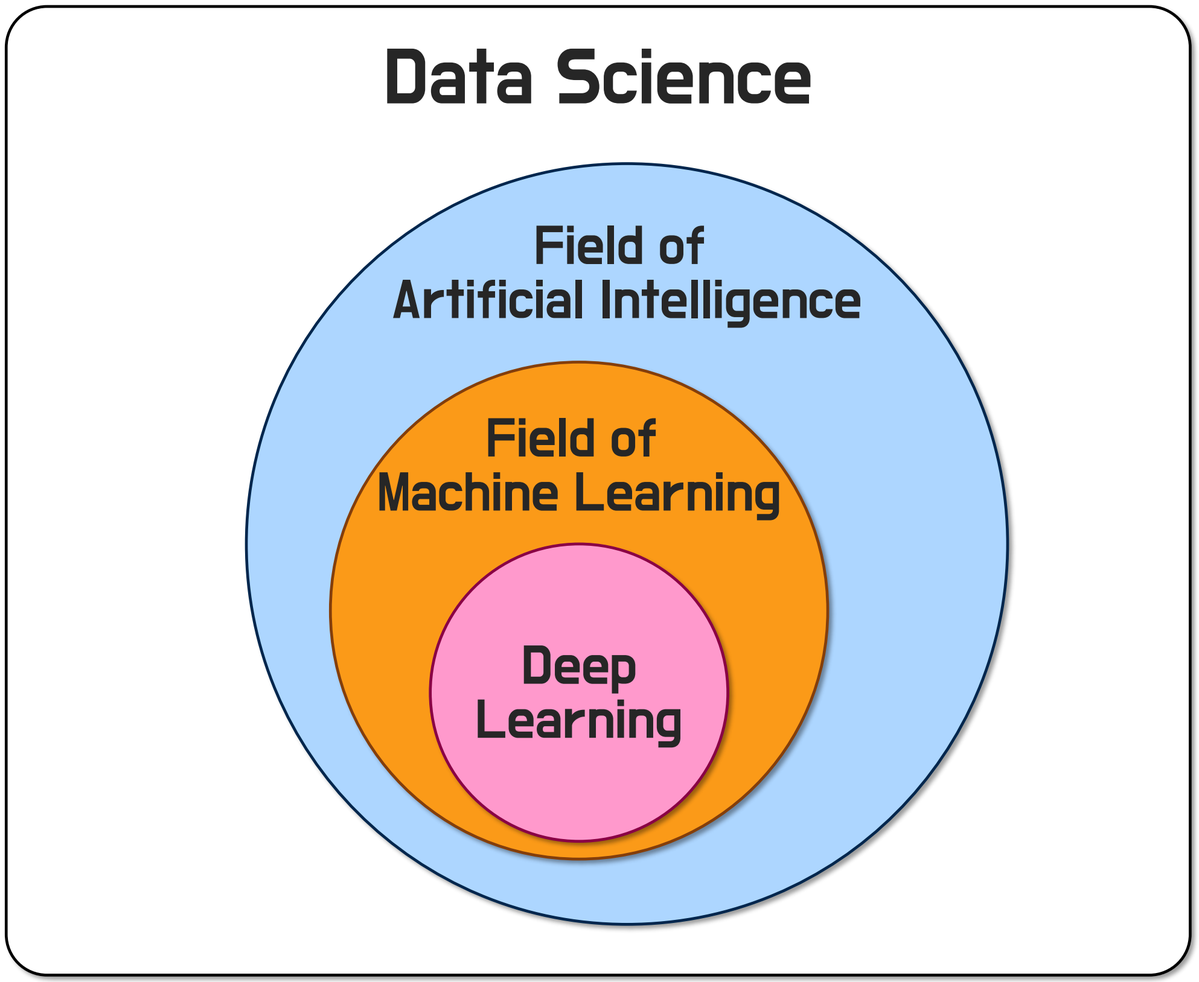
딥러닝 (Deep Learning) 기초

01 딥러닝이란?

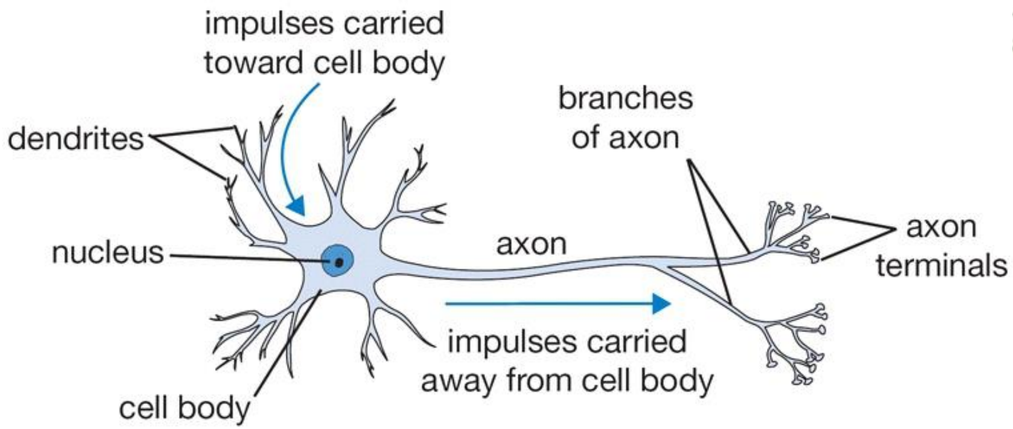
Deep & High Learning

딥러닝 (Deep Learning)

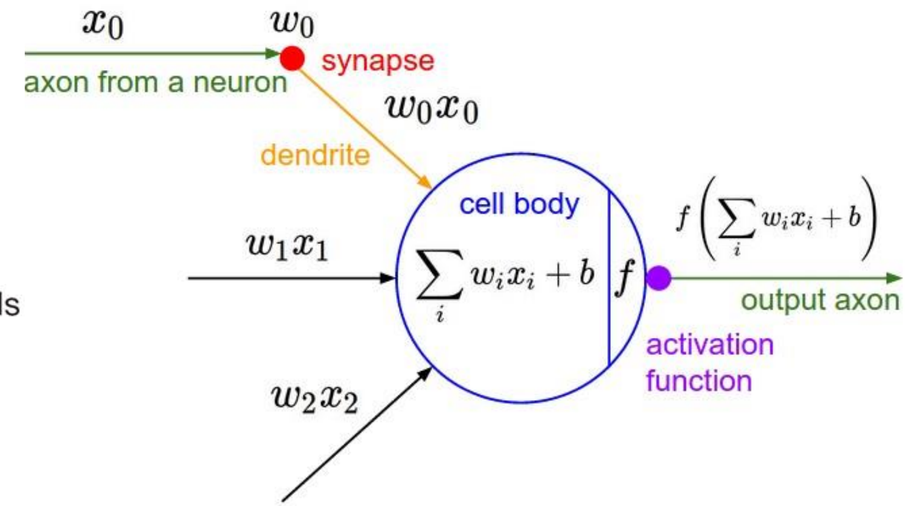


딥러닝 (Deep Learning)

"사람의 신경망을 모사한 인공 신경망 (Artificial neural network)에 기반하는 머신러닝 알고리즘의 한 종류"



<Neuron>

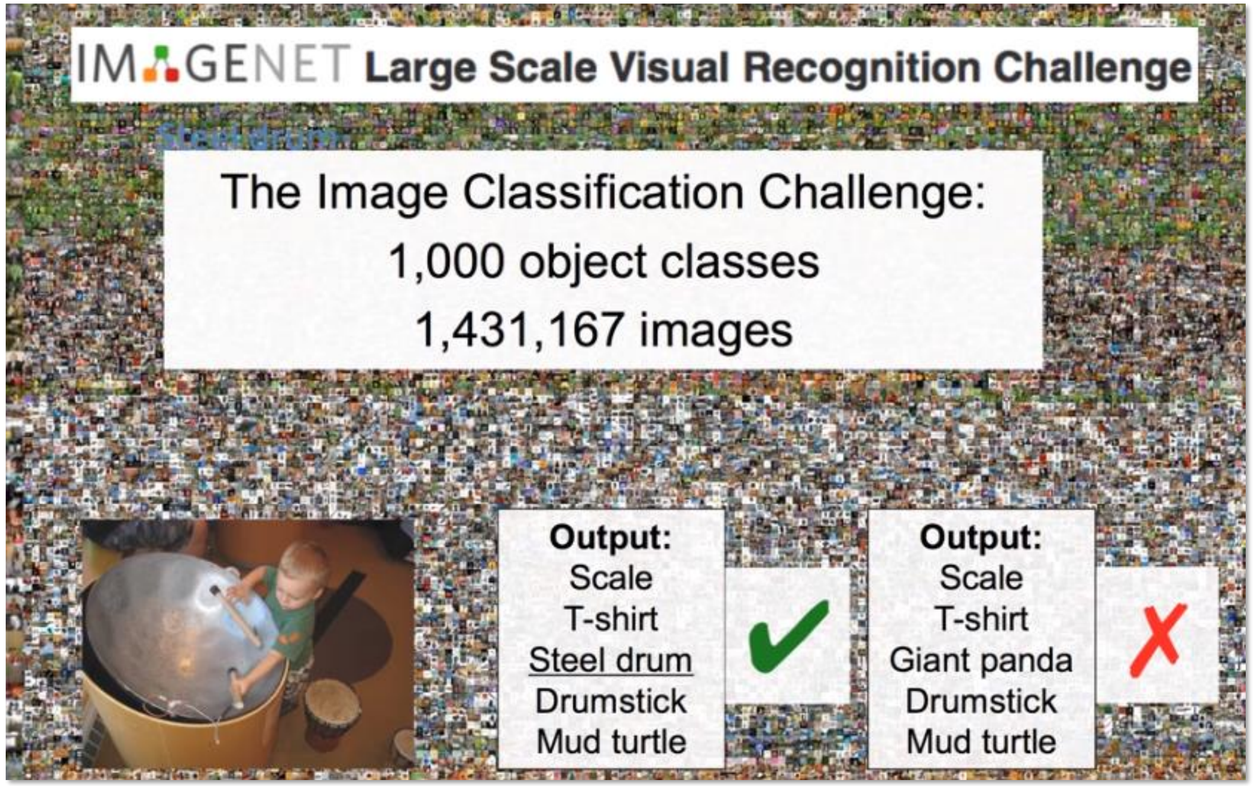


<Perceptron>

딥러닝 (Deep Learning)

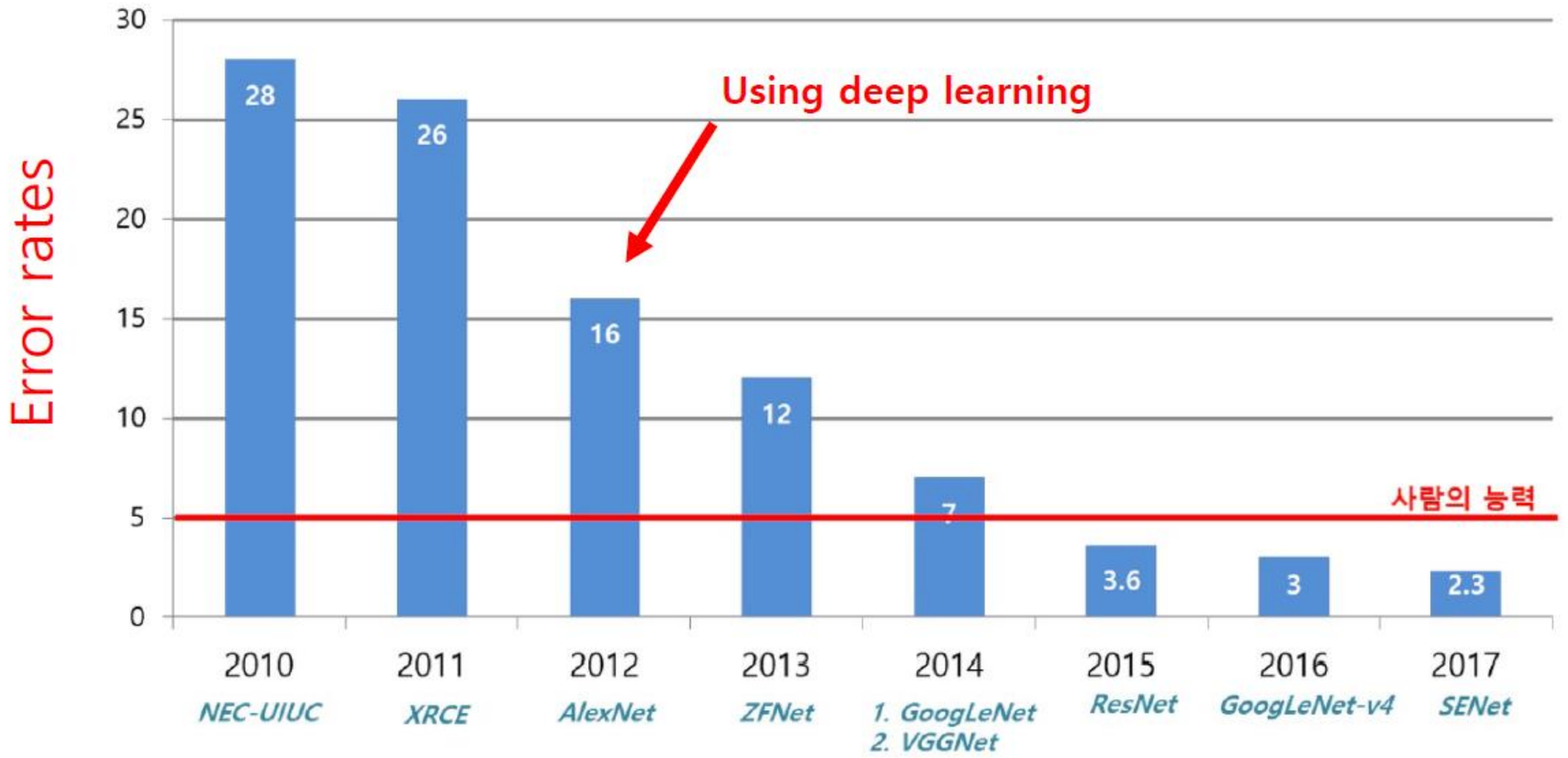
딥러닝이 부상한 이유?

ILSVRC – ImageNet Large Scale Visual Recognition Challenge



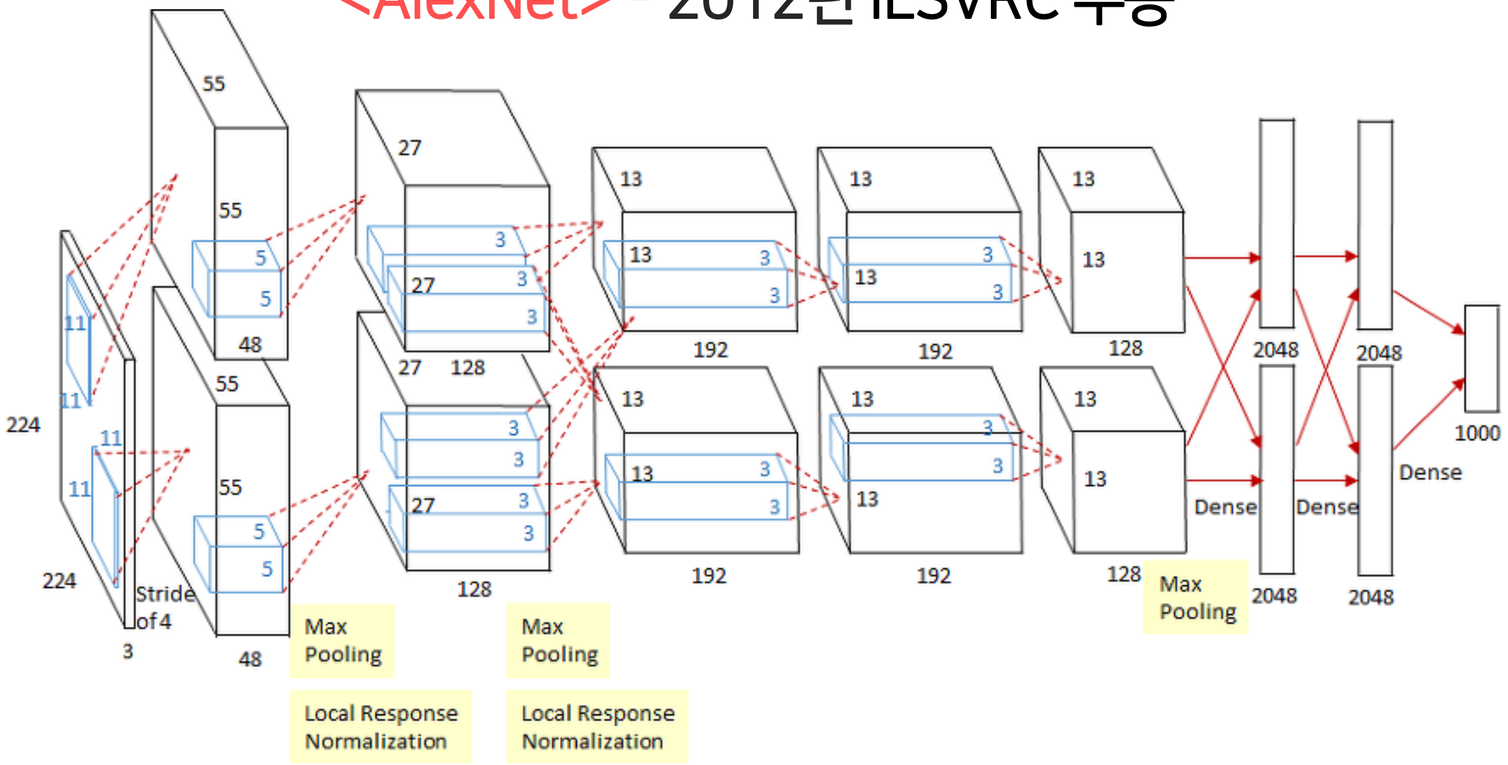
딥러닝 (Deep Learning)

ILSVRC 역대 우승 알고리즘들과 인식 에러율



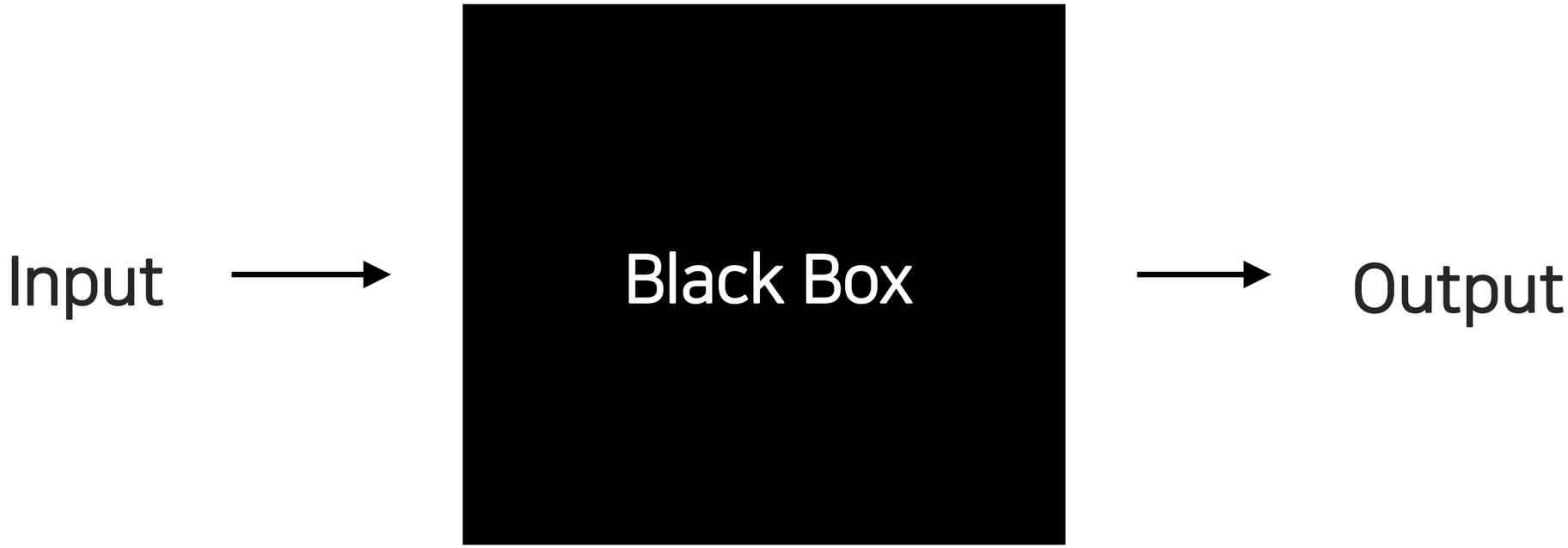
딥러닝 (Deep Learning)

<AlexNet> - 2012년 ILSVRC 우승



딥러닝의 특징

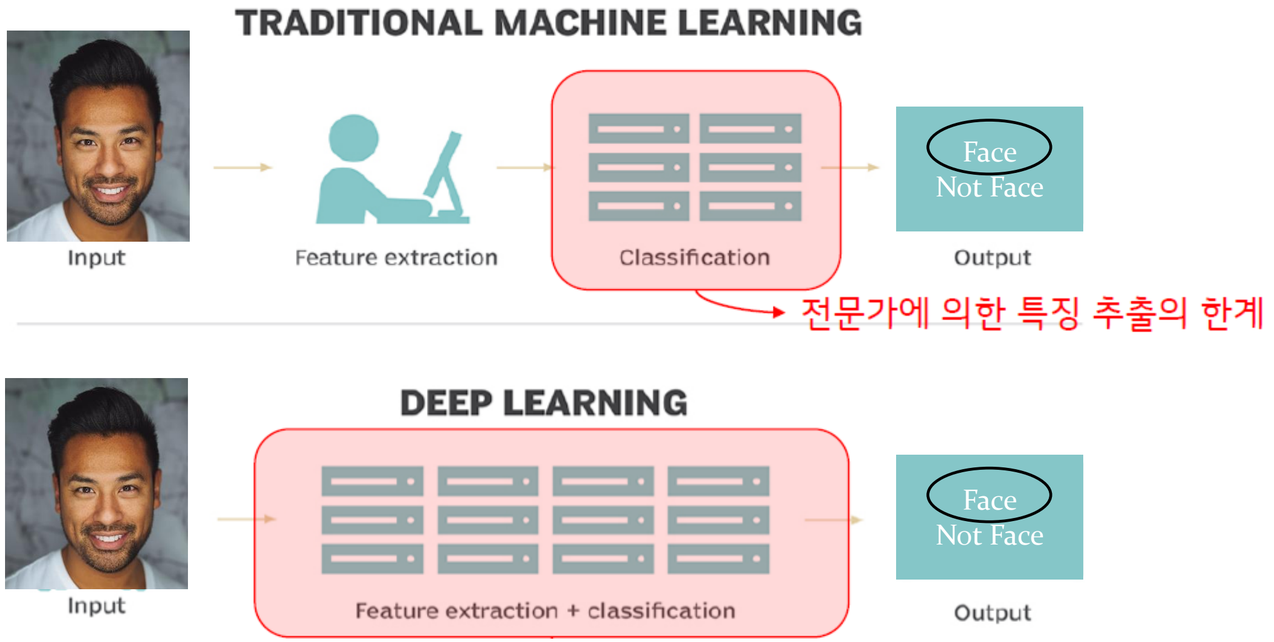
블랙박스 알고리즘?



딥러닝의 특징

기존 기계학습 : 특징 추출을 위해 사람이 직접 데이터를 가공해야 했음

딥러닝 : 별도 데이터 가공 없이 오직 데이터에 기반하여 End-to-End로 학습 가능

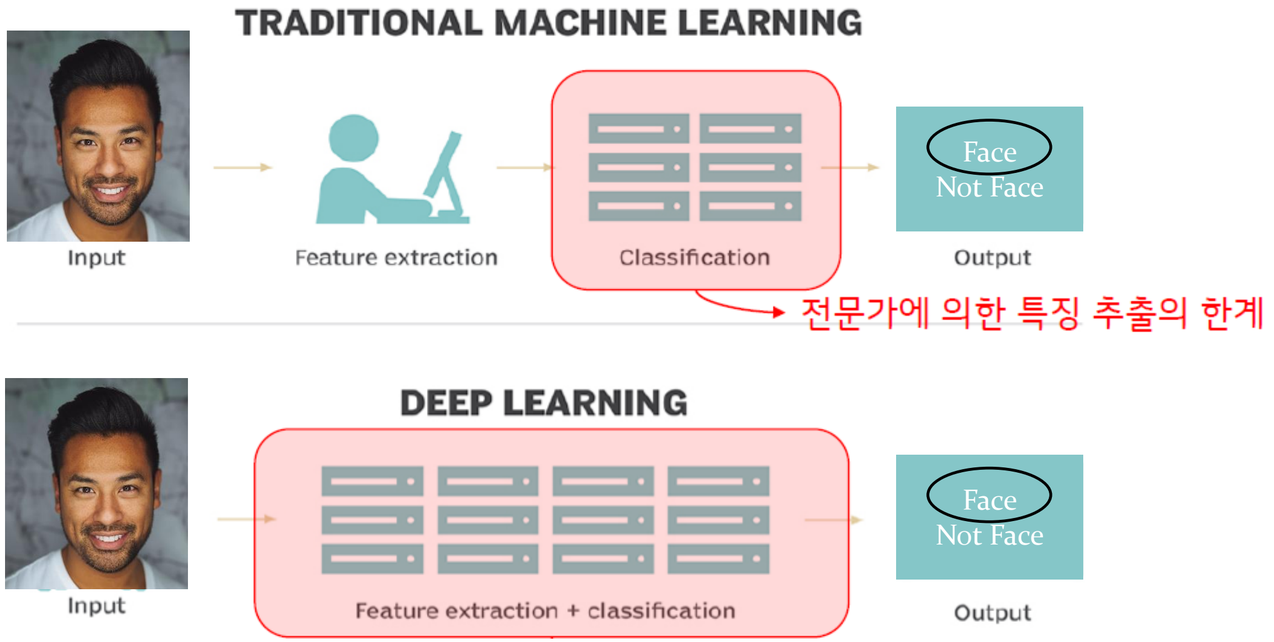


기계가 모든 특징을 추출
정보를 구분할 수 있는 최소단위 입력만으로 학습
(영상 픽셀, 음성 신호 등)

딥러닝의 특징

기존 기계학습 : 특징 추출을 위해 사람이 직접 데이터를 가공해야 했음

딥러닝 : 별도 데이터 가공 없이 오직 데이터에 기반하여 End-to-End로 학습 가능

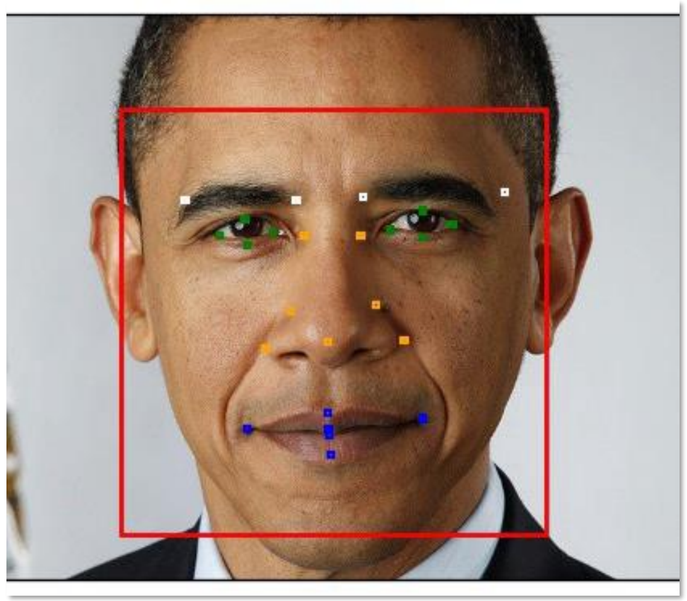


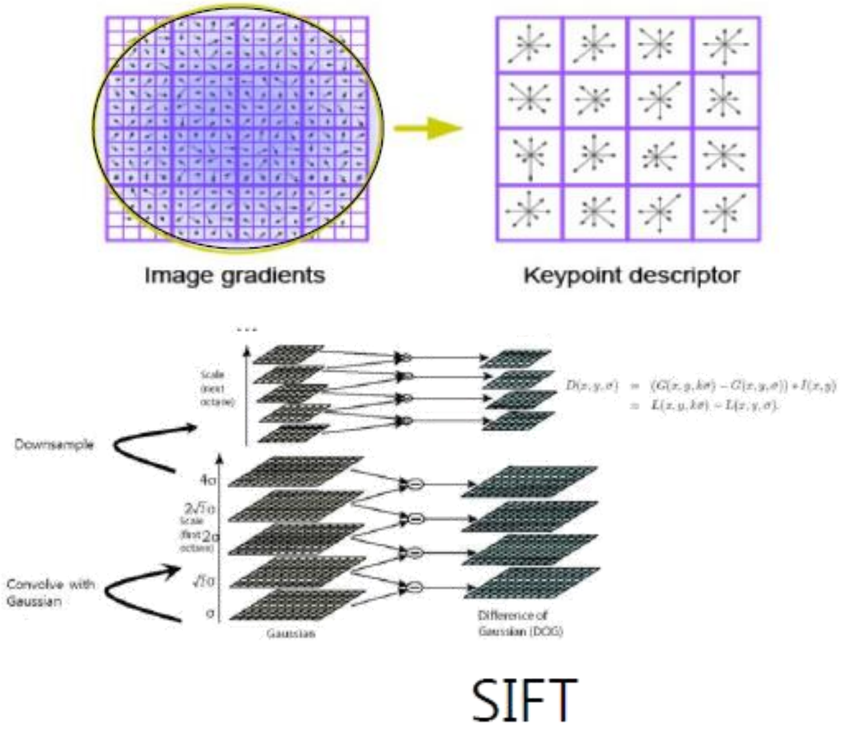
기계가 모든 특징을 추출
정보를 구분할 수 있는 최소단위 입력만으로 학습
(영상 픽셀, 음성 신호 등)

딥러닝의 특징

[특징 추출 단계를 대체하는 딥러닝]

- 기존의 머신러닝 기법들은 직접 추출한 (Hand-crafted features) feature에 기반





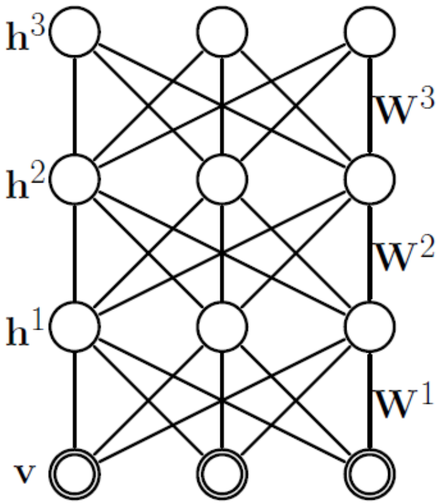
SIFT

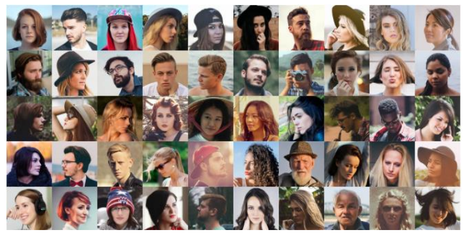
딥러닝의 특징

[특징 추출 단계를 대체하는 딥러닝]

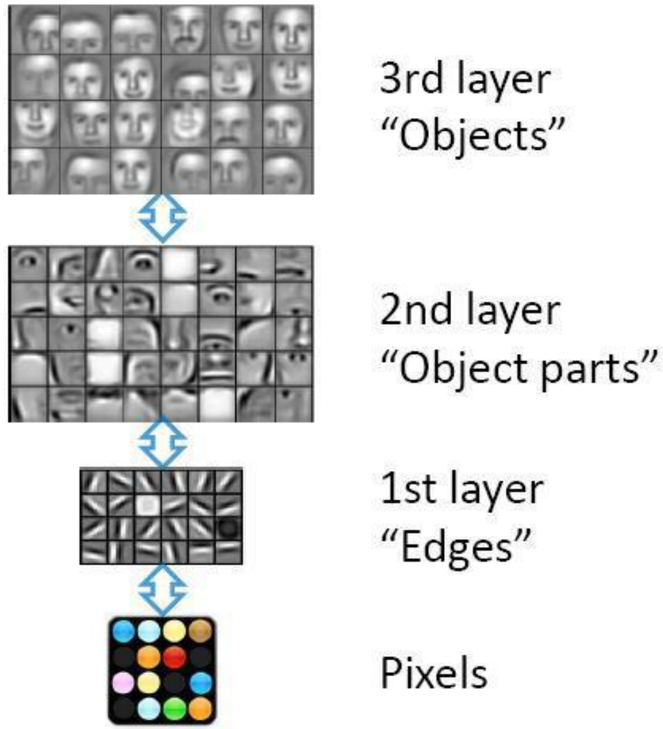
- 딥러닝은 데이터를 기반으로 자동적으로 특징 표현을 만들어 냄

Face





Feature representation

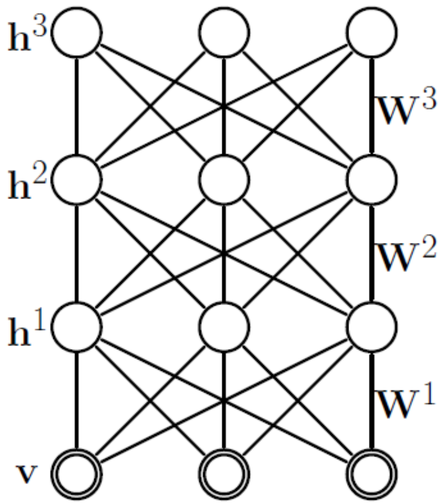


딥러닝의 특징

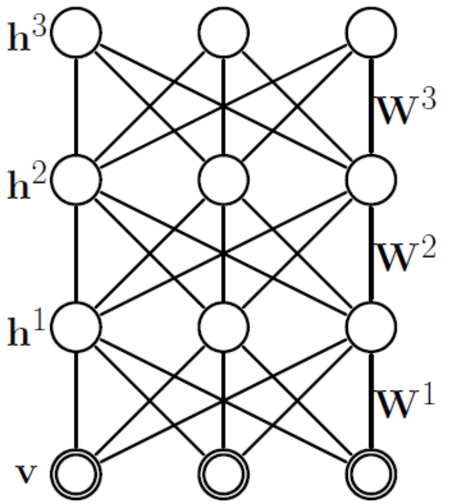
[특징 추출 단계를 대체하는 딥러닝]

- 딥러닝은 데이터를 기반으로 자동적으로 특징 표현을 만들어 냄

Not face!



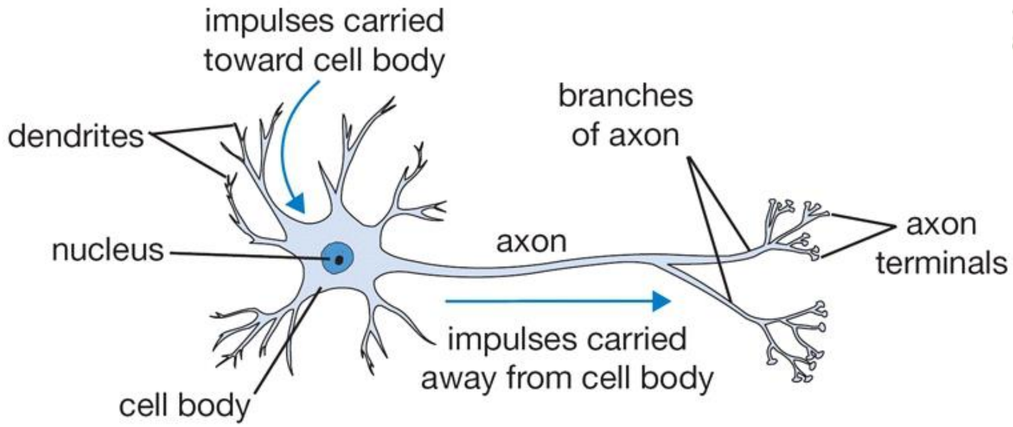
Face!



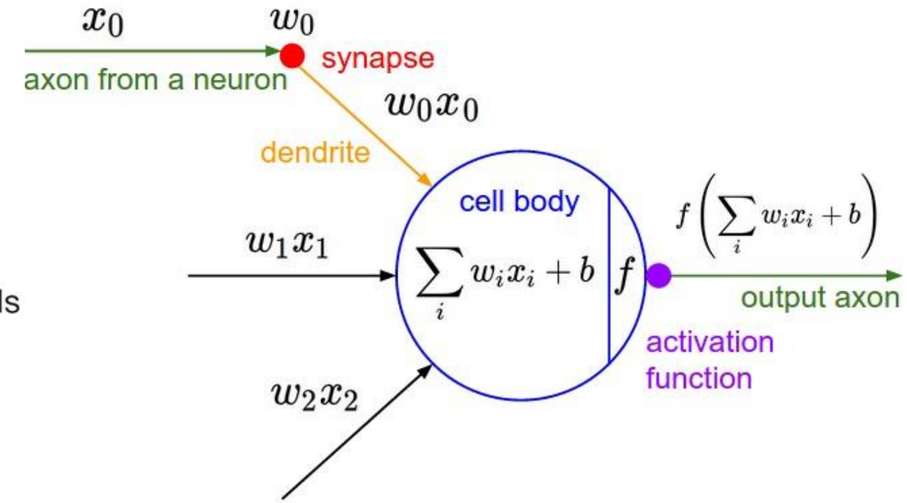
딥러닝의 특징

[특징 추출 단계를 대체하는 딥러닝]

- 딥러닝은 데이터를 기반으로 자동적으로 특징 표현을 만들어 냄
- 인공 신경망이 뭉치면 매우 강력해집니다!



<Neuron>



<Perceptron>

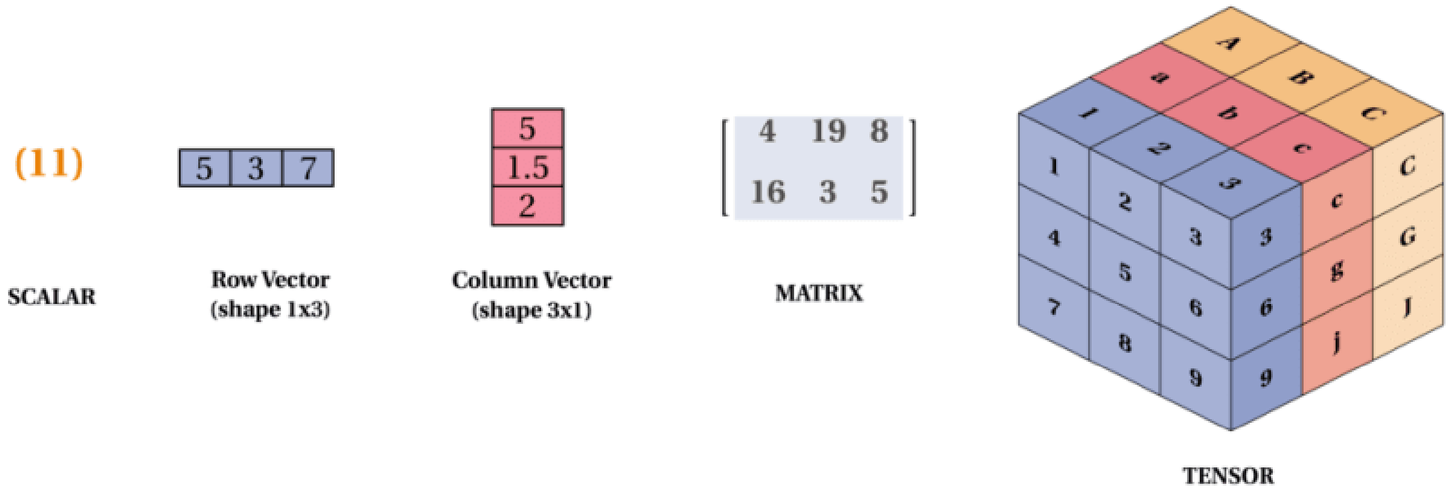
See you in the next lecture!

딥러닝 (Deep Learning) 기초

02 순전파 - 기초 수학 복습

Deep & High Learning

Scalar, Vector, Matrix, Tensor



Scalar Vector Matrix Tensor

1

$$\begin{bmatrix} 1 \\ 2 \end{bmatrix} \quad \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \quad \begin{bmatrix} \begin{bmatrix} 1 & 2 \\ 1 & 7 \end{bmatrix} & \begin{bmatrix} 3 & 2 \\ 5 & 4 \end{bmatrix} \end{bmatrix}$$

Vector

Scalar

24

Vector

$$\begin{bmatrix} 2 & -8 & 7 \end{bmatrix}$$

row

or column

$$\begin{bmatrix} -6 \\ -4 \\ 27 \end{bmatrix}$$

Matrix

$$\begin{bmatrix} 6 & 4 & 24 \\ 1 & -9 & 8 \end{bmatrix}$$

row(s) × column(s)

Vector

Sum

$$\vec{u} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \quad \vec{v} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}$$

$$\vec{u} + \vec{v} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} + \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}$$

$$\vec{u} + \vec{v} = \begin{bmatrix} u_1 + v_1 \\ u_2 + v_2 \\ u_3 + v_3 \end{bmatrix}$$

Dot Product

$$\mathbf{a} = \begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix} \quad \mathbf{b} = \begin{pmatrix} x_2 \\ y_2 \\ z_2 \end{pmatrix}$$

$$\mathbf{a}.\mathbf{b} = x_1x_2 + y_1y_2 + z_1z_2$$

$$\mathbf{a} = \begin{pmatrix} 2 \\ 4 \\ 6 \end{pmatrix} \quad \mathbf{b} = \begin{pmatrix} 3 \\ 5 \\ 7 \end{pmatrix}$$

$$\mathbf{a}.\mathbf{b} = 2(3) + 4(5) + 6(7)$$

$$= 68$$

Norm

$$x = [x_1, x_2, \ldots, x_n]$$

Define the p-Norm: $|x|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{\frac{1}{p}}$

L1-Norm is $|x|_1 = \sum_{i=1}^{n} |x_i|$

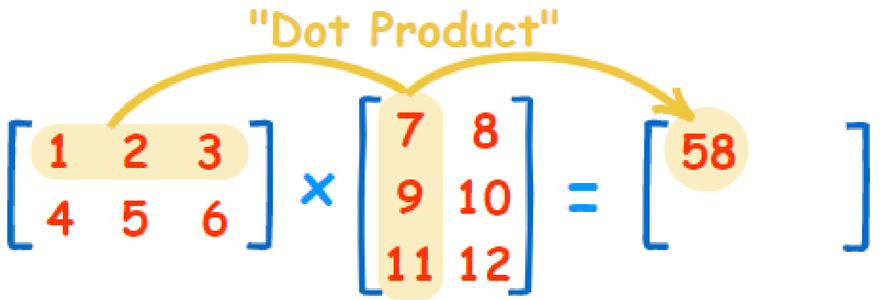
L2-Norm is $|x|_2 = \sqrt{\sum_{i=1}^{n} |x_i|^2}$

Matrix

Sum

$$\begin{pmatrix} 5 & 2 \\ 4 & 9 \\ 10 & -3 \end{pmatrix} + \begin{pmatrix} -11 & 0 \\ 7 & 1 \\ -6 & -8 \end{pmatrix} = \begin{pmatrix} 5+(-11) & 2+0 \\ 4+7 & 9+1 \\ 10+(-6) & -3+(-8) \end{pmatrix} = \begin{pmatrix} -6 & 2 \\ 11 & 10 \\ 4 & -11 \end{pmatrix}$$

Dot



Element wise multiplication

$$\underset{G}{\begin{bmatrix} 3 & 5 & 7 \\ 4 & 9 & 8 \end{bmatrix}} \circ \underset{H}{\begin{bmatrix} 1 & 6 & 3 \\ 0 & 2 & 9 \end{bmatrix}} = \underset{N}{\begin{bmatrix} 3\times1 & 5\times6 & 7\times3 \\ 4\times0 & 9\times2 & 8\times9 \end{bmatrix}}$$

See you in the next lecture!

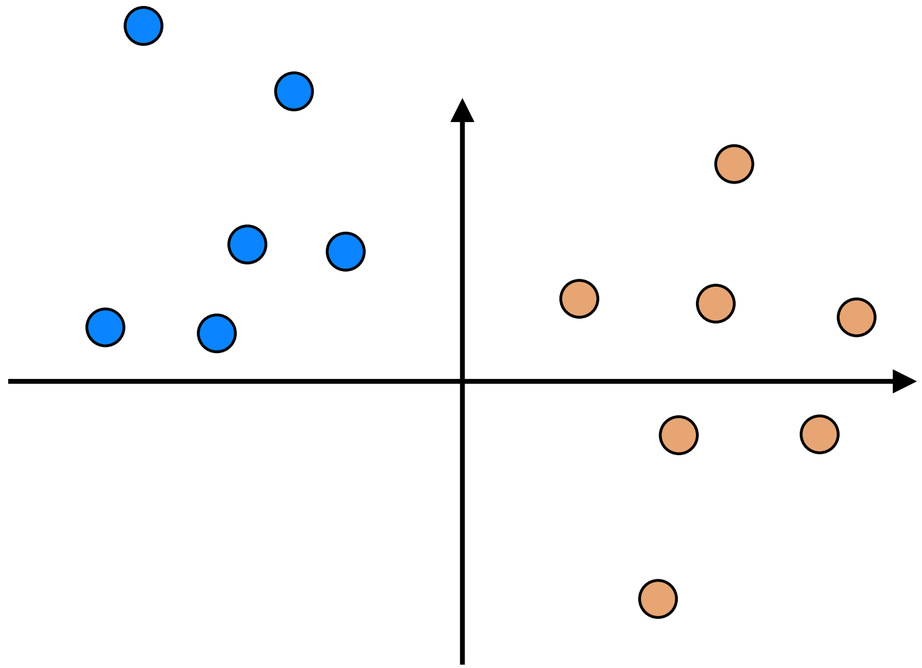
딥러닝 (Deep Learning) 기초

03 학습이란?

Deep & High Learning

학습이란?

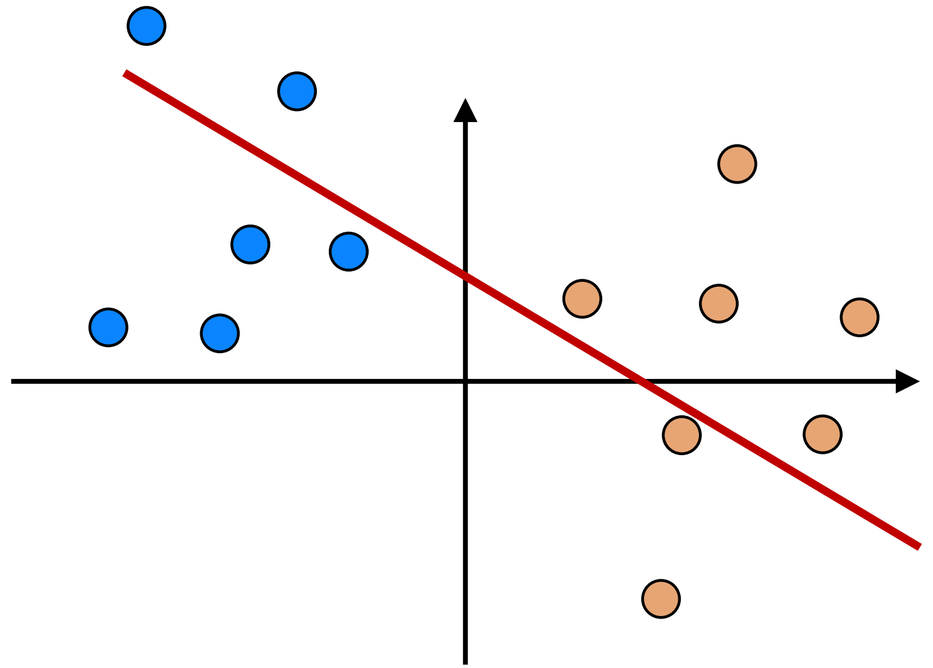
두 색깔이 다른 원들을 가르는 선을 그리기



학습이란?

두 색깔이 다른 원들을 가르는 선을 그리기

정확히 구분 실패

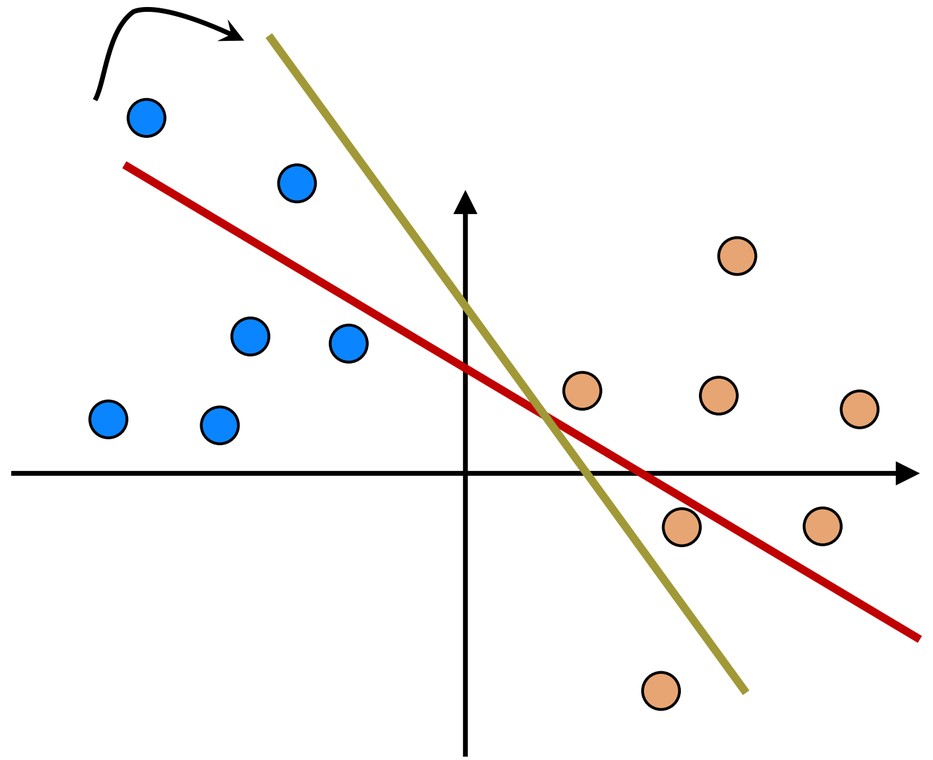


학습이란?

두 색깔이 다른 원들을 가르는 선을 그리기

정확히 구분 실패

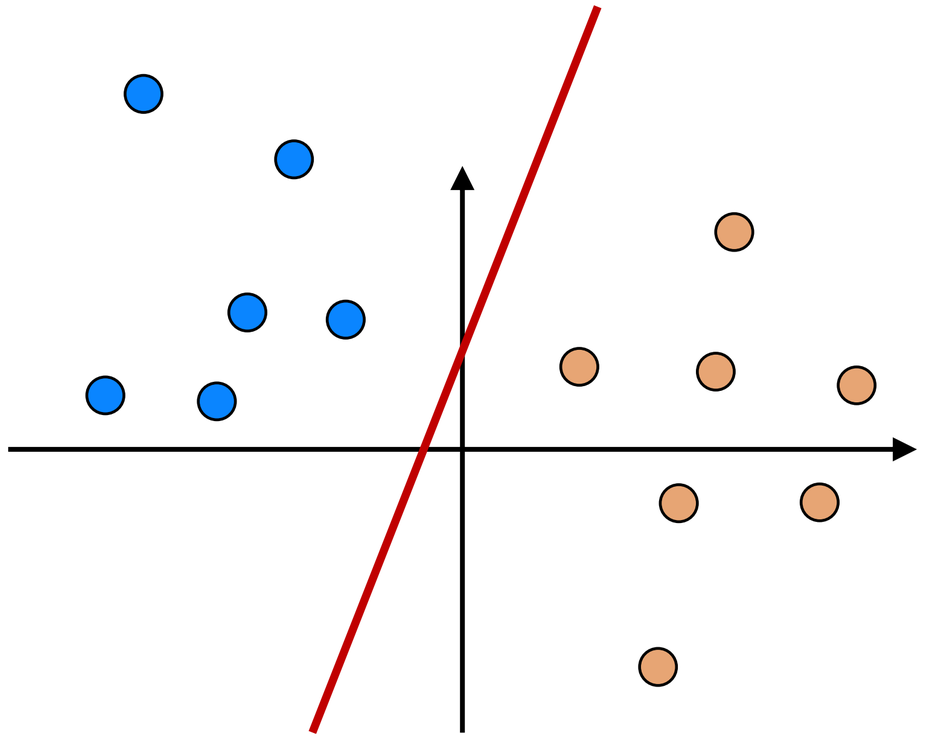
수정 : 시계 또는 반시계 방향으로 움직이기



학습이란?

두 색깔이 다른 원들을 가르는 선을 그리기

성공!



학습이란?

① 선 그리기

② 틀린 부분 찾기

③ 수정하기

학습이란?

① 선 그리기 → FeedForward
(순전파, Forward propagation)

② 틀린 부분 찾기 → Loss and Gradient
(역전파, Forward propagation)

③ 수정하기 → Update
(Optimize)

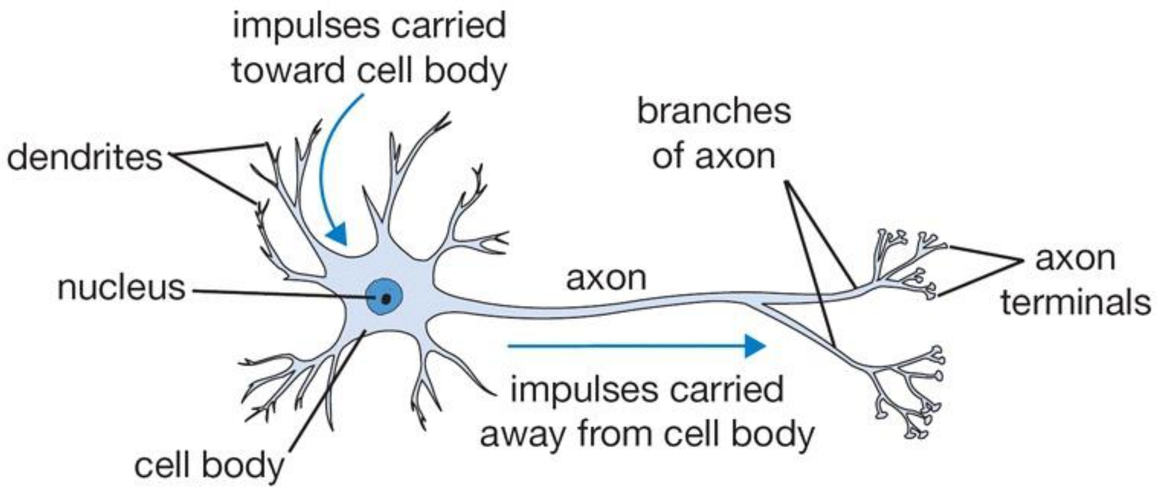
See you in the next lecture!

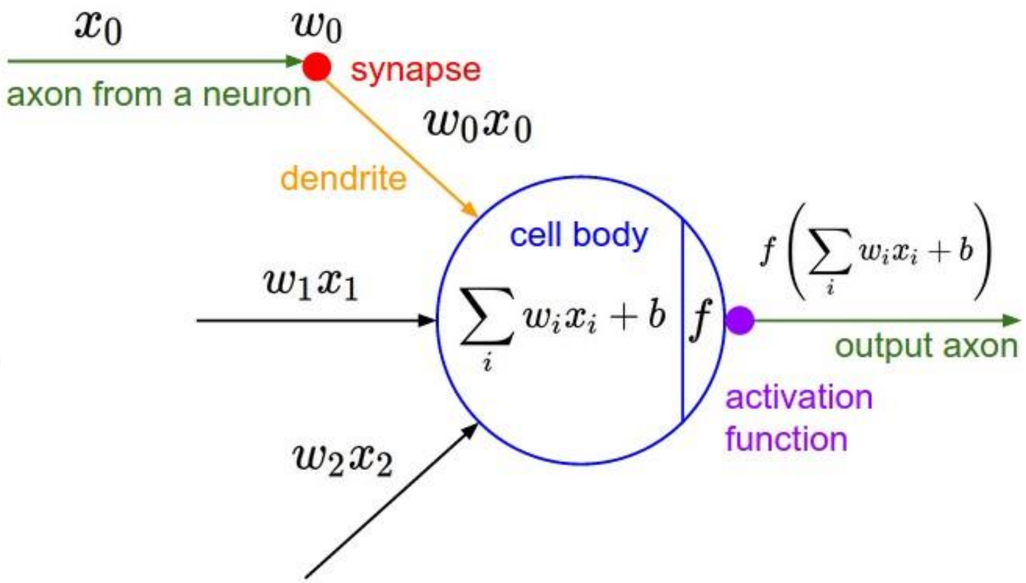
딥러닝 (Deep Learning) 기초

04 퍼셉트론과 활성화 함수

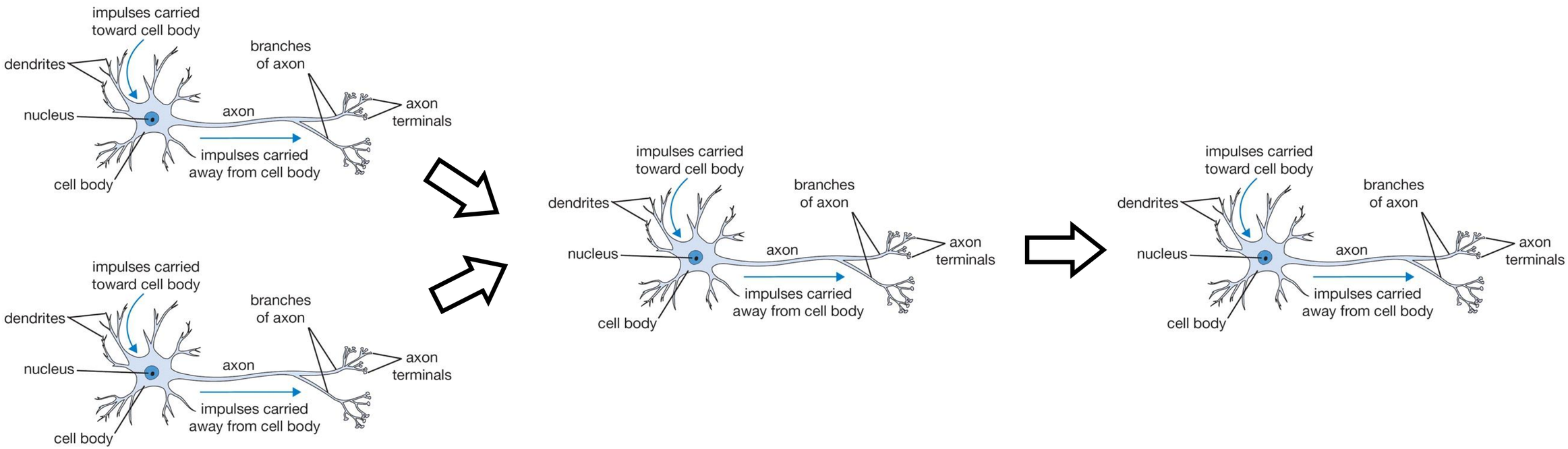
Deep & High Learning

Perceptron


impulses carried
toward cell body
dendrites
branches
of axon
nucleus
axon
axon
terminals
impulses carried
away from cell body
cell body


x_0
w_0
axon from a neuron
synapse
$w_0 x_0$
dendrite
cell body
$w_1 x_1$
$\sum_i w_i x_i + b$
f
$f\left(\sum_i w_i x_i + b\right)$
output axon
activation
function
$w_2 x_2$

Perceptron



① 이전 뉴런들로부터 신호 받기

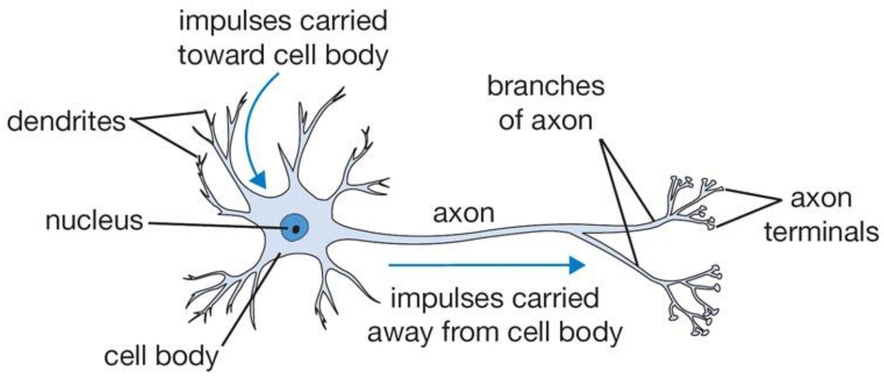
② 중요한 정보 고르기

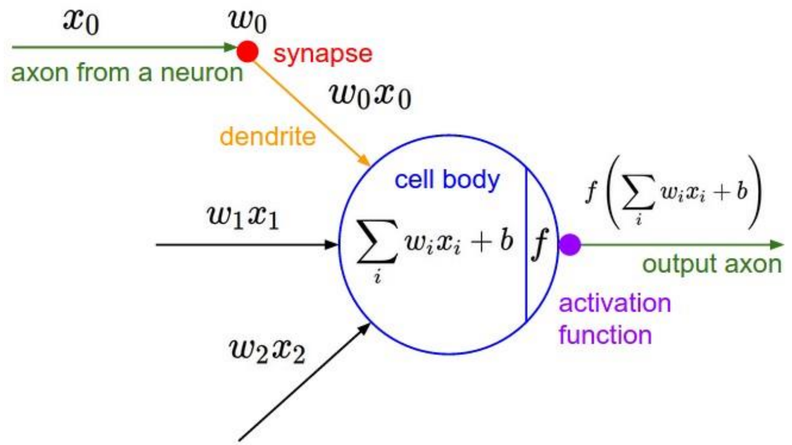
③ 받은 신호 처리하기

④ 다음 뉴런에게 전달할지 판단하기

⑤ 신호 전달하기

Perceptron





① 이전 뉴런들로부터 신호 받기

② 중요한 정보 고르기

③ 받은 신호 처리하기

④ 다음 뉴런에게 전달할지 판단하기

⑤ 신호 전달하기

① x_1, x_2

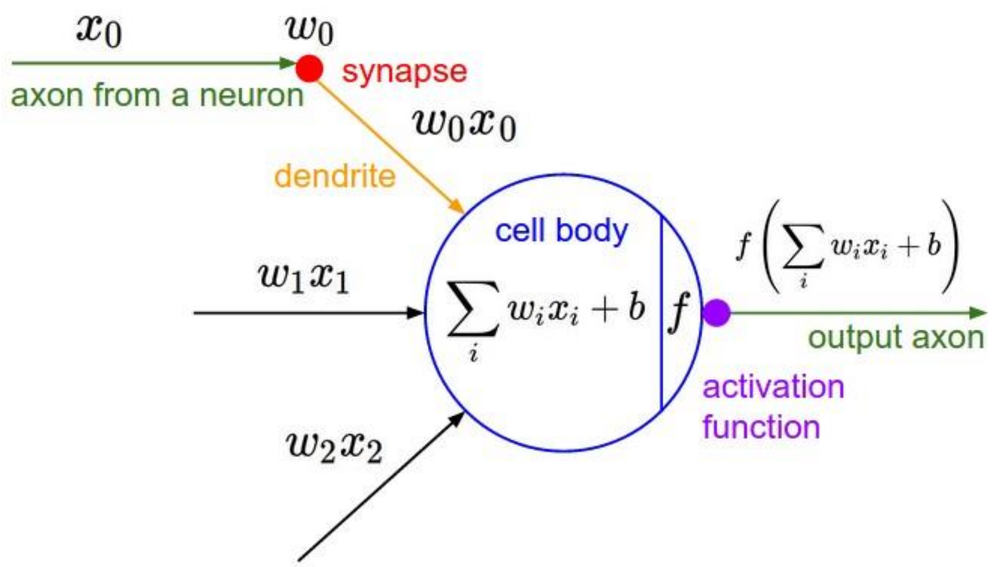
② w_1, w_2

③ $\sum w_i x_i + b$

④ Activation Function

⑤ Output

Perceptron



$$f\left(\sum_i w_i x_i + b\right)$$

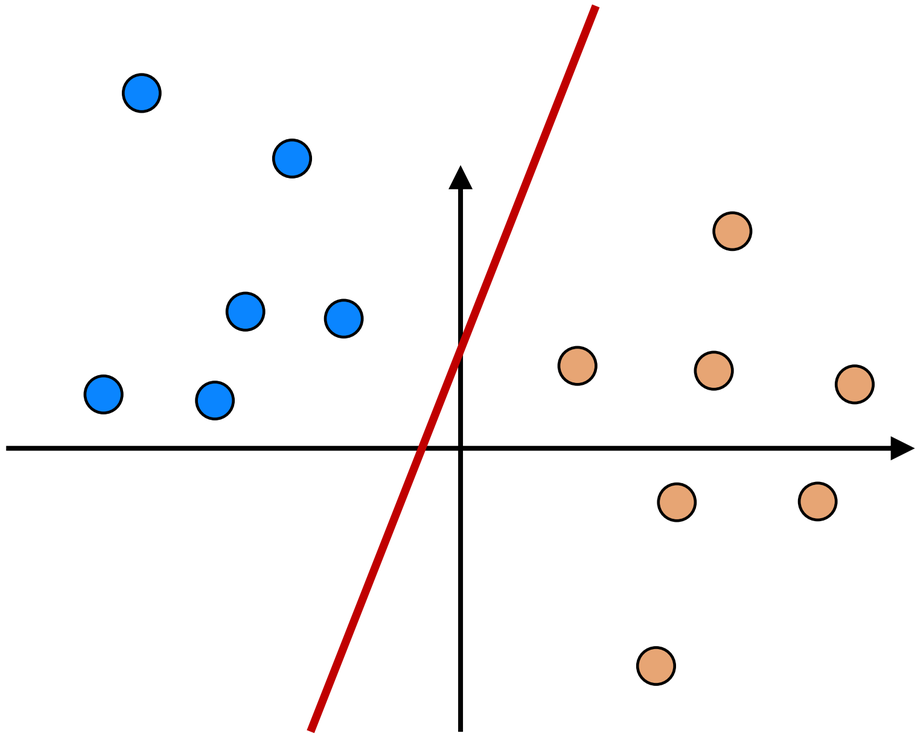
$$f\left(\begin{bmatrix} w_1 & w_2 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + b\right)$$

퍼셉트론에 들어오는 신호들(x_i)에 가중치를 곱하고($w_i x_i$), 이들과 편향(b)을 더한다.

이후 activation function (f)를 취해 출력값을 만든다.

목표 : 적절한 가중치(w_i)를 찾아내기!

퍼셉트론의 기하학적 의미



따라서 퍼셉트론은 선형 분류 문제를 풀 수 있음!

프랭크 로젠블랫
(Frank Rosenblatt)

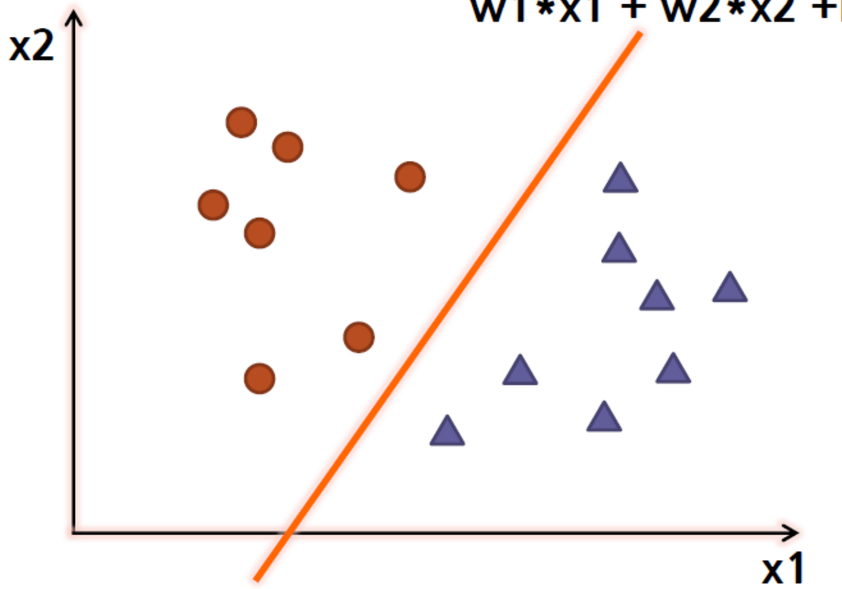
1962년,

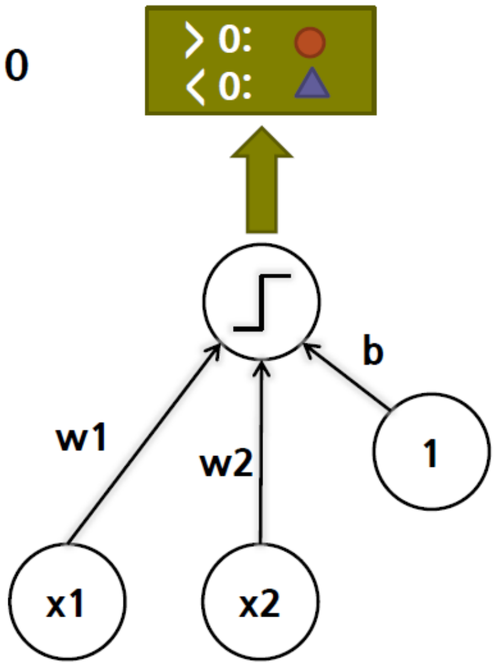
프랭크 로젠블랫에 의해

최초 제안됨

퍼셉트론의 기하학적 의미

$$f\left(\sum_i w_i x_i + b\right) \quad f\left(\begin{bmatrix} w_1 & w_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + b\right)$$



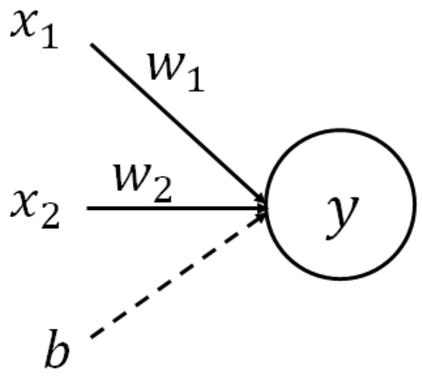


퍼셉트론 하나를 도식화하면

평면 안의 두 데이터를 가르는

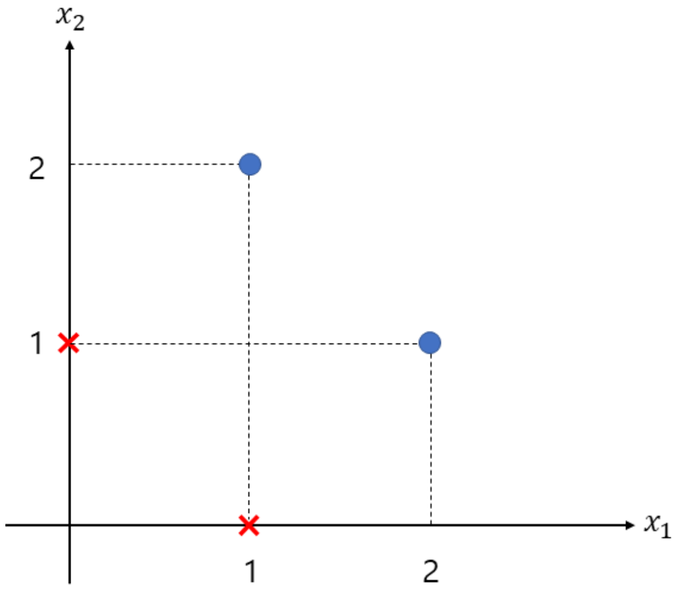
직선의 방정식이 됨

퍼셉트론의 기하학적 의미



$$y = f(w_1 x_1 + w_2 x_2 + b)$$

$$f(x) = \begin{cases} 1\ (x \geq 0) \\ 0\ (x < 0) \end{cases}$$



w_1	1
w_2	1
b	-2
f	단위 계단 함수

Activation Function (활성화 함수)

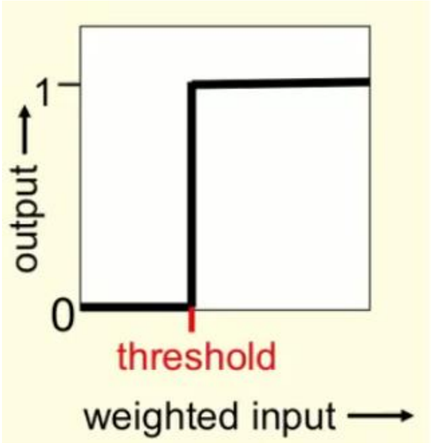
가중합($\sum w_i x_i$)이 임계값 (threshold) 이하이면 0(에 가까운 값)으로,

이상이면 1(에 가까운 값)으로 내보내는 함수

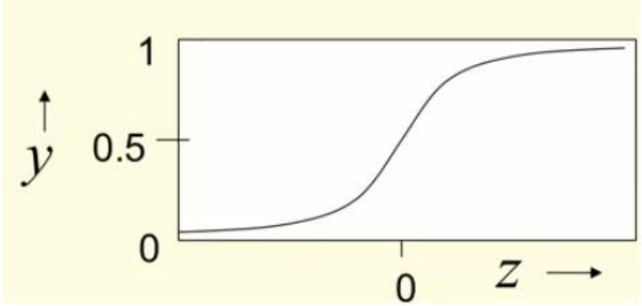
다음 퍼셉트론에 보낼 신호의 강도를 결정

비선형성을 주기 위하여 사용됨

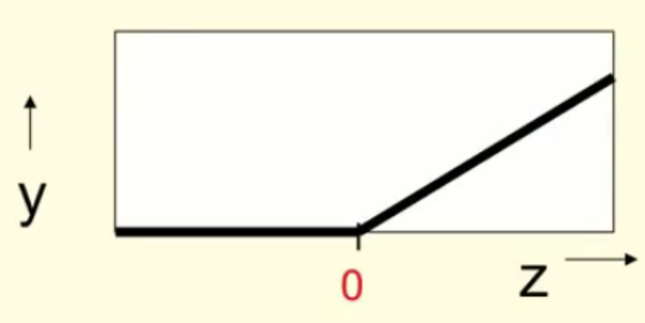
Step function



Sigmoid function

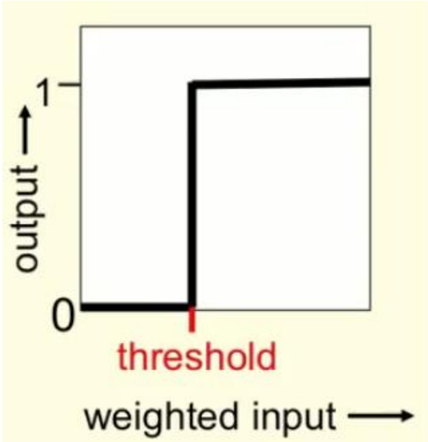


Rectified linear unit (ReLU)



Activation Function (활성화 함수)

Step function



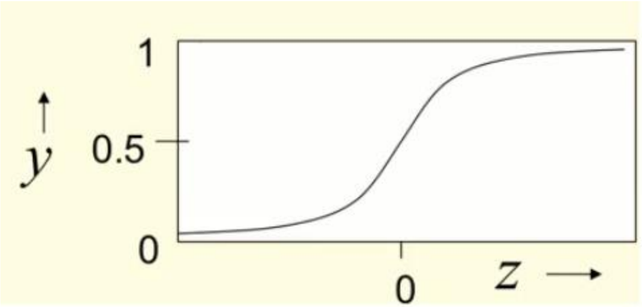
$$h(x) = \begin{cases} 0 & (x \leq 0) \\ 1 & (x > 0) \end{cases}$$

퍼셉트론에서 사용

미분값이 0 -> 한계 봉착

Activation Function (활성화 함수)

Sigmoid function

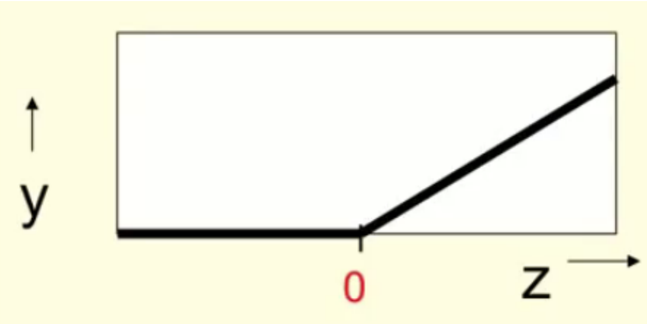


$$h(x) = \frac{1}{1 + \exp(-x)}$$

$h'(x) = h(x)(1 - h(x))$인 성질을 가짐

Activation Function (활성화 함수)

Rectified linear unit (ReLU)



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

미분이 매우 간단

현재 딥러닝 분야에서 가장 많이 사용

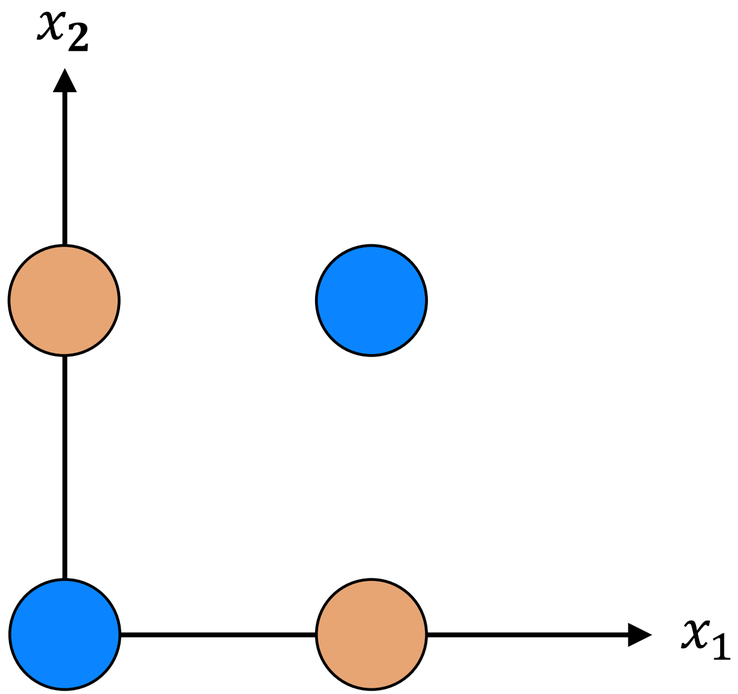
See you in the next lecture!

딥러닝 (Deep Learning) 기초

05 다층 퍼셉트론

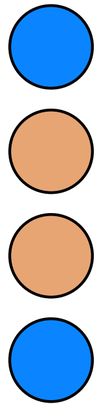
Deep & High Learning

단일 퍼셉트론의 한계

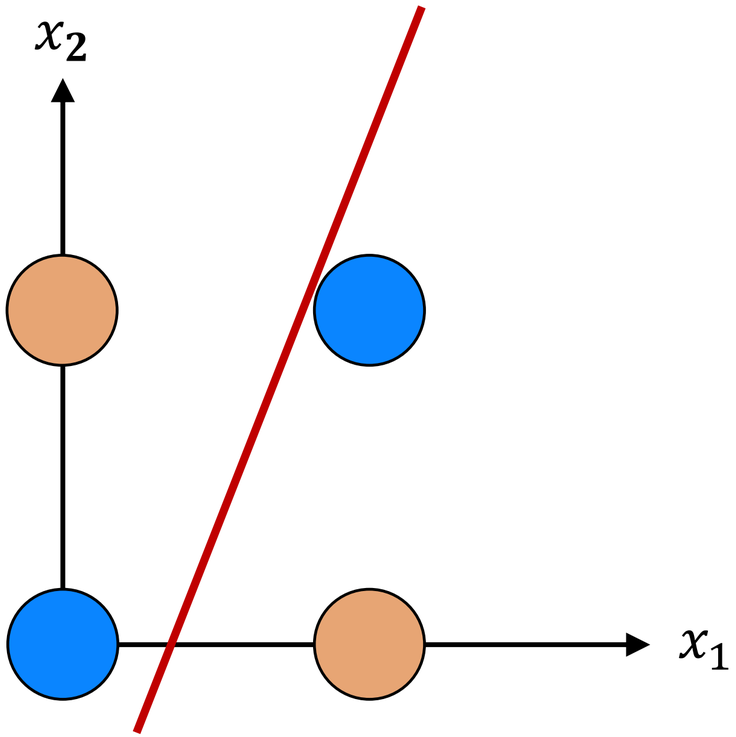


XOR 문제 : 두 값이 같으면 0, 다르면 1

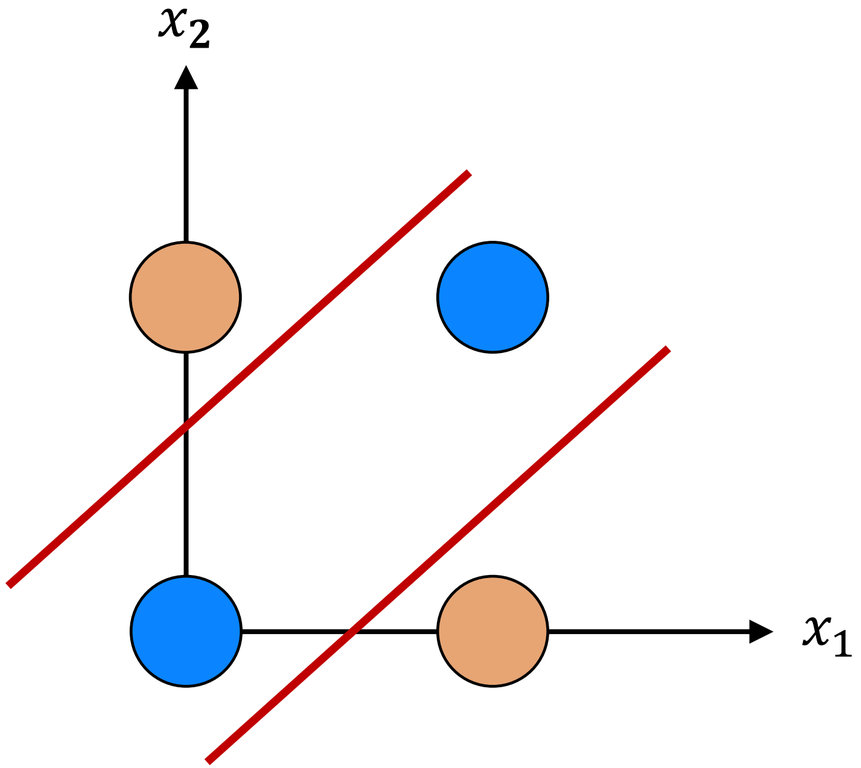
x_1	x_2	z(출력)
0	0	0
0	1	1
1	0	1
1	1	0



단일 퍼셉트론의 한계



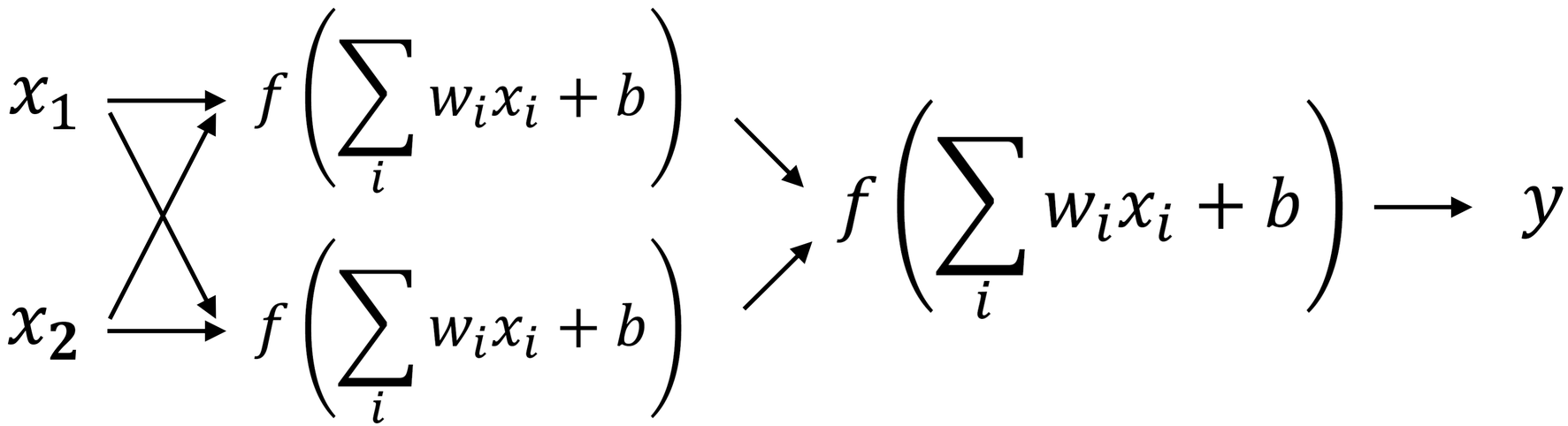
직선 하나로는 XOR 문제를 풀 수 없음



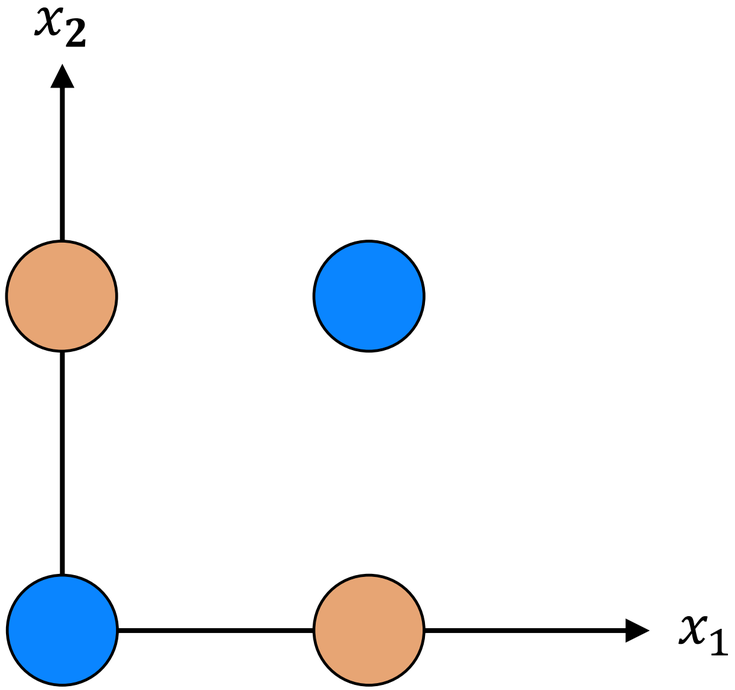
두 개의 직선으로는 XOR 문제를 풀 수 있음!

복수 개의 퍼셉트론이라면?

$$x_1 \searrow \\ \\ x_2 \nearrow \quad f\left(\sum_i w_i x_i + b\right) \longrightarrow y$$



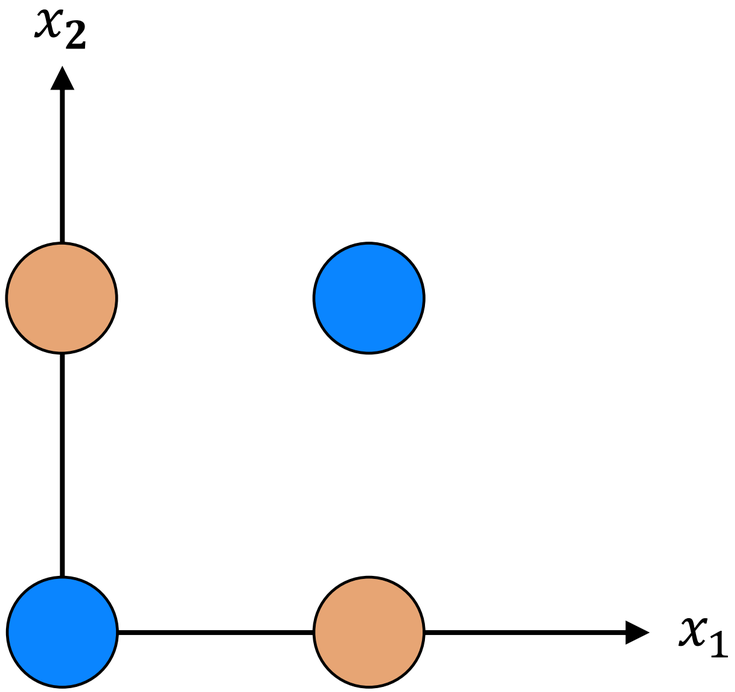
복수 개의 퍼셉트론이라면?



$$x_1 \rightarrow f\left(\sum_i w_i x_i + b\right) \rightarrow h_1$$
$$x_2 \rightarrow f\left(\sum_i w_i x_i + b\right) \rightarrow h_2$$
$$h_1, h_2 \rightarrow f\left(\sum_i w_i x_i + b\right) \rightarrow y$$

$$h_1 = f(x_1 - x_2 + 0.5)$$
$$h_2 = f(x_1 - x_2 - 0.5)$$

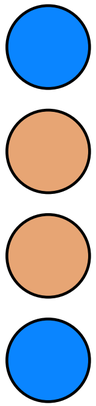
복수 개의 퍼셉트론이라면?



$$f = \begin{cases} 1\ (x \leq 0) \\ 0\ (x \geq 0) \end{cases}$$

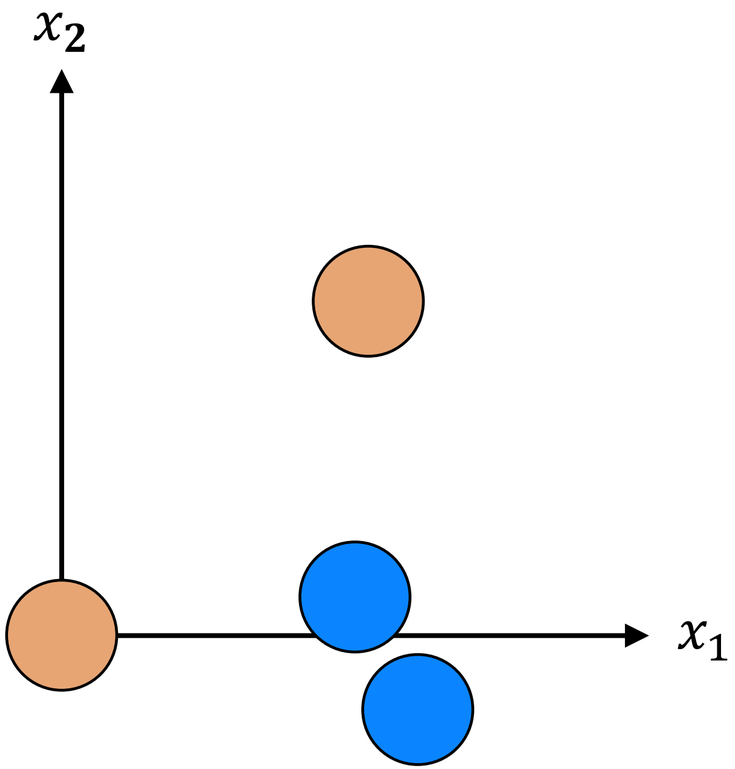
$$h_1 = f(x_1 - x_2 + 0.5)$$

$$h_2 = f(x_1 - x_2 - 0.5)$$



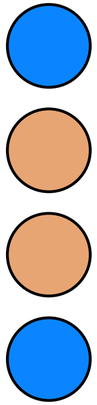
x	$wx + b$	h (출력)
(0, 0)	(0.5, -0.5)	(1, 0)
(0, 1)	(-0.5, -1.5)	(0, 0)
(1, 0)	(1.5, 0.5)	(1, 1)
(1, 1)	(0.5, -0.5)	(1, 0)

복수 개의 퍼셉트론이라면?



$$h_1 = f(x_1 - x_2 + 0.5)$$

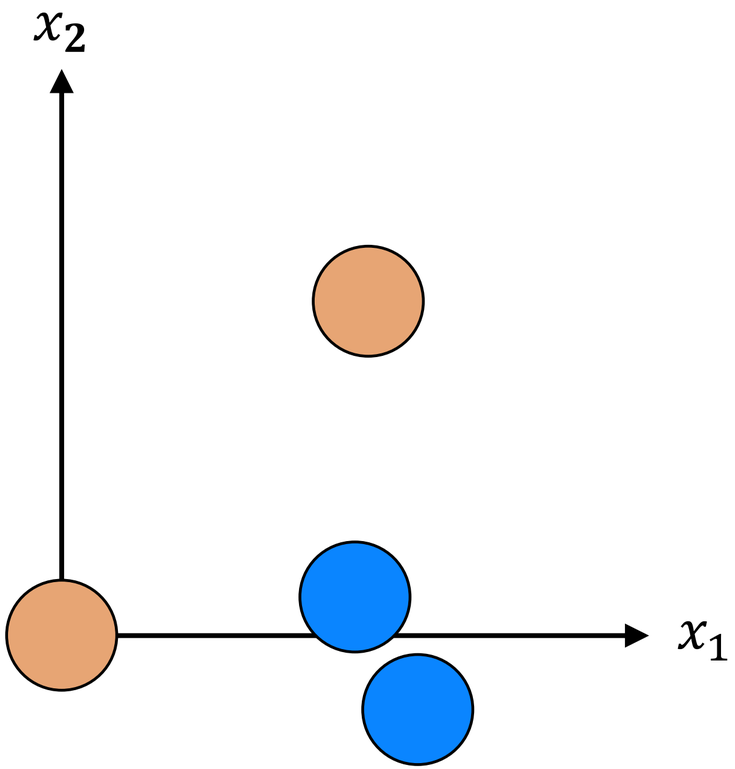
$$h_2 = f(x_1 - x_2 - 0.5)$$



x	$wx + b$	h (출력)
(0, 0)	(0.5, -0.5)	(1, 0)
(0, 1)	(-0.5, -1.5)	(0, 0)
(1, 0)	(1.5, 0.5)	(1, 1)
(1, 1)	(0.5, -0.5)	(1, 0)

희망이 생긴다!

복수 개의 퍼셉트론이라면?



$$f = \begin{cases} 1\ (x \leq 0) \\ 0\ (x \geq 0) \end{cases}$$

$$y = f(x_1 - x_2 - 0.5)$$

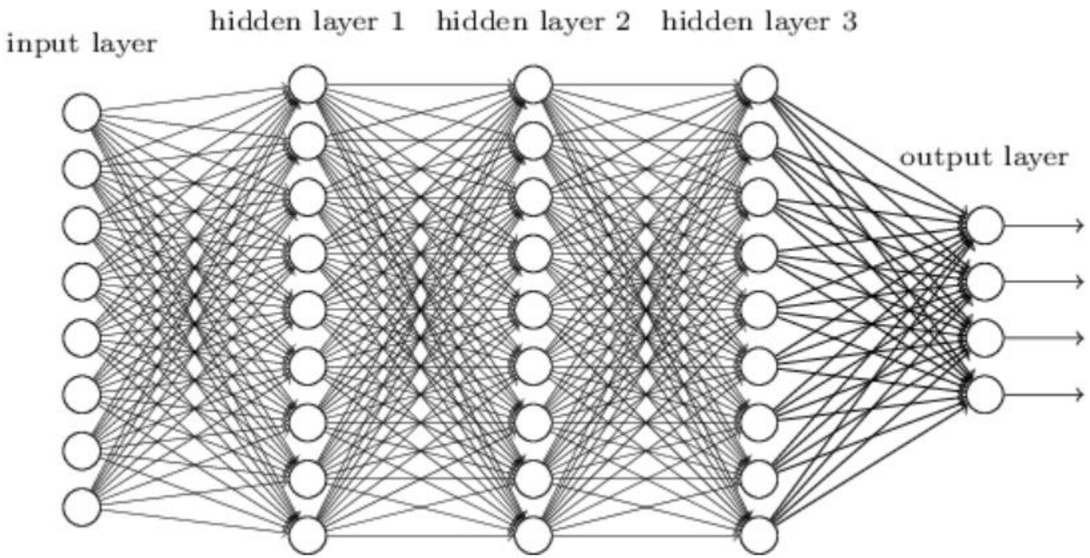
x	$wx + b$	y (출력)
(1, 0)	0.5	1
(0, 0)	-0.5	0
(1, 1)	-0.5	0
(1, 0)	0.5	1

Multi Layer Perceptron (MLP)

= Fully Connected Layer (FC layer)

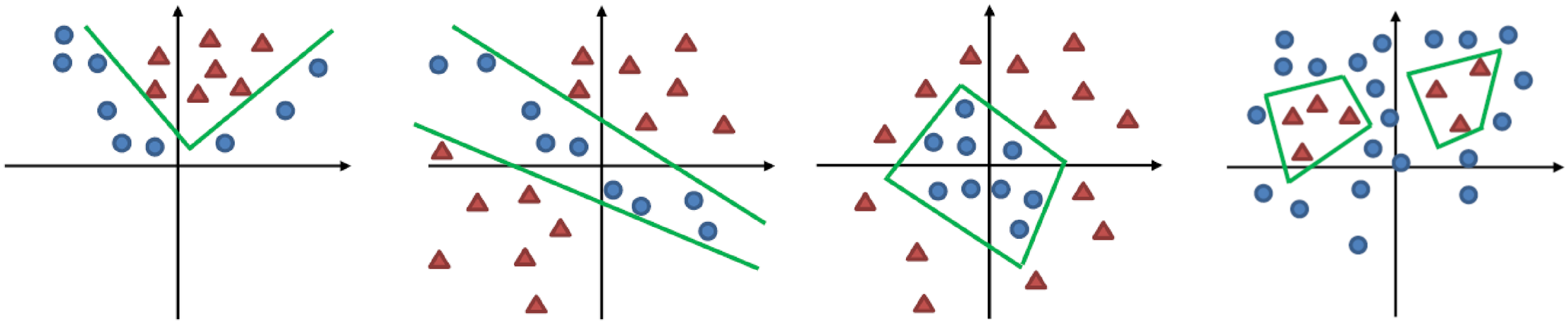
퍼셉트론이 여러 층 (layer)으로 쌓여 있는 구조, 가장 전통적이며 최소한의 성능 척도로 사용됨

현재도 많은 딥러닝 구조에 사용되며, 곱셈과 덧셈 연산으로만 이루어짐

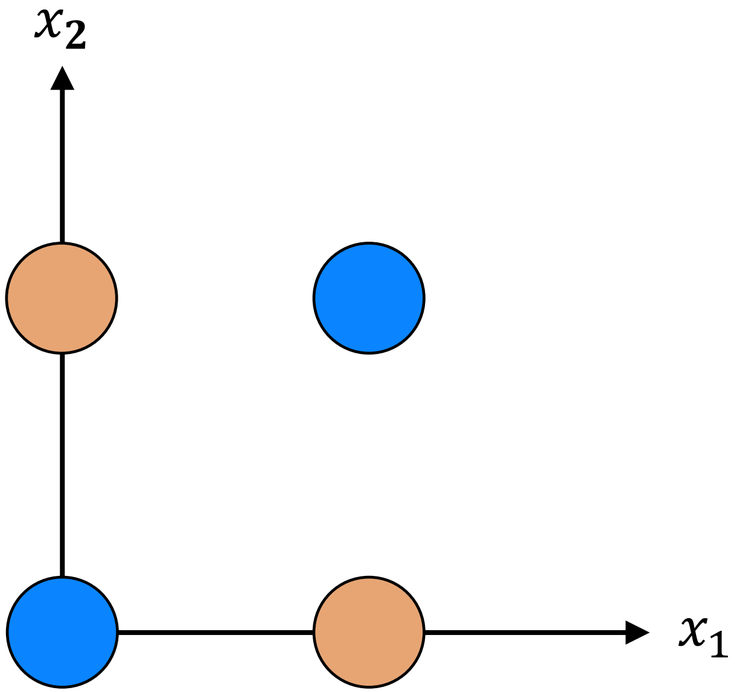


Multi Layer Perceptron (MLP)

= MLP를 활용하면 비선형 문제, 복잡한 문제도 해결 가능



복수 개의 퍼셉트론이라면?



$$x_1 \rightarrow f\left(\sum_i w_i x_i + b\right) \rightarrow h_1$$
$$x_2 \rightarrow f\left(\sum_i w_i x_i + b\right) \rightarrow h_2$$
$$h_1, h_2 \rightarrow f\left(\sum_i w_i x_i + b\right) \rightarrow y$$

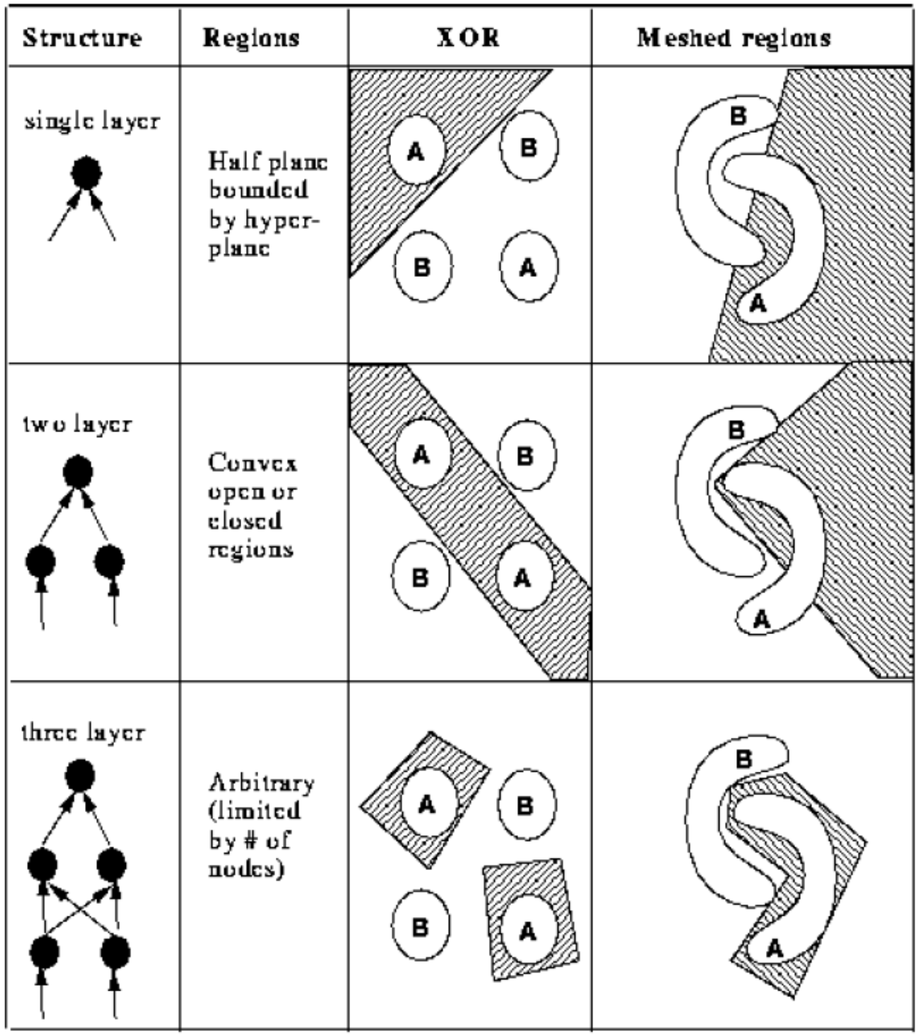
Multi Layer Perceptron (MLP)

여러 개의 경계가 필요하다

-> 다수의 Unit 사용

복잡한 경계선이 필요하다

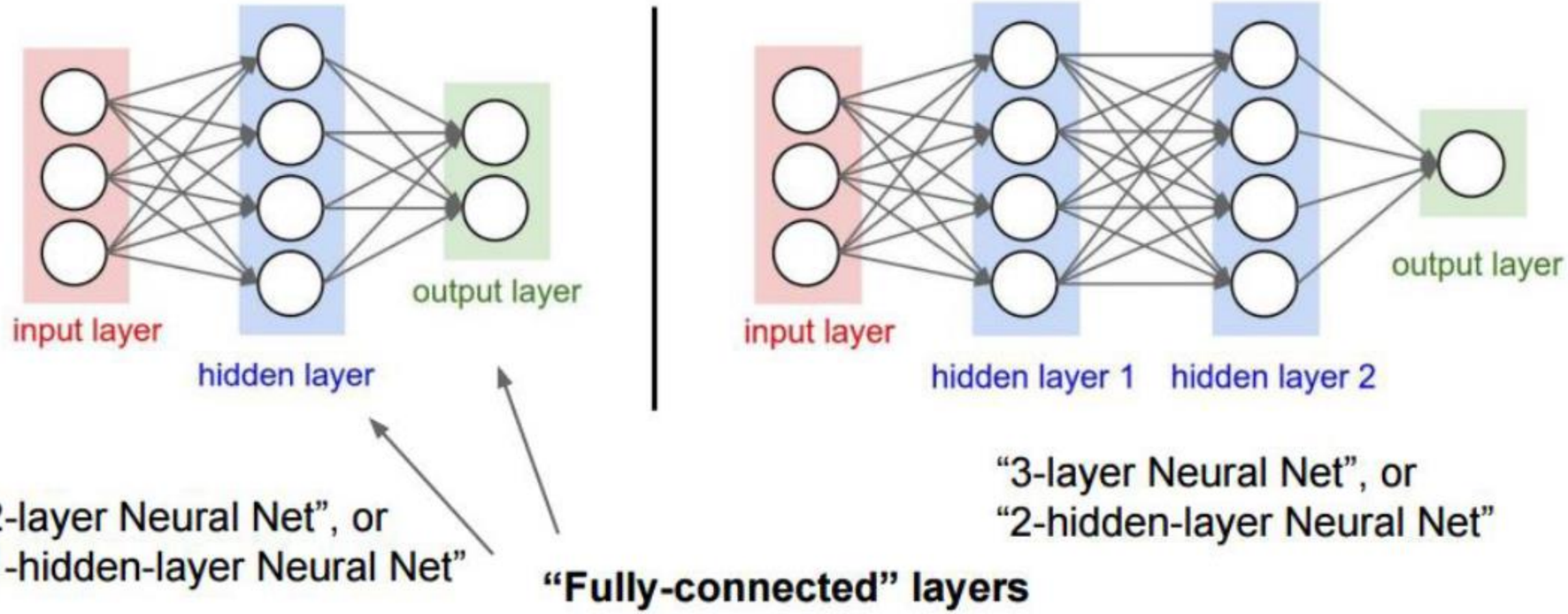
-> 다수의 Layer 사용



Multi Layer Perceptron (MLP)

of hidden layers <=1
→ Shallow neural network.

# of hidden layers >=2
→ deep neural network



학습이란?

① 선 그리기 → FeedForward (순전파, Forward propagation)

② 틀린 부분 찾기 → Loss and Gradient (역전파, Forward propagation)

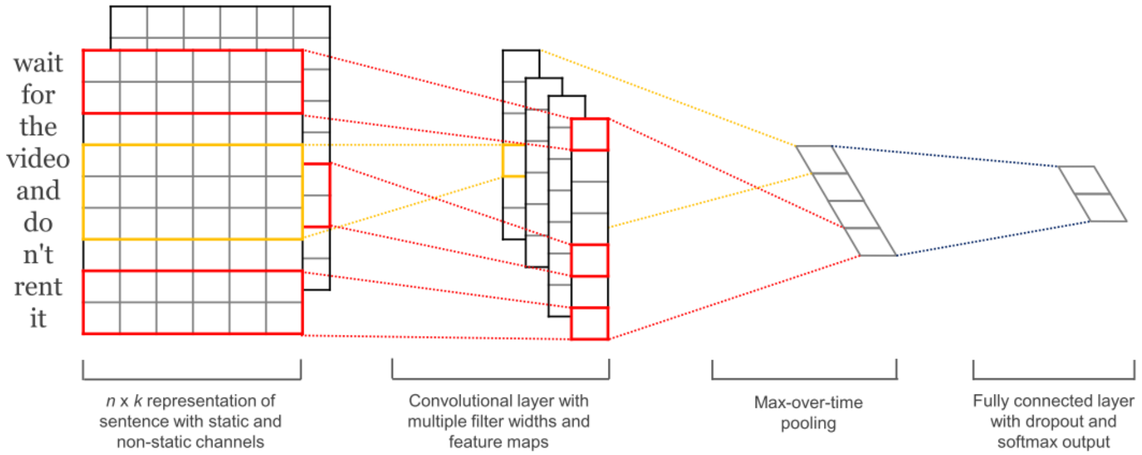
③ 수정하기 → Update (Optimize)

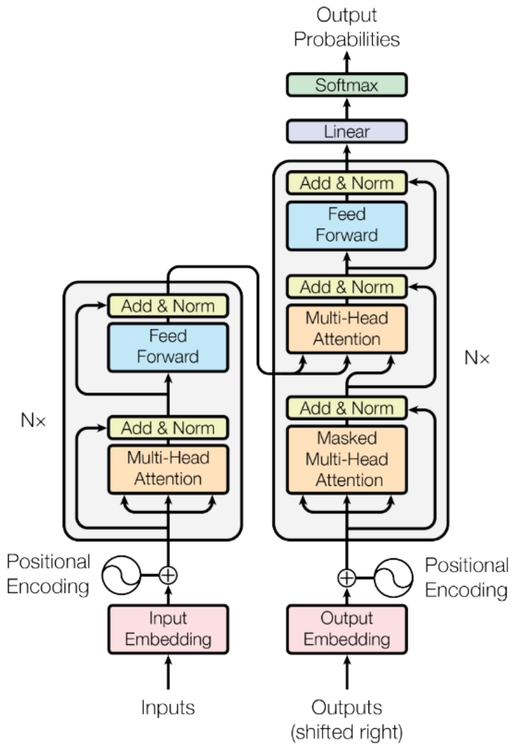
FeedForward (순전파, Forward Propagation)

네트워크가 **입력으로 데이터**를 받아 **출력**을 내보내는 과정

사용자가 설정한 네트워크 구조를 따라가며 진행

연산 과정은 **행렬**의 형태로 계산

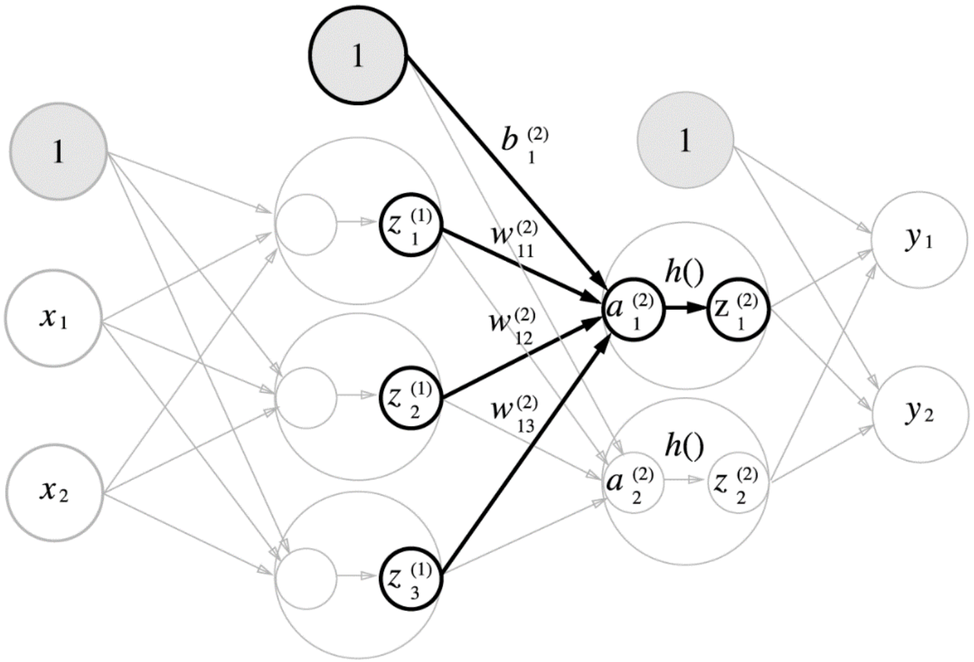


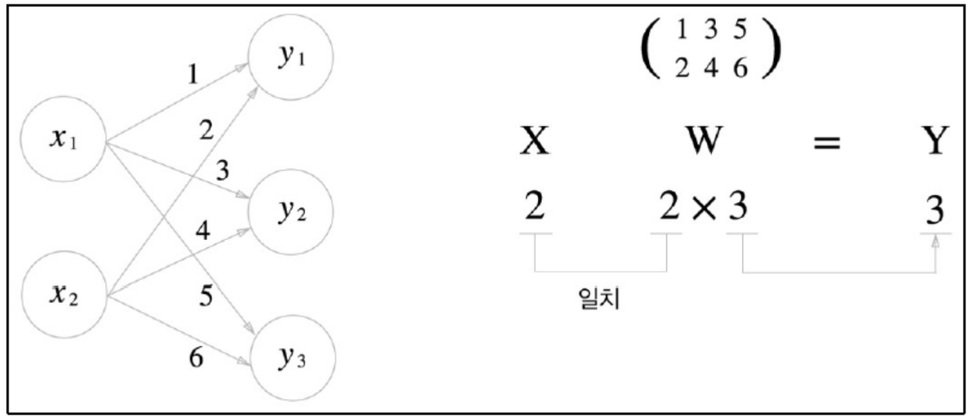


MLP FeedForward

입력과 가중치의 행렬곱을 이용해 출력을 계산하는 것

= Affine 연산

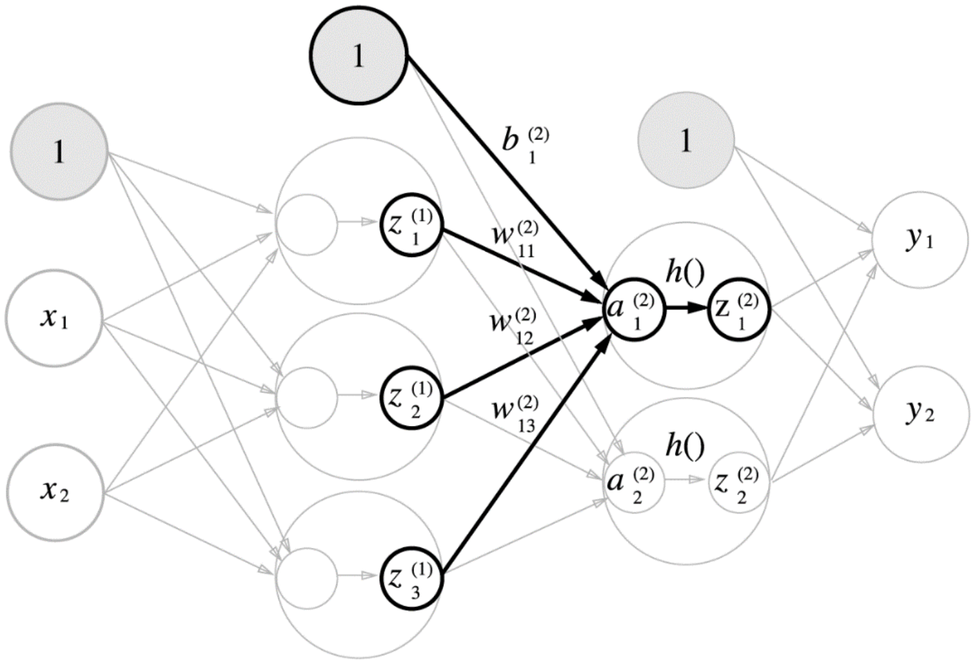




MLP FeedForward

모든 layer를 통과할 때까지 계산

최종적으로 문제에서 정의한 결과물의 점수 (score)가 나옴



Cat: 1.5

Dog: 3.7

See you in the next lecture!

딥러닝 (Deep Learning) 기초

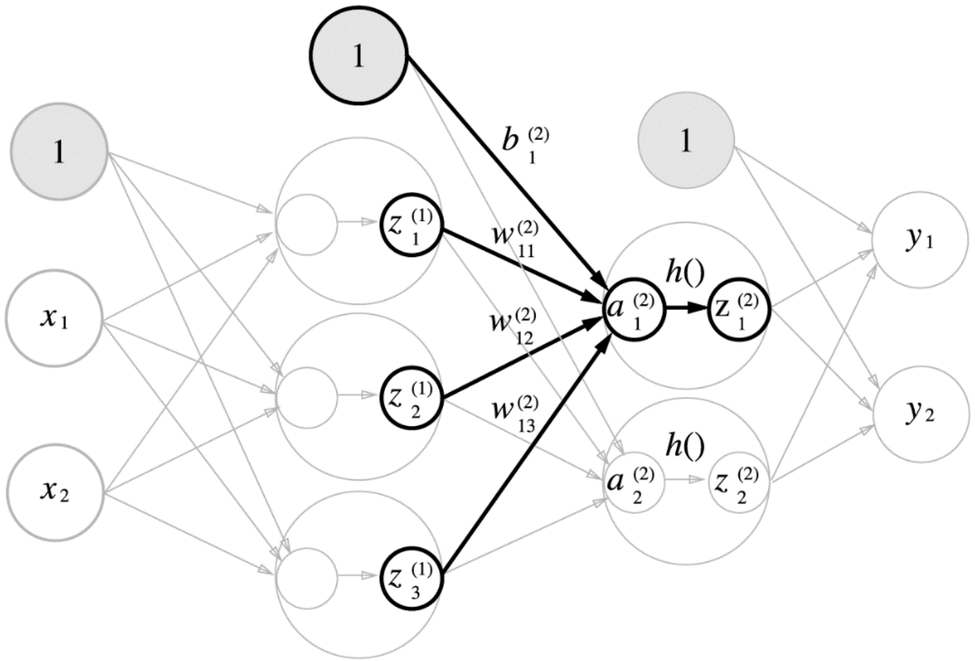
06 소프트맥스 함수와 손실 함수

Deep & High Learning

MLP FeedForward

모든 layer를 통과할 때까지 계산

최종적으로 문제에서 정의한 결과물의 점수 (score)가 나옴



Cat: 1.5

Dog: 3.7

학습이란?

① 선 그리기 → FeedForward (순전파, Forward propagation)

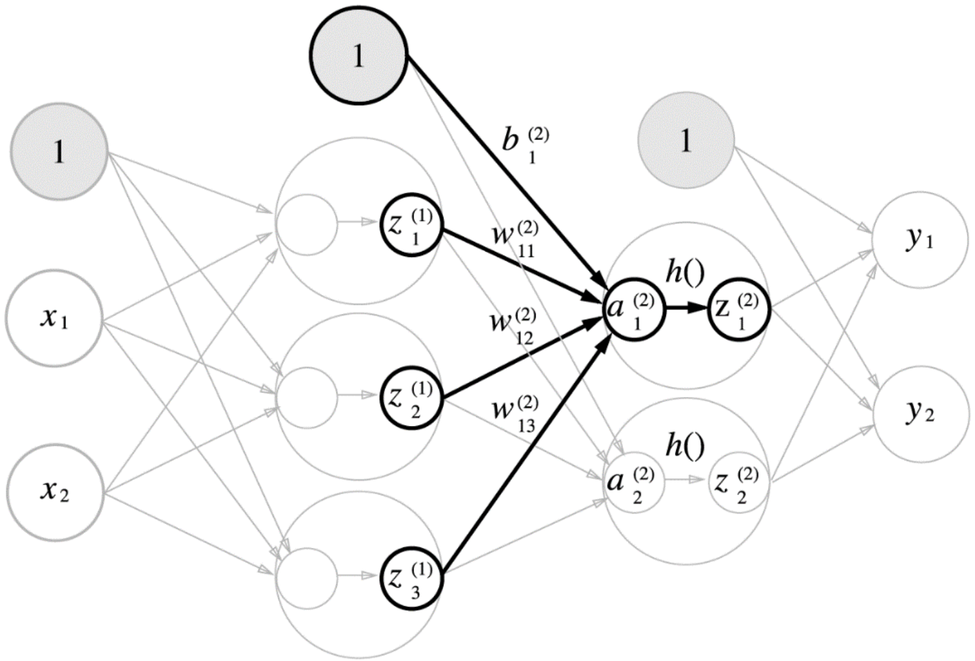
② 틀린 부분 찾기 → Loss and Gradient (역전파, Forward propagation)

③ 수정하기 → Update (Optimize)

MLP FeedForward

모든 layer를 통과할 때까지 계산

최종적으로 문제에서 정의한 결과물의 점수 (score)가 나옴



Cat: 1.5

Dog: 3.7

Softmax Function

분류 문제에 사용

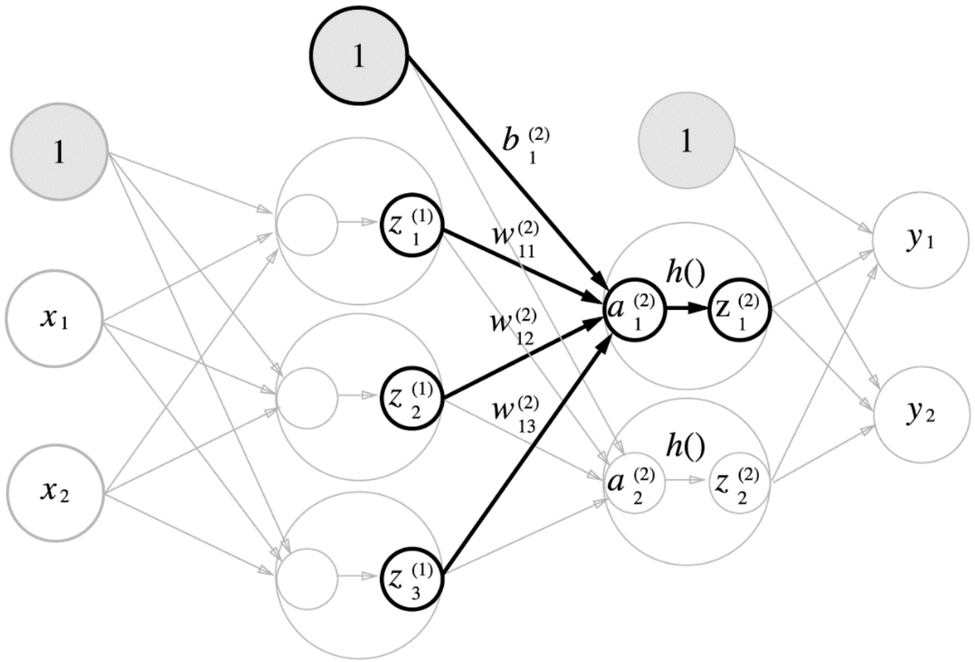
입력의 지수함수를 모든 입력의 지수 함수의 합으로 나눔

출력을 확률값으로 나타냄 (출력의 총합이 1)

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

Softmax Function

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



Cat: 1.5

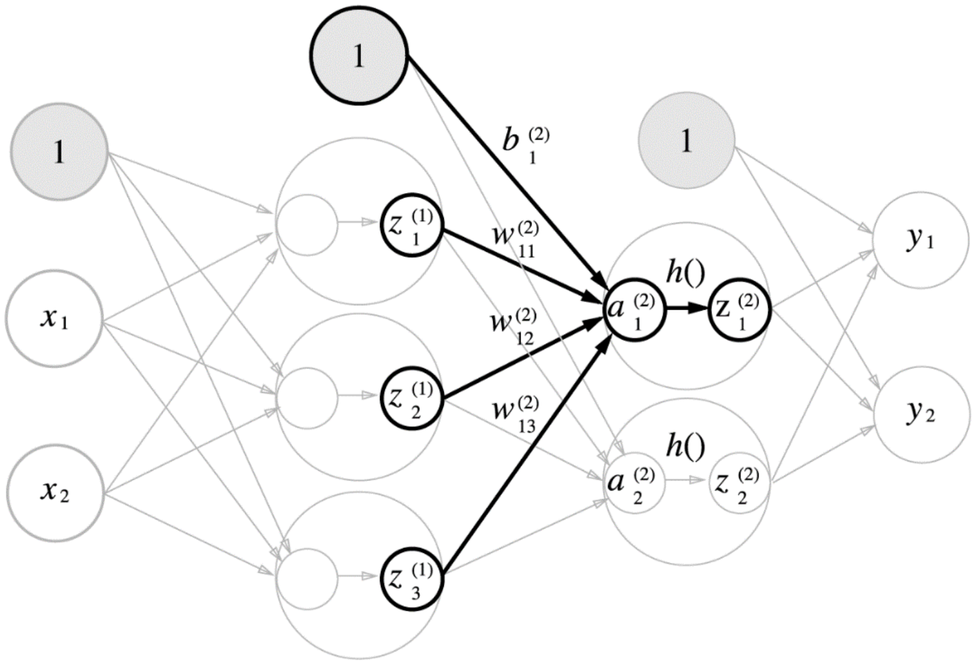
Dog: 3.7

$$\frac{e^{1.5}}{e^{1.5}+e^{3.7}} = 0.1$$

$$\frac{e^{3.7}}{e^{1.5}+e^{3.7}} = 0.9$$

Softmax Function

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



Cat: 1.5

Dog: 3.7

$$\frac{e^{1.5}}{e^{1.5}+e^{3.7}} = 0.1$$

$$\frac{e^{3.7}}{e^{1.5}+e^{3.7}} = 0.9$$

$$* \frac{1.5}{1.5+3.7} + \frac{3.7}{1.5+3.7} = 1$$ <-이렇게는 안되나요?

Loss Function (손실 함수)

출력으로 나온 데이터가 실제 정답과 얼마나 차이가 나는지를 나타냄

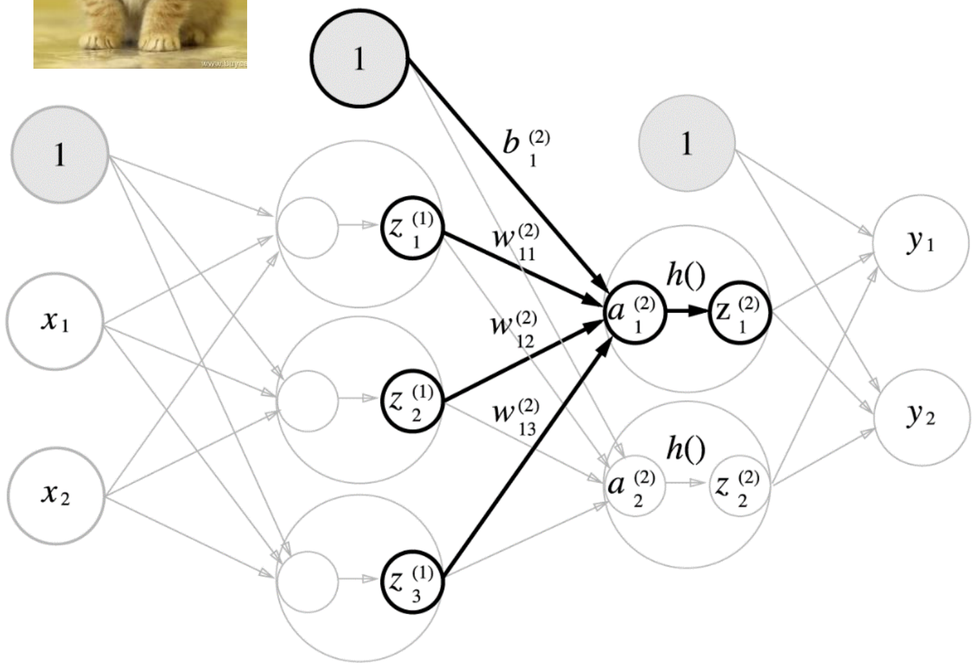
학습의 방향을 설정하는 아주 중요한 부분!

Loss Function의 결과값을 토대로 가중치들이 업데이트 됨
학습이 거듭 반복되면 해당 Loss function은 값이 작아짐

회귀(Regression) 문제 : Mean Squared Error
분류(Classification) 문제 : Cross Entropy

Mean Squared Error (MSE, 평균 제곱 오차)

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



Cat: 0.1 ⟷ Label: 1

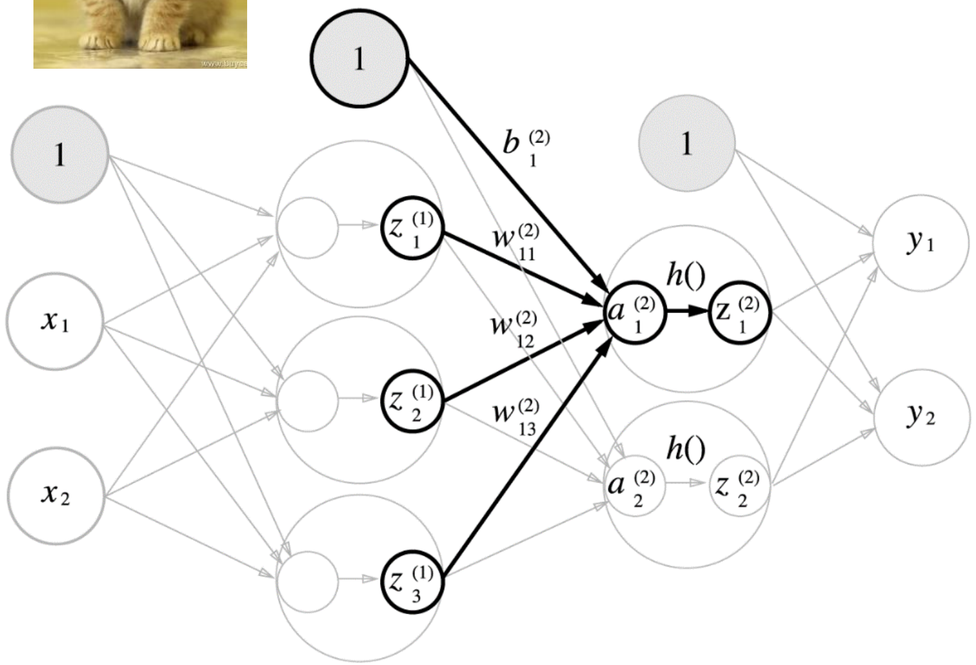
Dog: 0.9 ⟷ Label : 0

Error

$(0.1 - 1)^2$

$(0.9 - 0)^2$

Cross Entropy Error (CSE. 교차 엔트로피 오차)



$$E = -\sum_k t_k \log y_k$$

Cat: 0.1 ⟷ Label: 1

Dog: 0.9 ⟷ Label : 0

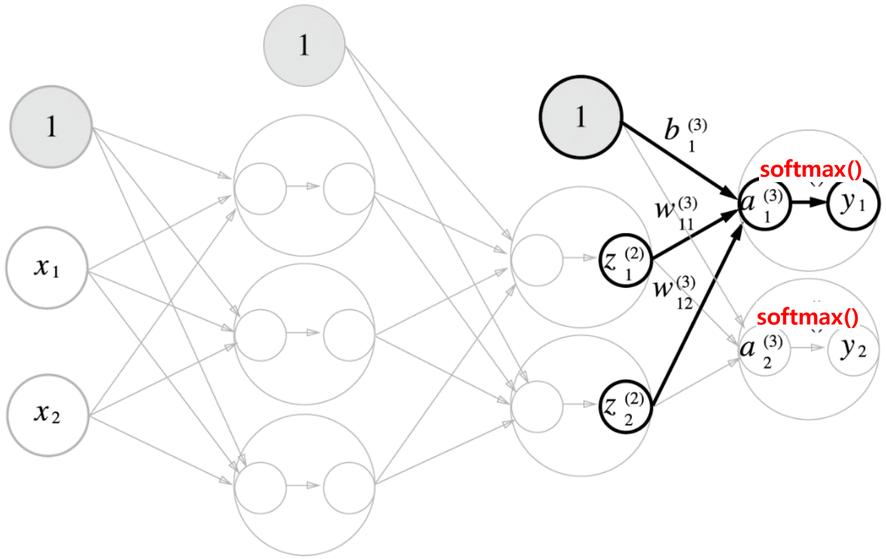
Error

$1 \times log0.1$

$0 \times log0.9$

정답일 때의 출력이 전체 값을 결정!

Loss 계산해 보기



Cat: ? ⟷ Label: 1

Dog: ? ⟷ Label : 0

Error

?

?

z_1	z_2	w_{11}	w_{12}	w_{21}	w_{22}	b_1	b_2	activation
0.5	0.1	0.2	0.3	0.1	0.4	0.7	0.2	softmax

Loss 계산해 보기

z_1	z_2	w_{11}	w_{12}	w_{21}	w_{22}	b_1	b_2	activation
0.5	0.1	0.2	0.3	0.1	0.4	0.7	0.2	softmax

$$z = [0.5\ 0.1]$$

$$w = \begin{bmatrix} 0.2 & 0.1 \\ 0.3 & 0.4 \end{bmatrix}$$

$$b = [0.7\ 0.2]$$

$$a = wz + b$$

$$= \begin{bmatrix} 0.2 & 0.1 \\ 0.3 & 0.4 \end{bmatrix} \begin{bmatrix} 0.5 \\ 0.1 \end{bmatrix} + \begin{bmatrix} 0.7 \\ 0.2 \end{bmatrix}$$

$$a = \begin{bmatrix} 0.11 \\ 0.19 \end{bmatrix} + \begin{bmatrix} 0.7 \\ 0.2 \end{bmatrix}$$

$$= \begin{bmatrix} 0.81 \\ 0.39 \end{bmatrix}$$

$$softmax(a) \approx \begin{bmatrix} 0.8 \\ 0.2 \end{bmatrix}$$

$$e^{0.81} \approx 2.25$$
$$e^{0.39} \approx 0.48$$

Loss 계산해 보기

z_1	z_2	w_{11}	w_{12}	w_{21}	w_{22}	b_1	b_2	activation
0.5	0.1	0.2	0.3	0.1	0.4	0.7	0.2	softmax

MSE Loss

$$=\frac{1}{2}\{(0.8-1)^2+(0.2-0)^2\}=0.04$$

Cross Entropy Loss

$$=-(1\times log0.8+0\times log0.2)=0.096$$

See you in the next lecture!

딥러닝 (Deep Learning) 기초

07 역전파 - 기초 수학 복습

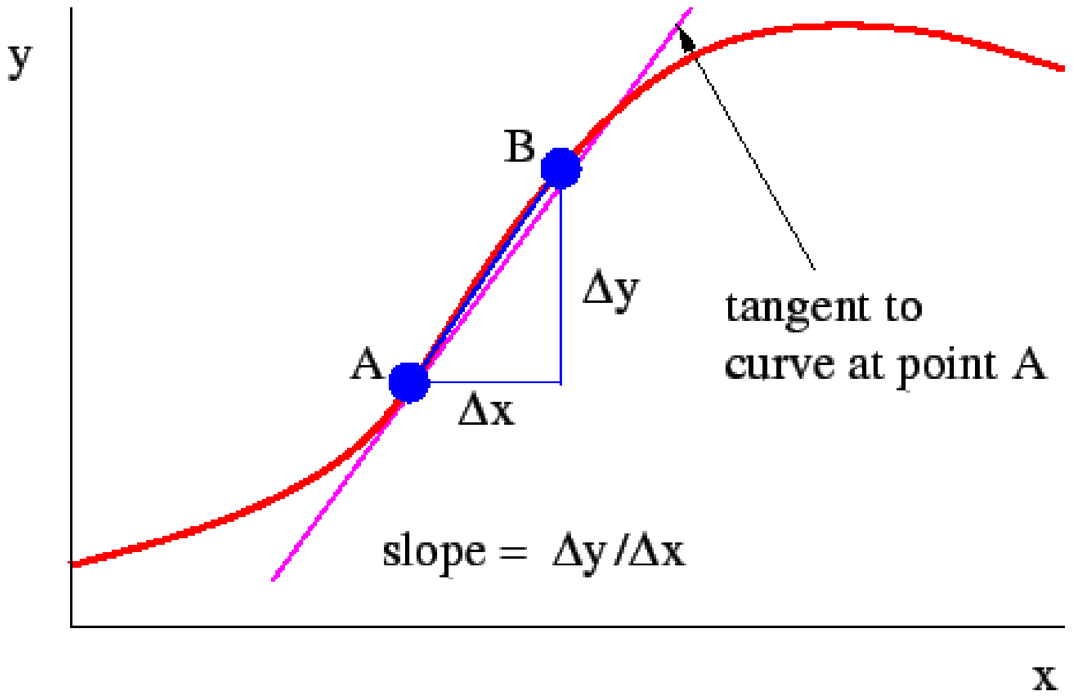
Deep & High Learning

기초 수학 복습 - 미분

미분 (Differentiation): 특정 함수 안에서 정의역에 속하는 한 점 x의 순간변화율

그래프로 표현된 함수에서 미분은 특정 지점의 기울기를 나타냄

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



https://en.wikibooks.org/wiki/Waves/Derivatives

기초 수학 복습 - 미분

$f(x) = ax^n$ 이라 할 때, $\frac{d}{dx} f(x) = f'(x) = n \times ax^{n-1}$

1) $f(x) = 3$

2) $f(x) = x$

3) $f(x) = 2x$

4) $f(x) = x + 3$

5) $f(x) = x^2$

6) $f(x) = 3x^4 - 2x^3 + 2$

https://en.wikibooks.org/wiki/Waves/Derivatives

기초 수학 복습 - 편미분

편미분 (Partial differentiation): 특정 함수가 2개 이상의 변수로 이루어진 경우,

다른 변수는 상수로 취급하고 특정 변수만을 이용해 미분하는 것

$z = f(x, y)$일 때,

z를 x로 편미분하면 $\frac{\partial z}{\partial x}$ 라고 표시

z를 y로 편미분하면 $\frac{\partial z}{\partial y}$ 라고 표시

https://en.wikibooks.org/wiki/Waves/Derivatives

기초 수학 복습 - 편미분

$z = f(x, y)$일 때, x로 편미분하면 y를 상수로 취급하고 미분

반대의 경우도 마찬가지

1) $f(x, y) = xy$

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

$$\frac{\partial f}{\partial y} =$$

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

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

$$\frac{\partial f}{\partial y} =$$

기초 수학 복습 - Chain Rule (연쇄법칙)

특정 함수가 2개 이상의 변수로 이루어진 상황에서, 변수들이 다른 매개변수로 해당 함수 안에서 표현될 때, **매개변수를 이용해 특정 변수의 편미분 값을 구하기 위해 사용**

$z = f_1(g)$ 이며 $g = f_2(x, y)$ 로 표현되는 경우

$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial g}\frac{\partial g}{\partial x'}$, $\frac{\partial z}{\partial y} = \frac{\partial z}{\partial g}\frac{\partial g}{\partial y}$ 를 이용하여 편미분

https://en.wikibooks.org/wiki/Waves/Derivatives

기초 수학 복습 - Chain Rule (연쇄법칙)

$z = g + 3, g = xy$인 경우,

$$\frac{\partial z}{\partial g} = 1, \frac{\partial g}{\partial x} = y, \frac{\partial g}{\partial y} = x$$

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial g}\frac{\partial g}{\partial x} = 1 \times y = y,$$

$$\frac{\partial z}{\partial y} = \frac{\partial z}{\partial g}\frac{\partial g}{\partial y} = 1 \times x = x.$$

https://en.wikibooks.org/wiki/Waves/Derivatives

기초 수학 복습 - Chain Rule (연쇄법칙)

$z = g + 3, g = xy$인 경우,

$$\frac{\partial z}{\partial g} = 1, \frac{\partial g}{\partial x} = y, \frac{\partial g}{\partial y} = x$$

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial g}\frac{\partial g}{\partial x} = 1 \times y = y,$$

$$\frac{\partial z}{\partial y} = \frac{\partial z}{\partial g}\frac{\partial g}{\partial y} = 1 \times x = x.$$

$\frac{\partial z}{\partial x}$를 구하고 싶다면,

$\frac{\partial z}{\partial g}$와 $\frac{\partial g}{\partial x}$를 따로 구해서 곱해서 구한다

https://en.wikibooks.org/wiki/Waves/Derivatives

기초 수학 복습 - Chain Rule (연쇄법칙)

$$f = g + 3$$
$$g = x + y$$

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

$$f = 2g$$
$$g = x - y$$

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

$$f = g - 3$$
$$g = xy$$

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

https://en.wikibooks.org/wiki/Waves/Derivatives

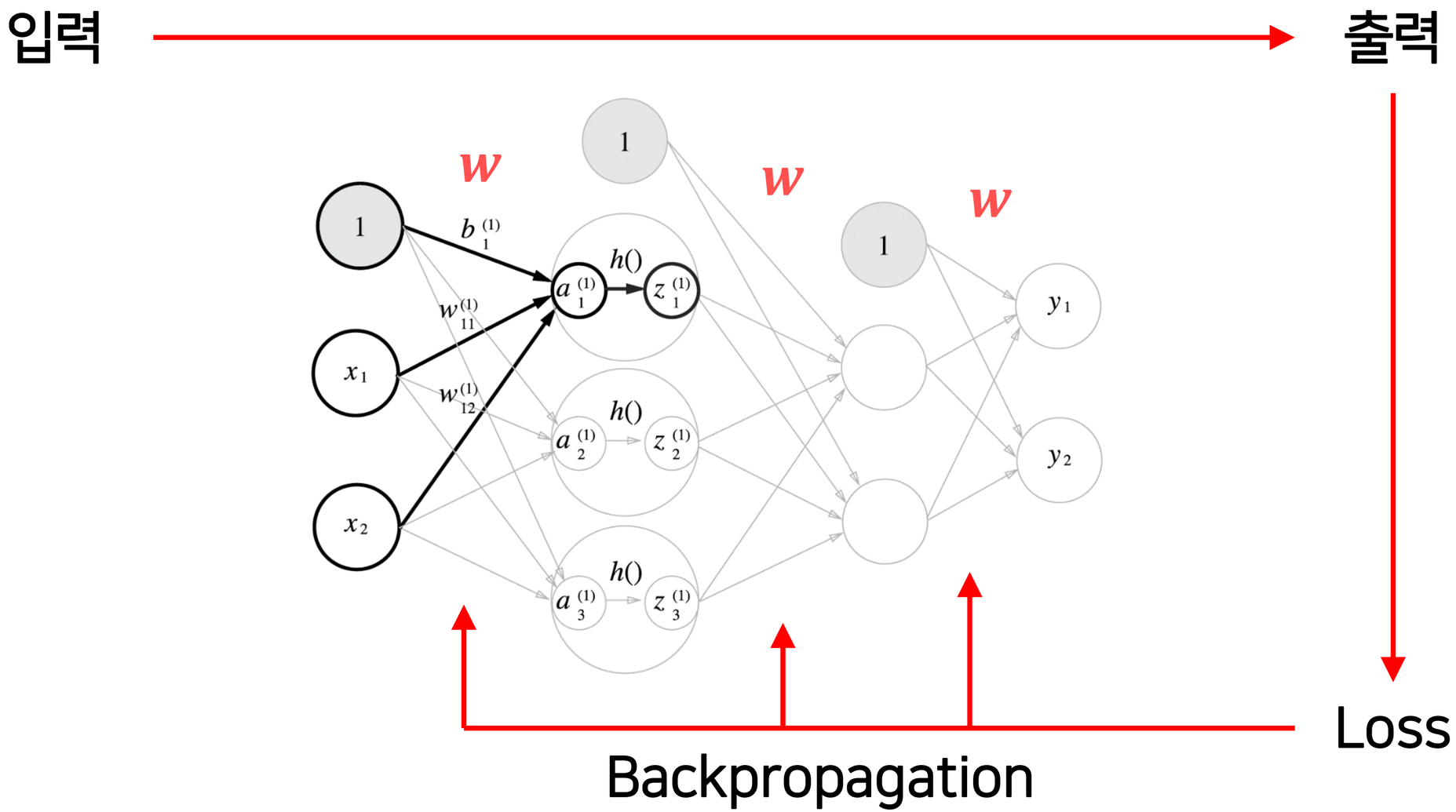
See you in the next lecture!

딥러닝 (Deep Learning) 기초

08 역전파와 경사 하강법

Deep & High Learning

Backpropagation (역전파)



Backpropagation (역전파)

출력과 Loss는 Feedfoward들의 결과로 나온 값

작은 Loss는 적절한 w들 덕분, 높은 Loss는 적절하지 않은 w들 때문

Loss를 바탕으로 각각의 $\boldsymbol{w}$ 하나 하나가 얼마나 영향을 주었는지 알아내고

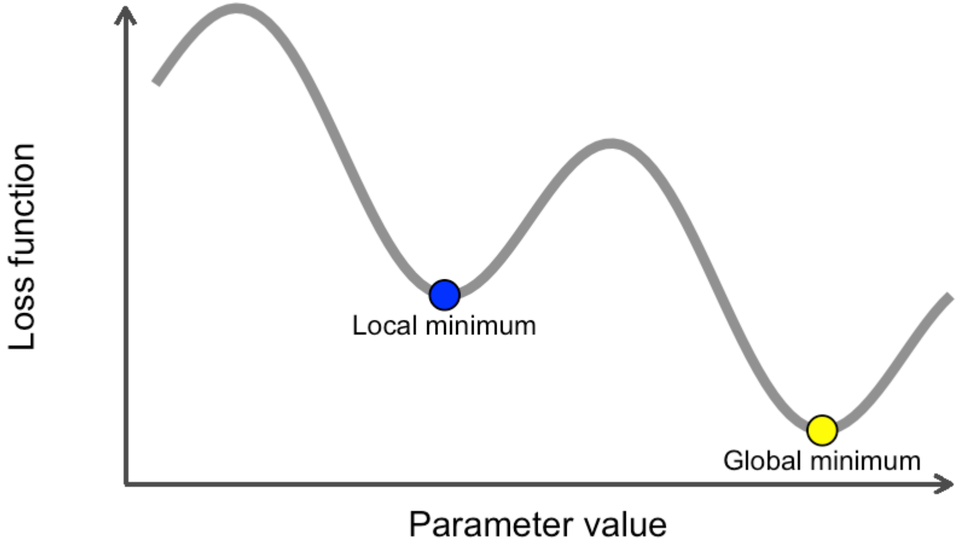
좀 더 적절한 방향으로 $\boldsymbol{w}$들을 변경해 나가는 것을 Backpropagation (역전파)이라 함

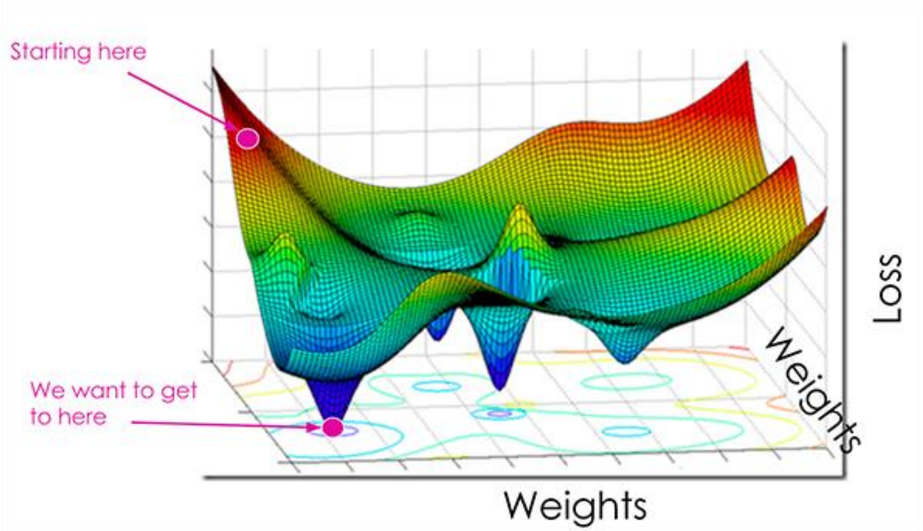
목표 : 적절한 가중치(w_i)를 찾아내기!

Backpropagation (역전파)

w와 loss의 관계를 이용해 좋은 w를 찾아가는 방법이 존재

그래프의 기울기를 이용한 Gradient Decent (경사 하강법)★★★★★

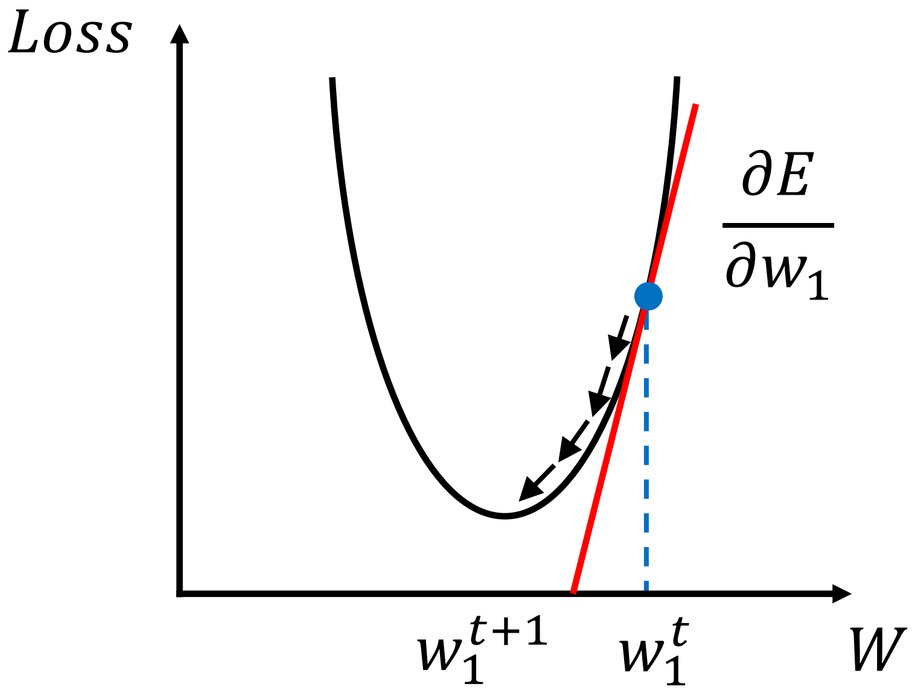




https://bradleyboehmke.github.io/HOML/deep-learning.html
https://www.pyimagesearch.com/2019/10/14/why-is-my-validation-loss-lower-than-my-training-loss/

Gradient Decent (경사 하강법)

임의의 한 지점으로부터 시작해, loss가 줄어드는 방향으로
w (weights=parameters)들을 갱신하는 방법



Gradient Decent (경사 하강법)

목적지 : 산 아래 지점

목적지 방향은 어떻게 알 수 있을까?

→ 발에 집중해 아래쪽으로 보며 방향을 찾음

→ 그 방향으로 얼만큼 움직여야 하는가?

→ 한 발짝

→ [기울어진 방향]으로 [한 발짝] 이동

Trask, Andrew W. Grokking deep learning. Simon and Schuster, 2019.

Gradient Decent (경사 하강법)

목적지 : 산 아래 지점

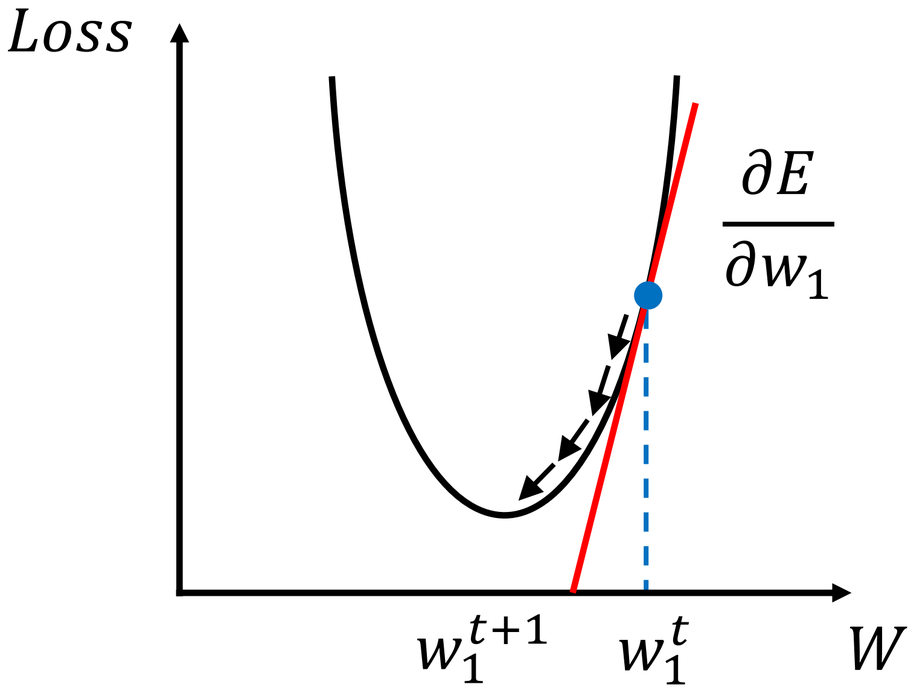
목적지 방향은 어떻게 알 수 있을까?

→ 발에 집중해 아래쪽으로 보며 방향을 찾음

→ 그 방향으로 얼만큼 움직여야 하는가?

→ 한 발짝

→ [기울어진 방향]으로 [한 발짝] 이동



Gradient Decent (경사 하강법)

$$w_1^{t+1} = w_1^t - \varepsilon \frac{\partial E}{\partial w_1}$$

목적지 : $Loss$가 최소값을 갖는 지점

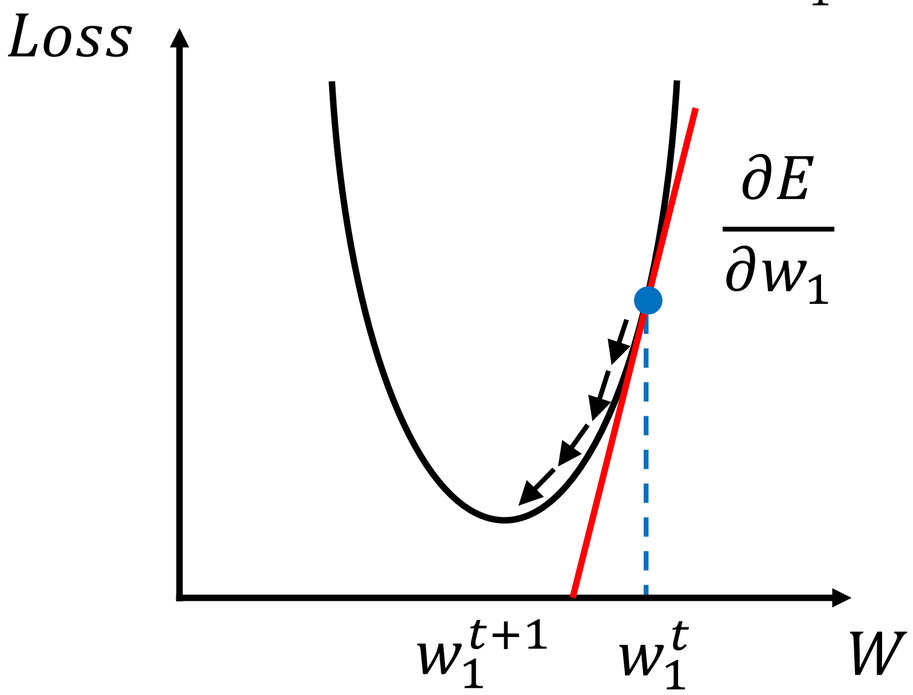
목적지 방향은 어떻게 알 수 있을까?

→ 미분을 이용 ($\frac{\partial E}{\partial w_i}$)

→ 그 방향으로 얼만큼 움직여야 하는가?

→ Learning Rate (ε)

→ $\frac{\partial E}{\partial w_i} \times \varepsilon$ 만큼 w_i들을 수정!



Gradient Decent (경사 하강법)

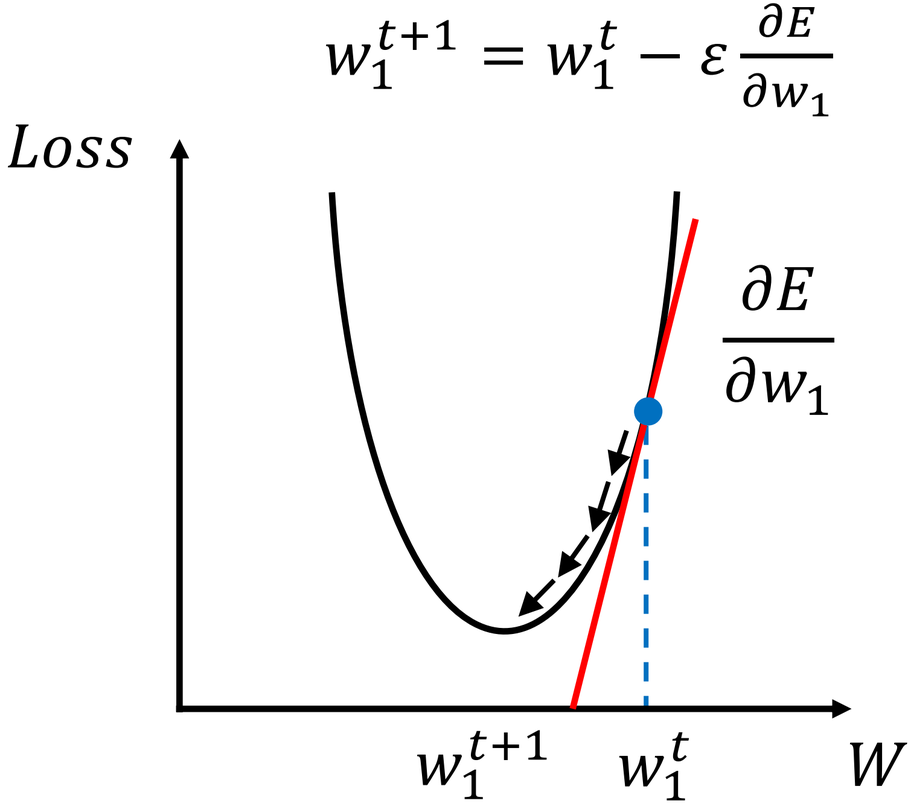
임의의 한 지점으로부터 시작해,

$Loss$가 줄어드는 방향으로 w들을 갱신하는 방법

$Loss$가 더이상 감소하지 않는 시점 (optimum)까지 ***w*를 음의 기울기 방향(-)** 으로 조금씩 움직이는 것 (step)을 여러 번 반복

$$w_1^{t+1} = w_1^t - \varepsilon \frac{\partial E}{\partial w_1}$$

(E: 모든 학습 샘플에 대해서 계산되는 오차)

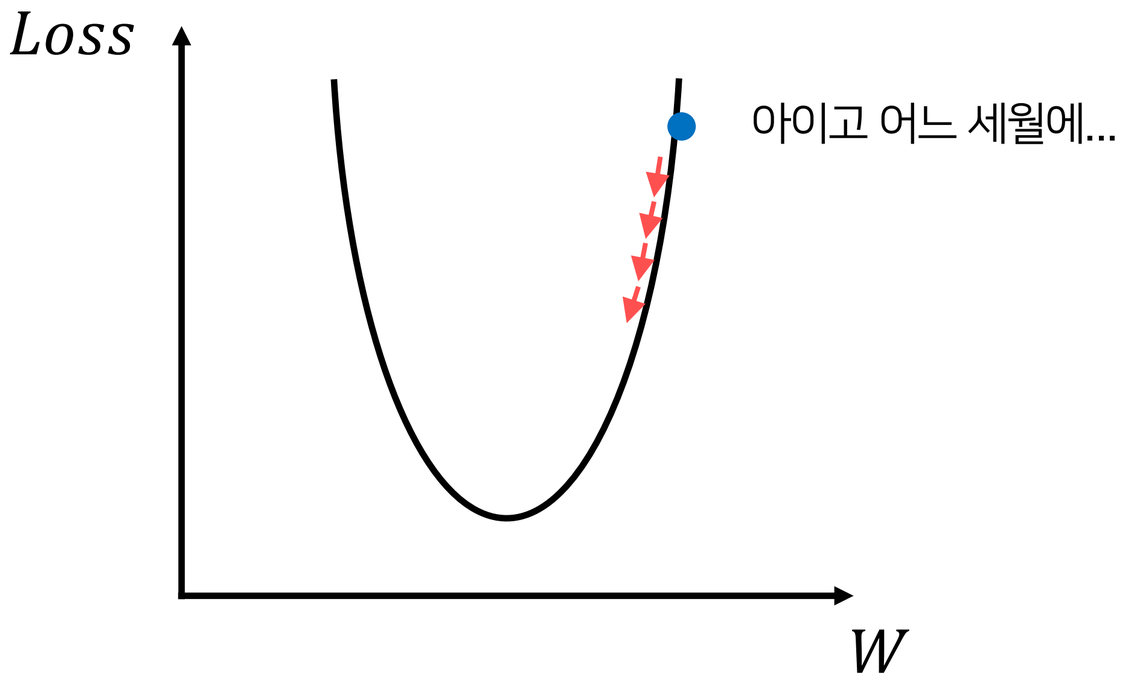


Gradient Decent (경사 하강법)

step이 너무 작다면?

optimum에 도달하는 데에

너무 오래 걸림

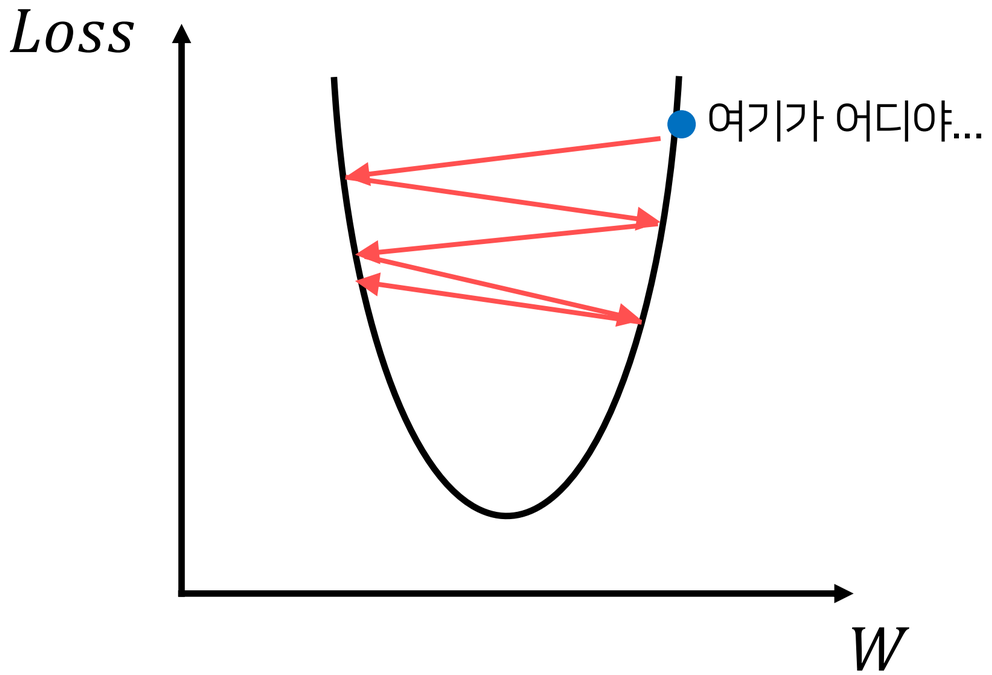


Gradient Decent (경사 하강법)

step이 너무 크다면?

반대편으로 건너뛰어

optimum을 찾기 힘듬



Chain-Rule 을 이용한 역전파

Forward 과정을 통해 나온 $Loss$에 각 w들이 끼친 영향을 알기 위해서는

전체 $Loss$를 각각의 w로 편미분해야 함

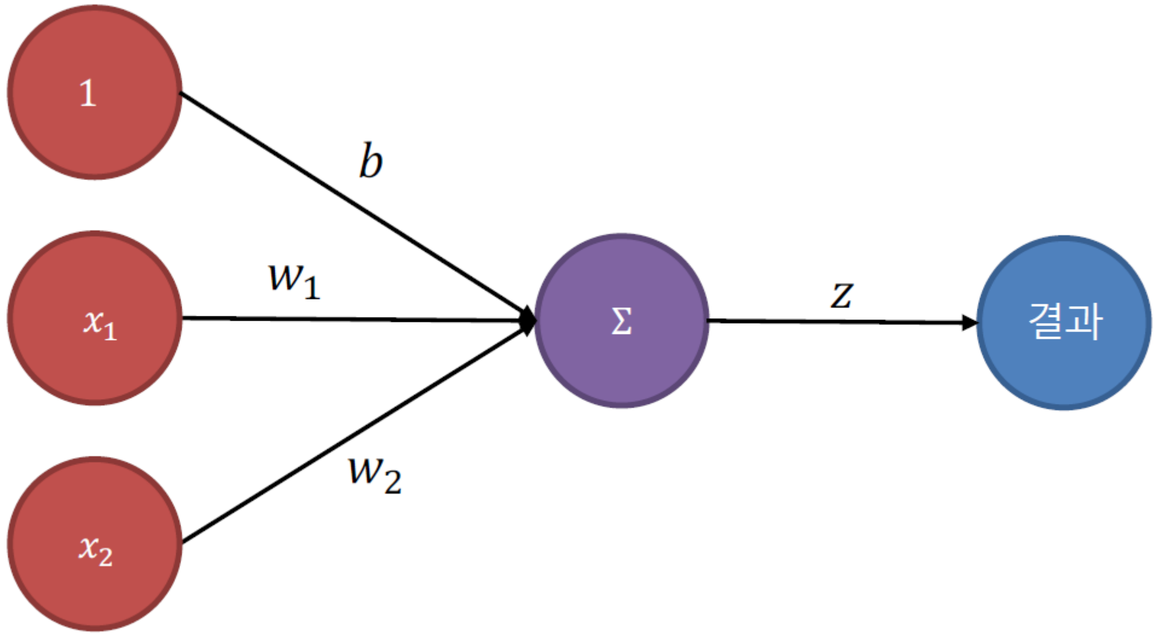
$$\frac{\partial Loss}{\partial w_i} = \frac{\partial E}{\partial w_i}$$

하지만 $Loss$를 직접적으로 w에 대해 미분하기란 쉽지 않음

Chain-Rule을 통한 해결!

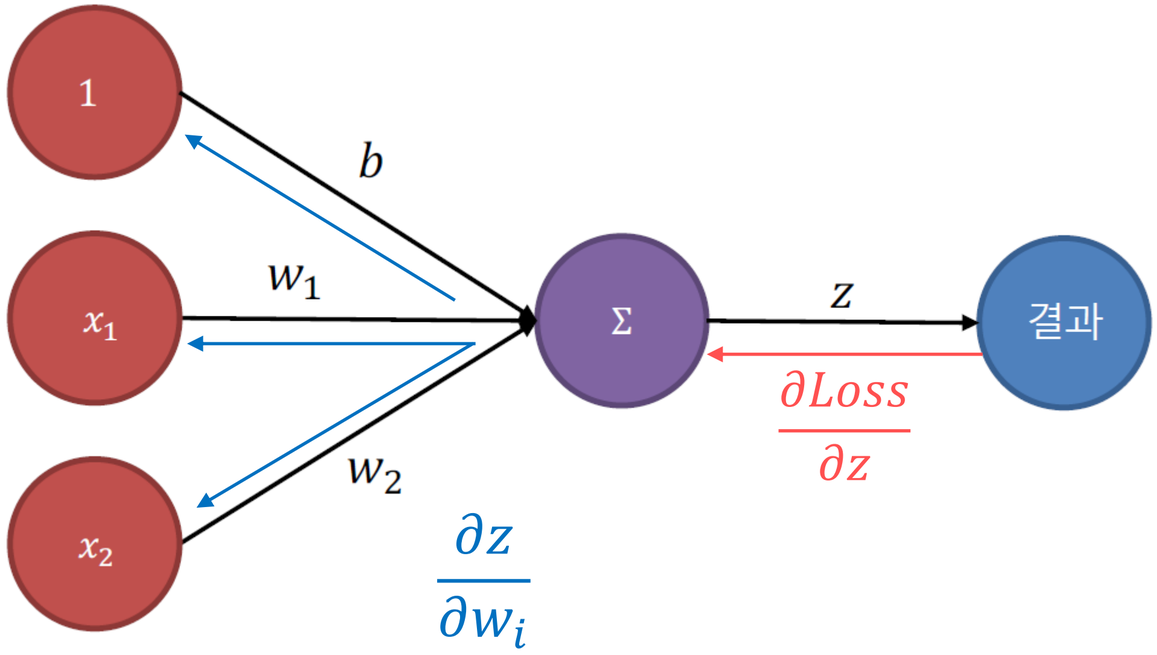
Chain-Rule 을 이용한 역전파

$$\frac{\partial Loss}{\partial w_i} = \frac{\partial Loss}{\partial z}\frac{\partial z}{\partial w_i}$$



Chain-Rule 을 이용한 역전파

$$\frac{\partial Loss}{\partial w_i} = \frac{\partial Loss}{\partial z}\frac{\partial z}{\partial w_i}$$



Chain-Rule 을 이용한 역전파

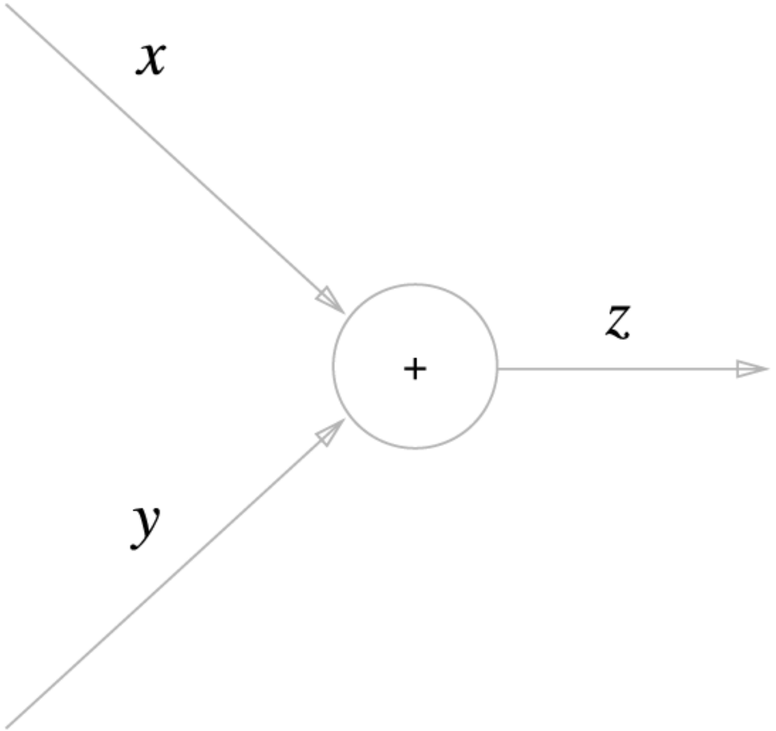
딥러닝 모델(네트워크)은 매우 많은 연산을 통해 결과를 도출함

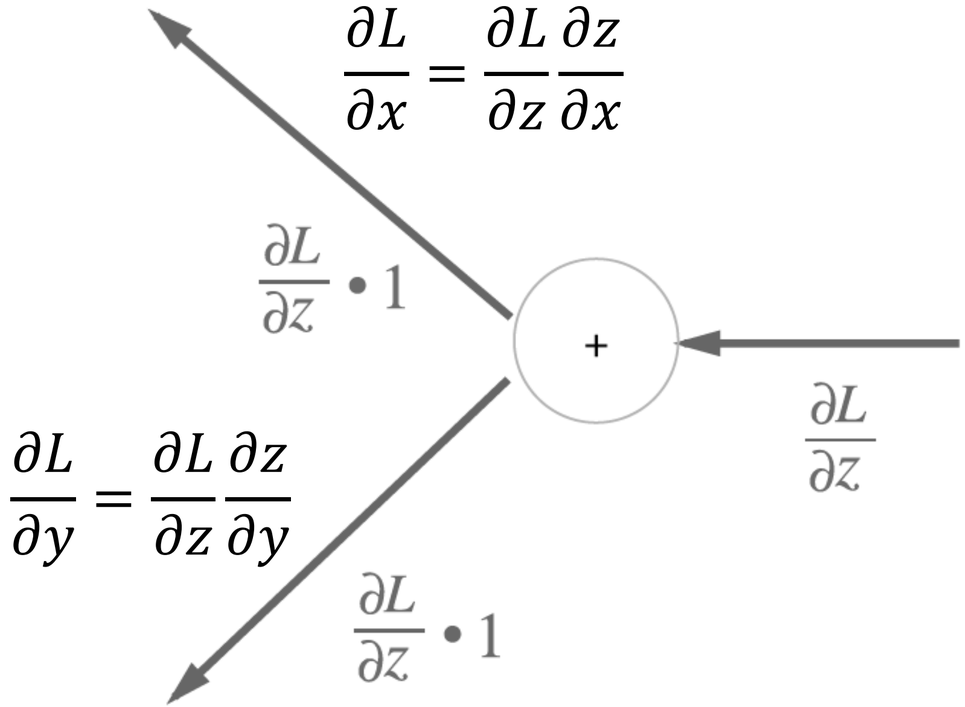
하지만 그 연산들은 잘게 쪼개면 더하기 및 곱하기로 이루어짐

Chain Rule은 복잡한 연산을 기본 연산으로 나누어 미분

결국 2가지(+,×) operation의 미분 과정만 알면 목적인 $\frac{\partial Loss}{\partial w_i}$ 를 알 수 있음

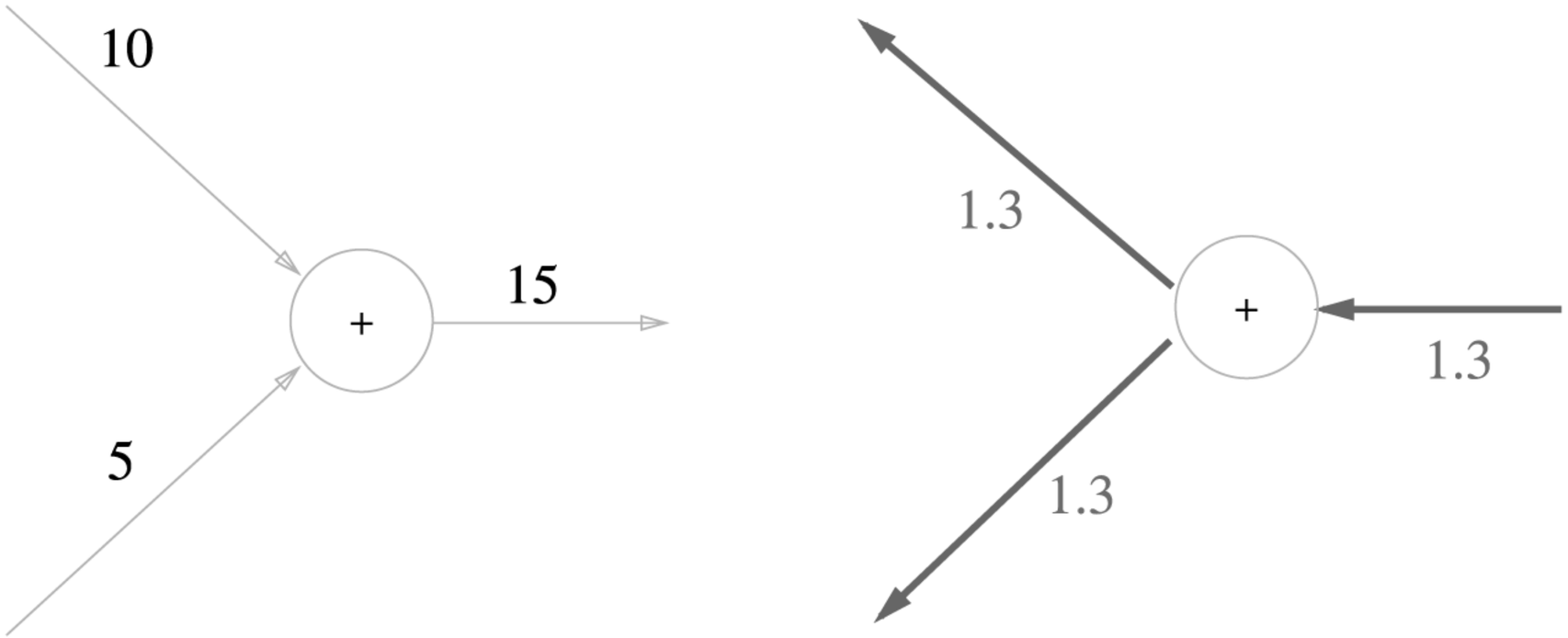
Chain-Rule 을 이용한 역전파



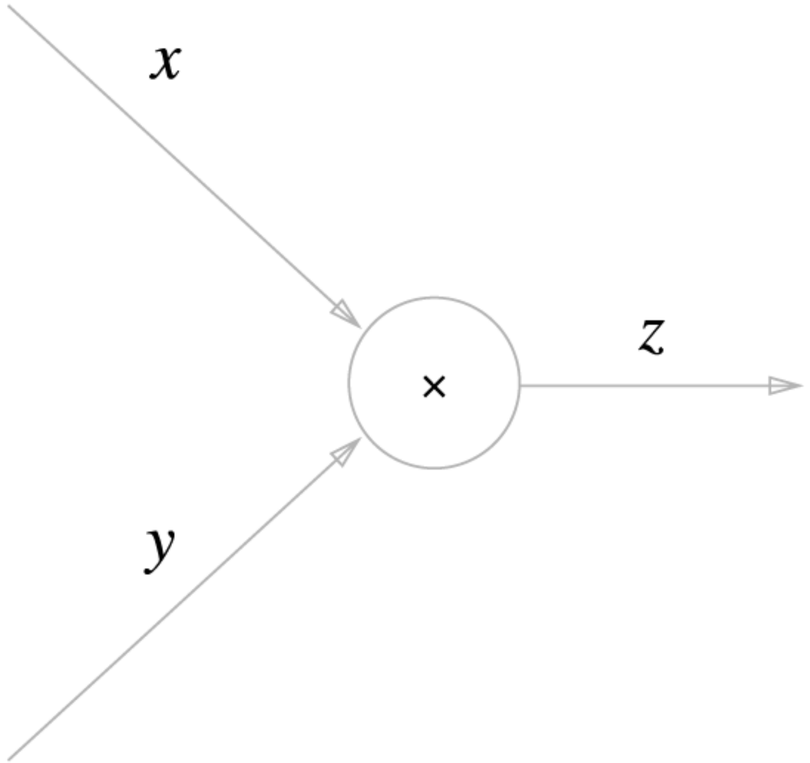


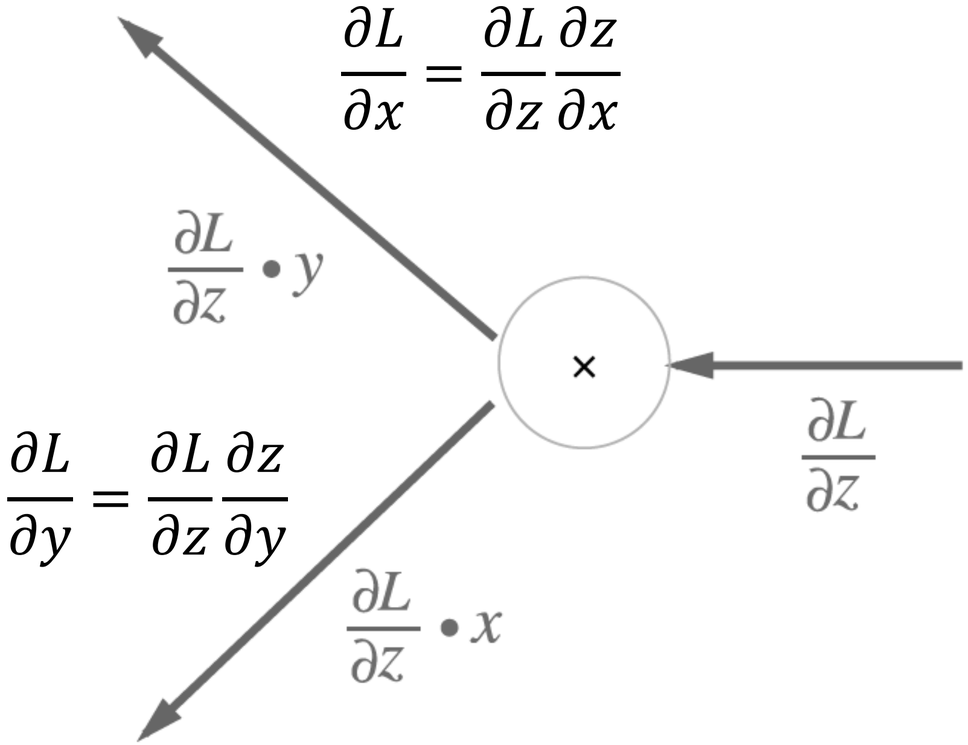
$$z = x + y \qquad \frac{\partial z}{\partial x} = 1 \qquad \frac{\partial z}{\partial y} = 1$$

Chain-Rule 을 이용한 역전파



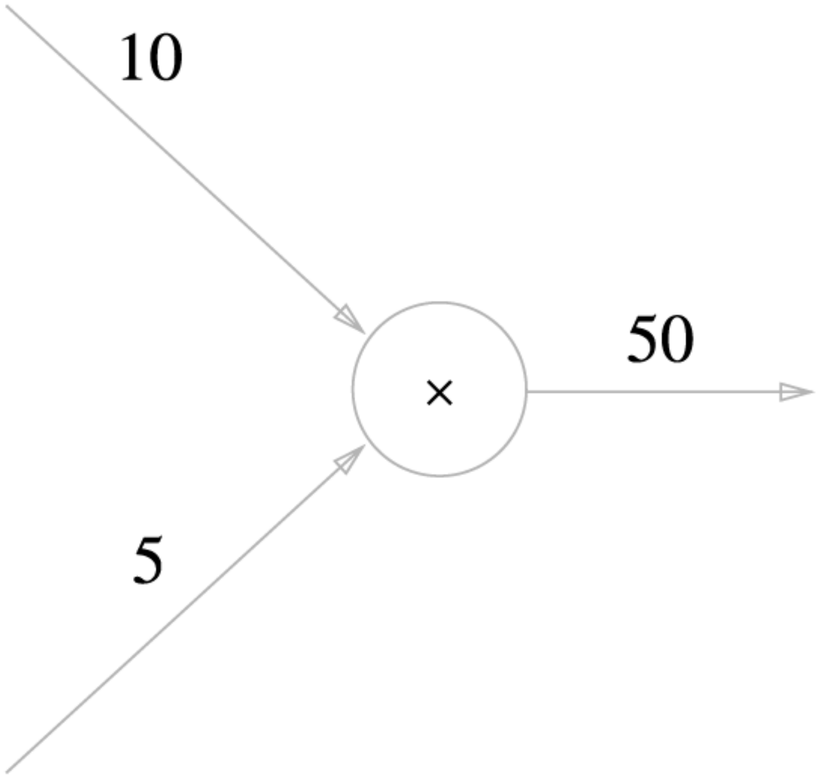
Chain-Rule 을 이용한 역전파

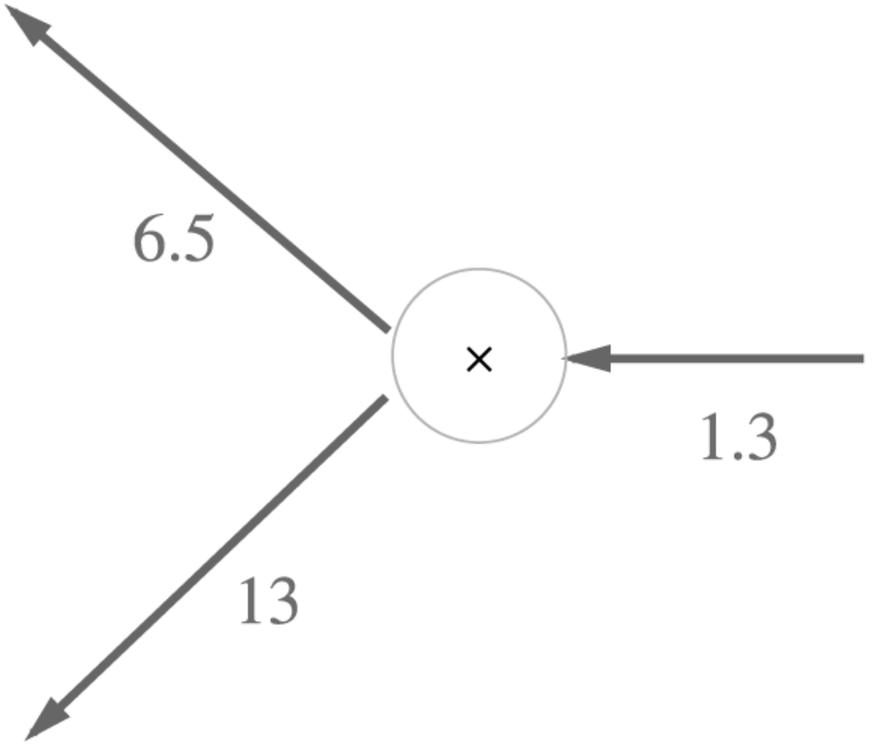




$$z = xy \qquad \frac{\partial z}{\partial x} = y \qquad \frac{\partial z}{\partial y} = x$$

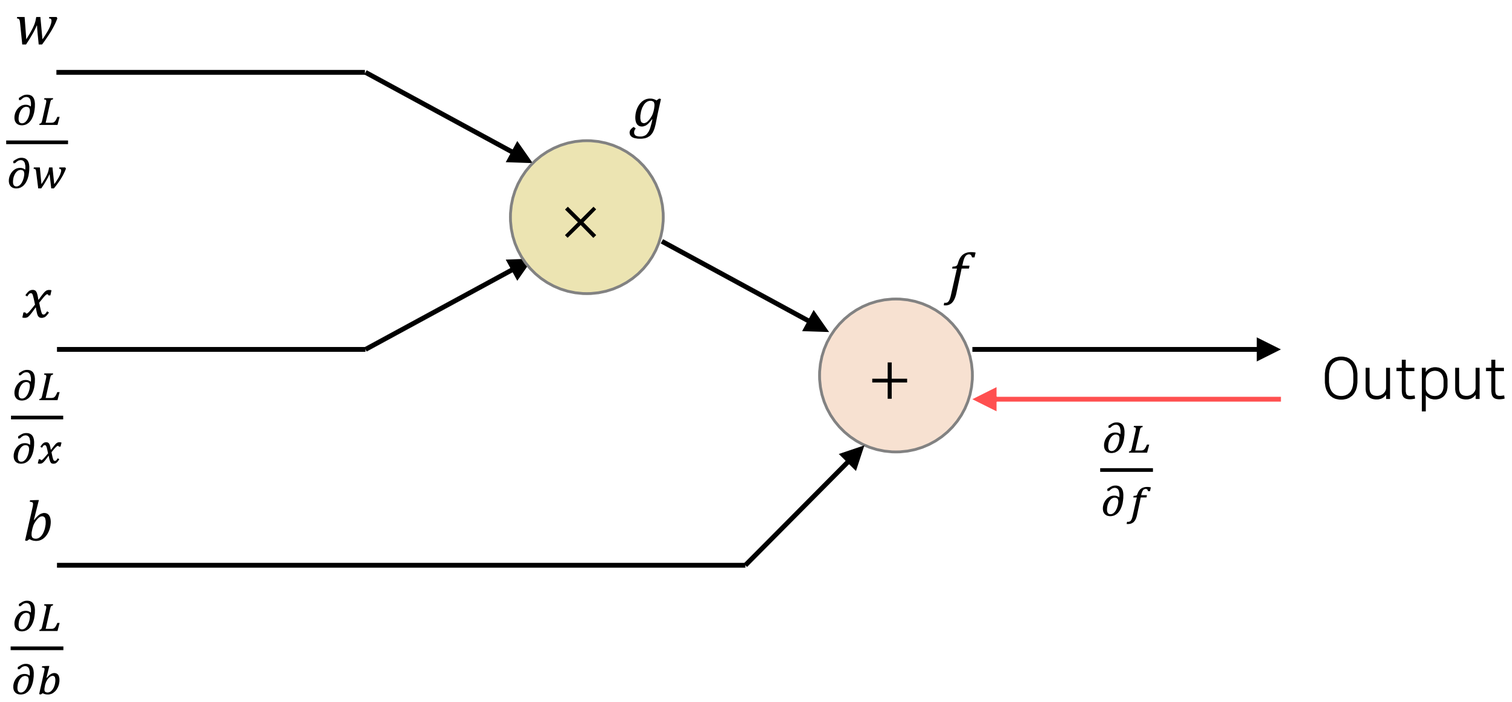
Chain-Rule 을 이용한 역전파



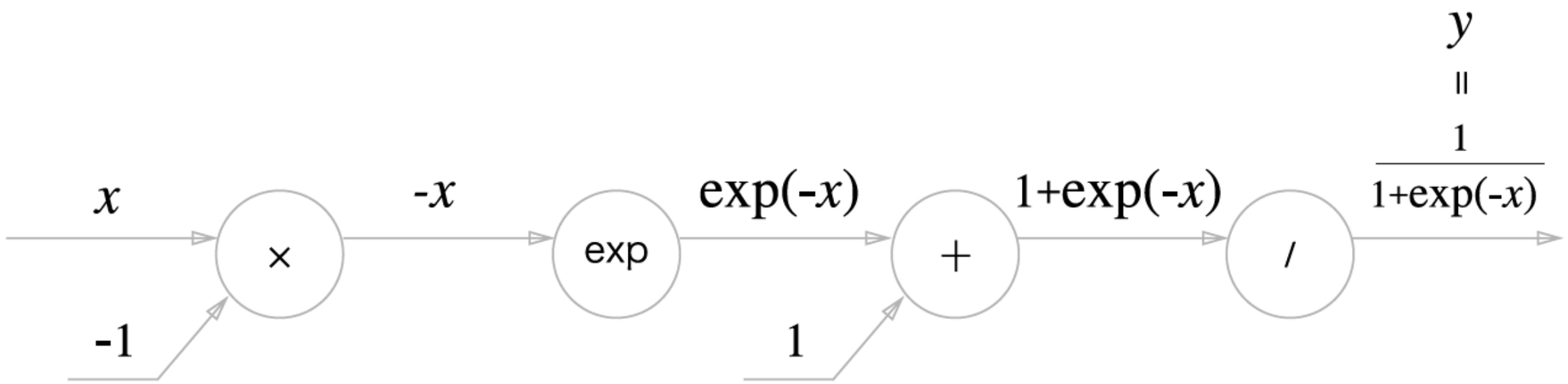


Chain-Rule 을 이용한 역전파

$$f = g + b, g = wx$$



Sigmoid 함수의 계산 그래프



$$f(x) = e^x \rightarrow \frac{df}{dx} = e^x$$

$$f_a(x) = ax \rightarrow \frac{df}{dx} = a$$

$$f(x) = \frac{1}{x} \rightarrow \frac{df}{dx} = -1/x^2$$

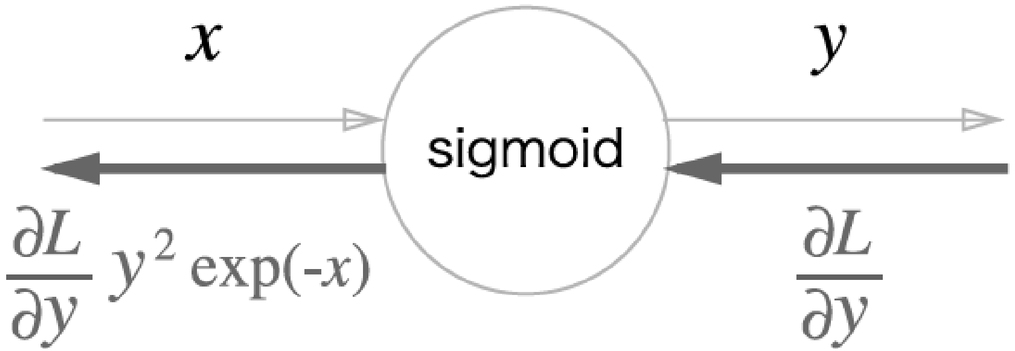
$$f_c(x) = c + x \rightarrow \frac{df}{dx} = 1$$

Sigmoid 함수의 계산 그래프

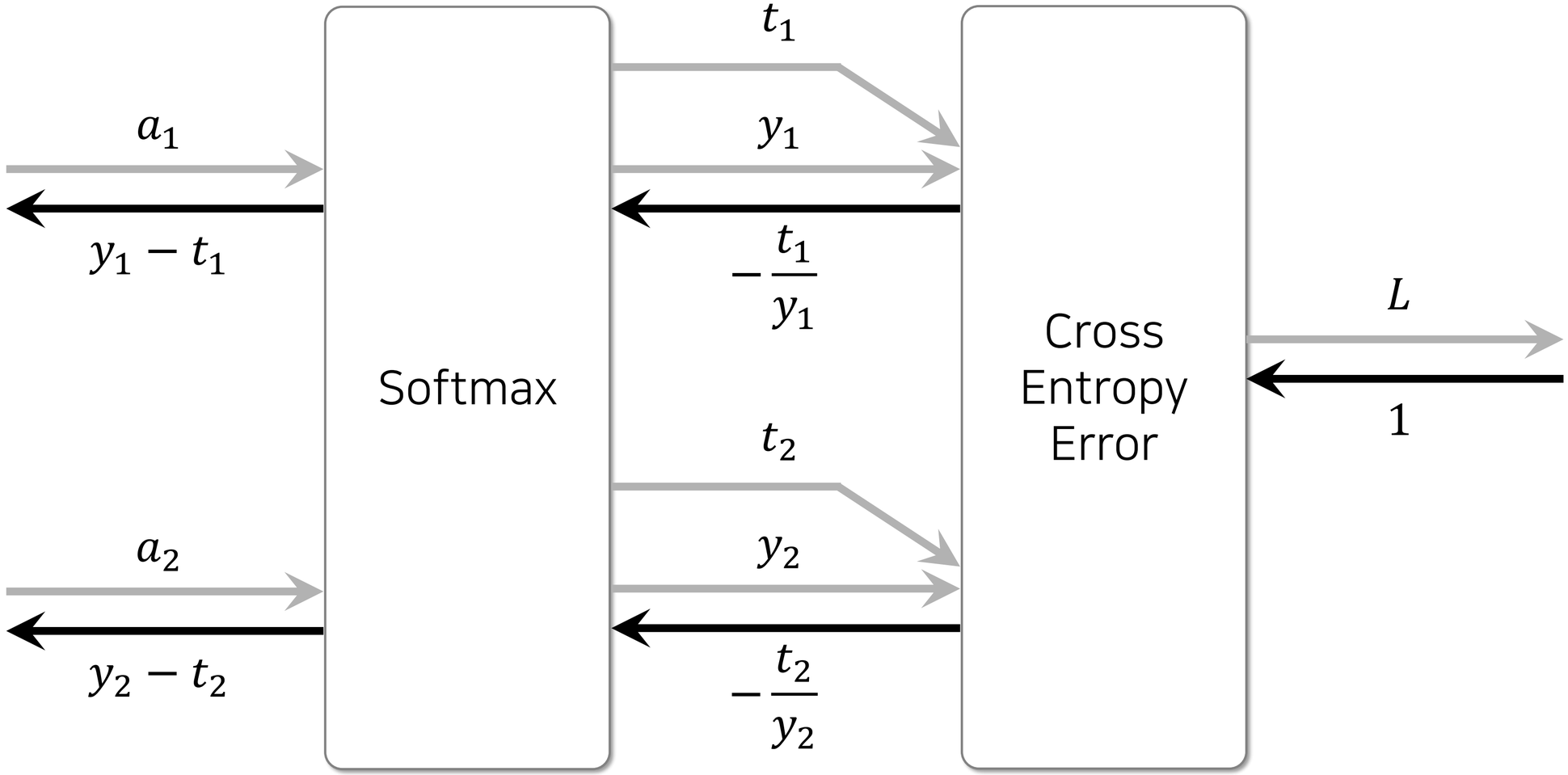
$$\frac{\partial L}{\partial y} y^2 \exp(-x) = \frac{\partial L}{\partial y} \frac{1}{(1+\exp(-x))^2} \exp(-x)$$

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

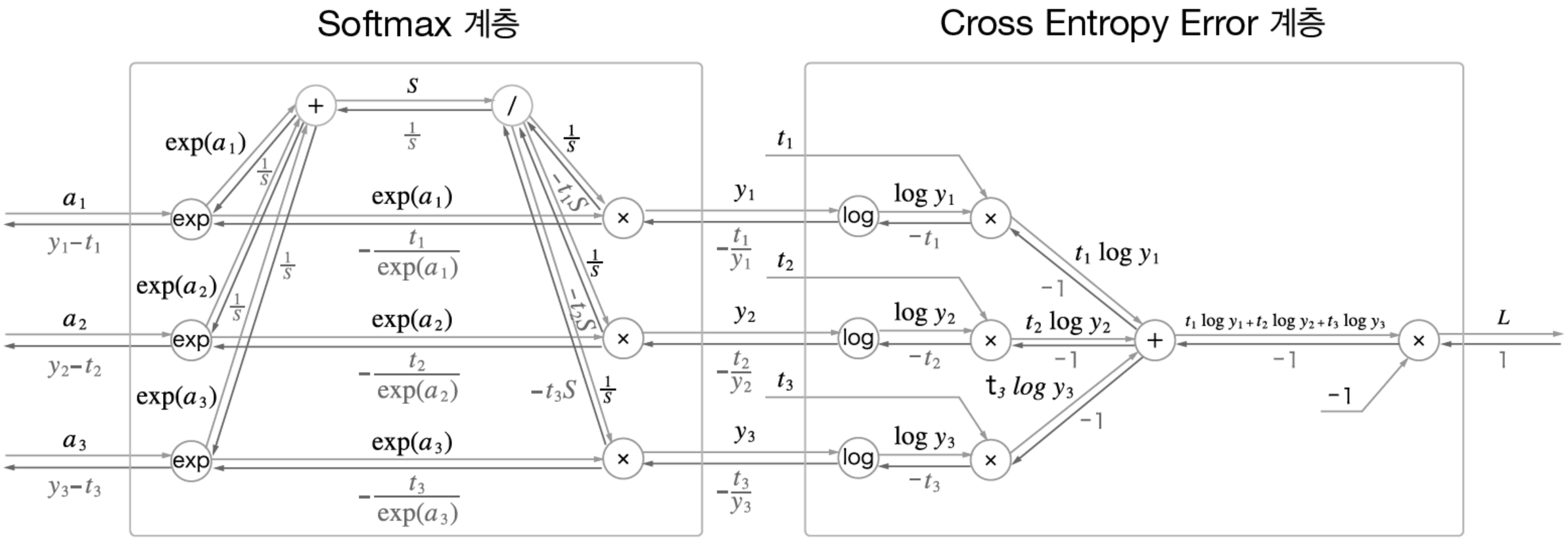
$$= \frac{\partial L}{\partial y} y(1-y)$$



Softmax + Cross Entropy loss계산 그래프



Softmax + Cross Entropy loss계산 그래프



역전파를 이용한 Loss 감소 예시

```
2020-09-25-08-58: epoch: 1 |loss 10.925624 |
2020-09-25-09-07: epoch: 2 |loss 4.139618 |
2020-09-25-09-15: epoch: 3 |loss 3.199270 |
2020-09-25-09-24: epoch: 4 |loss 2.842443 |
2020-09-25-09-32: epoch: 5 |loss 2.605526 |
2020-09-25-09-46: epoch: 6 |loss 2.507510 |
2020-09-25-09-55: epoch: 7 |loss 2.367399 |
2020-09-25-10-03: epoch: 8 |loss 2.319310 |
2020-09-25-10-12: epoch: 9 |loss 2.210096 |
2020-09-25-10-20: epoch: 10 |loss 2.159435 |
2020-09-25-10-34: epoch: 11 |loss 2.153410 |
2020-09-25-10-43: epoch: 12 |loss 2.059602 |
2020-09-25-10-51: epoch: 13 |loss 2.048363 |
2020-09-25-11-00: epoch: 14 |loss 1.964805 |
2020-09-25-11-08: epoch: 15 |loss 1.877977 |
2020-09-25-11-22: epoch: 16 |loss 1.833441 |
2020-09-25-11-31: epoch: 17 |loss 1.580934 |
2020-09-25-11-39: epoch: 18 |loss 1.458391 |
2020-09-25-11-48: epoch: 19 |loss 1.372296 |
2020-09-25-11-57: epoch: 20 |loss 1.357357 |
2020-09-25-12-10: epoch: 21 |loss 1.306156 |
2020-09-25-12-19: epoch: 22 |loss 1.245406 |
2020-09-25-12-28: epoch: 23 |loss 1.174226 |
2020-09-25-12-36: epoch: 24 |loss 1.218089 |
2020-09-25-12-45: epoch: 25 |loss 1.166818 |
2020-09-25-12-59: epoch: 26 |loss 1.144130 |
2020-09-25-13-07: epoch: 27 |loss 1.120341 |
2020-09-25-13-16: epoch: 28 |loss 1.095250 |
```

```
2020-09-26-05-11: epoch: 127 |loss 0.471236 |
2020-09-26-05-20: epoch: 128 |loss 0.474254 |
2020-09-26-05-29: epoch: 129 |loss 0.468066 |
2020-09-26-05-37: epoch: 130 |loss 0.467732 |
2020-09-26-05-51: epoch: 131 |loss 0.477968 |
2020-09-26-06-00: epoch: 132 |loss 0.466950 |
2020-09-26-06-08: epoch: 133 |loss 0.473362 |
2020-09-26-06-17: epoch: 134 |loss 0.464963 |
2020-09-26-06-26: epoch: 135 |loss 0.467376 |
2020-09-26-06-40: epoch: 136 |loss 0.473385 |
2020-09-26-06-48: epoch: 137 |loss 0.462927 |
2020-09-26-06-57: epoch: 138 |loss 0.466065 |
2020-09-26-07-05: epoch: 139 |loss 0.455414 |
2020-09-26-07-14: epoch: 140 |loss 0.461836 |
```

See you in the next lecture!

딥러닝 (Deep Learning) 기초

09 미니배치와 최적화 알고리즘들

Deep & High Learning

Optimization (최적화)

역전파를 통해 weights(= parameters)을 업데이트 시키는 방법

학습의 안정성 및 효율성을 위해 여러 Optimization 기법들이 제안됨

- Stochastic Gradient Decent (SGD)
- AdaGrad
- RMSProp
- Momentum
- Adam

Batch Learning (Epoch Learning)

전체 훈련 데이터를 사용

대규모 데이터셋에 적용하기 힘듦

데이터 구성이 항상 같기 때문에 local minima에 빠질 가능성이 있음

$$w_1^{t+1} = w_1^t - \varepsilon \frac{\partial E}{\partial w_1}$$

(E: 모든 학습 샘플에 대해서 계산되는 오차 함수의 기울기)

Stochastic Gradient Descent (확률적 경사 하강법, SGD)

Batch learning과는 달리 샘플의 일부만을 사용하여 파라미터를 업데이트하는 방법

빠른 학습 가능, local minima에 빠질 위험이 적음

노이즈데이터로 인한 변동이 큼

Epoch마다 무작위로 샘플을 선택하면 그 효과를 극대화할 수 있고, 이 때문에 Stochastic Gradient decent라 부름

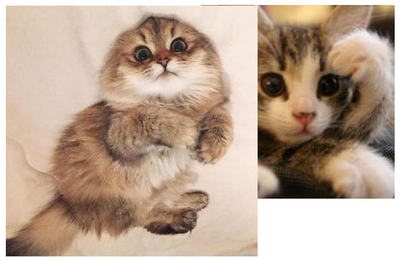
Mini-batch (미니배치)

몇 개의 샘플을 하나의 소규모 집합 단위로 하여 가중치를 업데이트

복수의 샘플을 묶은 작은 집합을 미니배치 (Mini-batch)라고 함

Mini-batch#1

Mini-batch#2



Mini-batch#3

Mini-batch#4

Mini-batch (미니배치)

몇 개의 샘플을 하나의 소규모 집합 단위로 하여 가중치를 업데이트

복수의 샘플을 묶은 작은 집합을 미니배치 (Mini-batch)라고 함

$$E_t = \frac{1}{N_t} \sum_{n \in D_t} E_n$$

$$w_1^{t+1} = w_1^t - \varepsilon \frac{\partial E_t}{\partial w_1}$$

(일부 학습 샘플에 대해서
계산되는 오차 함수의 기울기)

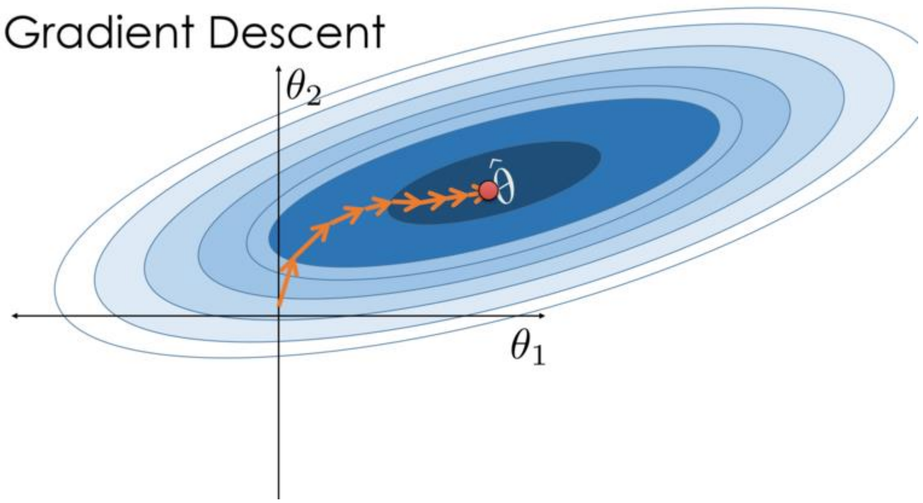
D_t: t번째의 Mini-batch

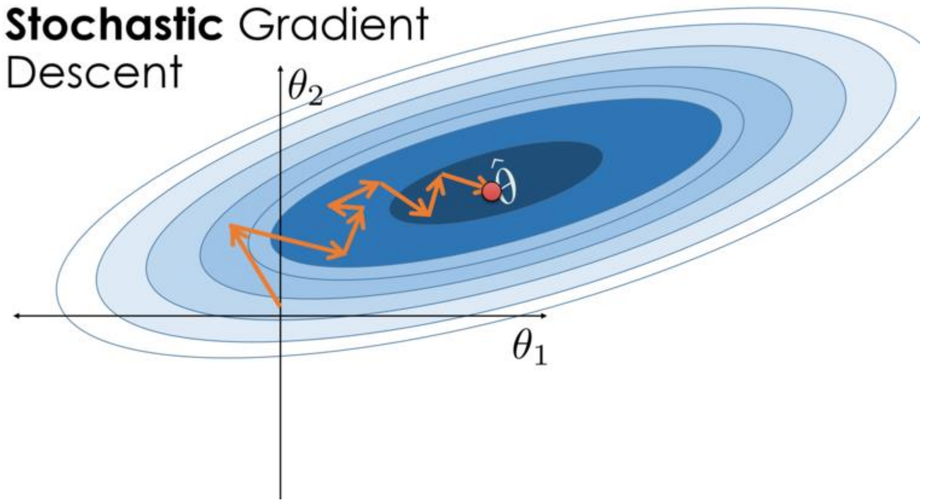
N_t: D_t의 샘플 개수

E_n: 각 샘플에 대한 loss

E_t: t번째 Mini-batch에 대한 loss

Stochastic Gradient Descent (확률적 경사 하강법, SGD)





https://optimization.cbe.cornell.edu/index.php?title=Stochastic_gradient_descent

Adagrad

"가중치별로 학습 속도를 조정해볼 수 없을까?"

많이 갱신된 가중치는 학습률을 낮추는 학습 방법

SGD

$$w_1^{t+1} = w_1^t - \varepsilon \frac{\partial E_t}{\partial w_1}$$

Adagrad

$$h = h + \left(\frac{\partial E}{\partial w_1}\right)^2$$

$$w_1^{t+1} = w_1^t - \varepsilon \frac{1}{\sqrt{h}} \frac{\partial E_t}{\partial w_1}$$

Duchi, John, Elad Hazan, and Yoram Singer. "Adaptive subgradient methods for online learning and stochastic optimization." Journal of machine learning research 12.7 (2011).

RMSProp

"h가 너무 커지면 학습이 잘 안되는 현상을 개선해볼까?"

지수가중이동평균을 사용하여 최신 기울기를 더 크게 반영하는 학습 방법

Adagrad

$$h = h + (\frac{\partial E}{\partial w_1})^2$$

$$w_1^{t+1} = w_1^t - \varepsilon \frac{1}{\sqrt{h}} \frac{\partial E_t}{\partial w_1}$$

RMSProp

$$h = \rho h + (1-\rho)(\frac{\partial E}{\partial w_1})^2$$

$$w_1^{t+1} = w_1^t - \varepsilon \frac{1}{\sqrt{h}} \frac{\partial E_t}{\partial w_1}$$

Momentum

"이전의 학습 정도를 고려해 관성을 추가해 볼까?"

이전 step의 학습 정도를 반영하는 학습 방법

SGD

$$w_1^{t+1} = w_1^t - \varepsilon \frac{\partial E_t}{\partial w_1}$$

Momentum

$$w_1^{t+1} = w_1^t - \varepsilon \frac{\partial E_t}{\partial w_1} + v$$

$$v = v - \alpha \frac{\partial E_t}{\partial w_1}$$

Sutskever, Ilya, et al. "On the importance of initialization and momentum in deep learning." International conference on machine learning. PMLR, 2013.

Adadelta

Adagrad에서 gradient가 너무 작아지는 것을 방지하는 학습 방법

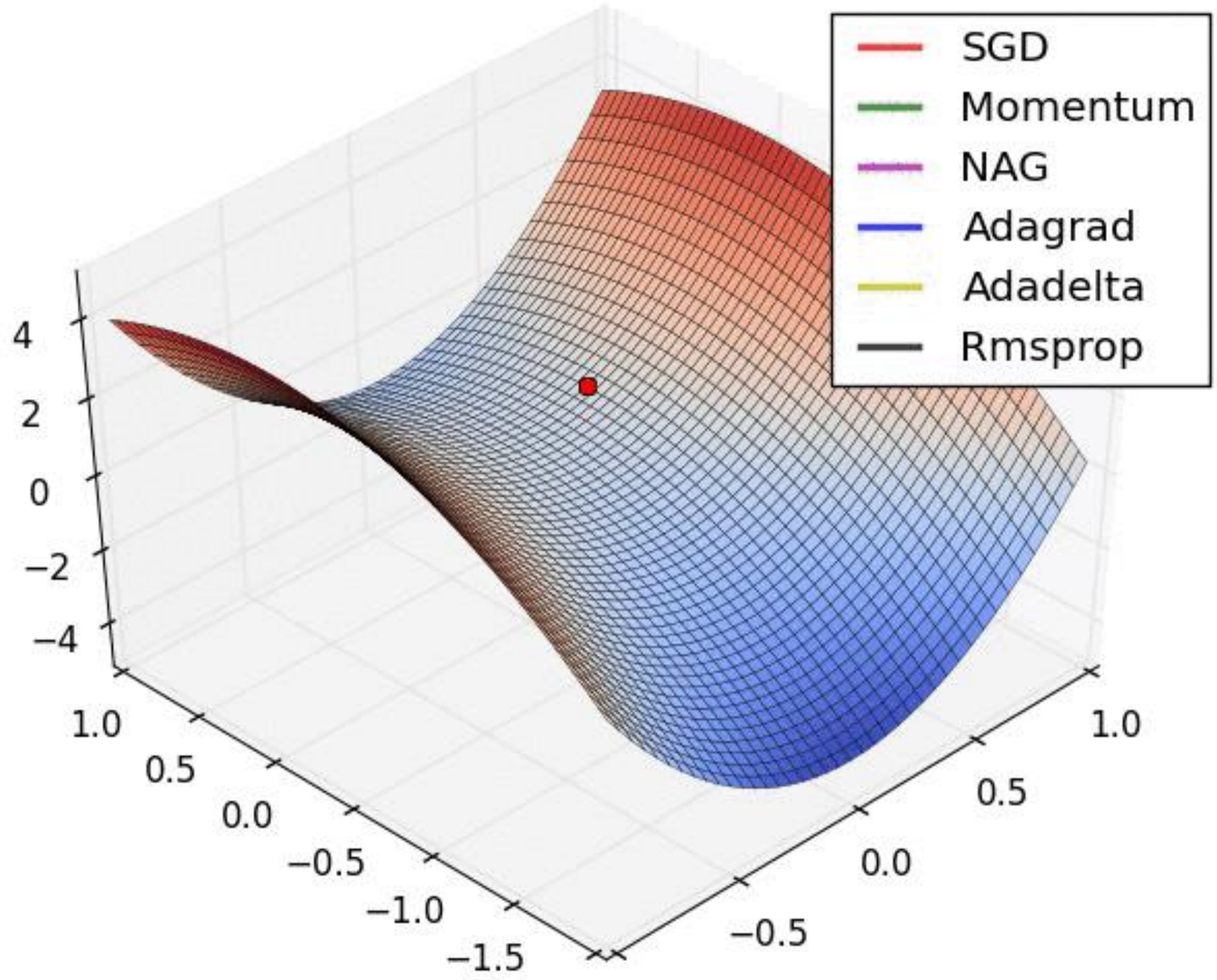
Adam

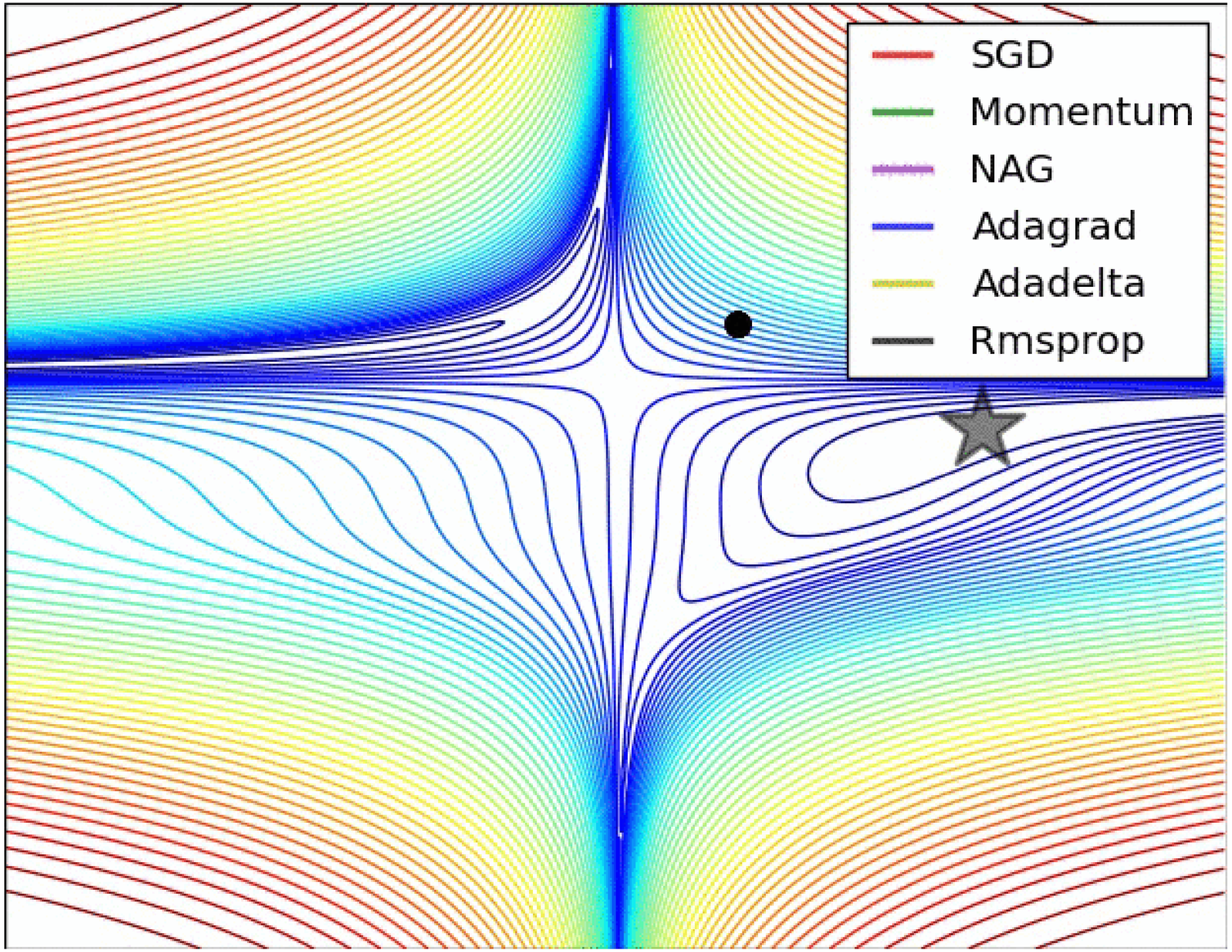
Adagrad, Momentum, RmsProp의 방법론을 종합한 학습 방법

RAdam

Adam이 초반에 local minima에 너무 일찍 도달해 학습이 더 이상 일어나지 않는 한계 개선

- 일반적으로 Adam 가장 많이 사용!


SGD
Momentum
NAG
Adagrad
Adadelta
Rmsprop
4
2
0
−2
−4
1.0
0.5
0.0
−0.5
−1.0
−1.5
1.0
0.5
0.0
−0.5


SGD
Momentum
NAG
Adagrad
Adadelta
Rmsprop

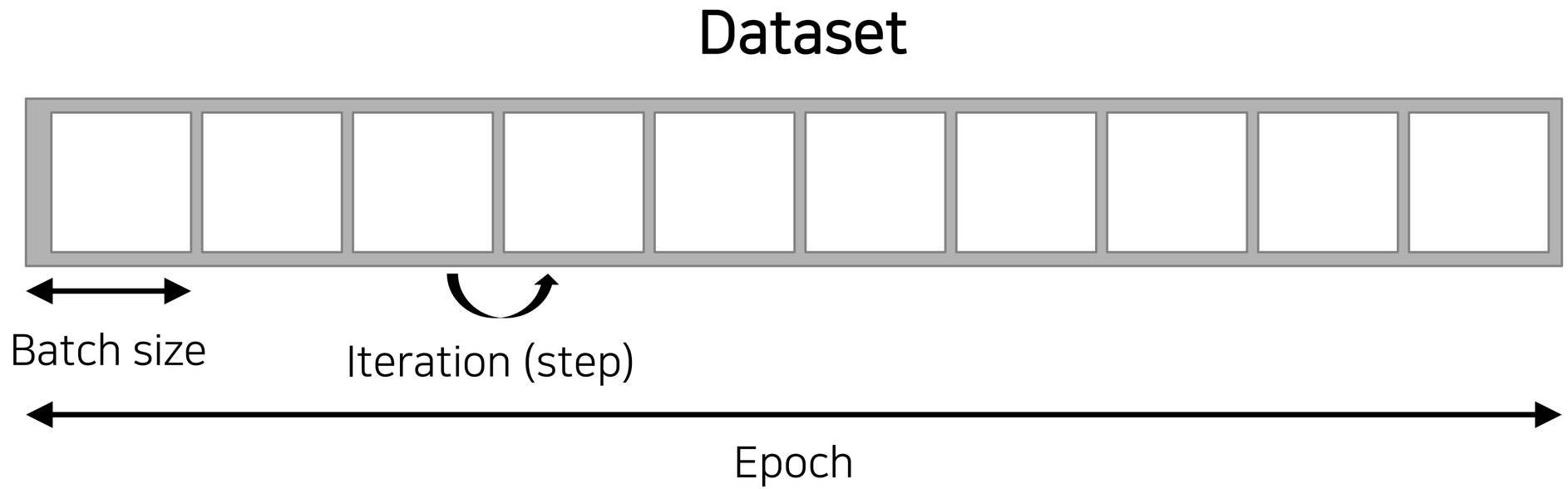
See you in the next lecture!

딥러닝 (Deep Learning) 기초

10 데이터셋 구성하기

Deep & High Learning

Epoch, Batch size, Iteration (step)



Example)

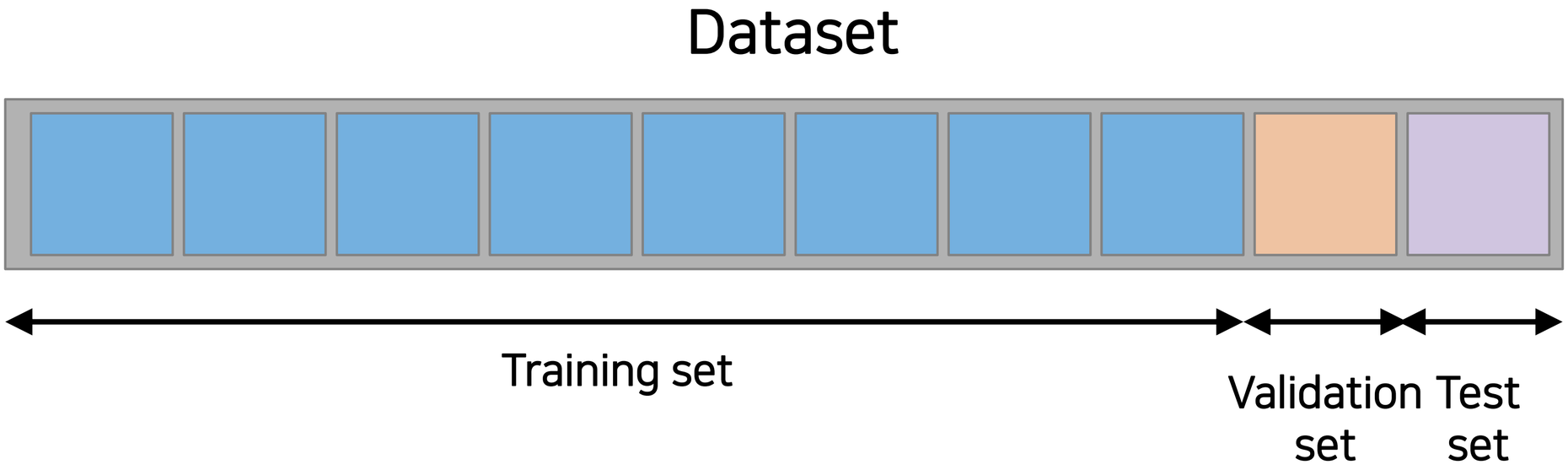
Dataset 전체 샘플 개수 : 500

Batch size : 10

Iteration (step) : 50

1epoch = 50 iterations

학습을 위한 데이터셋 구성



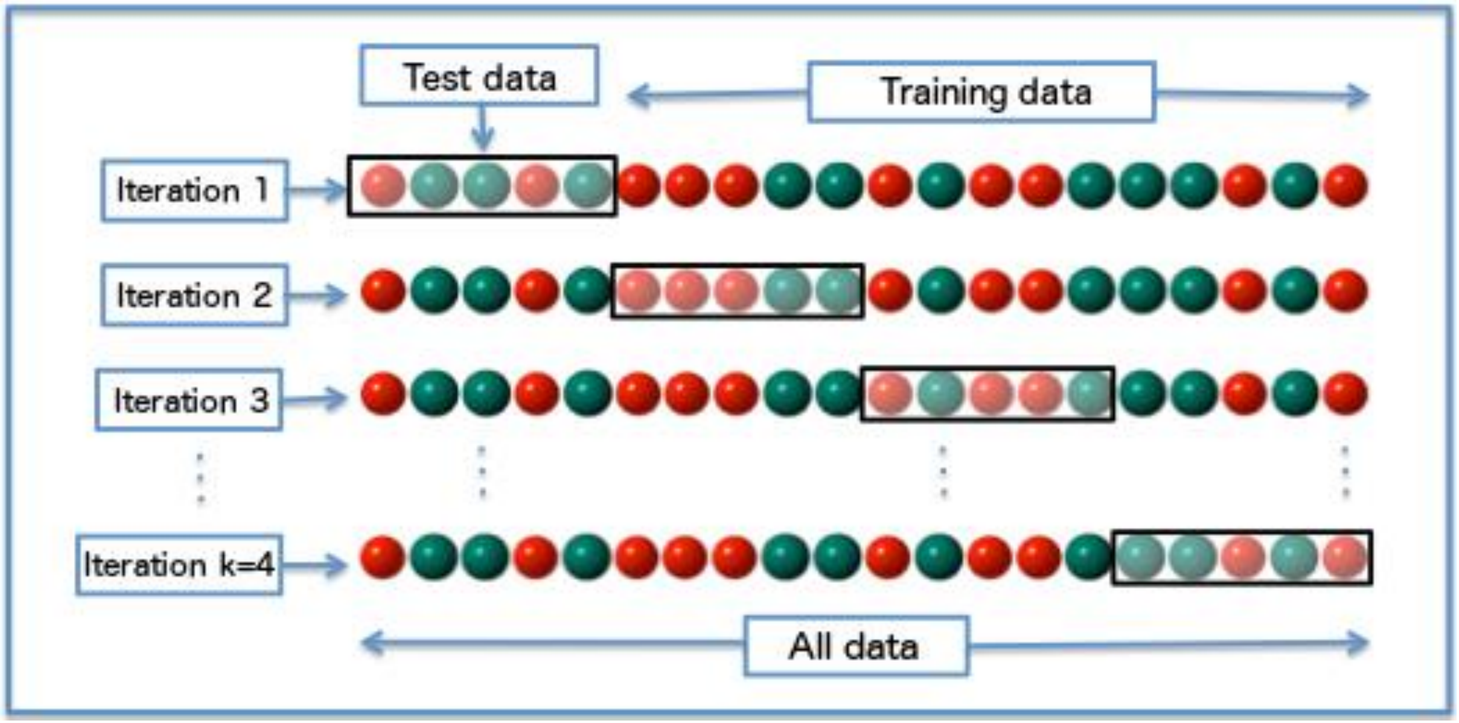
Training set : 실제 학습에 사용되는 데이터 (일반적으로, 전체의 약 80%)

Validation set : 학습 중간에 사용되는 평가 데이터 (일반적으로, 전체의 약 10%)

Test set : 실제 학습된 모델의 성능을 평가하는 데이터 (일반적으로, 전체의 약 10%)

학습 과정에서는 사용되지 않음.

Cross validation



Test data가 따로 없을 경우, 데이터셋을 k개로 쪼갠 다음,

training data의 일부분을 test data로 만들어 모델을 학습 후,

모든 k개의 partition들에 대해 반복한다.

See you in the next lecture!

딥러닝 (Deep Learning) 기초

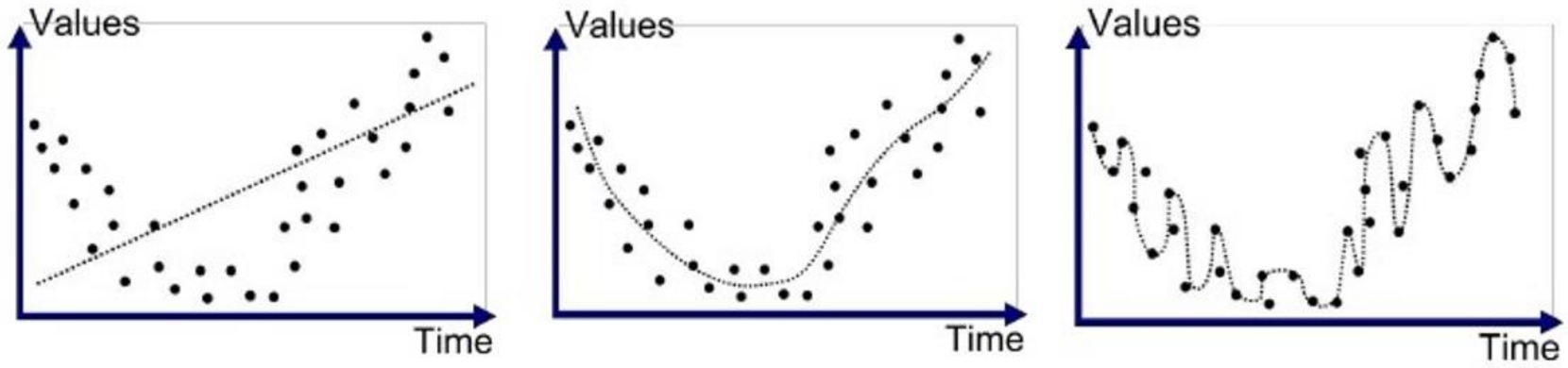
11 오버피팅 방지 기법들

Deep & High Learning

Overfitting (오버피팅)

모델이 training data에 지나치게 집중하면서 실제 test 데이터에서 결과가 나빠지는 현상

학습 데이터로 학습하고 테스트 데이터에서 성능이 좋은 것을 일반화 (Generalization)가 되었다고 함



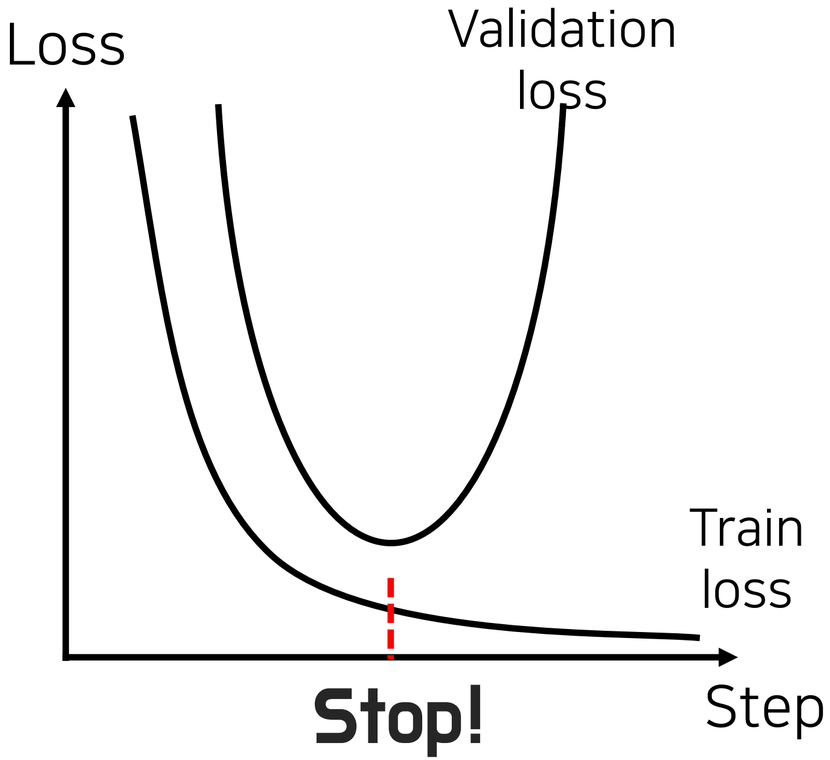
Underfitted　　Good Fit/Robust　　Overfitted

Overfitting 방지 방법

1) Early Stopping
2) Dropout
3) Weight decay
4) Weight restriction
5) Data augmentation
6) 기타 다른 방법들

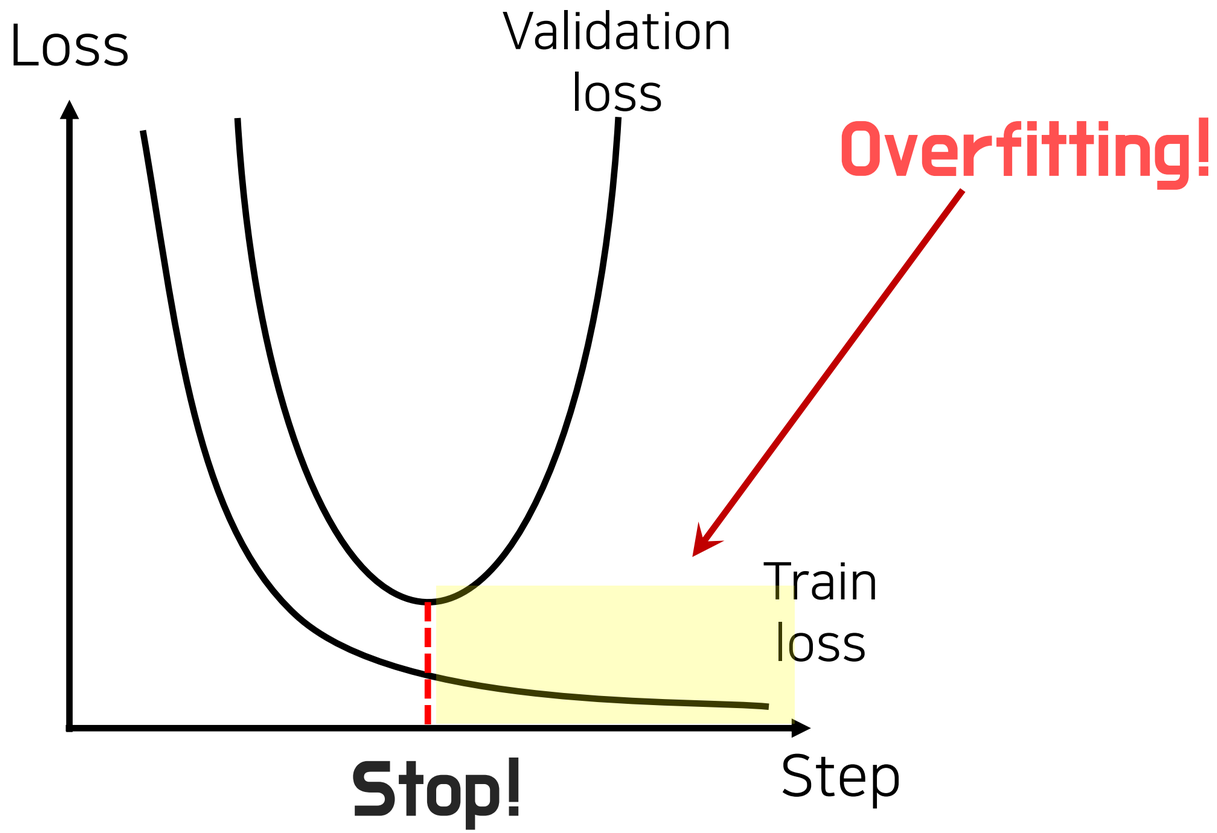
Early Stopping

Train loss & Eval loss graph 상에서 최적 포인트를 자동으로 찾아 지정한 epoch까지 모두 학습하는 것이 아니라 최적 포인트에서 학습 과정을 미리 (Early) 멈추는 (Stopping) 기법



* Loss가 아닌, Accuracy 등 모델의 성능을 측정할 수 있는 지표에 대해서도 적용 가능

Early Stopping



Dropout

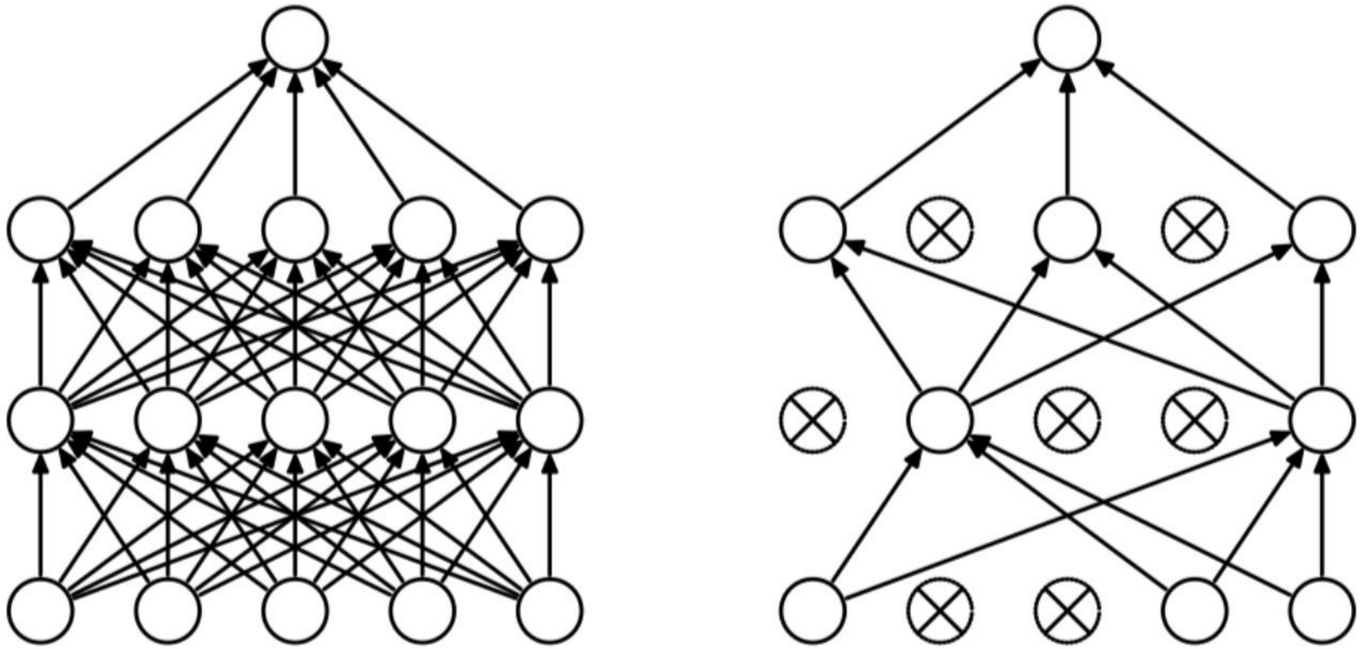
네트워크의 일부를 생략하는 것.

학습 중 Forward 과정에서 일부 perceptron을 특정 확률(p)로 사용/미사용 함

(p값은 hyperparameter)

평가 과정에서는 Dropout을 사용하지 않고 모든 perceptron을 사용

Dropout



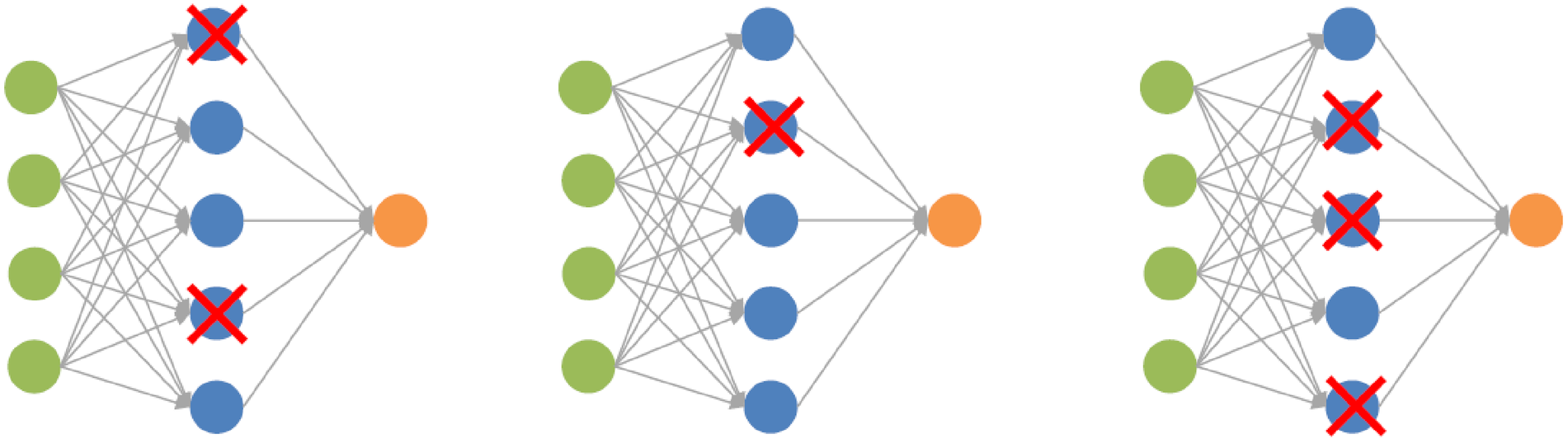
(a) Standard Neural Net (b) After applying dropout.

Dropout — Srivastava et al. (2014)

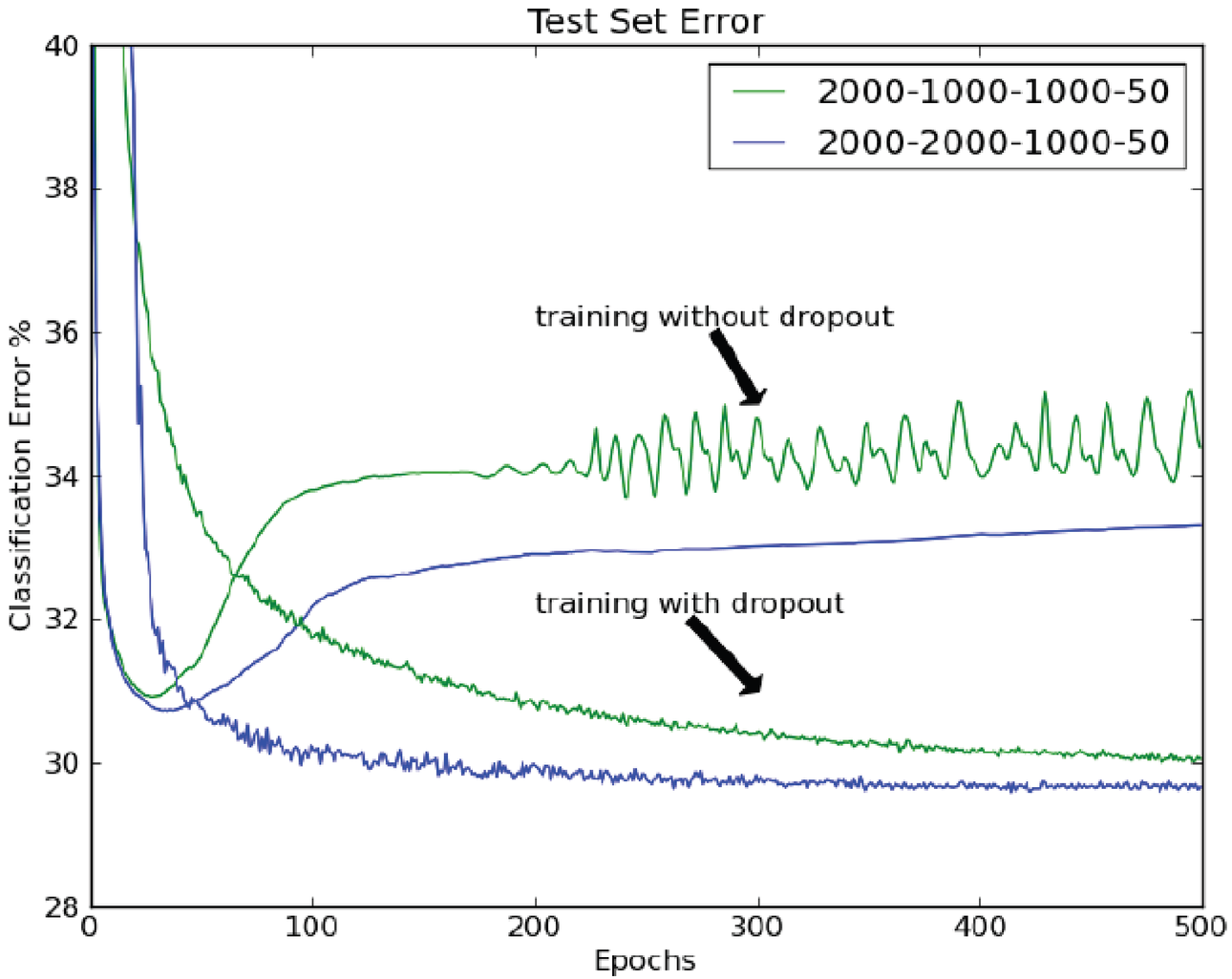
https://medium.com/konvergen/understanding-dropout-ddb60c9f98aa

Dropout

정해진 데이터셋으로 한 번 학습 하는 과정으로 여러 네트워크를 이용한 최적 네트워크 탐색 과정의 효과를 얻을 수 있음



Dropout


Test Set Error
2000-1000-1000-50
2000-2000-1000-50
training without dropout
training with dropout
Classification Error %
Epochs
40
38
36
34
32
30
28
0
100
200
300
400
500

Weight decay & Weight restriction

Overfitting 된 weight들은 보통 그 크기가 매우 큼

따라서 그 weight들의 값을 너무 키우지 않는 방법으로 Overfitting을 방지

Weight decay (가중치 감쇠)

$$E = E + \frac{\lambda}{2}||w||^2$$

L2-regularization

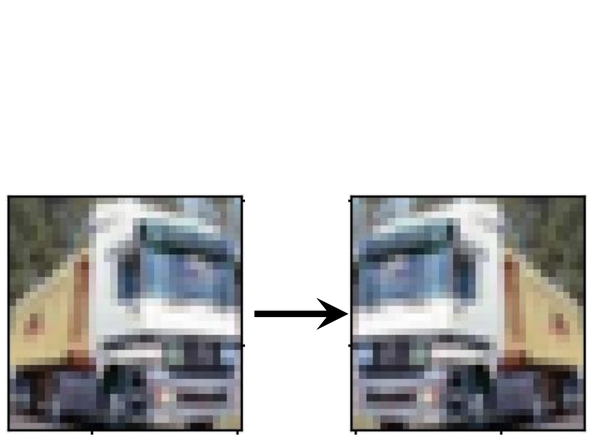
Weight restriction (가중치 제한)

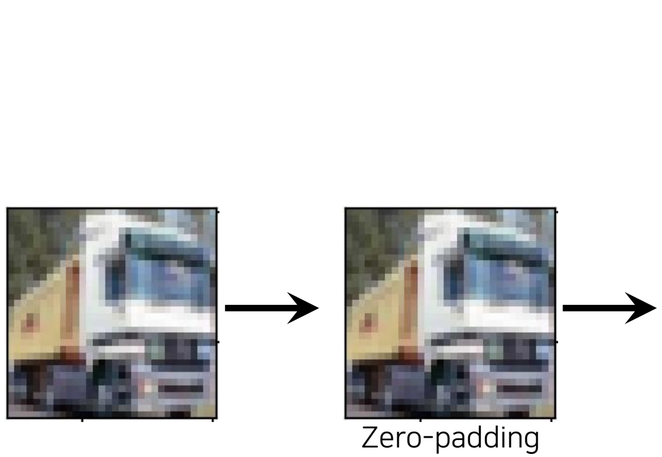
$$||w||^2 < c$$

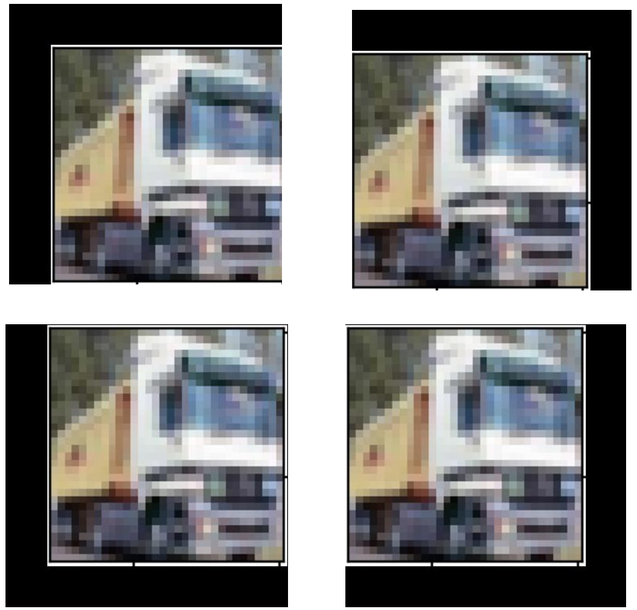
Data augmentation (데이터 증강)

데이터의 특징 학습에 도움이 되도록 데이터에 변형을 가하여 양을 늘리는 기법

데이터의 크기가 작을 때 잘 일어날 수 있는 overfitting 방지



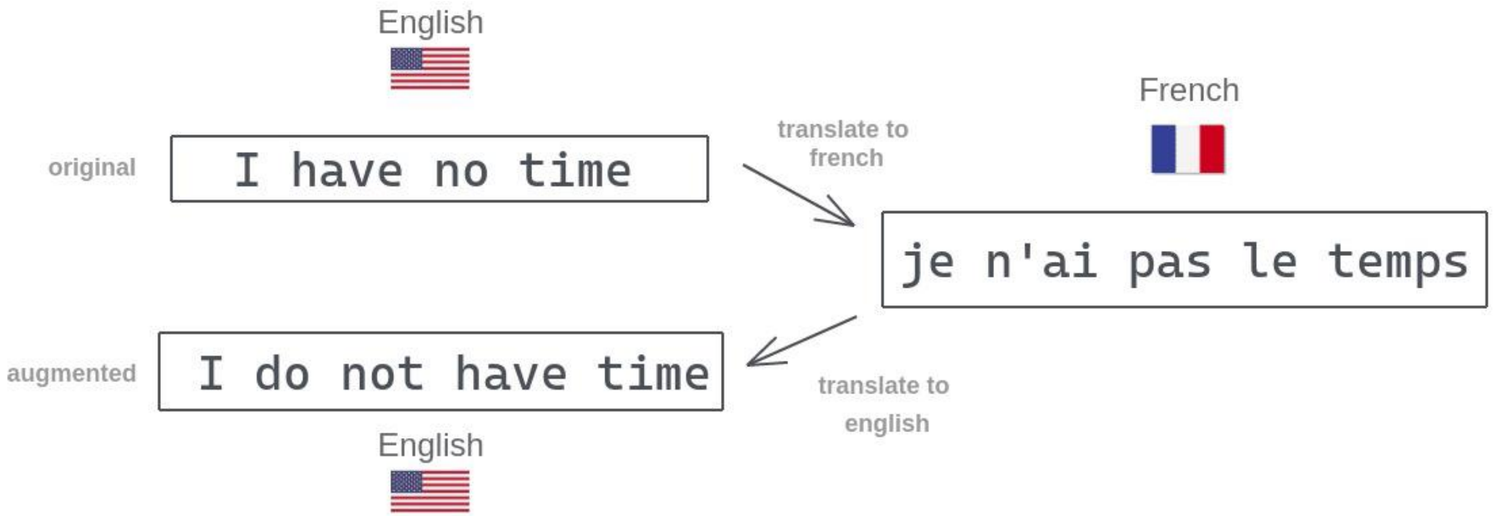




Data augmentation (데이터 증강)

데이터의 특징 학습에 도움이 되도록 데이터에 변형을 가하여 양을 늘리는 기법

데이터의 크기가 작을 때 잘 일어날 수 있는 overfitting 방지



Weight Initialization (가중치 초기화)

시작 weight의 값이 좋아야 학습도 잘 된다!

특정 분포를 정해두고 해당 분포에서 sampling한 값들을 weight의 초기값으로 설정

정규분포에서 응용된 분포들을 주로 사용

sampling을 하기 때문에 매번 학습할 때마다 결과가 조금씩 다르게 나옴

Xavier Initialization

표준편차가 $\frac{1}{\sqrt{n}}$인 정규분포로 초기화 ($n = \sqrt{\frac{2}{n_{in}+n_{out}}}$, n은 노드 수)

활성화 함수가 sigmoid일 때 사용

He Initialization

표준편차가 $\frac{\sqrt{2}}{\sqrt{n_{in}}}$인 정규분포로 초기화 ($n$은 노드 수)

활성화 함수가 ReLU일 때 사용

See you in the next lecture!

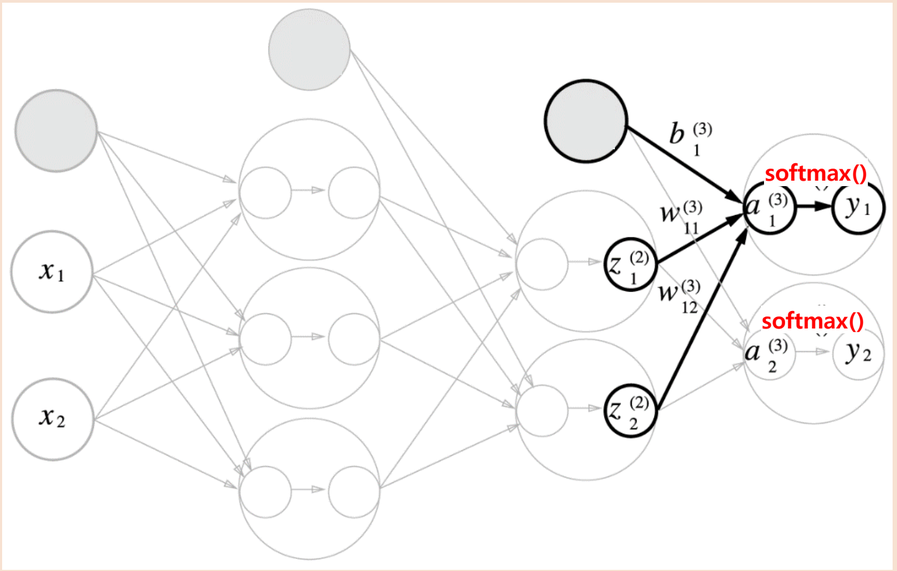
딥러닝 (Deep Learning) 기초

★ 연습 문제

Deep & High Learning

연습문제 1

Weight와 Bias가 아래 표와 같이 주어져 있을 때, Error 값을 계산하세요.



Cat: ? ⟷ Label: 1

Dog: ? ⟷ Label : 0

Error

?
?

z_1	z_2	w_{11}	w_{21}	w_{12}	w_{22}	b_1	b_2	activation
0.2	0.4	0.1	0.2	0.9	0.2	0.1	0.9	softmax

$$e^{0.48} \approx 1.62$$
$$e^{1.02} \approx 2.77$$
$$log 0.37 \approx -0.4318$$

연습문제 1 풀이

z_1	z_2	w_{11}	w_{21}	w_{12}	w_{22}	b_1	b_2	activation
0.2	0.4	0.1	0.2	0.9	0.2	0.1	0.9	softmax

$$e^{0.48} \approx 1.62$$
$$e^{1.02} \approx 2.77$$
$$log0.37 \approx -0.4318$$

$$z = [0.2\ 0.4]$$

$$w = \begin{bmatrix} 0.1 & 0.9 \\ 0.2 & 0.2 \end{bmatrix}$$

$$b = [0.1\ 0.9]$$

$$a = wz + b$$

$$= \begin{bmatrix} 0.1 & 0.9 \\ 0.2 & 0.2 \end{bmatrix} \begin{bmatrix} 0.2 \\ 0.4 \end{bmatrix} + \begin{bmatrix} 0.1 \\ 0.9 \end{bmatrix}$$

$$a = \begin{bmatrix} 0.38 \\ 0.12 \end{bmatrix} + \begin{bmatrix} 0.1 \\ 0.9 \end{bmatrix}$$

$$= \begin{bmatrix} 0.48 \\ 1.02 \end{bmatrix}$$

$$softmax(a) \approx \begin{bmatrix} 0.37 \\ 0.63 \end{bmatrix}$$

연습문제 1 풀이

z_1	z_2	w_{11}	w_{21}	w_{12}	w_{22}	b_1	b_2	activation
0.2	0.4	0.1	0.2	0.9	0.2	0.1	0.9	softmax

$$e^{0.48} \approx 1.62$$
$$e^{1.02} \approx 2.77$$
$$log0.37 \approx -0.4318$$

MSE Loss

$$=\frac{1}{2}\{(0.37-1)^2+(0.63-0)^2\}=\frac{1}{2}(0.396+0.396)=0.396$$

Cross Entropy Loss

$$=-(1\times log0.37+0\times log0.63)=0.4318$$

연습문제 2

$f = ReLU(x) = \begin{cases} x\ (x > 0) \\ 0\ (x \leq 0) \end{cases}$ 일 때, 계산 그래프를 그리고 $\frac{\partial L}{\partial x}$ 를 $\frac{\partial L}{\partial f}$와 변수들을 사용하여 나타내세요.

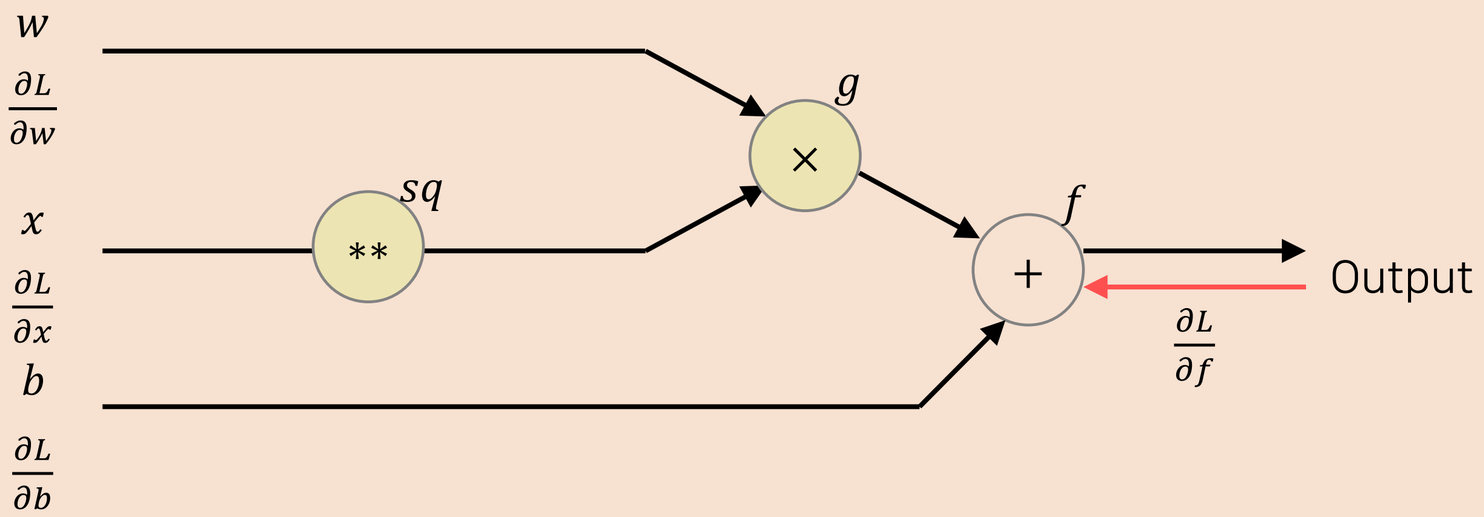
연습문제 3

$f = g + b, g = wx^2$ 일 때, 계산 그래프를 그리고 $\frac{\partial L}{\partial w'}, \frac{\partial L}{\partial x'}, \frac{\partial L}{\partial b}$ 를 $\frac{\partial L}{\partial f}$와 변수들을 사용하여 나타내세요.

연습문제 3 풀이

$f = g + b, g = wx^2$ 일 때, 계산 그래프를 그리고 $\frac{\partial L}{\partial w}, \frac{\partial L}{\partial x}, \frac{\partial L}{\partial b}$ 를 $\frac{\partial L}{\partial f}$와 변수들을 사용하여 나타내세요.

$f = g + b, g = wx^2$



Thank You!