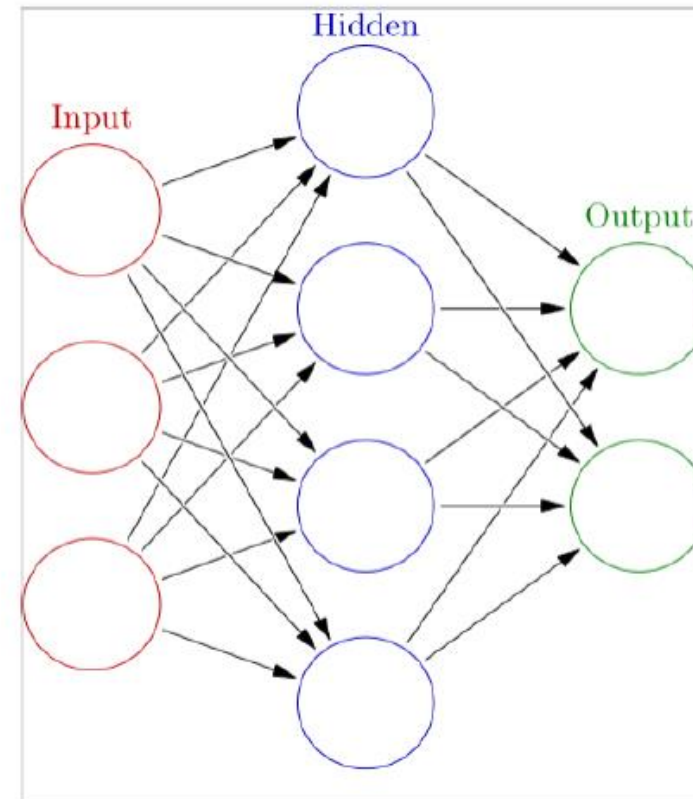
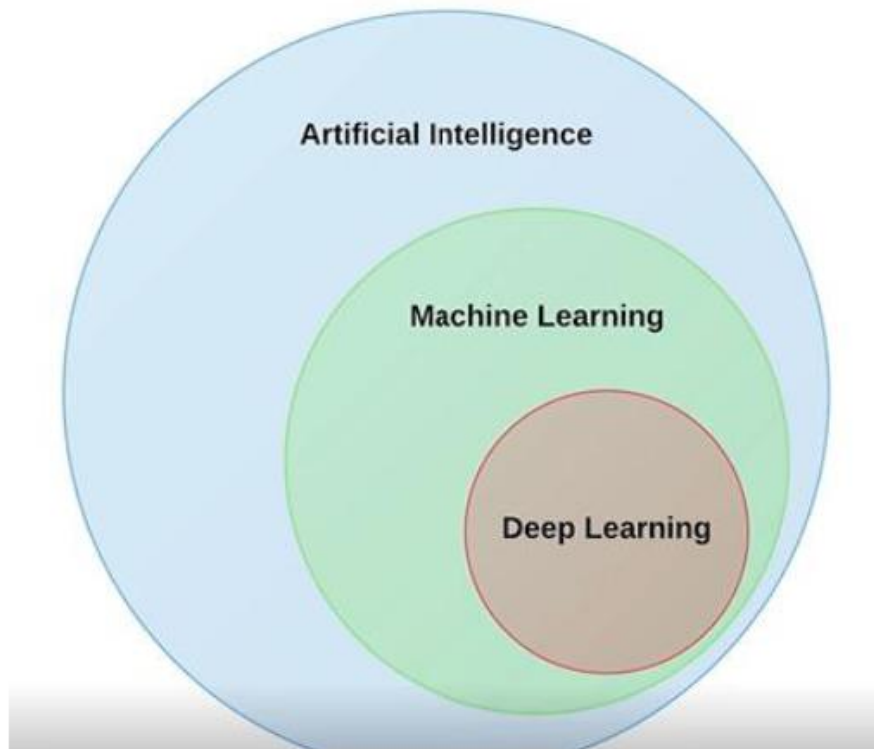
Unit 02 | Neural Network – 기본 개념

Unit 03 | 퍼셉트론

Unit 04 | Neural Network – Forward

Unit 05 | Neural Network – Backpropagation

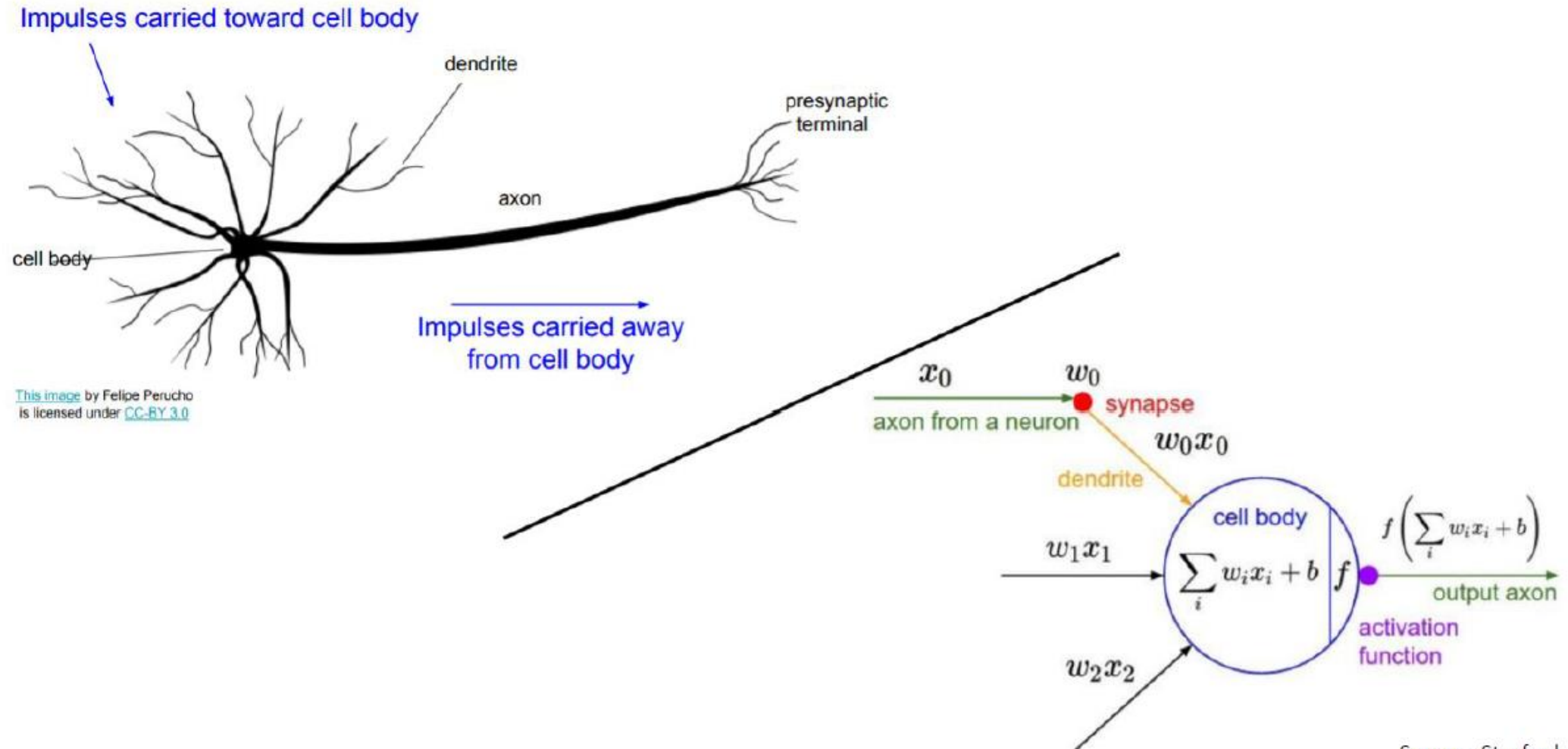
Unit 01 | 딥러닝이란?



Artificial Neural Network


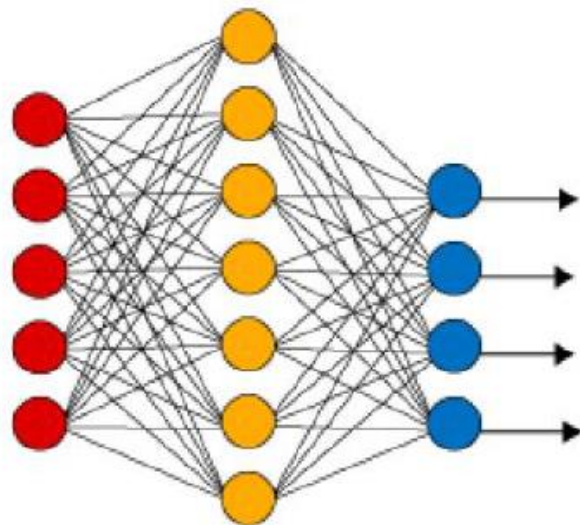

Source : <https://www.youtube.com/watch?v=aF03asAmQbY&feature=youtu.be>
https://en.wikipedia.org/wiki/Artificial_neural_network

Unit 01 | 딥러닝이란?

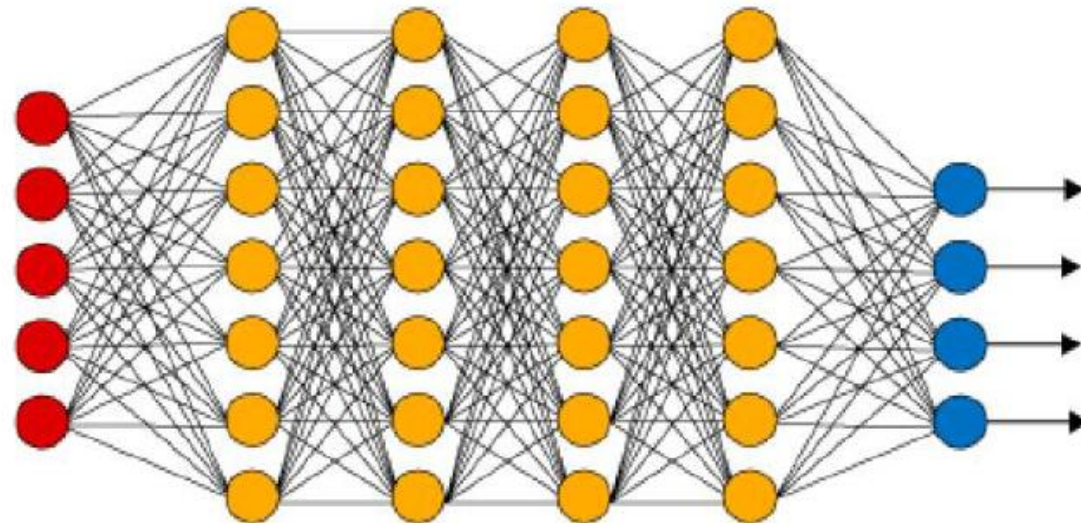


Unit 01 | 딥러닝이란?

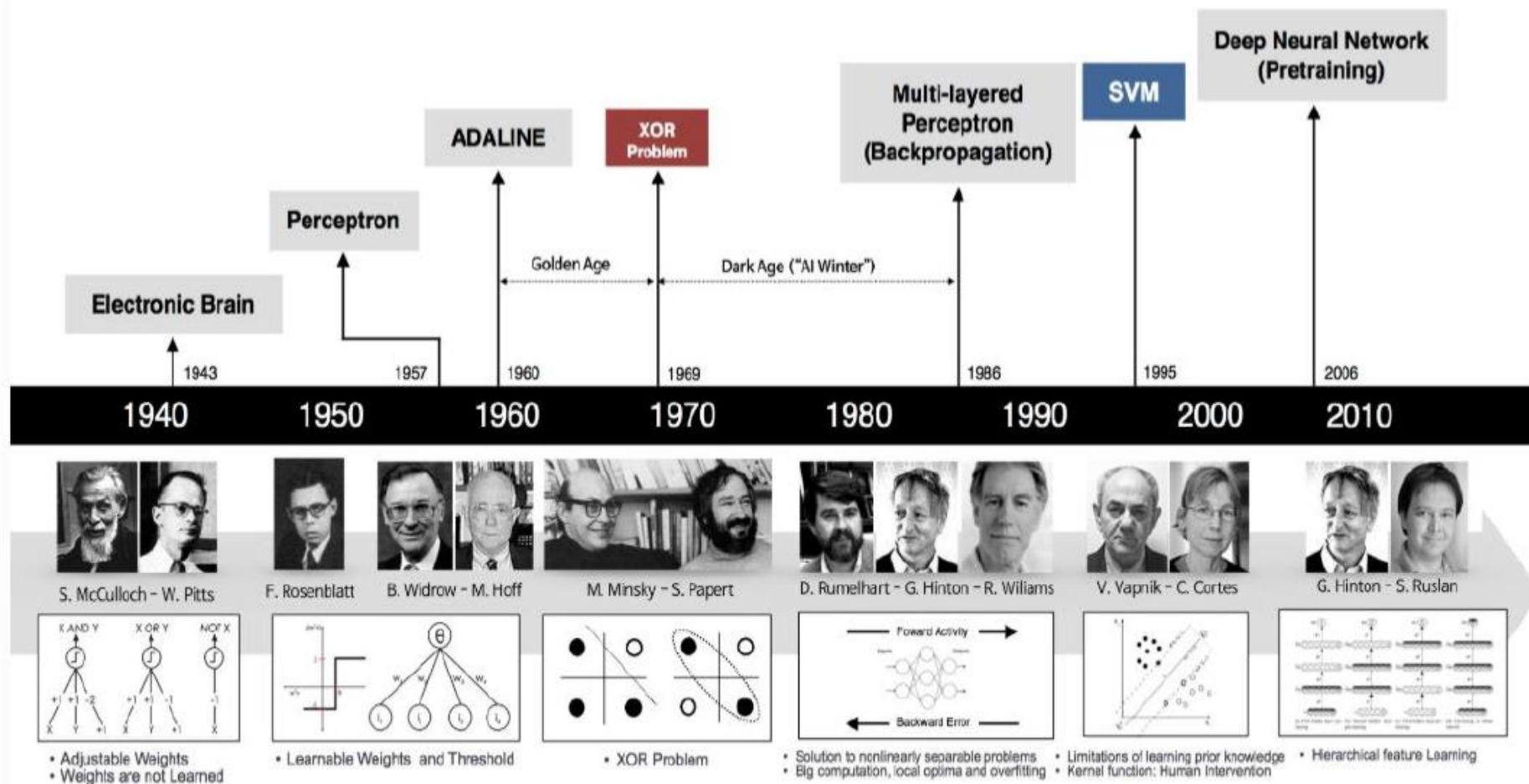
Simple Neural Network

 Input Layer Hidden Layer Output Layer

Deep Learning Neural Network



Unit 01 | 딥러닝이란?



Unit 01 | 딥러닝이란?

Why is Deep Learning Hot **Now**?

1. Big Data

- Larger Datasets
- Easier Collection & Storage

IMAGENET



2. Hardware

- Graphics Processing Units (GPUs)
- Massively Parallelizable



3. Software

- Improved Techniques
- New Models
- Toolboxes



Unit 01 | 딥러닝이란?

Deep Learning Success: Vision

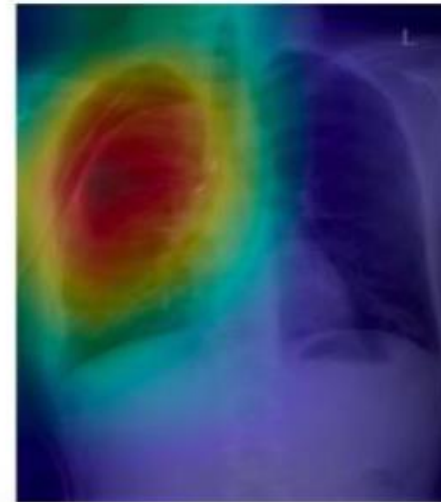
Image Recognition



Unit 01 | 딥러닝이란?

Deep Learning Success: Vision

Detect pneumothorax in real X-Ray scans



Unit 01 | 딥러닝이란?

Deep Learning Success: Audio

Music Generation



Unit 01 | 딥러닝이란?

Deep Learning Success

And so many more...



Unit 01 | 딥러닝이란?

동영상 시청:
남세동의 인공지능(딥러닝) 이야기

<https://www.youtube.com/watch?v=kMGEpIYPCiM>

Unit 02 | NN-기본개념

그래서 Neural Network가 뭔데?

Linear + Activation + ... = 분류기

Unit 02 | NN-기본개념

Linear + Activation +

Linear + Activation +

... +

Linear + 분류기

Unit 02 | NN-기본 개념

Linear function

$$W_{11}X_1 + W_{12}X_2 + W_{13}X_3 + \cdots + b_1 = H_1$$

$$W_{21}X_1 + W_{22}X_2 + W_{23}X_3 + \cdots + b_2 = H_2$$

•
•
•

$$W_{H1}X_1 + W_{H2}X_2 + W_{H3}X_3 + \cdots + b_H = H_H$$

결국 여기서도 회귀식이!

X 변수 개수만큼 가중치 생성

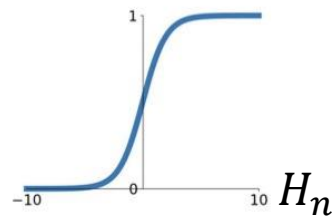
만드는 만큼 만들 수 있다!

Unit 02 | NN-기본개념

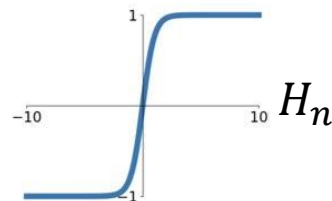
Activation function

Sigmoid

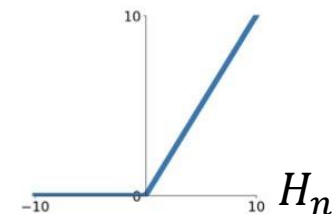
$$\sigma(x) = \frac{1}{1+e^{-x}}$$

**tanh**

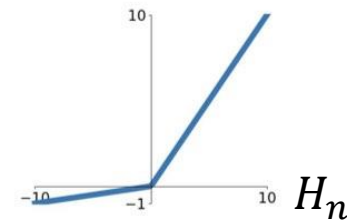
$$\tanh(x)$$

**ReLU**

$$\max(0, x)$$

**Leaky ReLU**

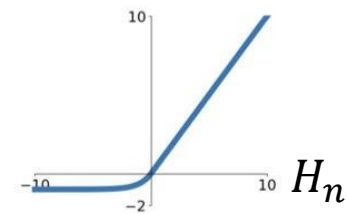
$$\max(0.1x, x)$$

**Maxout**

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



선형과 비선형이
만난다는 것

Unit 02 | NN-기본개념

분류기

Softmax Classifier (Multinomial Logistic Regression)



$$W_{11}X_1 + W_{12}X_2 + W_{13}X_3 + \dots + b_1 = H_1$$

cat

car

frog

3.2

5.1

-1.7

exp

24.5

164.0

0.18

normalize

0.13

0.87

0.00

unnormalized log probabilities

unnormalized probabilities

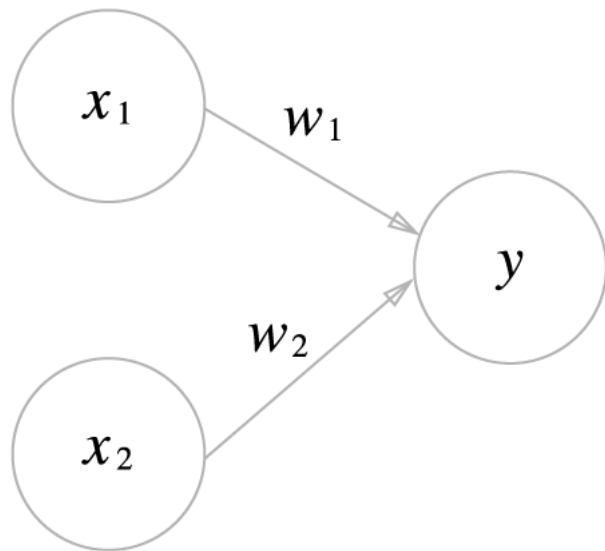
$$L_i = -\log\left(\frac{e^{sy_i}}{\sum_j e^{sj}}\right)$$

$$p_i = \frac{e^{s_i}}{\sum e^{s_j}} = \frac{24.5}{24.5 + 164 + 0.18} = 0.13$$

probabilities

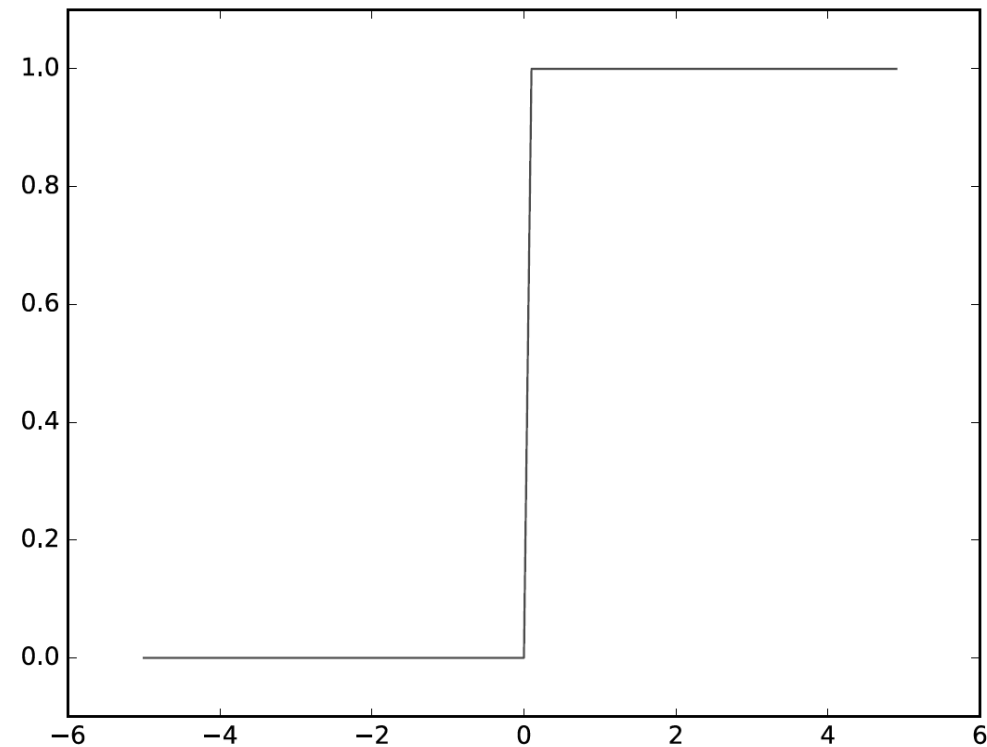
Sigmoid 분류기의
일반화

Unit 03 | 퍼셉트론



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

Linear function



Activation function : 계단 함수

Unit 03 | 퍼셉트론

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

AND 문제

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

NAND 문제

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

OR 문제

Unit 03 | 퍼셉트론

$$w_1 = 0.5, w_2 = 0.5, b = -0.7$$

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

AND 문제

$$w_1 = -0.5, w_2 = -0.5, b = 0.7$$

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

NAND 문제

$$w_1 = 0.5, w_2 = 0.5, b = -0.2$$

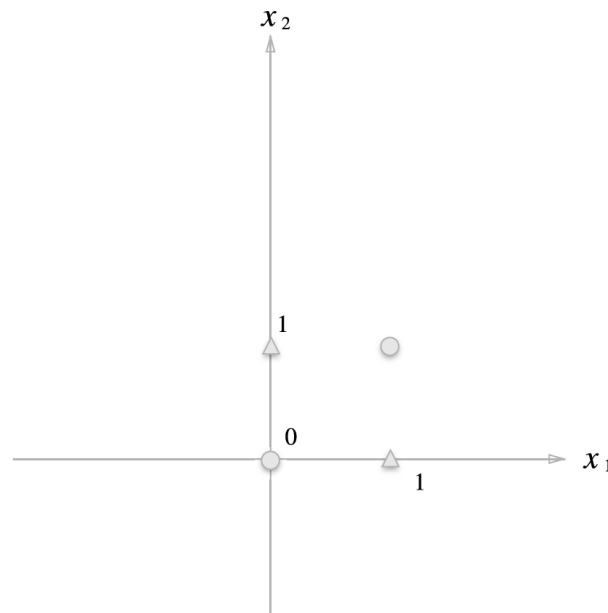
x_1	x_2	y
0	0	0
1	0	1
0	1	1
1	1	1

OR 문제

Unit 03 | 퍼셉트론

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

XOR 문제



어떤 선형으로도 XOR
문제를 풀 수 없다.

인공지능 암흑기의
첫 원인

층을 쌓는 것으로 해결!

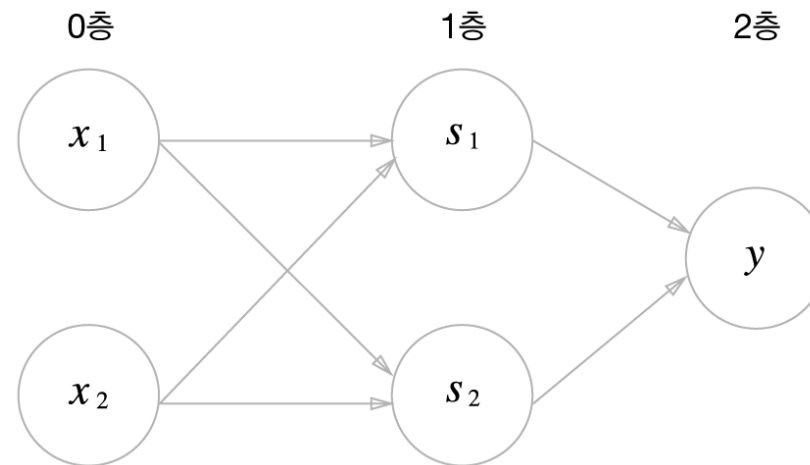
Unit 03 | 퍼셉트론

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

$$s_1 \Rightarrow w_1 = -0.5, w_2 = -0.5, b = 0.7$$

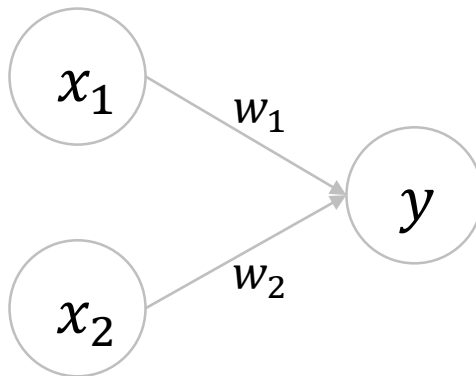
$$s_2 \Rightarrow w_1 = 0.5, w_2 = 0.5, b = -0.2$$

$$y \Rightarrow w_1 = 0.5, w_2 = 0.5, b = -0.7$$



Unit 04 | NN - Forward

Network



X1	X2	Y
1	2	1

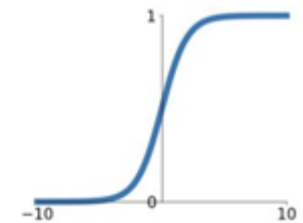
Linear

$$W_{11}X_1 + W_{12}X_2 = H$$

Activation/분류기

Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

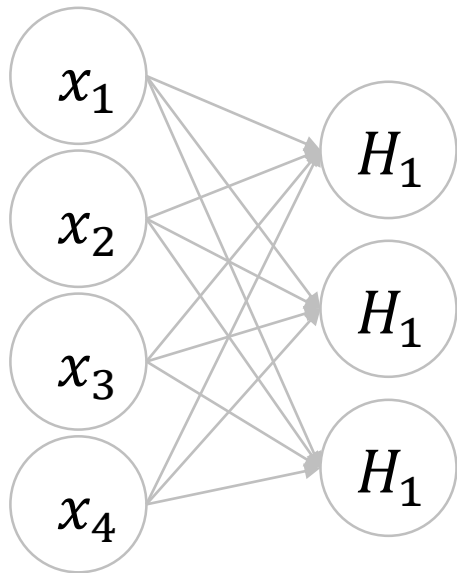


W1	W2	H	p
2	3	8	0.99966
-4	-3	-10	0.0000454

Unit 04 | NN - Forward

번외

Network



Linear

$$w_{11}x_1 + w_{12}x_2 + w_{13}x_3 + w_{14}x_4 + b_1 = H_1$$

$$w_{21}x_1 + w_{22}x_2 + w_{23}x_3 + w_{24}x_4 + b_2 = H_2$$

$$w_{31}x_1 + w_{32}x_2 + w_{33}x_3 + w_{34}x_4 + b_3 = H_3$$

Softmax

$$p_i = \frac{e^{H_i}}{\sum e^{H_j}}$$

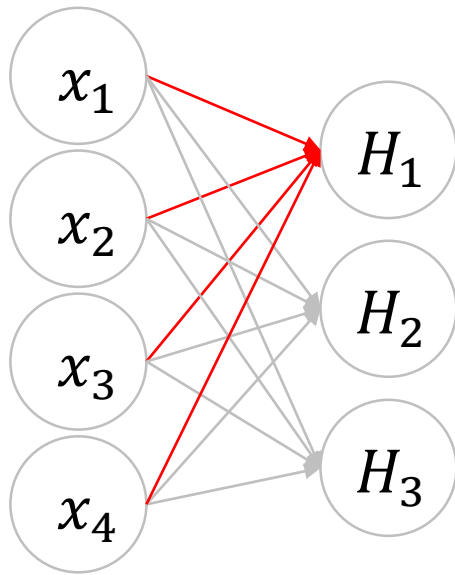
x1	x2	x3	x4	label
6.3	2.7	4.9	1.8	2

w11	w12	w13	w14	b1	H1	exp(H1)	P1
0.231595	0.613372	-0.89624	-0.41208	1.123884	-0.89428	0.408902	0.035734
w21	w22	w23	w24	b2	H2	exp(H2)	P2
0.076107	-0.22382	0.20313	-0.05043	0.319879	1.099584	3.002917	0.262425
w31	w32	w33	w34	b3	H3	exp(H3)	P3
-0.30772	-0.38941	0.693168	0.462622	0.844128	2.083326	8.031135	0.701841

Unit 04 | NN - Forward

번외

Network



Linear

$$w_{11}x_1 + w_{12}x_2 + w_{13}x_3 + w_{14}x_4 + b_1 = H_1$$

$$w_{21}x_1 + w_{22}x_2 + w_{23}x_3 + w_{24}x_4 + b_2 = H_2$$

$$w_{31}x_1 + w_{32}x_2 + w_{33}x_3 + w_{34}x_4 + b_3 = H_3$$

Softmax

$$p_i = \frac{e^{H_i}}{\sum e^{H_j}}$$

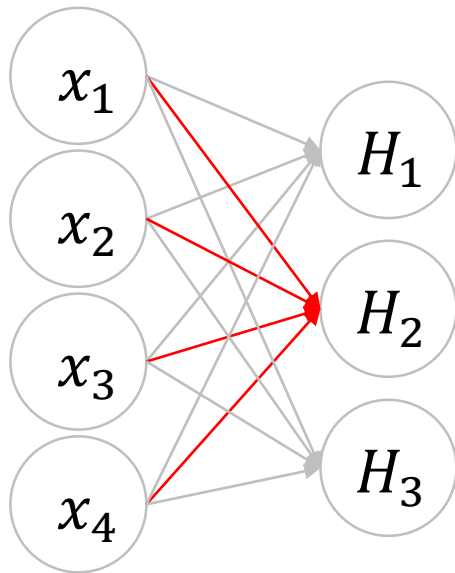
x1	x2	x3	x4	label
6.3	2.7	4.9	1.8	2

w11	w12	w13	w14	b1	H1	exp(H1)	P1
0.231595	0.613372	-0.89624	-0.41208	1.123884	-0.89428	0.408902	0.035734
w21	w22	w23	w24	b2	H2	exp(H2)	P2
0.076107	-0.22382	0.20313	-0.05043	0.319879	1.099584	3.002917	0.262425
w31	w32	w33	w34	b3	H3	exp(H3)	P3
-0.30772	-0.38941	0.693168	0.462622	0.844128	2.083326	8.031135	0.701841

Unit 04 | NN - Forward

번외

Network



Linear

$$w_{11}x_1 + w_{12}x_2 + w_{13}x_3 + w_{14}x_4 + b_1 = H_1$$

$$w_{21}x_1 + w_{22}x_2 + w_{23}x_3 + w_{24}x_4 + b_2 = H_2$$

$$w_{31}x_1 + w_{32}x_2 + w_{33}x_3 + w_{34}x_4 + b_3 = H_3$$

Softmax

$$p_i = \frac{e^{H_i}}{\sum e^{H_j}}$$

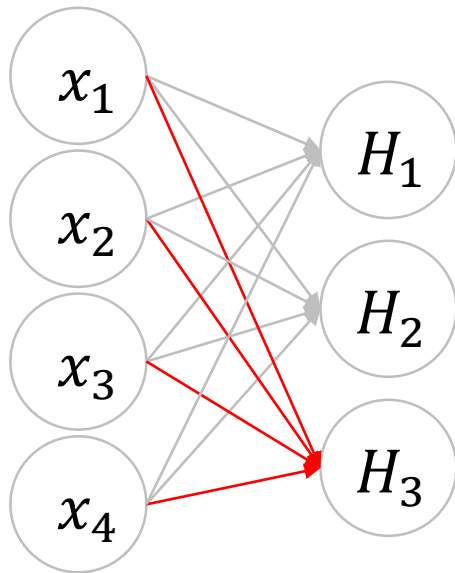
x1	x2	x3	x4	label
6.3	2.7	4.9	1.8	2

w11	w12	w13	w14	b1	H1	exp(H1)	P1
0.231595	0.613372	-0.89624	-0.41208	1.123884	-0.89428	0.408902	0.035734
w21	w22	w23	w24	b2	H2	exp(H2)	P2
0.076107	-0.22382	0.20313	-0.05043	0.319879	1.099584	3.002917	0.262425
w31	w32	w33	w34	b3	H3	exp(H3)	P3
-0.30772	-0.38941	0.693168	0.462622	0.844128	2.083326	8.031135	0.701841

Unit 04 | NN - Forward

번외

Network



Linear

$$w_{11}x_1 + w_{12}x_2 + w_{13}x_3 + w_{14}x_4 + b_1 = H_1$$

$$w_{21}x_1 + w_{22}x_2 + w_{23}x_3 + w_{24}x_4 + b_2 = H_2$$

$$w_{31}x_1 + w_{32}x_2 + w_{33}x_3 + w_{34}x_4 + b_3 = H_3$$

Softmax

$$p_i = \frac{e^{H_i}}{\sum e^{H_j}}$$

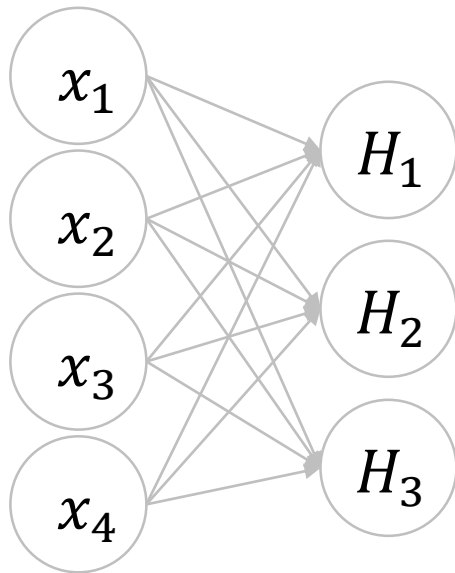
x1	x2	x3	x4	label
6.3	2.7	4.9	1.8	2

w11	w12	w13	w14	b1	H1	exp(H1)	P1
0.231595	0.613372	-0.89624	-0.41208	1.123884	-0.89428	0.408902	0.035734
w21	w22	w23	w24	b2	H2	exp(H2)	P2
0.076107	-0.22382	0.20313	-0.05043	0.319879	1.099584	3.002917	0.262425
w31	w32	w33	w34	b3	H3	exp(H3)	P3
-0.30772	-0.38941	0.693168	0.462622	0.844128	2.083326	8.031135	0.701841

Unit 04 | NN - Forward

번외

Network



Linear

$$w_{11}x_1 + w_{12}x_2 + w_{13}x_3 + w_{14}x_4 + b_1 = H_1$$

$$w_{21}x_1 + w_{22}x_2 + w_{23}x_3 + w_{24}x_4 + b_2 = H_2$$

$$w_{31}x_1 + w_{32}x_2 + w_{33}x_3 + w_{34}x_4 + b_3 = H_3$$

Softmax

$$p_i = \frac{e^{H_i}}{\sum e^{H_j}}$$

만약 b2에 +1을 한다면?

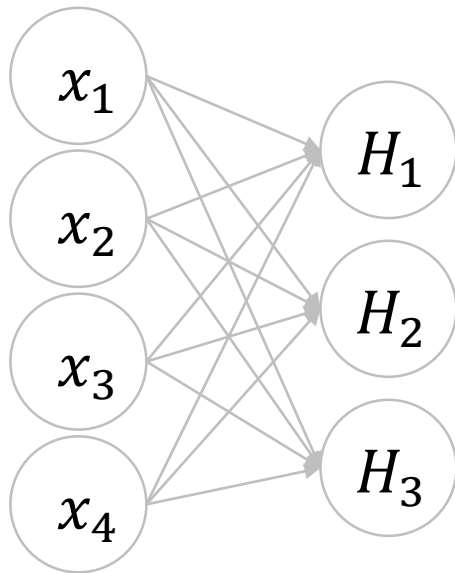
x1	x2	x3	x4	label
6.3	2.7	4.9	1.8	2

w11	w12	w13	w14	b1	H1	exp(H1)	P1
0.231595	0.613372	-0.89624	-0.41208	1.123884	-0.89428	0.408902	0.035734
w21	w22	w23	w24	b2	H2	exp(H2)	P2
0.076107	-0.22382	0.20313	-0.05043	0.319879	1.099584	3.002917	0.262425
w31	w32	w33	w34	b3	H3	exp(H3)	P3
-0.30772	-0.38941	0.693168	0.462622	0.844128	2.083326	8.031135	0.701841

Unit 04 | NN - Forward

번외

Network



Linear

$$w_{11}x_1 + w_{12}x_2 + w_{13}x_3 + w_{14}x_4 + b_1 = H_1$$

$$w_{21}x_1 + w_{22}x_2 + w_{23}x_3 + w_{24}x_4 + b_2 = H_2$$

$$w_{31}x_1 + w_{32}x_2 + w_{33}x_3 + w_{34}x_4 + b_3 = H_3$$

Softmax

$$p_i = \frac{e^{H_i}}{\sum e^{H_j}}$$

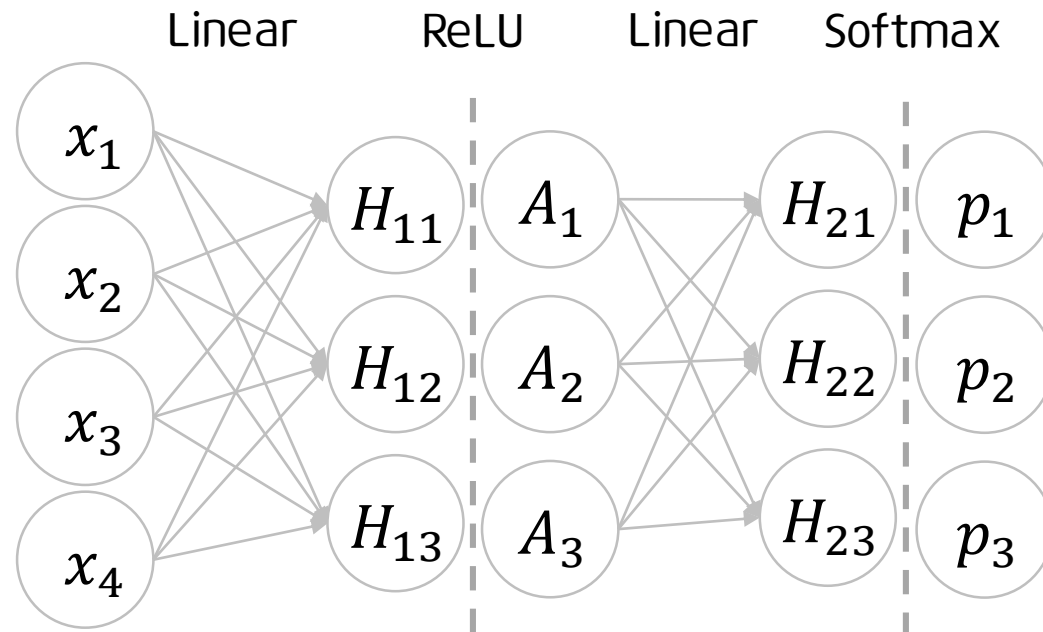
만약 b2에 +1을 한다면?

x1	x2	x3	x4	label
6.3	2.7	4.9	1.8	2

w11	w12	w13	w14	b1	H1	exp(H1)	P1
0.231595	0.613372	-0.89624	-0.41208	1.123884	-0.89428	0.408902	0.024628
w21	w22	w23	w24	b2	H2	exp(H2)	P2
0.076107	-0.22382	0.20313	-0.05043	1.319879	2.099584	8.162775	0.49165
w31	w32	w33	w34	b3	H3	exp(H3)	P3
-0.30772	-0.38941	0.693168	0.462622	0.844128	2.083326	8.031135	0.483721

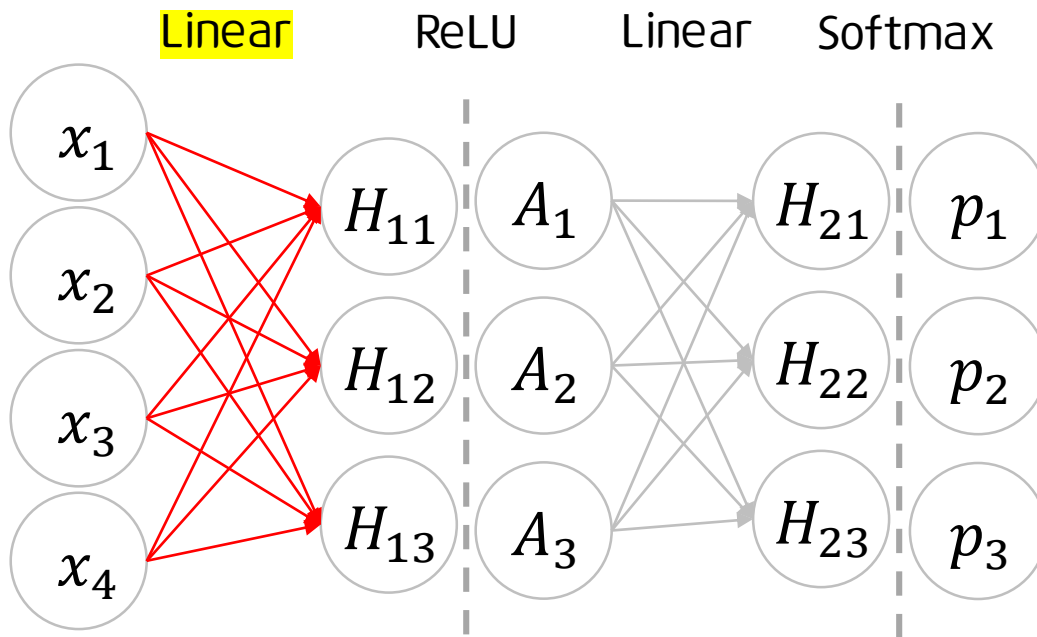
Unit 04 | NN - Forward

Network



Unit 04 | NN - Forward

Network



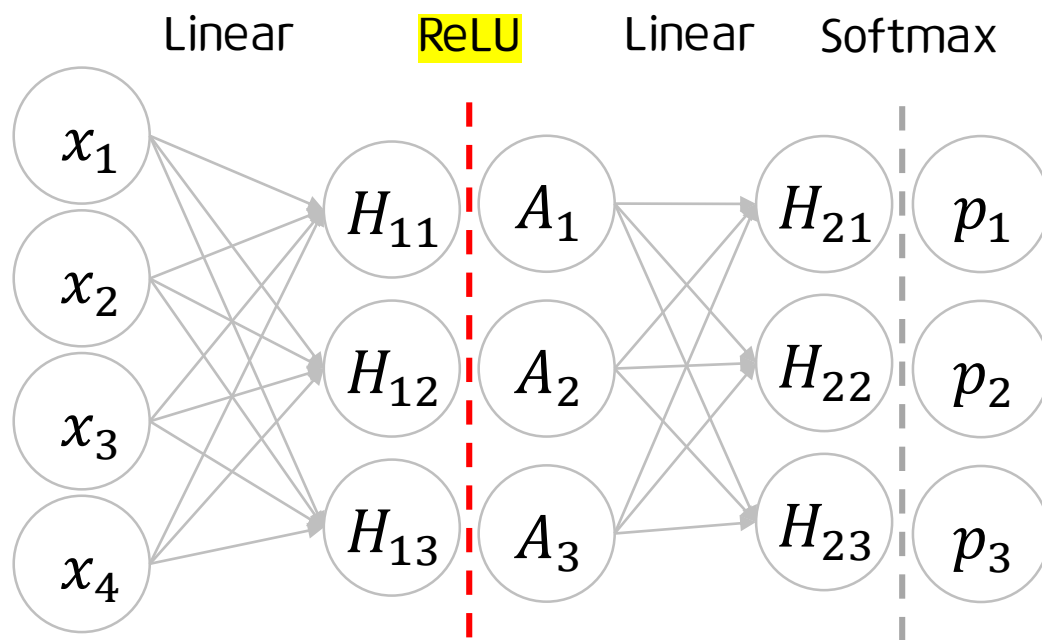
$$w_{11}x_1 + w_{12}x_2 + w_{13}x_3 + w_{14}x_4 + b_1 = H_{11}$$

$$w_{21}x_1 + w_{22}x_2 + w_{23}x_3 + w_{24}x_4 + b_2 = H_{12}$$

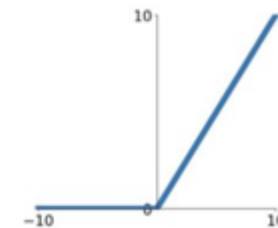
$$w_{31}x_1 + w_{32}x_2 + w_{33}x_3 + w_{34}x_4 + b_3 = H_{13}$$

Unit 04 | NN - Forward

Network

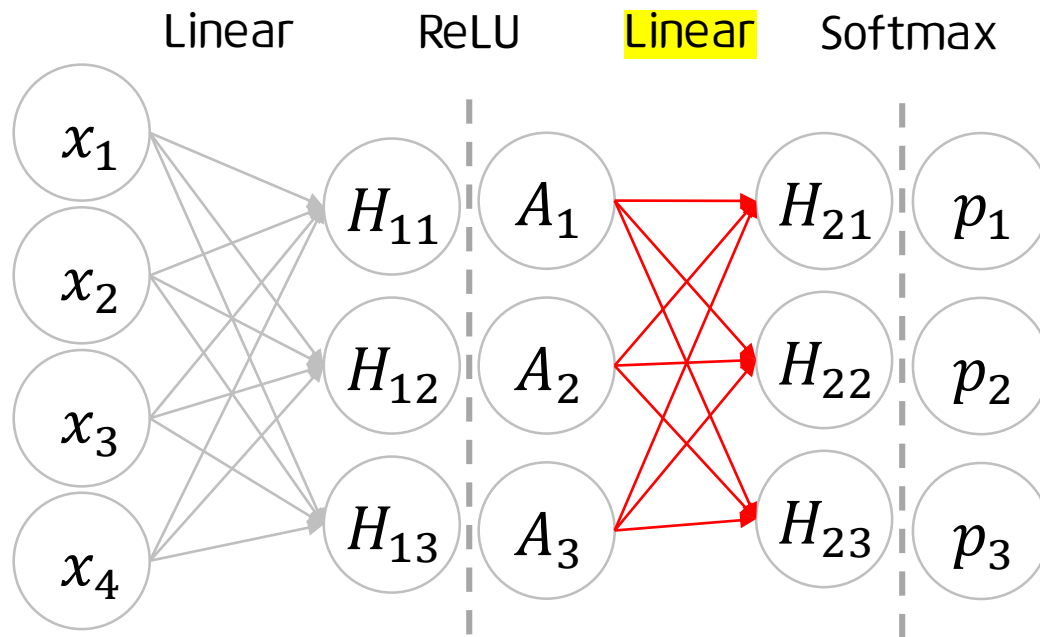


ReLU
 $\max(0, x)$



Unit 04 | NN - Forward

Network



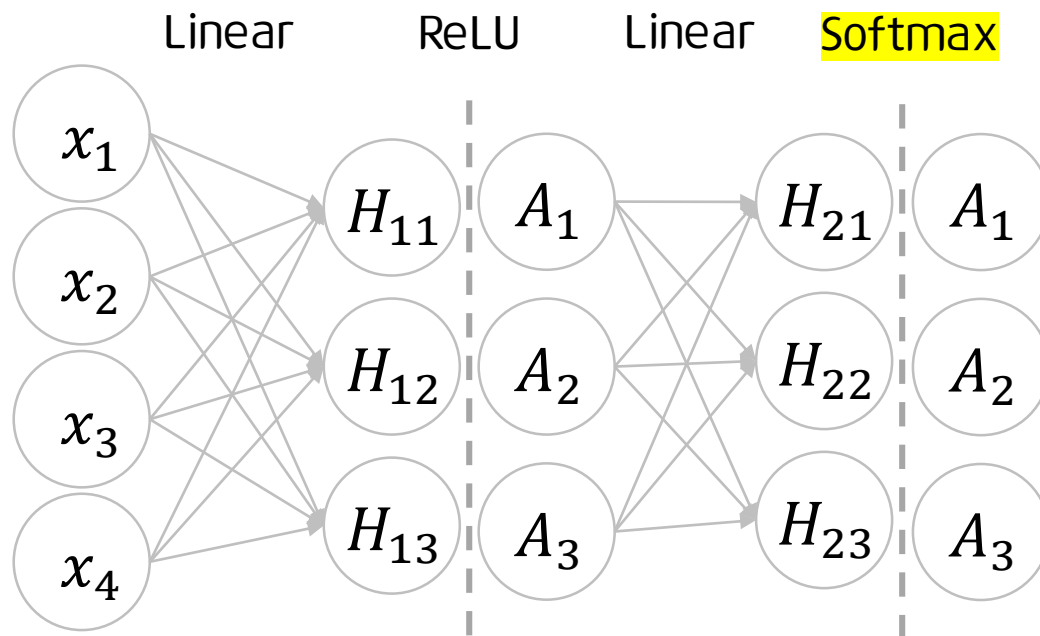
$$w'_{11}x_1 + w'_{12}x_2 + w'_{13}x_3 + b'_1 = H_{21}$$

$$w'_{21}A_1 + w'_{22}A_2 + w'_{23}A_3 + b'_2 = H_{22}$$

$$w'_{31}A_1 + w'_{32}A_2 + w'_{33}A_3 + b'_3 = H_{23}$$

Unit 04 | NN - Forward

Network



$$p_i = \frac{e^{H_i}}{\sum e^{H_j}}$$

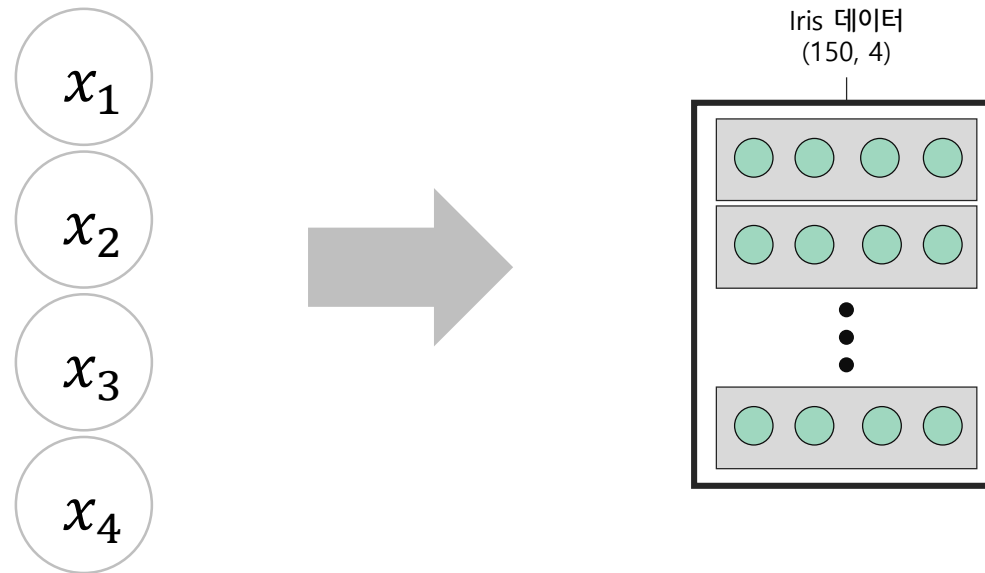
Unit 04 | NN - Forward

Network



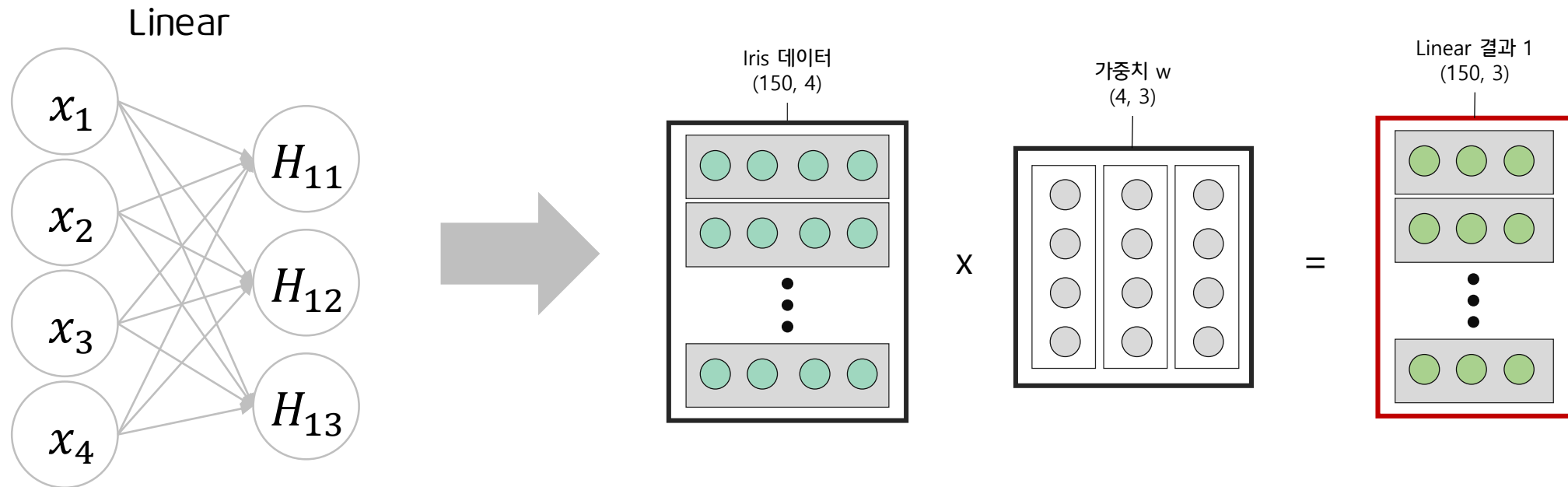
Unit 04 | NN - Forward

데이터는 이렇게 표현된다.



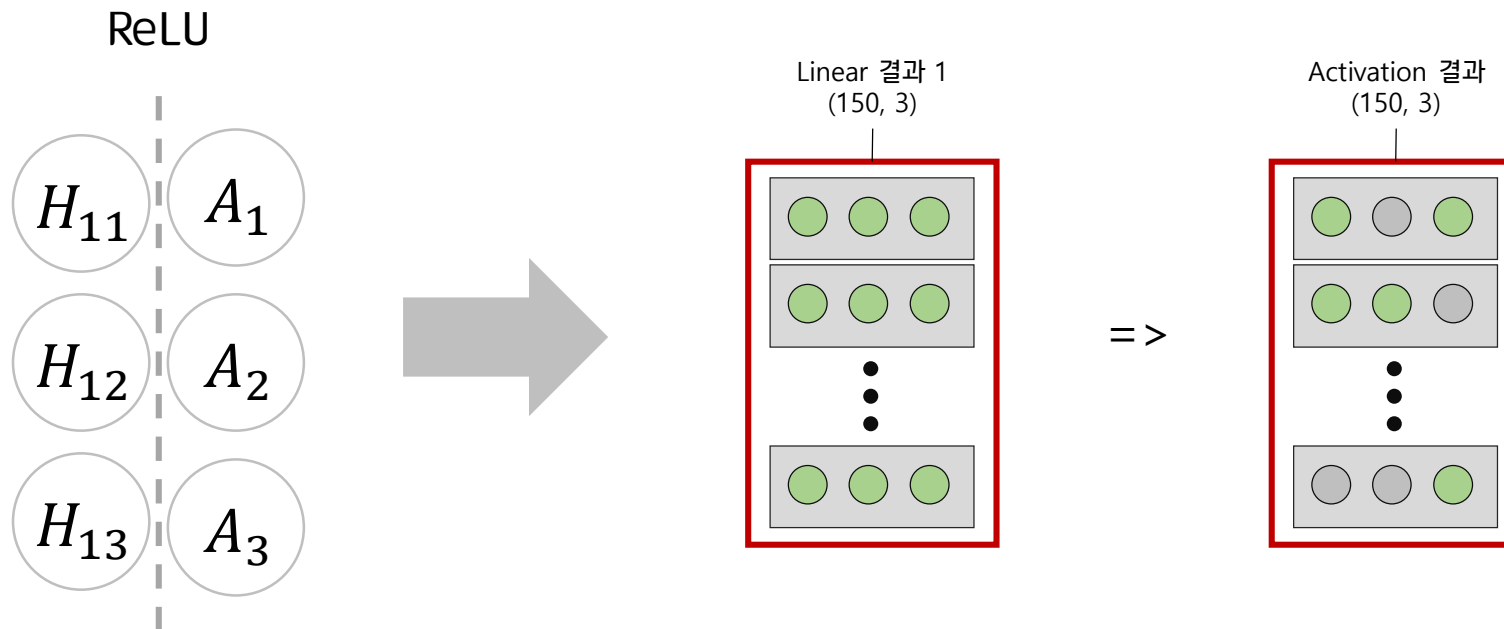
Unit 04 | NN - Forward

Linear는 이렇게 표현된다.



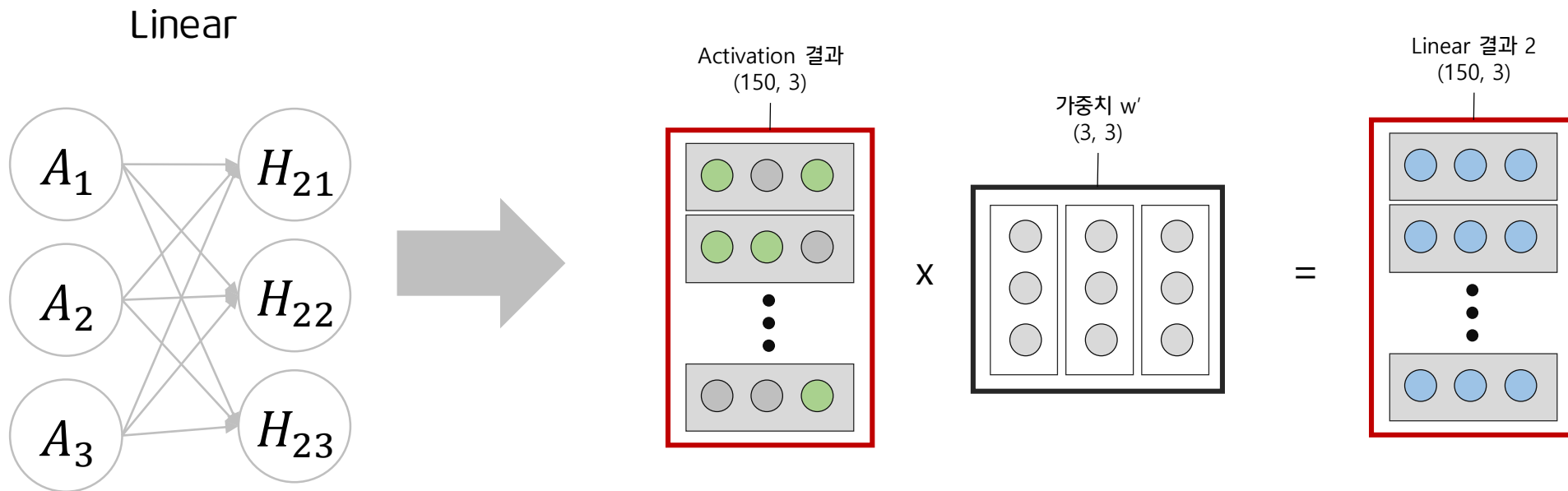
Unit 04 | NN - Forward

Activation는 이렇게 표현된다.



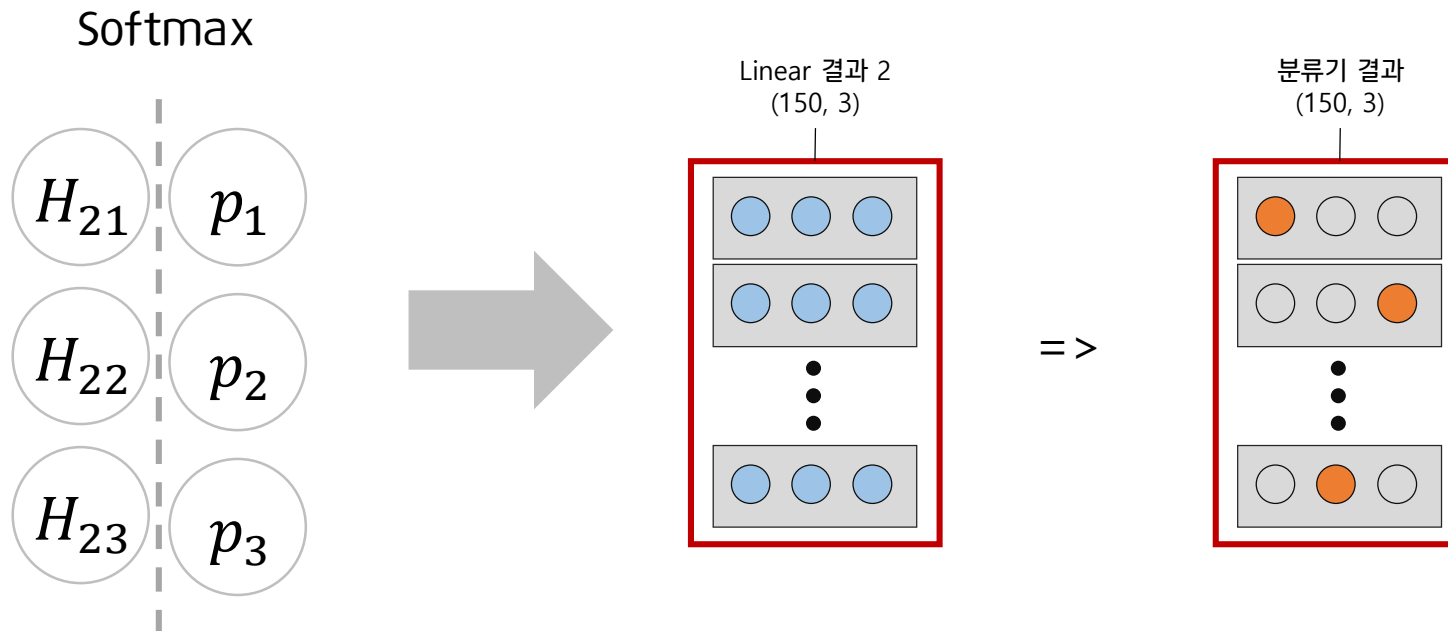
Unit 04 | NN - Forward

다시 Linear는 이렇게 표현된다.



Unit 04 | NN - Forward

분류기 결과는 이렇게 표현된다.



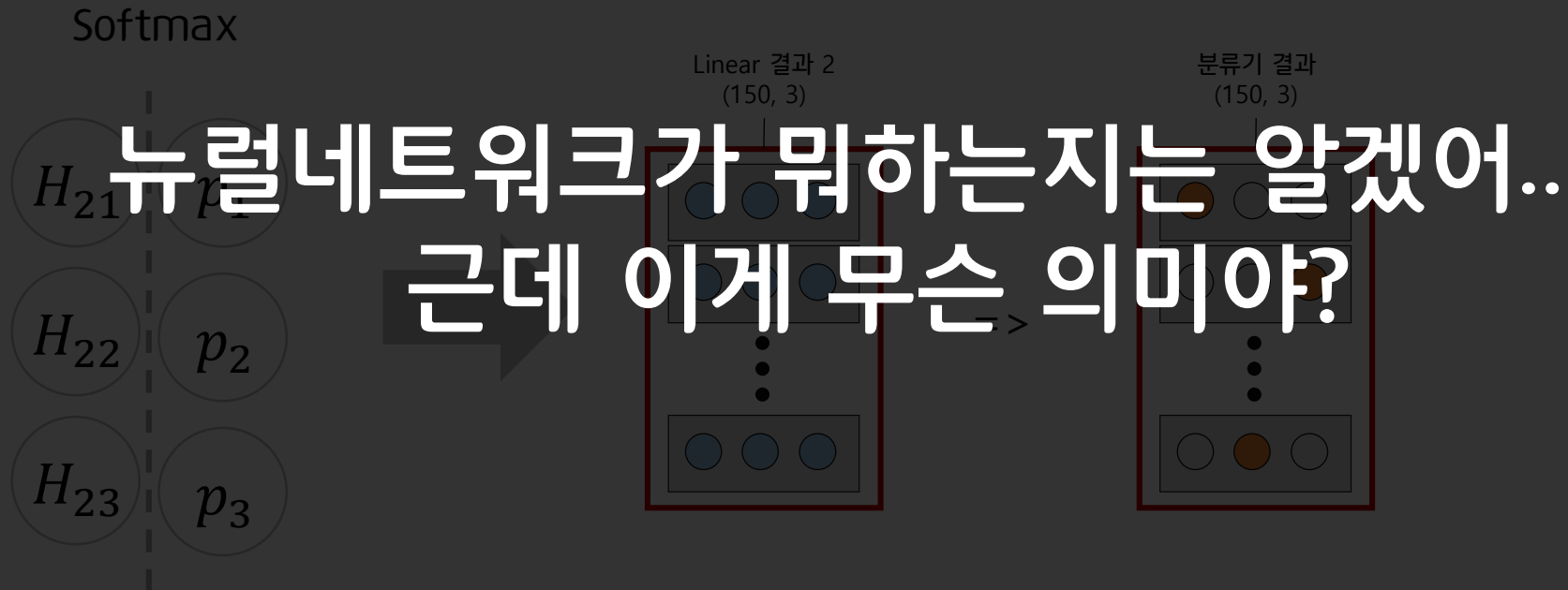
Unit 04 | NN - Forward

Forward 코드 예시

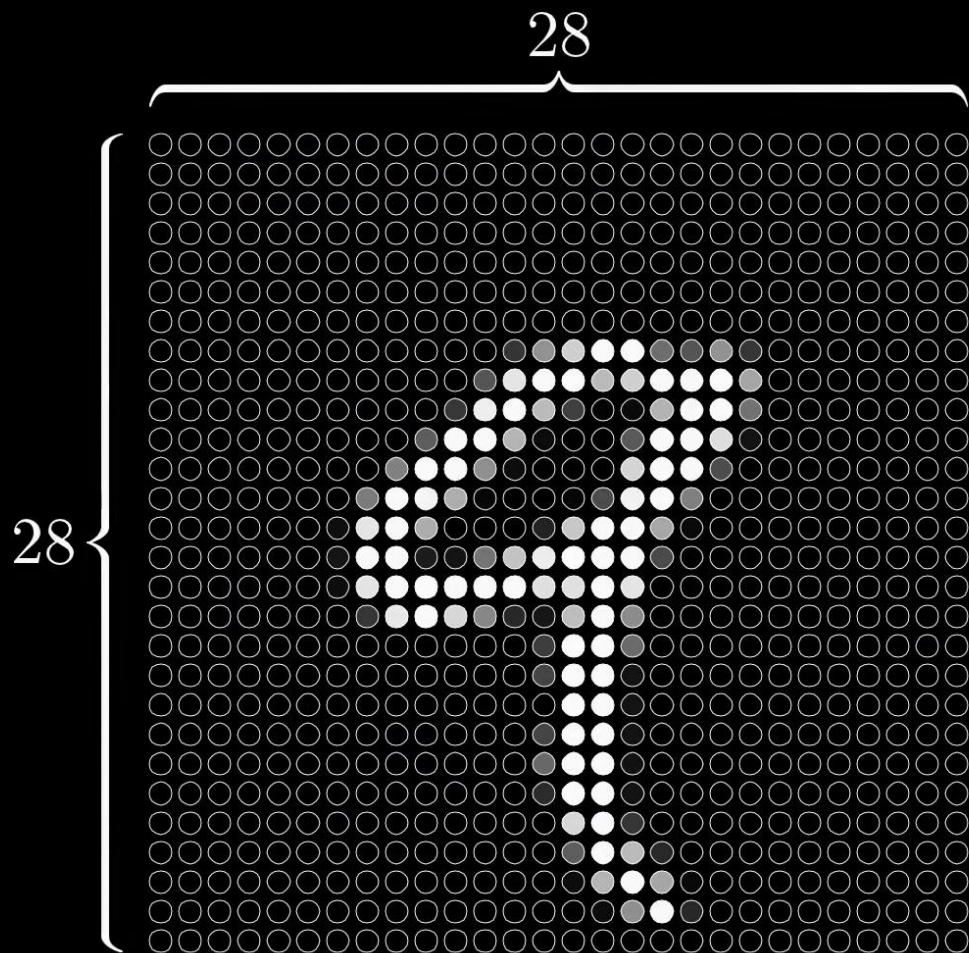
```
def __init__(self):  
  
    self.params = {}  
    self.params['W'] = 0.0001 * np.random.randn(4, 3)  
    self.params['b'] = np.ones(3)  
  
def forward(self, X):  
    W = self.params['W']  
    b = self.params['b']  
  
    h = np.dot(X, W) + b  
    a = np.exp(h)  
    #stable_a = np.exp(h - np.max(h, axis = 1).reshape(-1,1))  
    p = a/np.sum(a, axis = 1).reshape(-1,1)  
    return p
```

Unit 04 | NN - Forward

분류기 결과는 이렇게 표현된다.



Unit 04 | NN - Forward

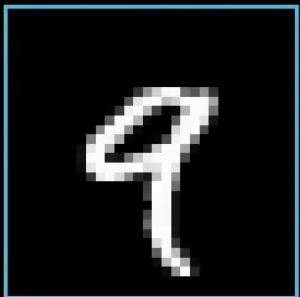


$$28 \times 28 = 784$$

MNIST 데이터
0~9 손글씨 숫자 데이터
진할수록 1에 가깝다



Unit 04 | NN - Forward



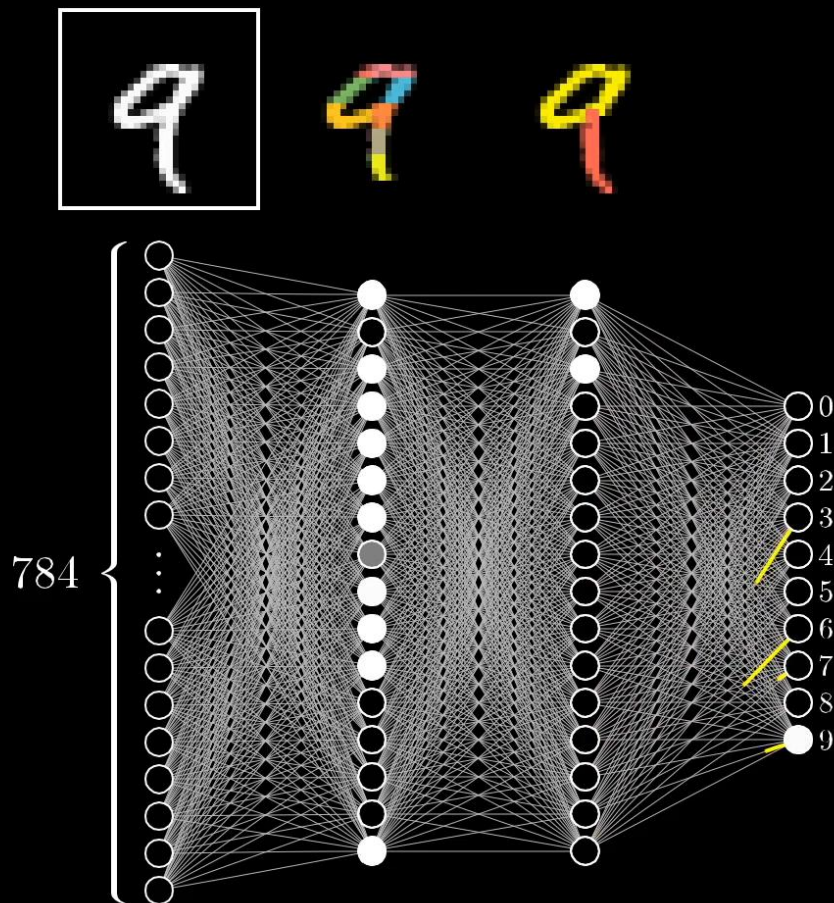
784



28 * 28 데이터를
쭉 늘여서
784개의 변수로 만든다.

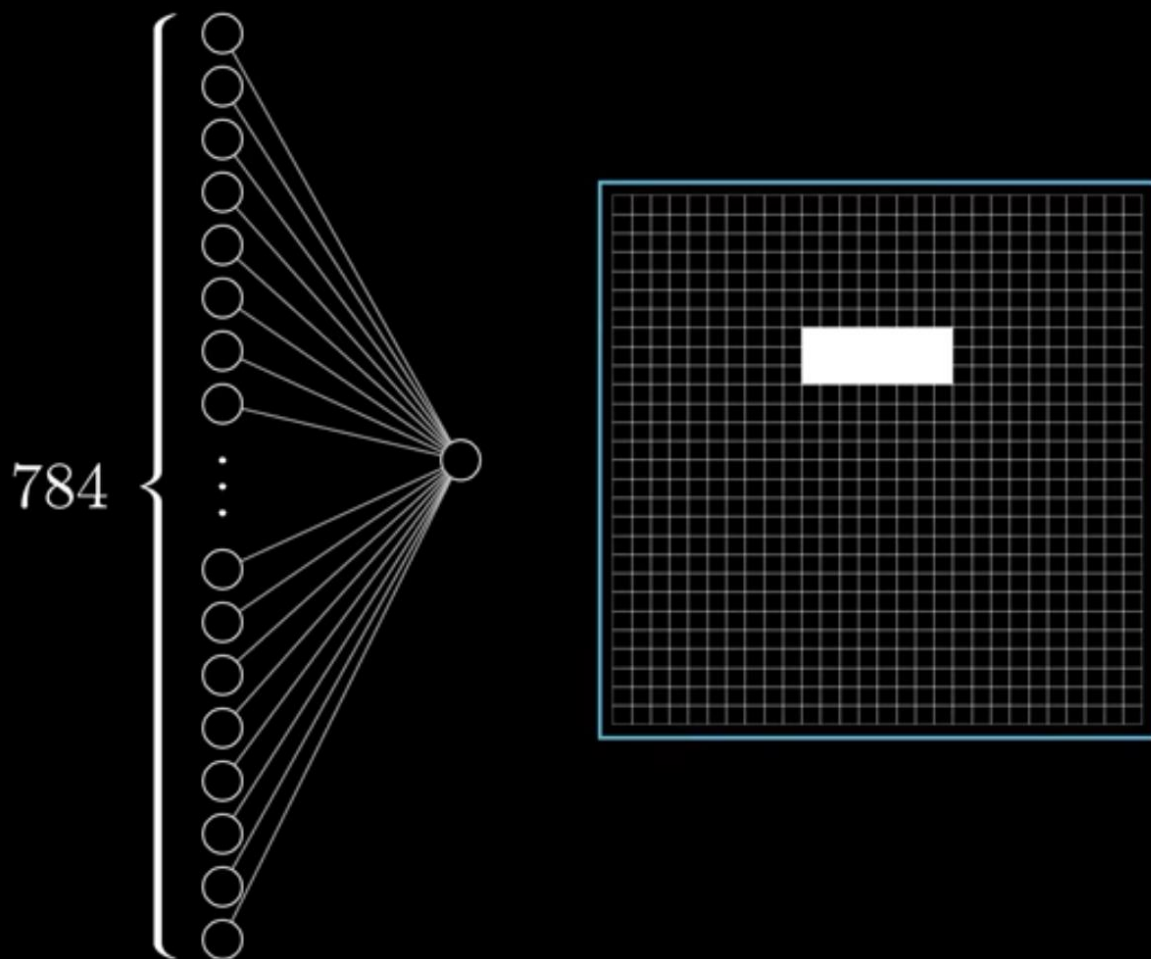


Unit 04 | NN - Forward



각 반응하는 hidden은
특정 부분에 집중해서
활성화된다.

Unit 04 | NN - Forward



특정 부분의 가중치가
784개의 변수중
어디에 집중할지를 판단.



Unit 05 | NN - Backward

좋은 가중치 w 를 어떻게 찾을 수 있을까?

기존 가중치를 이용하여 확률 p 를 구할 수 있다.
해당 p 를 이용하여 현재의 Loss를 구할 수 있다.

이 Loss를 줄이자!!

$$p_i = \frac{\exp(h_i)}{\sum \exp(h_k)}$$

$$L = -\sum y_j \log(p_j)$$

Y1	P1	log(P1)
0	0.0357	-1.44692
Y2	P2	log(P2)
0	0.2624	-0.58099
Y3	P3	log(P3)
1	0.7018	-0.15376

$$L = 0.15376$$

Y1	P1	log(P1)
0	0.0246	-1.60857
Y2	P2	log(P2)
0	0.4917	-0.30834
Y3	P3	log(P3)
1	0.4837	-0.31541

$$L = 0.31541$$

Unit 05 | NN - Backward

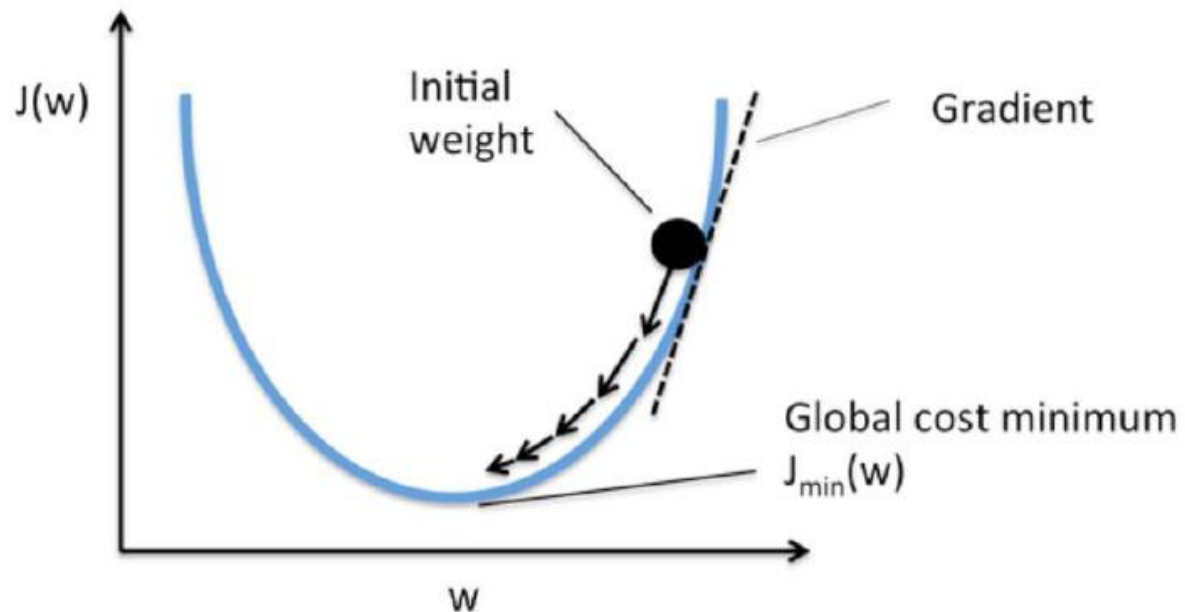
Loss 코드 예시

```
def loss(self, X, T):  
  
    p = self.forward(X)  
  
    n = T.shape[0]  
    log_likelihood = 0  
    log_likelihood -= np.log(p[np.arange(n), T]).sum()  
    Loss = log_likelihood / n  
  
    return Loss
```


Unit 05 | NN - Backward

Gradient Descent : loss function을 최소화 하기 위해 사용하는 알고리즘

- 1) Weight에 random 한 숫자로 초기값 부여
- 2) Loss를 minimize 하는 방향으로 w를 업데이트 !

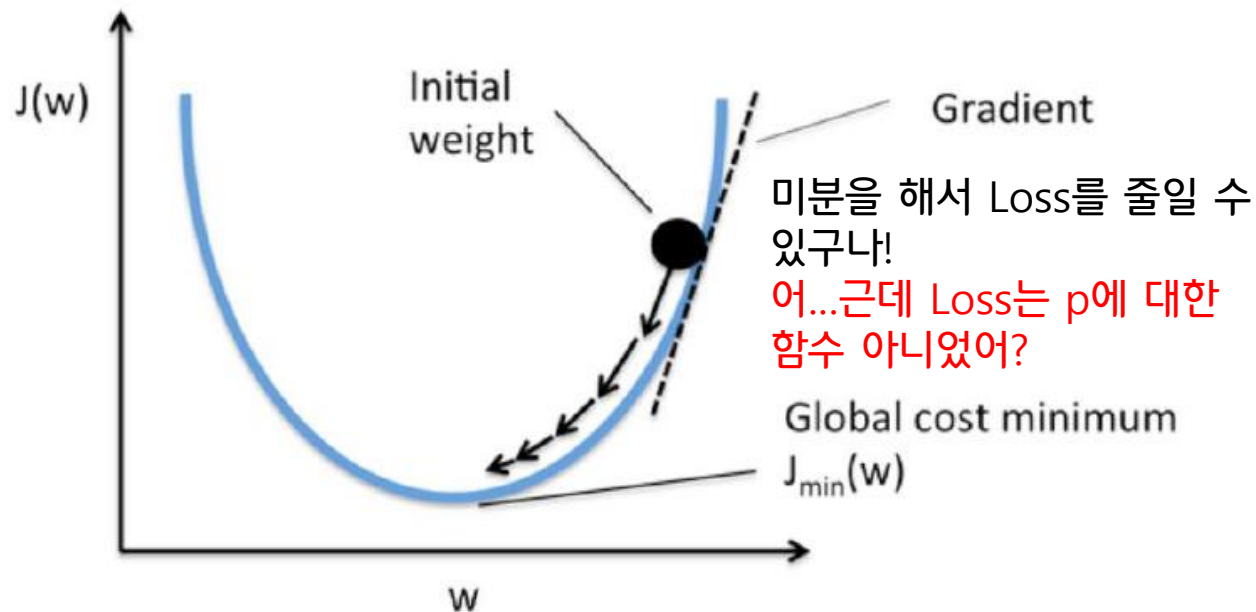


$$W := W - \underbrace{\alpha}_{\text{learning rate}} \frac{d}{dw} \text{loss}(W)$$

Unit 05 | NN - Backward

Gradient Descent : loss function을 최소화 하기 위해 사용하는 알고리즘

- 1) Weight에 random 한 숫자로 초기값 부여
- 2) Loss를 minimize 하는 방향으로 w를 업데이트 !



$$W := W - \alpha \frac{d}{dw} \text{loss}(W)$$

learning rate

Unit 05 | NN - Backward

Chain Rule을 통해 Loss를 타고 타고 w 에 대해서 미분하자!

Chain Rule

$$y = 5x + 3$$
$$z = 4y^2$$

$$\frac{dz}{dy} = 8y, \frac{dy}{dx} = 5$$

$$\frac{dz}{dx} = \frac{dz}{dy} \frac{dy}{dx} = 8y * 5 = 40y = 40(5x + 3)$$

$$H = XW$$
$$P = \text{Softmax}(H)$$
$$L = -\text{loglikelihood}(P)$$

$$\frac{dL}{dH} = P - y$$

(y : 실제 레이블)

$$\frac{dH}{dW} = X^T$$

$$\frac{dL}{dH} = \frac{dL}{dH} \frac{dH}{dW} = X^T (P - y)$$

Unit 05 | NN - Backward

Backpropagation: a simple example

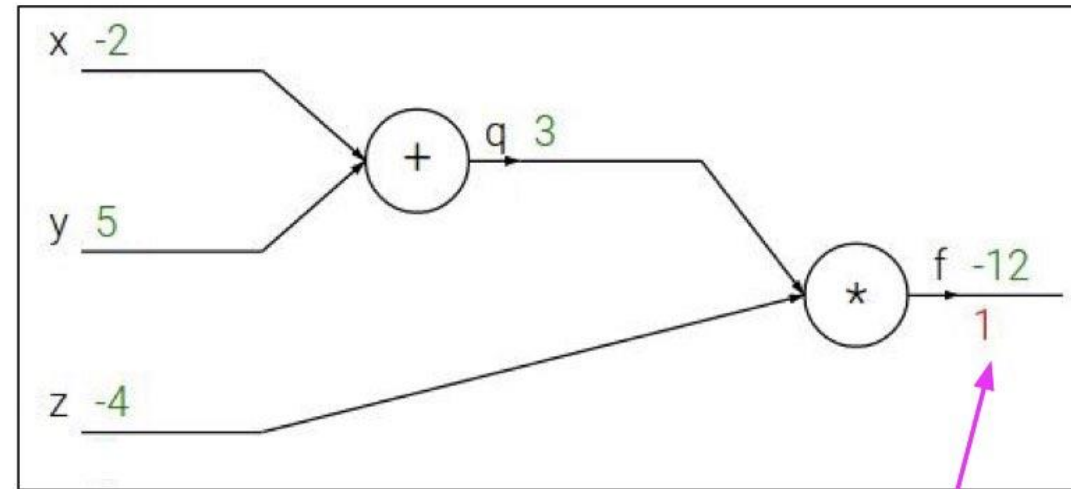
$$f(x, y, z) = (x + y)z$$

e.g. $x = -2, y = 5, z = -4$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



$$\frac{\partial f}{\partial f}$$

Unit 05 | NN - Backward

Backpropagation: a simple example

$$f(x, y, z) = (x + y)z$$

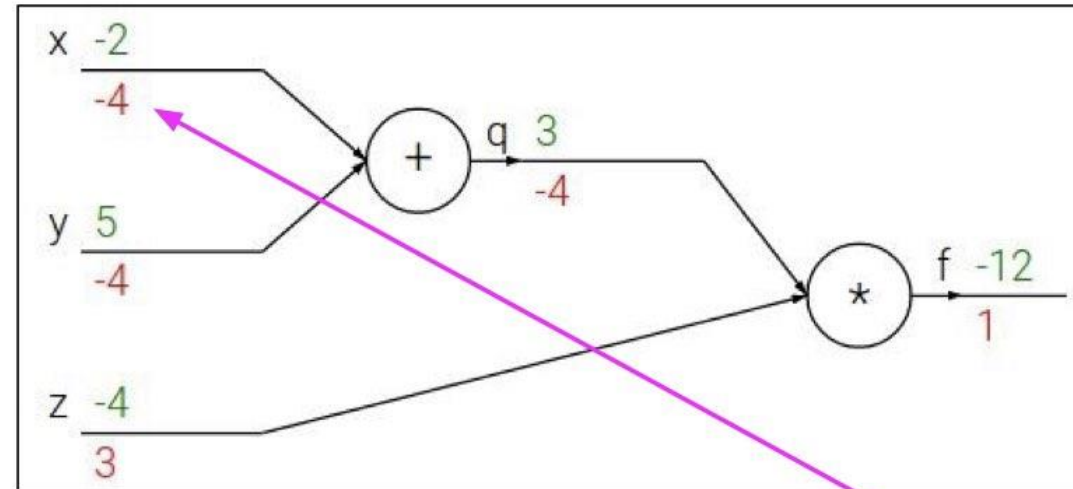
e.g. $x = -2, y = 5, z = -4$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

덧셈 게이트에서는 그대로 넘긴다!

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

곱셈 게이트에서는 상대방을 곱한다!

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$ 

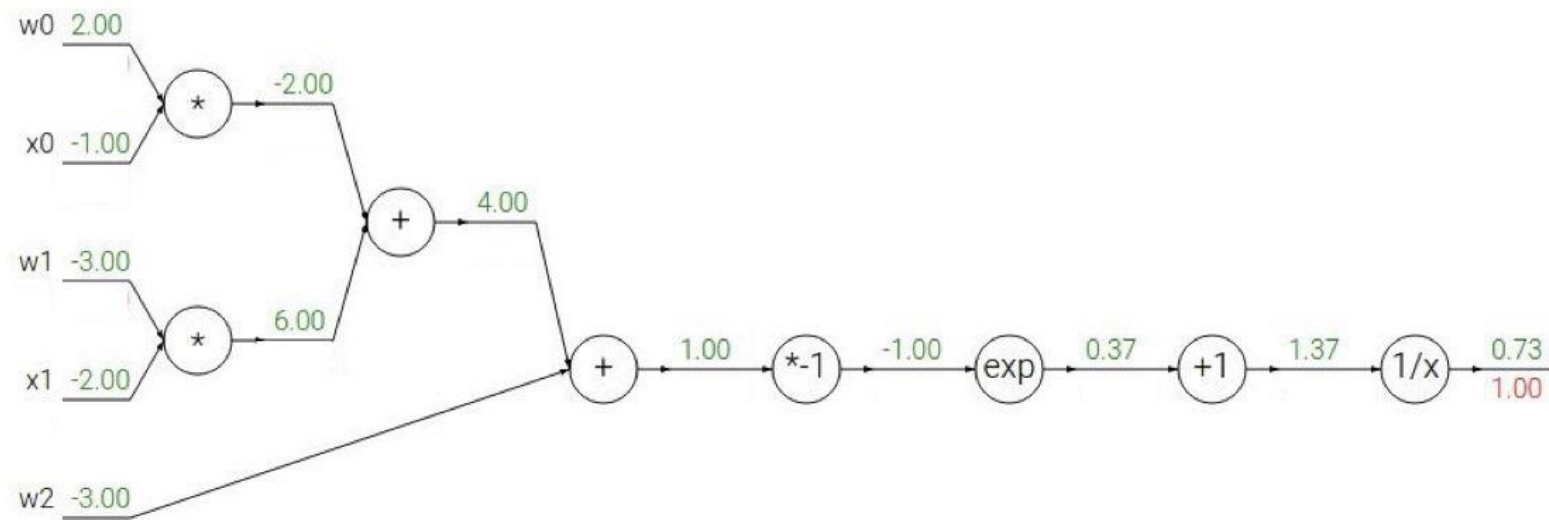
$$\frac{\partial f}{\partial x}$$

Chain rule:

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q} \frac{\partial q}{\partial x}$$

Unit 05 | NN - Backward

Another example: $f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2x_2)}}$



$f(x) = e^x$	→	$\frac{df}{dx} = e^x$		$f(x) = \frac{1}{x}$	→	$\frac{df}{dx} = -1/x^2$
$f_a(x) = ax$	→	$\frac{df}{dx} = a$		$f_c(x) = c + x$	→	$\frac{df}{dx} = 1$

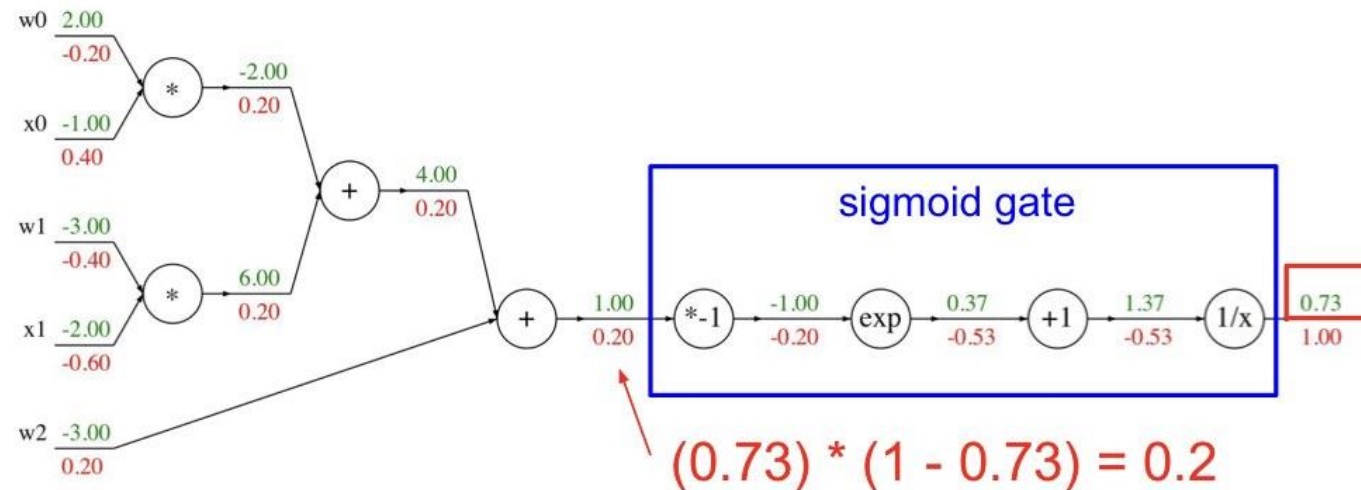
Unit 05 | NN - Backward

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2x_2)}}$$

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

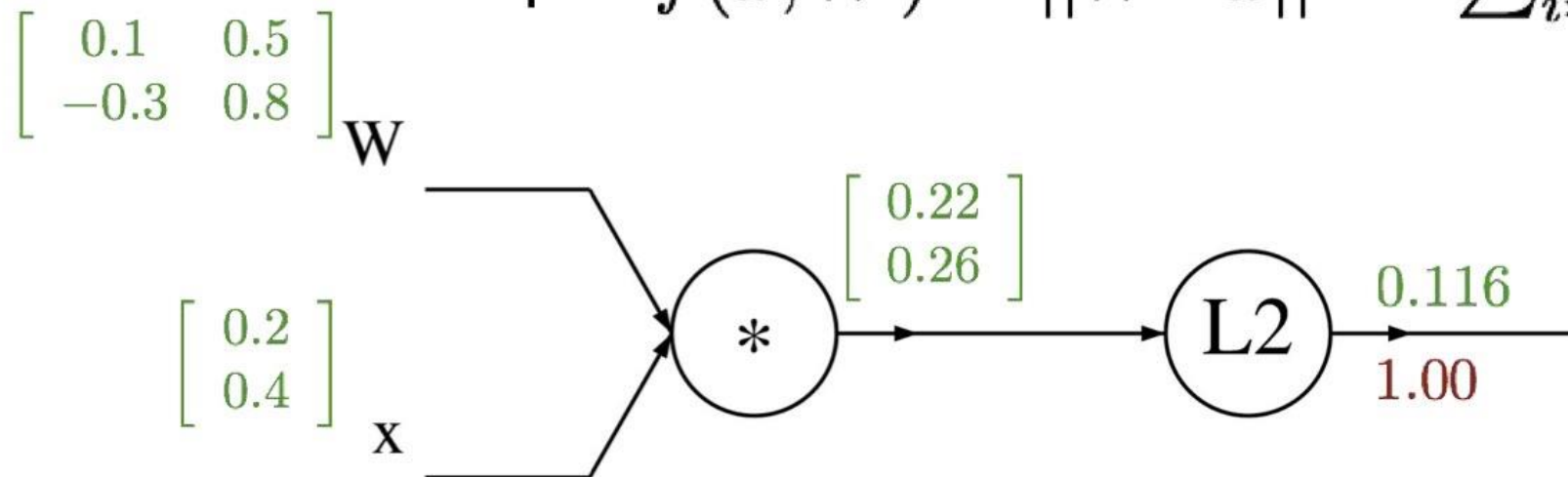
sigmoid function

$$\frac{d\sigma(x)}{dx} = \frac{e^{-x}}{(1 + e^{-x})^2} = \left(\frac{1 + e^{-x} - 1}{1 + e^{-x}} \right) \left(\frac{1}{1 + e^{-x}} \right) = (1 - \sigma(x)) \sigma(x)$$



Unit 05 | NN - Backward

A vectorized example: $f(x, W) = ||W \cdot x||^2 = \sum_{i=1}^n (W \cdot x)_i^2$



$$q = W \cdot x = \begin{pmatrix} W_{1,1}x_1 + \cdots + W_{1,n}x_n \\ \vdots \\ W_{n,1}x_1 + \cdots + W_{n,n}x_n \end{pmatrix}$$

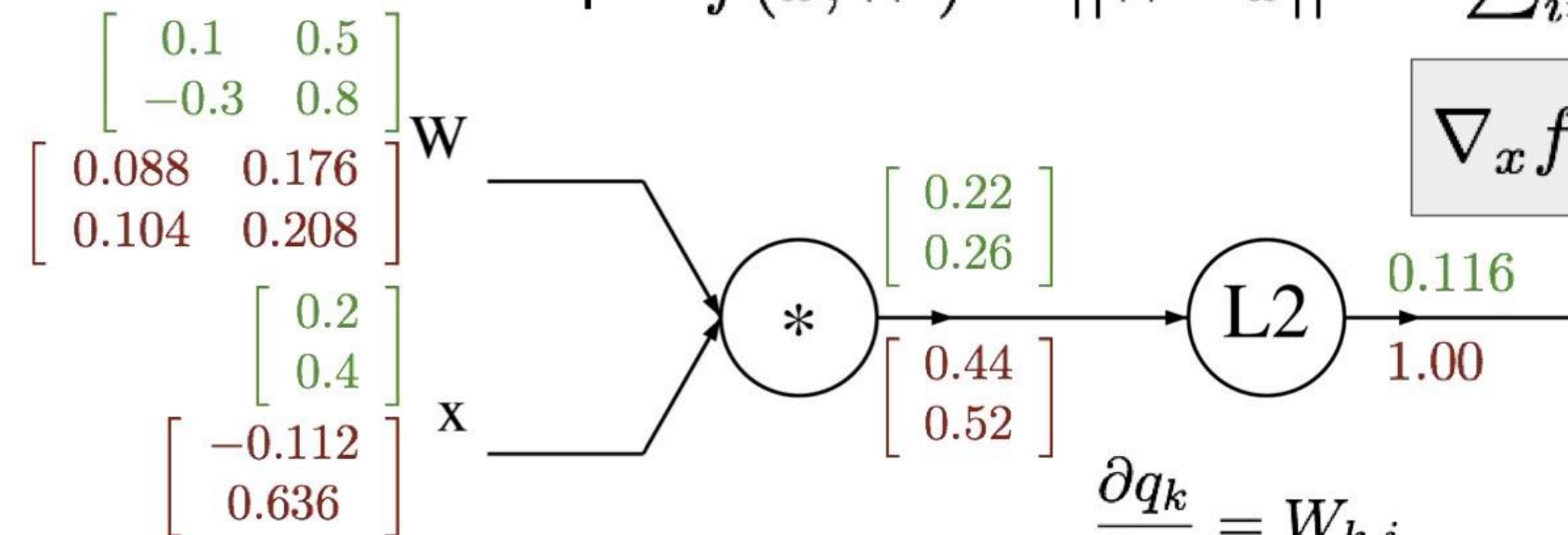
$$f(q) = ||q||^2 = q_1^2 + \cdots + q_n^2$$

$$\frac{\partial f}{\partial q_i} = 2q_i$$

$$\nabla_q f = 2q$$

Unit 05 | NN - Backward

A vectorized example: $f(x, W) = \|W \cdot x\|^2 = \sum_{i=1}^n (W \cdot x)_i^2$

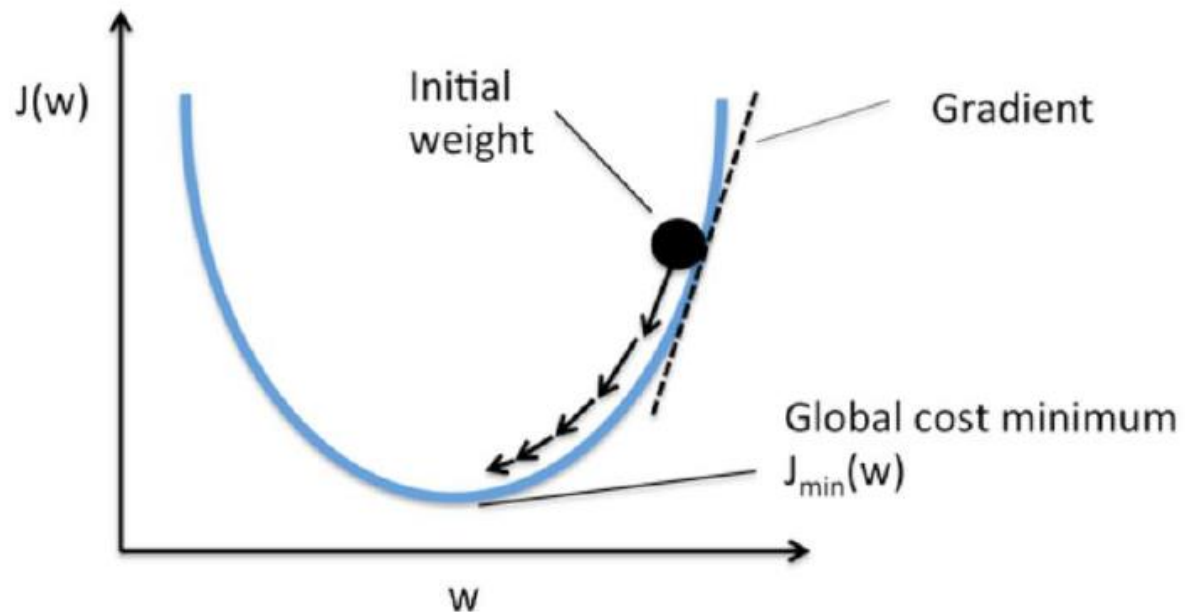


$$\begin{aligned} \frac{\partial q_k}{\partial x_i} &= W_{k,i} \\ \frac{\partial f}{\partial x_i} &= \sum_k \frac{\partial f}{\partial q_k} \frac{\partial q_k}{\partial x_i} \\ &= \sum_k 2q_k W_{k,i} \end{aligned}$$

Unit 05 | NN - Backward

Gradient Descent : loss function을 최소화 하기 위해 사용하는 알고리즘

- 1) Weight에 random 한 숫자로 초기값 부여
- 2) Loss를 minimize 하는 방향으로 w를 업데이트 !



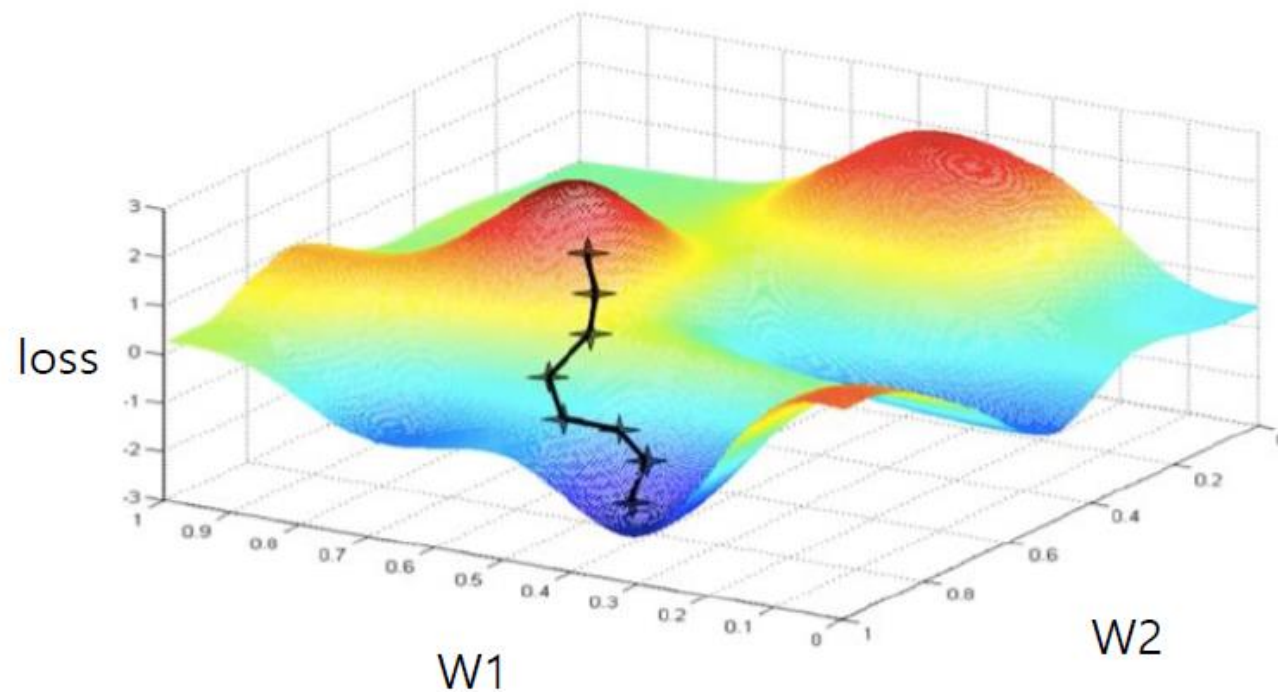
결국 미분한 값을
Learning_rate를 곱해서
현재 W 가중치에서 빼준다!

$$W := W - \alpha \frac{d}{dw} \text{loss}(W)$$

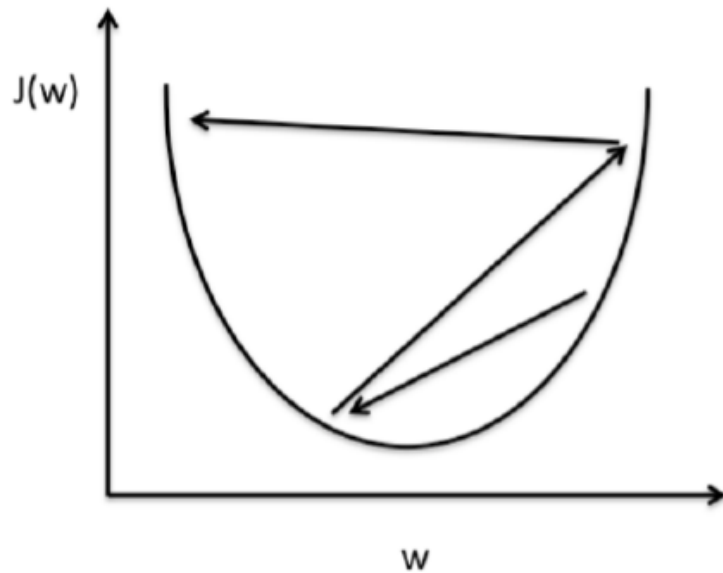
learning rate

Unit 05 | NN - Backward

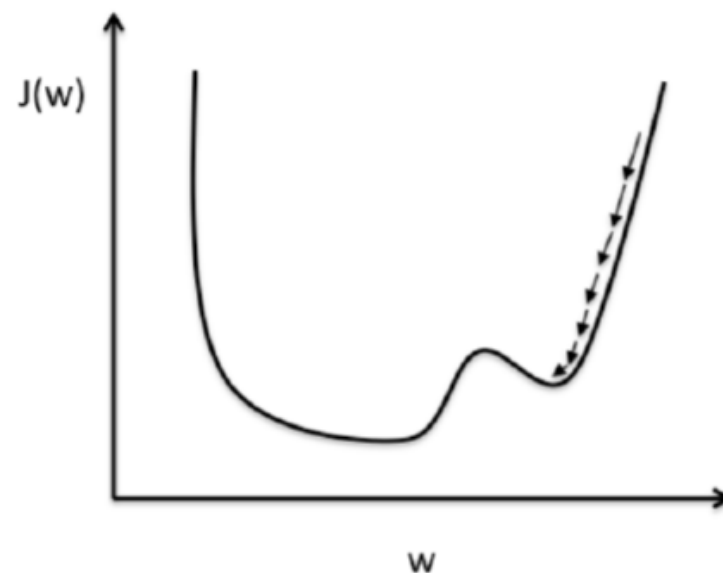
Gradient Descent : loss function을 최소화 하기 위해 사용하는 알고리즘



Unit 05 | NN - Backward

Learning rate (α)

Learning rate 가 **큰** 경우
-> overshooting



Learning rate 가 **작은** 경우
-> 학습하는데 시간이 오래 걸린다
최저점이 아닌 데서 멈추거나,
local minimum에 빠진다.

Unit 05 | NN - Backward

GD 코드 예시

```
def gradient(self, X, T, learning_rate = 0.0001):

    p = self.forward(X)

    t = np.zeros((T.shape[0], 3))
    t[np.arange(T.shape[0]), T] = 1
    #t는 인덱스 레이블 T를 one hot 벡터로 바꾼 것

    dp = p.copy()
    dp[np.arange(len(T)), T] -= 1

    #목적함수에 대한 가중치 미분값을 담은 zero array 생성
    grads = {}
    grads['W'] = np.zeros((4, 3))
    grads['b'] = np.zeros(10)
    #목적함수에 대한 가중치 미분값 합 구하기
    grads['W'] = (1/len(T)) * np.dot(X.T, p-t)
    grads['b'] = (1/len(T)) * np.sum(p-t, axis = 0)
    #p-t 대신 dp 사용 가능

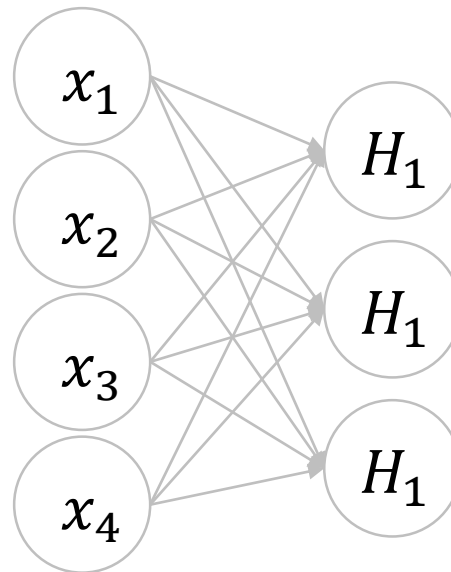
    self.params['W'] -= learning_rate * grads['W']
    self.params['b'] -= learning_rate * grads['b']
```

Unit 05 | NN - Backward

```
[[ -8.74615472e-05  3.03732950e-05  4.65428693e-05]
 [ -3.23668516e-05  4.42560055e-06 -2.56037962e-05]
 [ -2.43068233e-05  9.68563102e-05 -6.81574199e-05]
 [ -1.63824136e-04  2.12286420e-04  7.43973291e-05]]
[[ -0.00321581  0.00061467  0.0025906 ]
 [  0.02404027 -0.01065741 -0.01343641]
 [ -0.05779136  0.01472877  0.04306699]
 [ -0.0255728   0.00370264  0.02199303]]
[[  0.00861248 -0.00040284 -0.00822019]
 [  0.05412096 -0.02178615 -0.03238836]
 [ -0.10186908  0.02754678  0.07432669]
 [ -0.04607576  0.00652109  0.03967753]]
[[  0.02155924 -0.00163555 -0.01993423]
 [  0.08324693 -0.03262187 -0.0506786 ]
 [ -0.1418079   0.03937622  0.10243607]
 [ -0.06481271  0.00892456  0.05601101]]
[[  0.03370442 -0.00258894 -0.03112603]
 [  0.11051824 -0.0428869  -0.06768488]
 [ -0.17908648  0.05050046  0.12859041]
 [ -0.08228968  0.01100061  0.07141193]]
```

Gradient Descent로 가중치가 변하는 실제 예시

Network



Iris 데이터에 적용한 것

가중치는 $4 * 3$ 매트릭스150개 데이터에 대해 5000번씩
학습시키고,

1000번째마다 가중치를 찍어본 것

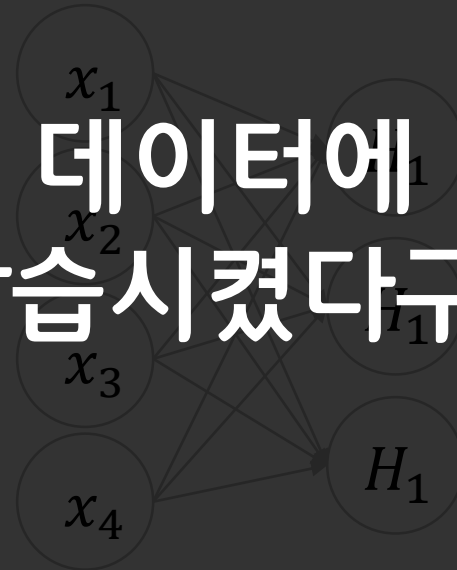
Unit 05 | NN - Backward

```
[[ -8.74615472e-05  3.03732950e-05  4.65428693e-05]
 [ -3.23668516e-05  4.42560055e-06 -2.56037962e-05]
 [ -2.43068233e-05  9.68563102e-05 -6.81574199e-05]
 [ -1.63824136e-04  2.12286420e-04  7.43973291e-05]]
[[ -0.00321581  0.00061467  0.0025906 ]
 [ 0.02404027 -0.01065741 -0.01343641]
 [ -0.05779136  0.01472877  0.00000000]
 [ -0.0255728  0.00370264  0.00000000]
 [[ 0.00861248 -0.00040284 -0.00000000]
 [ 0.05412096 -0.02178615 -0.03238555]
 [ -0.10186908  0.02754678  0.07432341]
 [ -0.04607576  0.00652109  0.03967753]]
[[ 0.02155924 -0.00163555 -0.01993423]
 [ 0.08324693 -0.03262187 -0.0506786 ]
 [ -0.1418079  0.03937622  0.10243607]
 [ -0.06481271  0.00892456  0.05601101]]
[[ 0.03370442 -0.00258894 -0.03112603]
 [ 0.11051824 -0.0428869  -0.06768488]
 [ -0.17908648  0.05050046  0.12859041]
 [ -0.08228968  0.01100061  0.07141193]]
```

방금.. 150개 데이터에 대해
5000번 학습시켰다구?

Gradient Descent로 가중치가 변하는 실제 예시

Network



Iris 데이터에 적용한 것

가중치는 $4 * 3$ 매트릭스

150개 데이터에 대해 5000번씩
학습시키고,

1000번째마다 가중치를 찍어본 것

Unit 05 | NN - Backward

Epoch :

150개 데이터에 대해서 1바퀴 다 도는 것을 뜻한다.
방금 예시에서는 Epoch가 5000

Batch_size :

150개가 너무 많아 데이터를 나눠서 1Epoch를 학습하는 것을 뜻한다.
Ex) Batch_size 가 50이면 1Epoch에 3번 돈다.

즉 학습에 대해 Epoch for문이 있고,
그 안에 데이터갯수/Batch_size(위 예시에서 3번) 만큼 for문이 돈다.

참고문헌

밑바닥부터 시작하는 딥러닝, 사이토고키, 개앞맵시

https://www.youtube.com/channel/UCYO_jab_esuFRV4b17AJtAw (유튜브 영상), 3 Brown 1 Blue

<http://cs231n.stanford.edu/2017/> (CS231N 강의), 스탠포드 대학교

9기 임소정 자료

Q & A

들어주셔서 감사합니다.