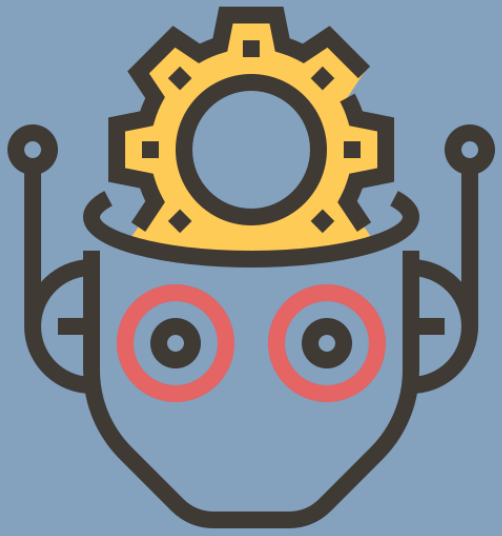
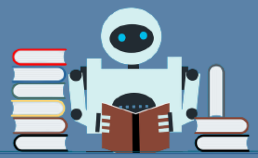


Deep Learning

스마트인재개발원
Smart Human Resources Development

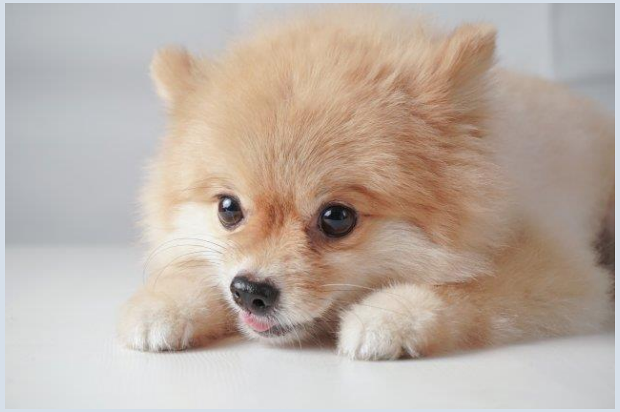
정 봉 균 연구원

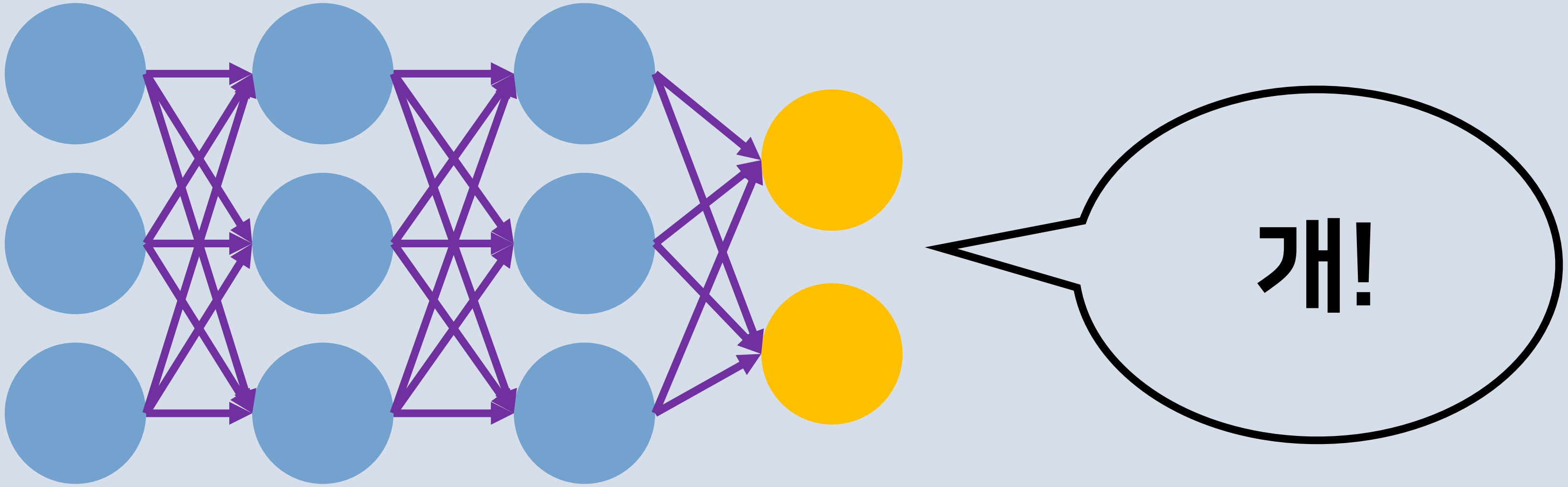


1. 활성화 함수의 개념을 이해하고 종류를 알 수 있다.
2. 오차역전파의 개념을 이해할 수 있다.
3. 다양한 경사하강법 종류를 알 수 있다.
4. 인공신경망이 학습하는 과정에 대해 이해할 수 있다.

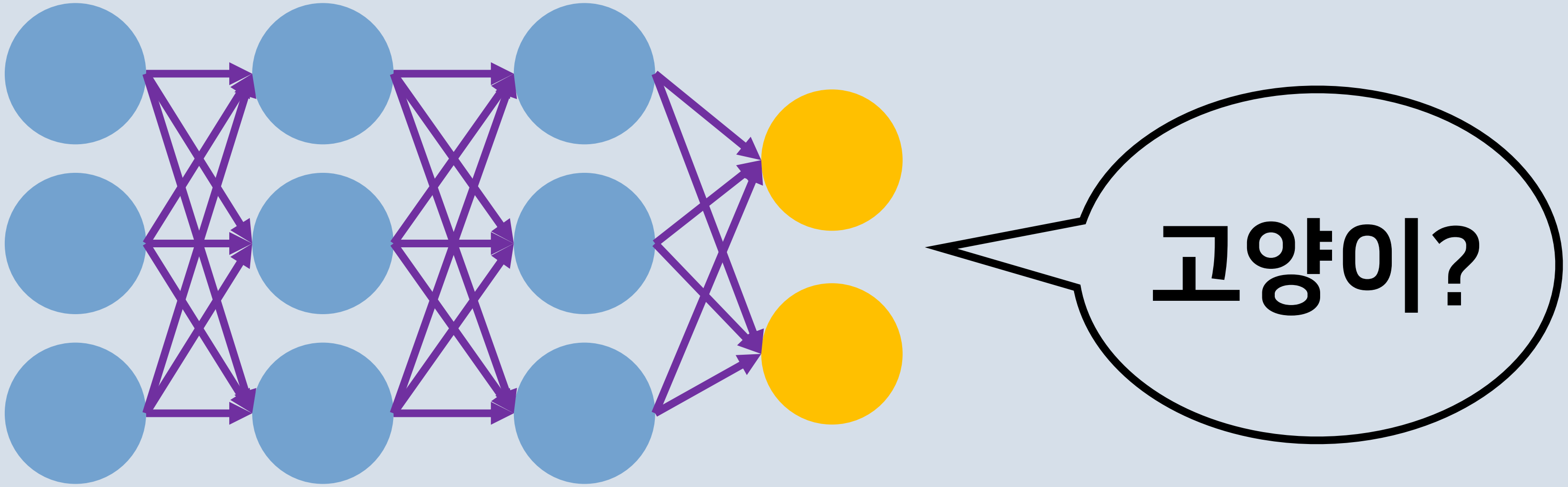
퍼셉트론 학습

퍼셉트론 학습

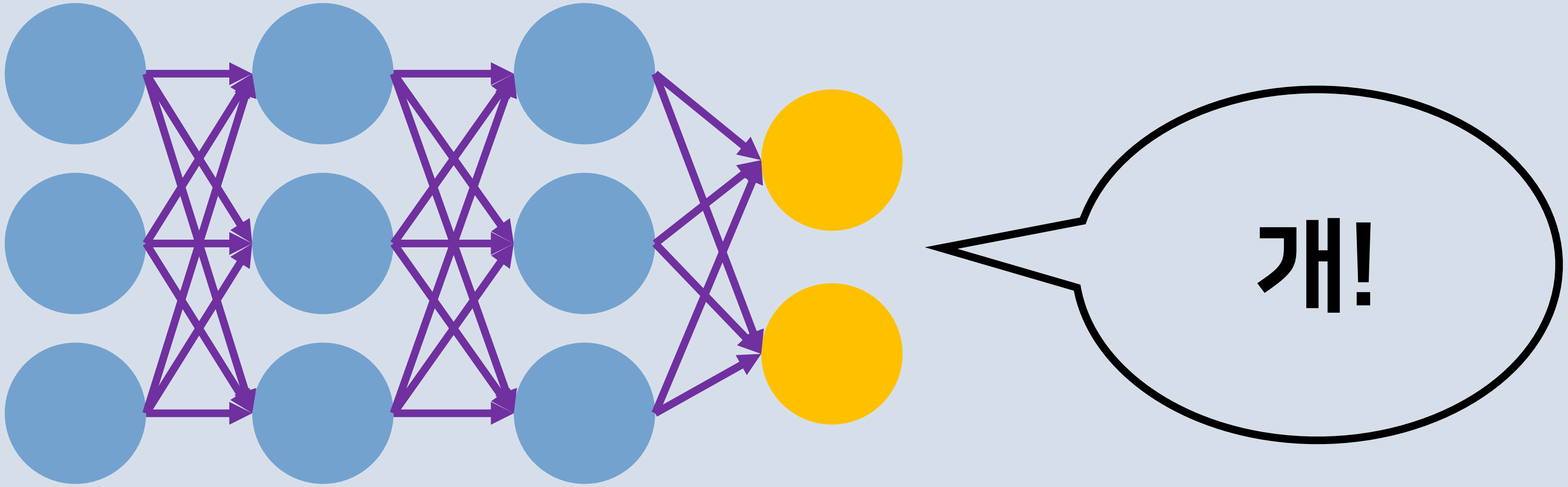




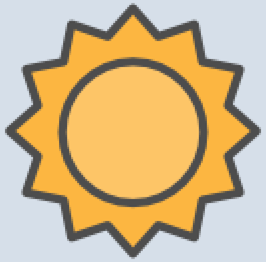
퍼셉트론 학습



퍼셉트론 학습



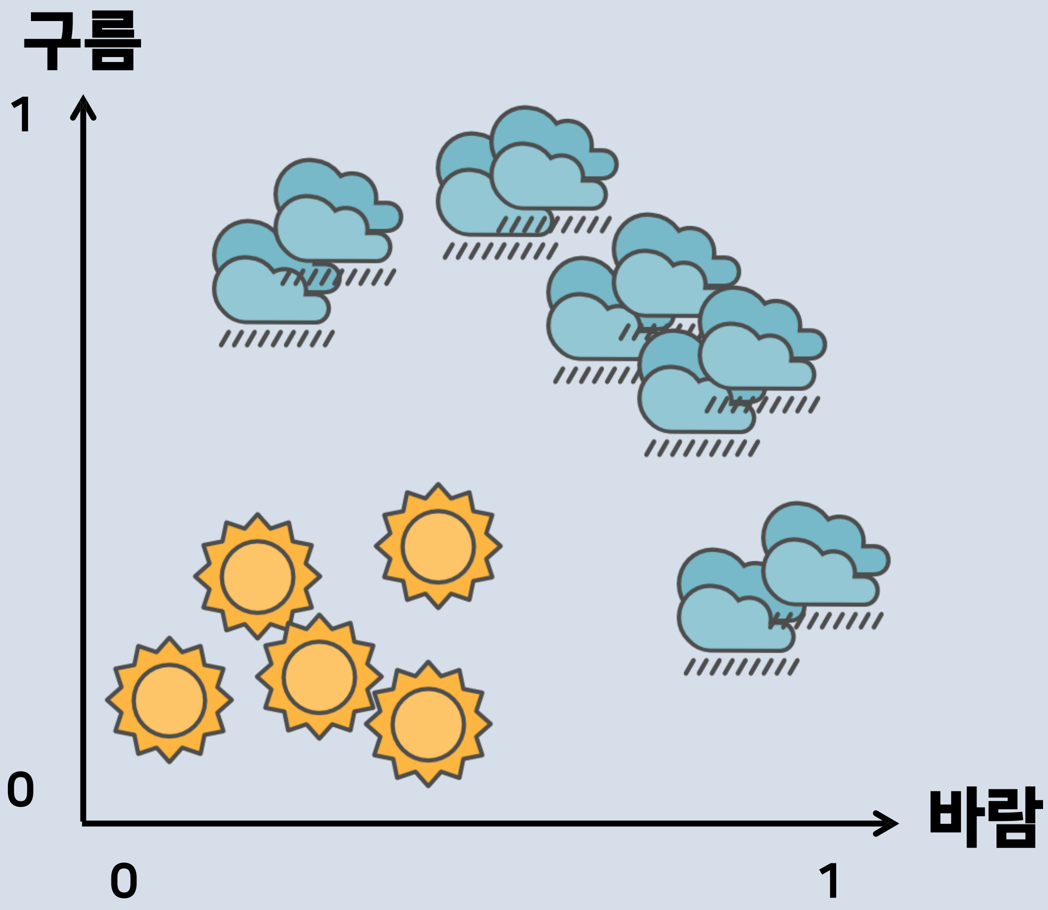
퍼셉트론 학습



아침에 구름이 없고 바람이 약하면, 맑은 날이 될 확률이 높다

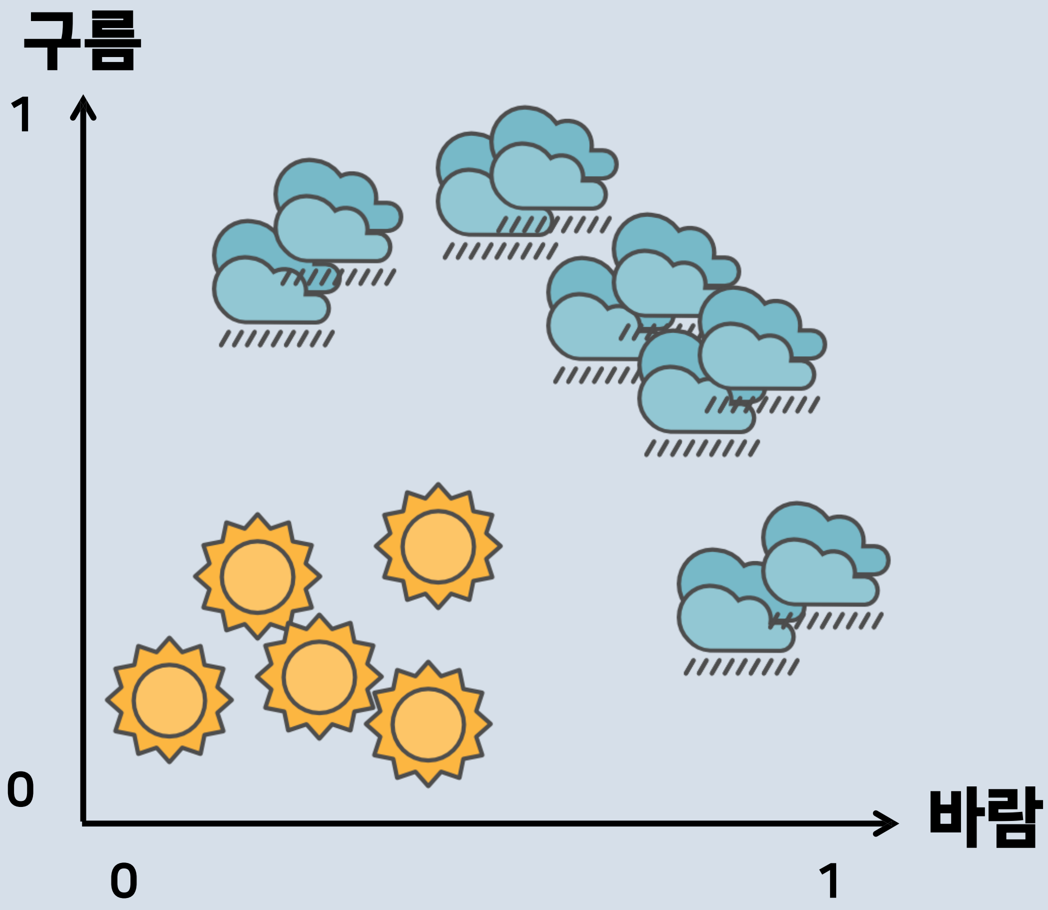
아침에 구름이 많고 바람이 강하면, 비가 올 확률이 높다

퍼셉트론 학습



구름	바람	날씨
0.8	0.6	☔
0.6	0.9	☔
0.1	0.2	🌞
0.3	0.1	🌞
0.6	0.6	☔
0.4	0.3	🌞
0.1	0.2	🌞

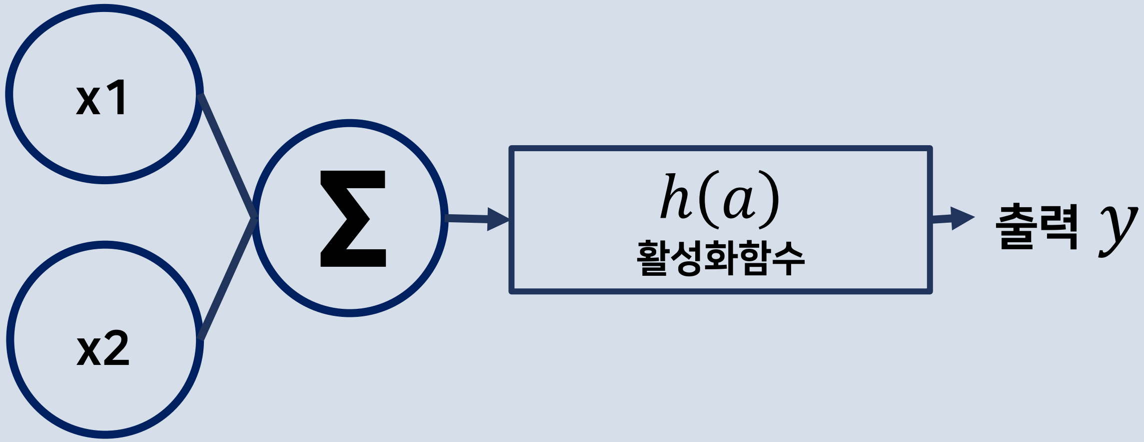
퍼셉트론 학습



x1	x2	y
0.8	0.6	1
0.6	0.9	1
0.1	0.2	0
0.3	0.1	0
0.6	0.6	1
0.4	0.3	0
0.1	0.2	0

퍼셉트론 학습

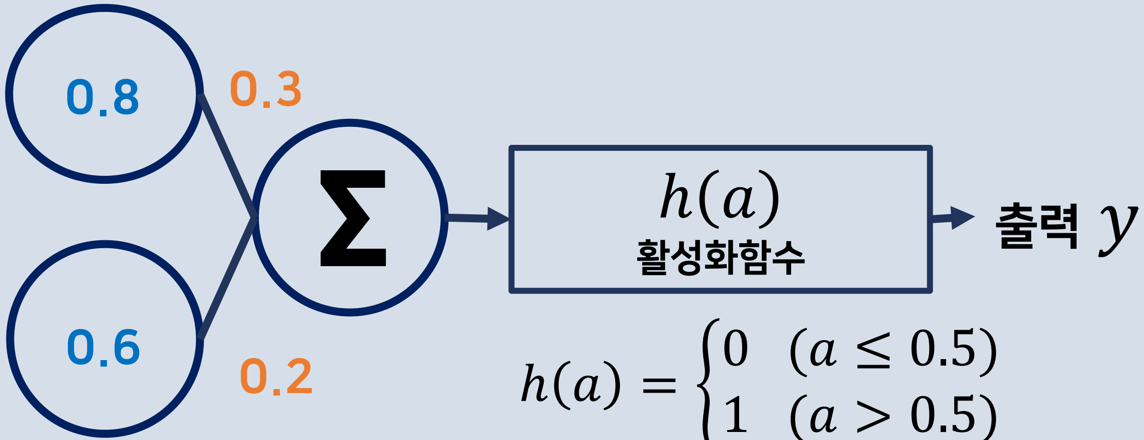
x1	x2
0.8	0.6
0.6	0.9
0.1	0.2
0.3	0.1
0.6	0.6
0.4	0.3
0.1	0.2



y
1
1
0
0
1
0
0

퍼셉트론 학습

x1	x2
0.8	0.6
0.6	0.9
0.1	0.2
0.3	0.1
0.6	0.6
0.4	0.3
0.1	0.2



$$h(a) = \begin{cases} 0 & (a \leq 0.5) \\ 1 & (a > 0.5) \end{cases}$$

0.8 * 0.3 = 0.24
0.6 * 0.2 = 0.12 +

0.36

y	$\hat{y}$
1	0
1	
0	
0	
1	
0	
0	

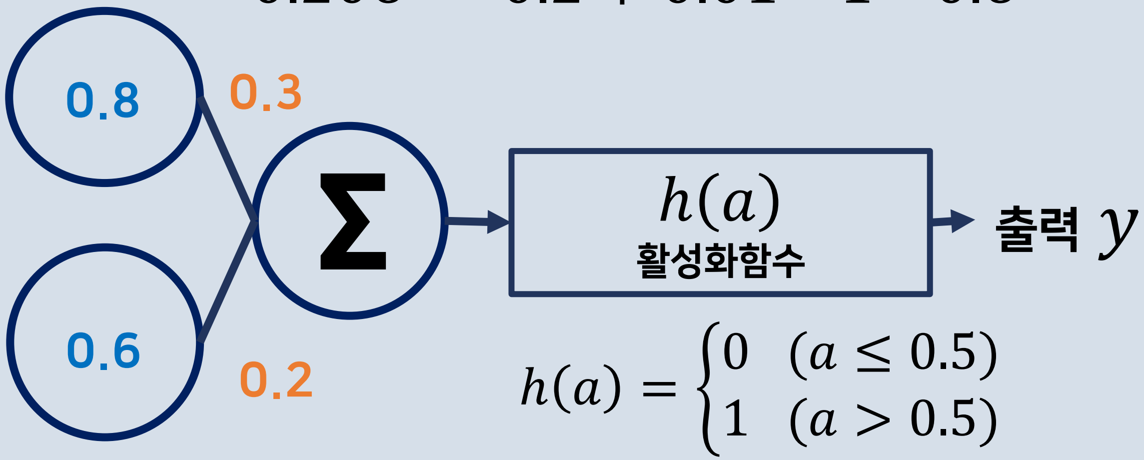
퍼셉트론 학습

$$w_i := w_i + \alpha * (y - \hat{y}) * x_i$$

$$0.308 := 0.3 + 0.01 * 1 * 0.8$$

$$0.206 := 0.2 + 0.01 * 1 * 0.6$$

x1	x2
0.8	0.6
0.6	0.9
0.1	0.2
0.3	0.1
0.6	0.6
0.4	0.3
0.1	0.2



$$h(a) = \begin{cases} 0 & (a \leq 0.5) \\ 1 & (a > 0.5) \end{cases}$$

0.8 * 0.3 = 0.24
0.6 * 0.2 = 0.12 +

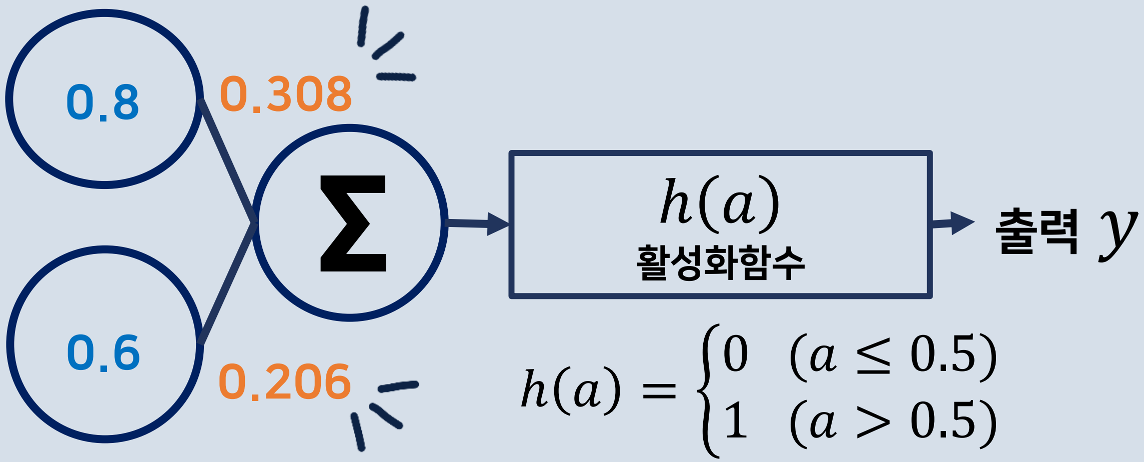
0.36

y	$\hat{y}$
1	0
1	
0	
0	
1	
0	
0	

퍼셉트론 학습

$$w_i := w_i + \alpha * (y - \hat{y}) * x_i$$

x1	x2
0.8	0.6
0.6	0.9
0.1	0.2
0.3	0.1
0.6	0.6
0.4	0.3
0.1	0.2



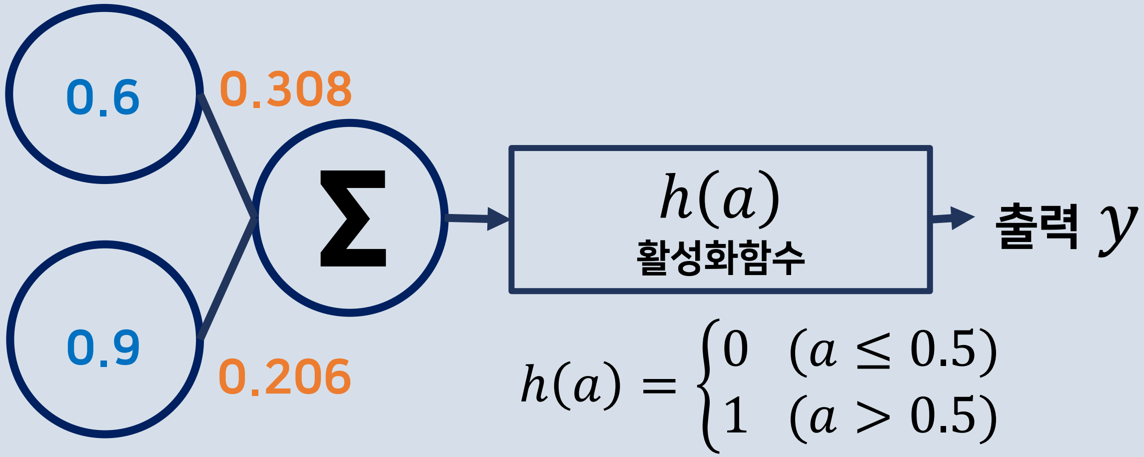
$$h(a) = \begin{cases} 0 & (a \leq 0.5) \\ 1 & (a > 0.5) \end{cases}$$

y	$\hat{y}$
1	0
1	
0	
0	
1	
0	
0	

퍼셉트론 학습

$$w_i := w_i + \alpha * (y - \hat{y}) * x_i$$

x1	x2
0.8	0.6
0.6	0.9
0.1	0.2
0.3	0.1
0.6	0.6
0.4	0.3
0.1	0.2



$$h(a) = \begin{cases} 0 & (a \leq 0.5) \\ 1 & (a > 0.5) \end{cases}$$

0.6 * 0.308 = 0.1848
0.9 * 0.206 = 0.1854 +

0.3702

y	$\hat{y}$
1	0
1	
0	
0	
1	
0	
0	

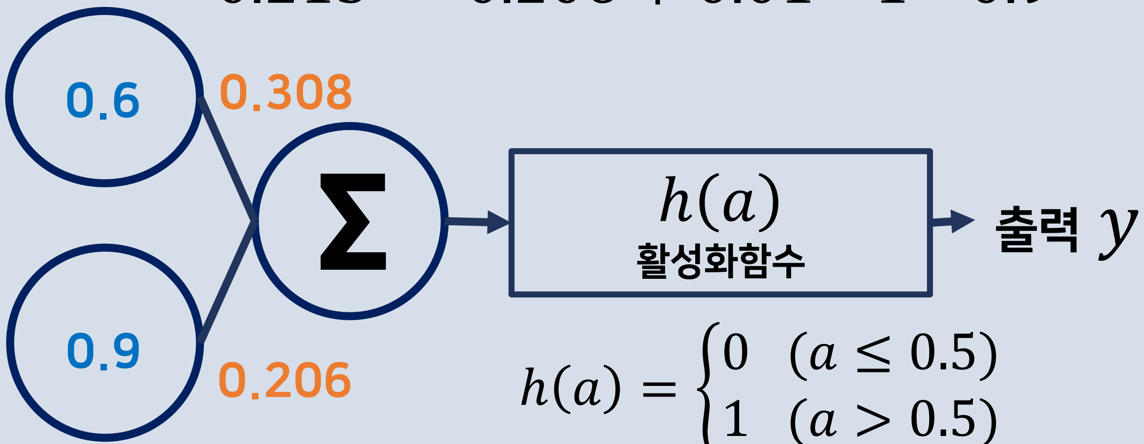
퍼셉트론 학습

x1	x2
0.8	0.6
0.6	0.9
0.1	0.2
0.3	0.1
0.6	0.6
0.4	0.3
0.1	0.2

$$w_i := w_i + \alpha * (y - \hat{y}) * x_i$$

$$0.314 := 0.308 + 0.01 * 1 * 0.6$$

$$0.215 := 0.206 + 0.01 * 1 * 0.9$$



$$h(a) = \begin{cases} 0 & (a \leq 0.5) \\ 1 & (a > 0.5) \end{cases}$$

0.6 * 0.308 = 0.1848

0.9 * 0.206 = 0.1854 +

0.3702

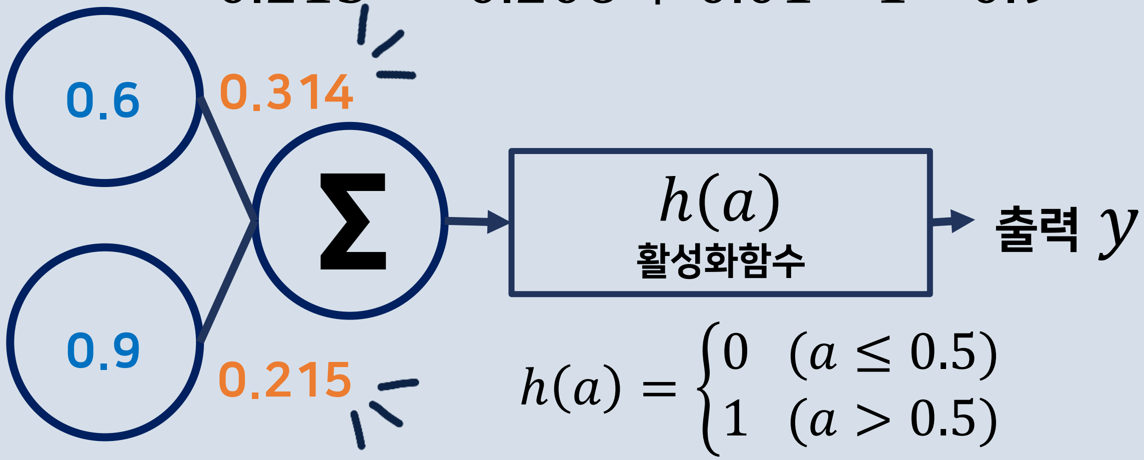
y	$\hat{y}$
1	0
1	0
0	
0	
1	
0	
0	

퍼셉트론 학습

$$w_i := w_i + \alpha * (y - \hat{y}) * x_i$$

x1	x2
0.8	0.6
0.6	0.9
0.1	0.2
0.3	0.1
0.6	0.6
0.4	0.3
0.1	0.2

$$0.314 := 0.308 + 0.01 * 1 * 0.6$$
$$0.215 := 0.206 + 0.01 * 1 * 0.9$$



$$h(a) = \begin{cases} 0 & (a \le 0.5) \\ 1 & (a > 0.5) \end{cases}$$

0.6 * 0.308 = 0.1848
0.9 * 0.206 = 0.1854 +

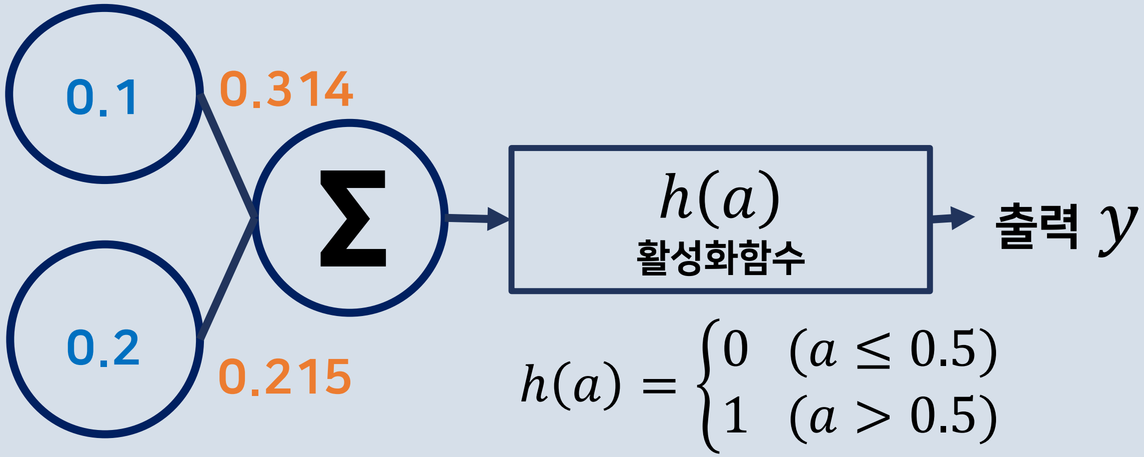
0.3702

y	$\hat{y}$
1	0
1	0
0	
0	
1	
0	
0	

퍼셉트론 학습

$$w_i := w_i + \alpha * (y - \hat{y}) * x_i$$

x1	x2
0.8	0.6
0.6	0.9
0.1	0.2
0.3	0.1
0.6	0.6
0.4	0.3
0.1	0.2



$$h(a) = \begin{cases} 0 & (a \leq 0.5) \\ 1 & (a > 0.5) \end{cases}$$

0.1 * 0.314 = 0.0314
0.2 * 0.215 = 0.0430 +

0.0744

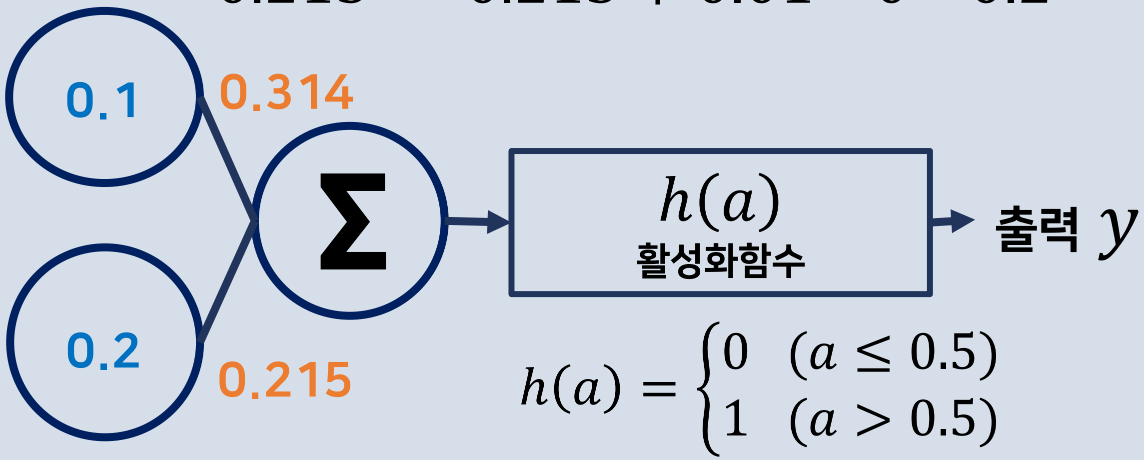
y	$\hat{y}$
1	0
1	0
0	
0	
1	
0	
0	

퍼셉트론 학습

$$w_i := w_i + \alpha * (y - \hat{y}) * x_i$$

$$0.314 := 0.314 + 0.01 * 0 * 0.1$$
$$0.215 := 0.215 + 0.01 * 0 * 0.2$$

x1	x2
0.8	0.6
0.6	0.9
0.1	0.2
0.3	0.1
0.6	0.6
0.4	0.3
0.1	0.2



$$h(a) = \begin{cases} 0 & (a \leq 0.5) \\ 1 & (a > 0.5) \end{cases}$$

y	$\hat{y}$
1	0
1	0
0	0
0	
1	
0	
0	

0.1 * 0.314 = 0.0314
0.2 * 0.215 = 0.0430 +

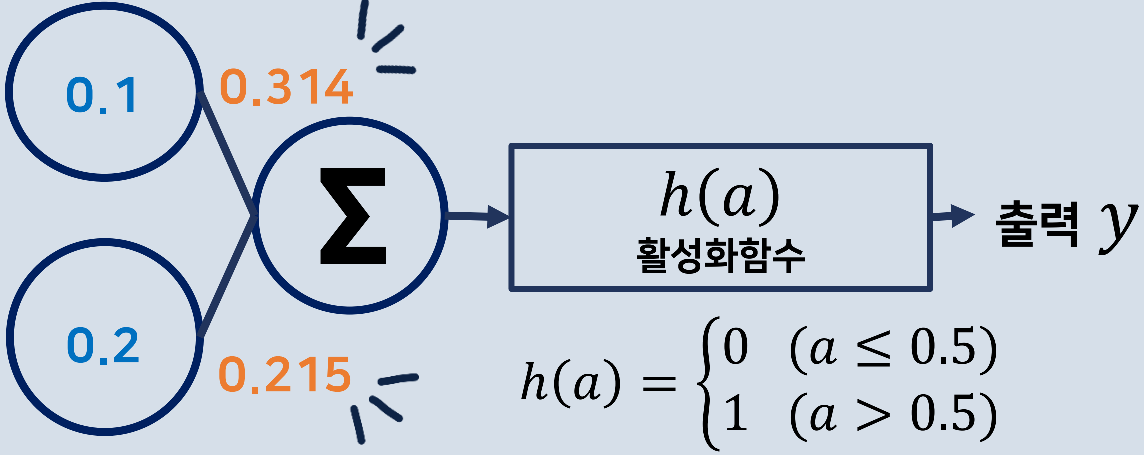
0.0744

퍼셉트론 학습

$$w_i := w_i + \alpha * (y - \hat{y}) * x_i$$

x1	x2
0.8	0.6
0.6	0.9
0.1	0.2
0.3	0.1
0.6	0.6
0.4	0.3
0.1	0.2

$$0.314 := 0.314 + 0.01 * 0 * 0.1$$
$$0.215 := 0.215 + 0.01 * 0 * 0.2$$



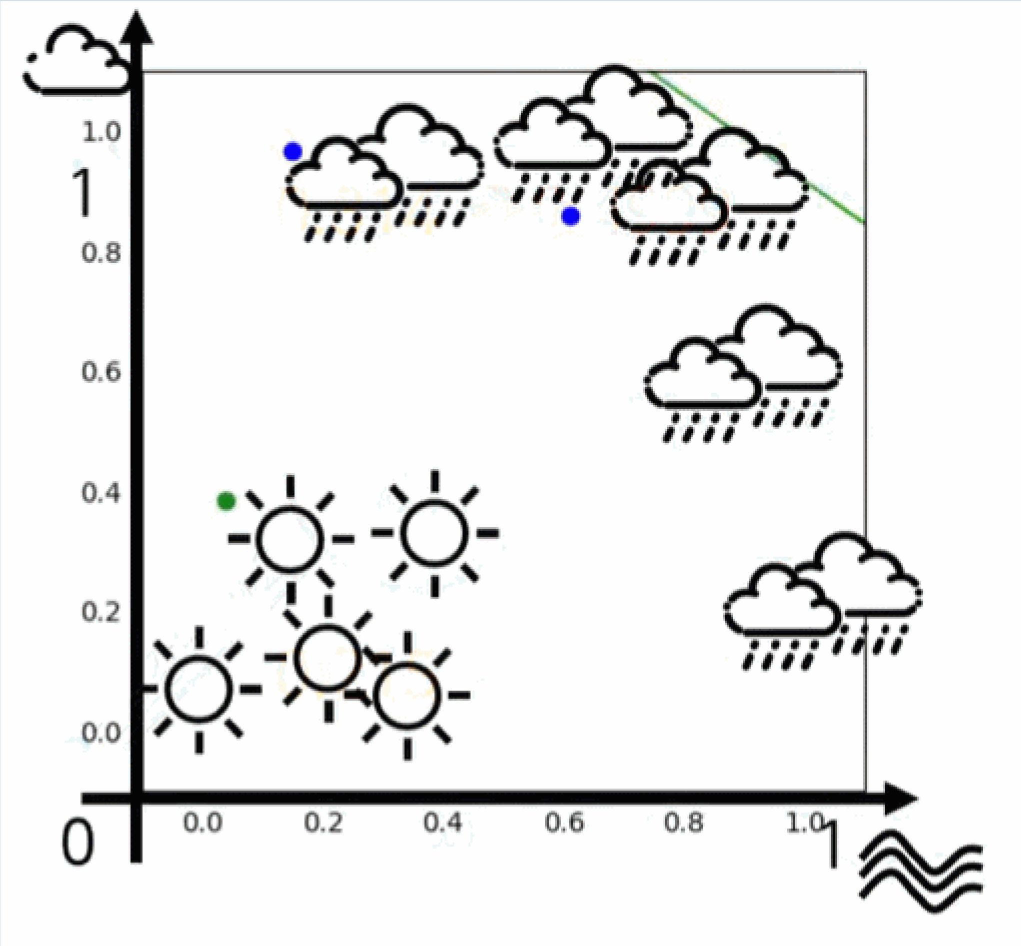
$$h(a) = \begin{cases} 0 & (a \le 0.5) \\ 1 & (a > 0.5) \end{cases}$$

0.1 * 0.314 = 0.0314
0.2 * 0.215 = 0.0430 +

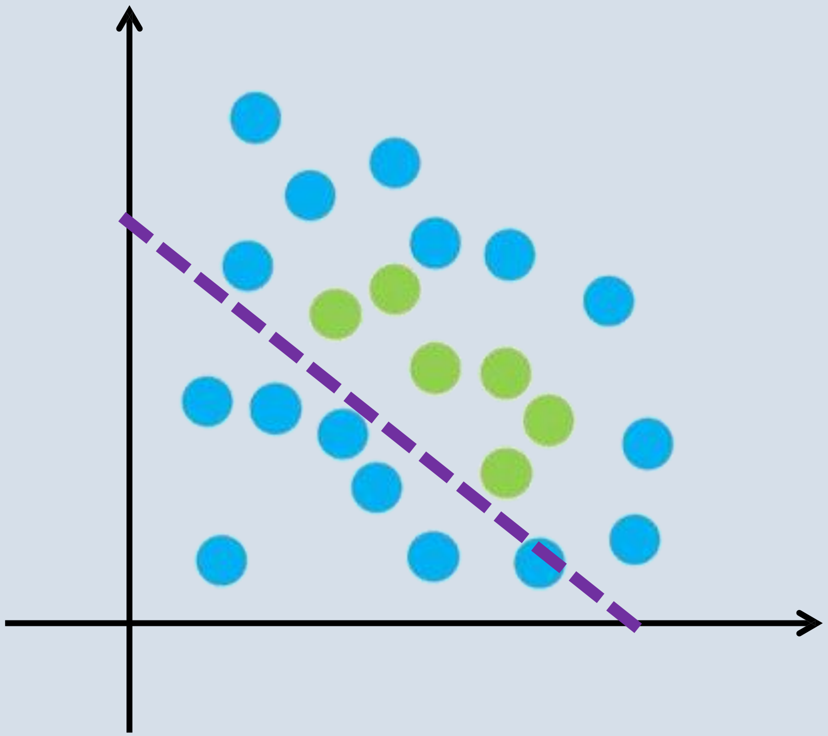
0.0744

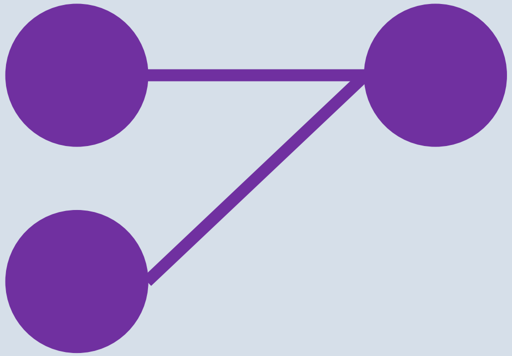
y	$\hat{y}$
1	0
1	0
0	0
0	
1	
0	
0	

퍼셉트론 학습

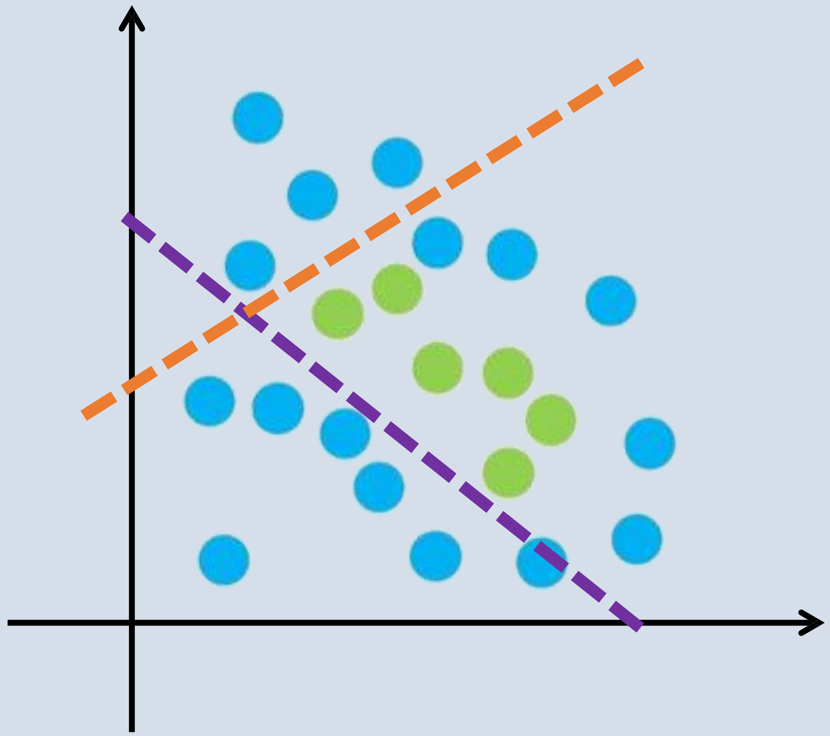


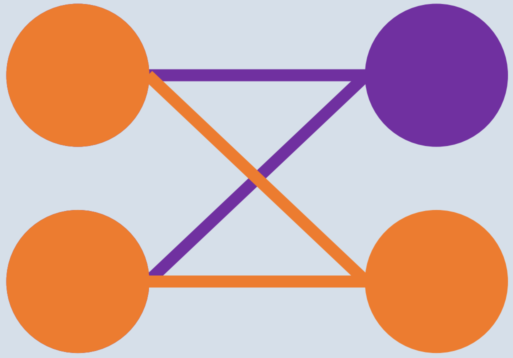
인공신경망 학습



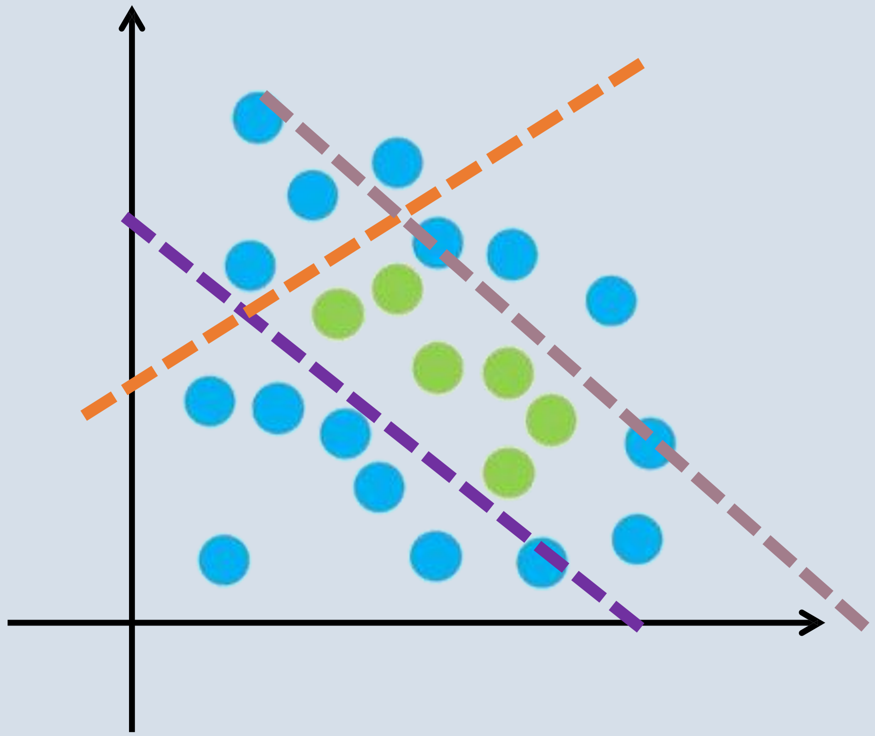


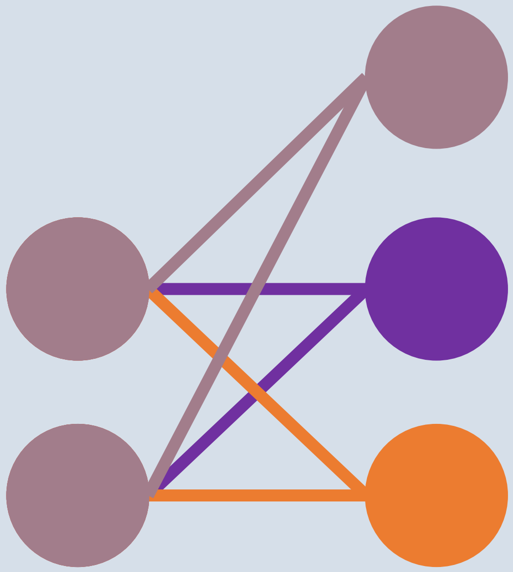
인공신경망 학습



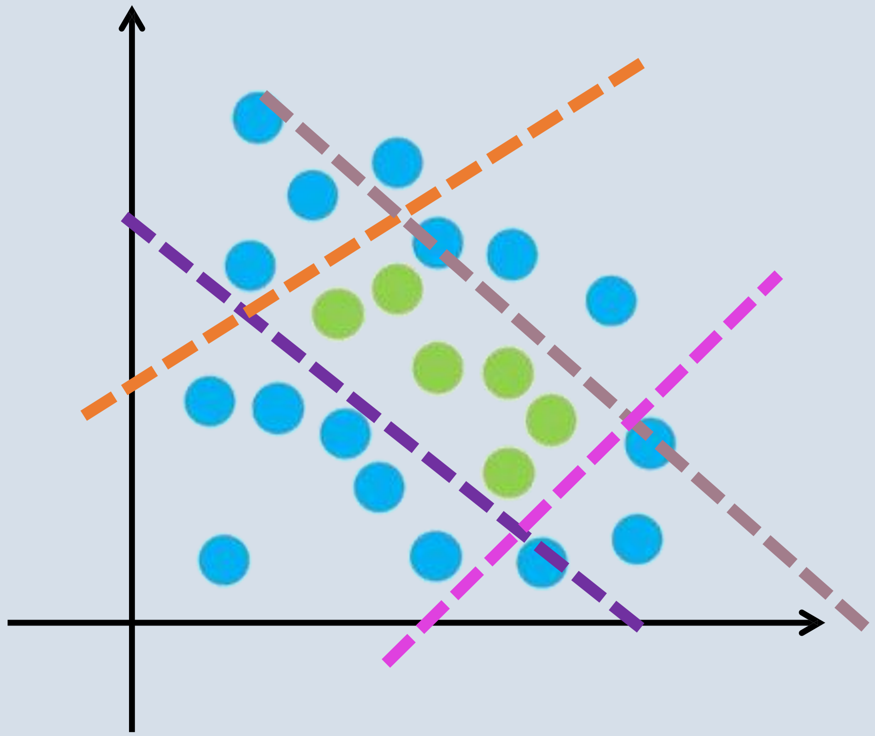


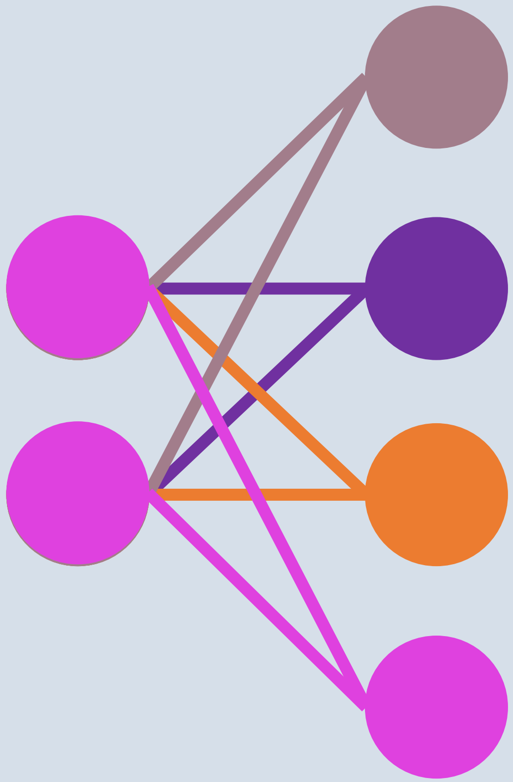
인공신경망 학습



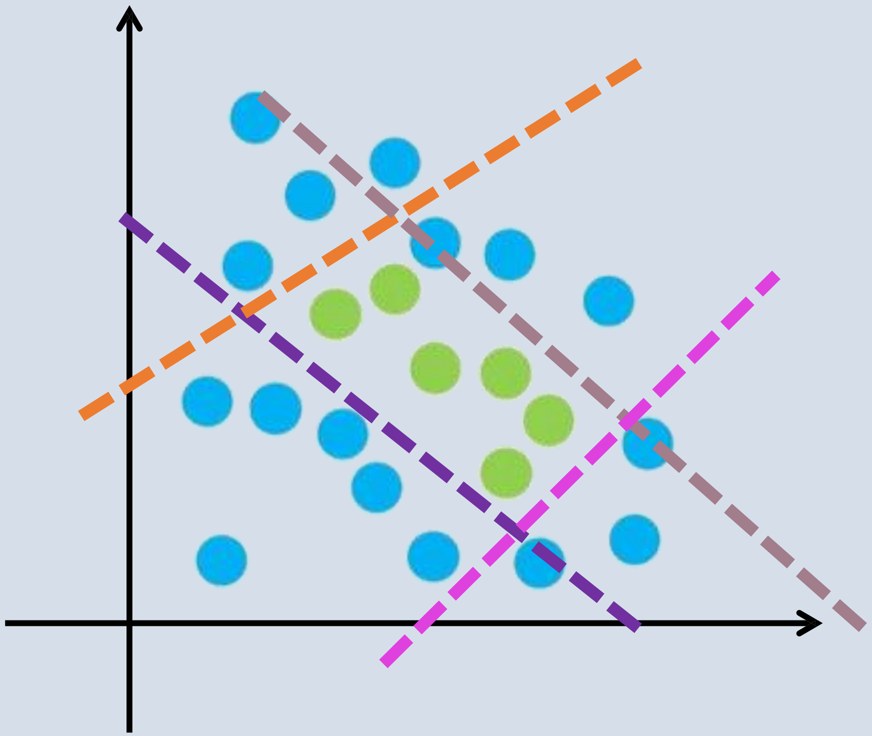


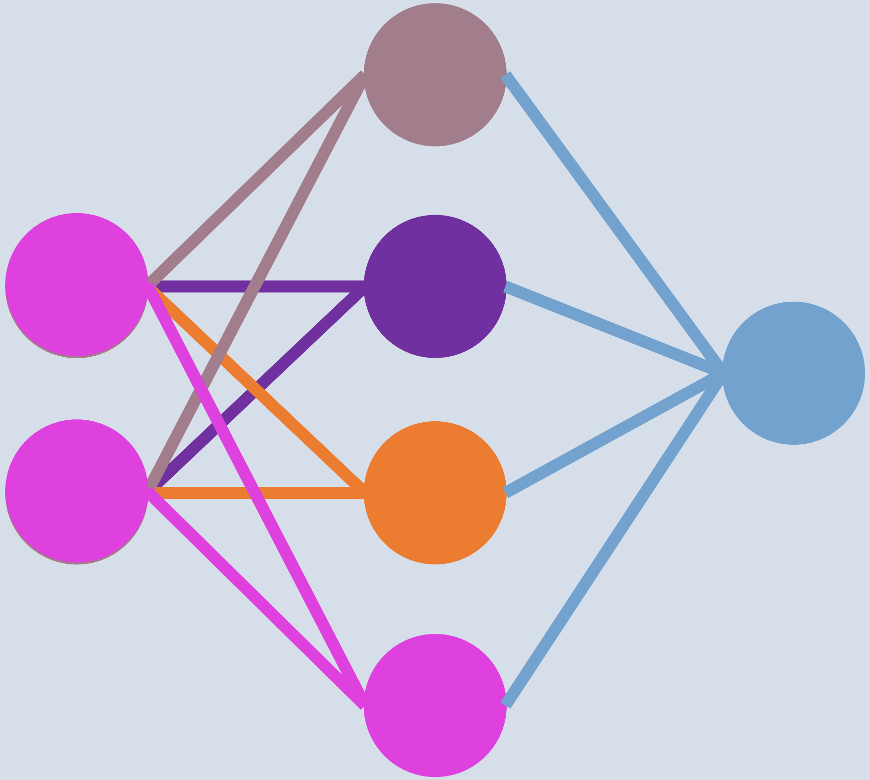
인공신경망 학습



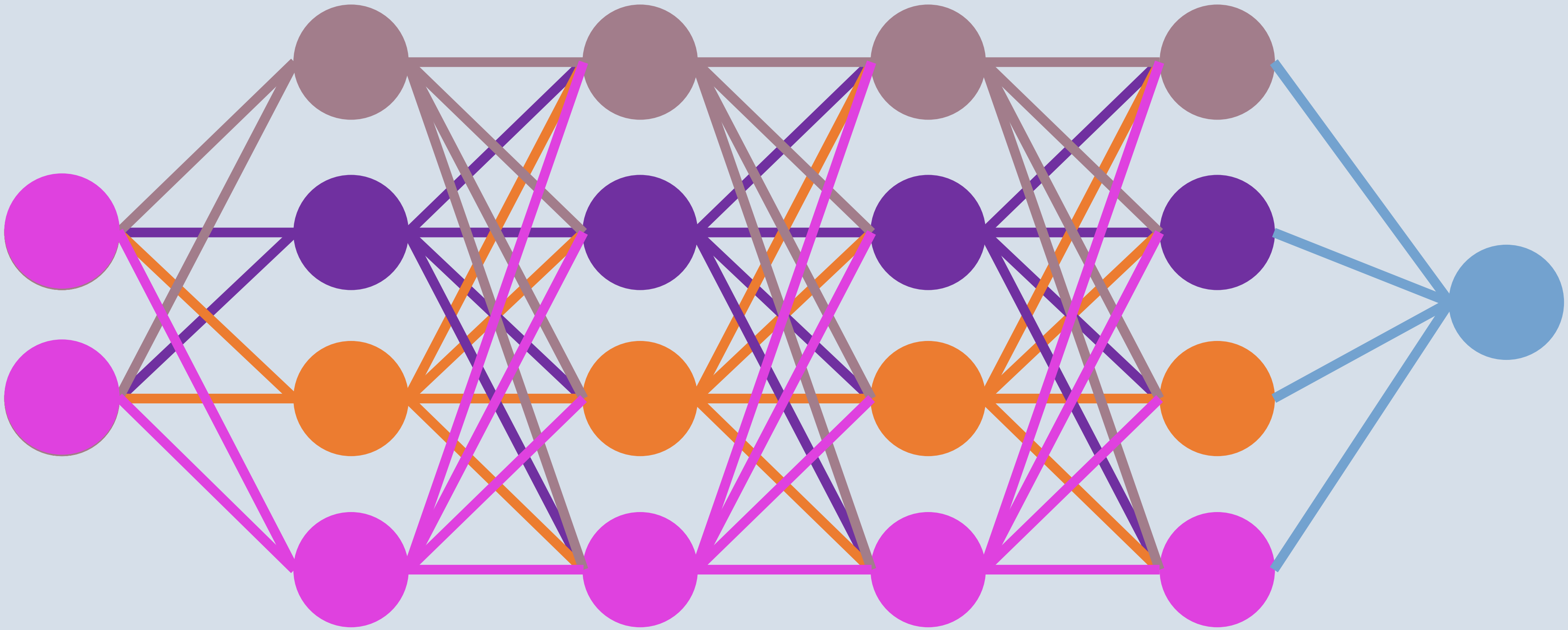


인공신경망 학습

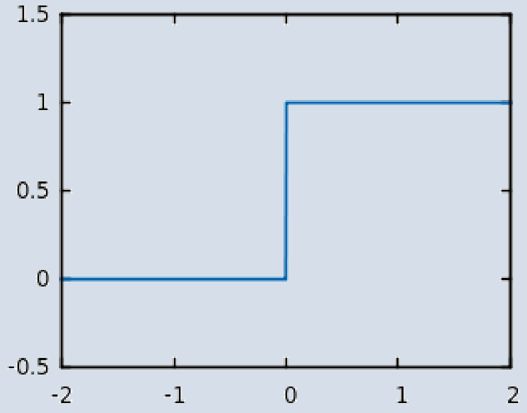




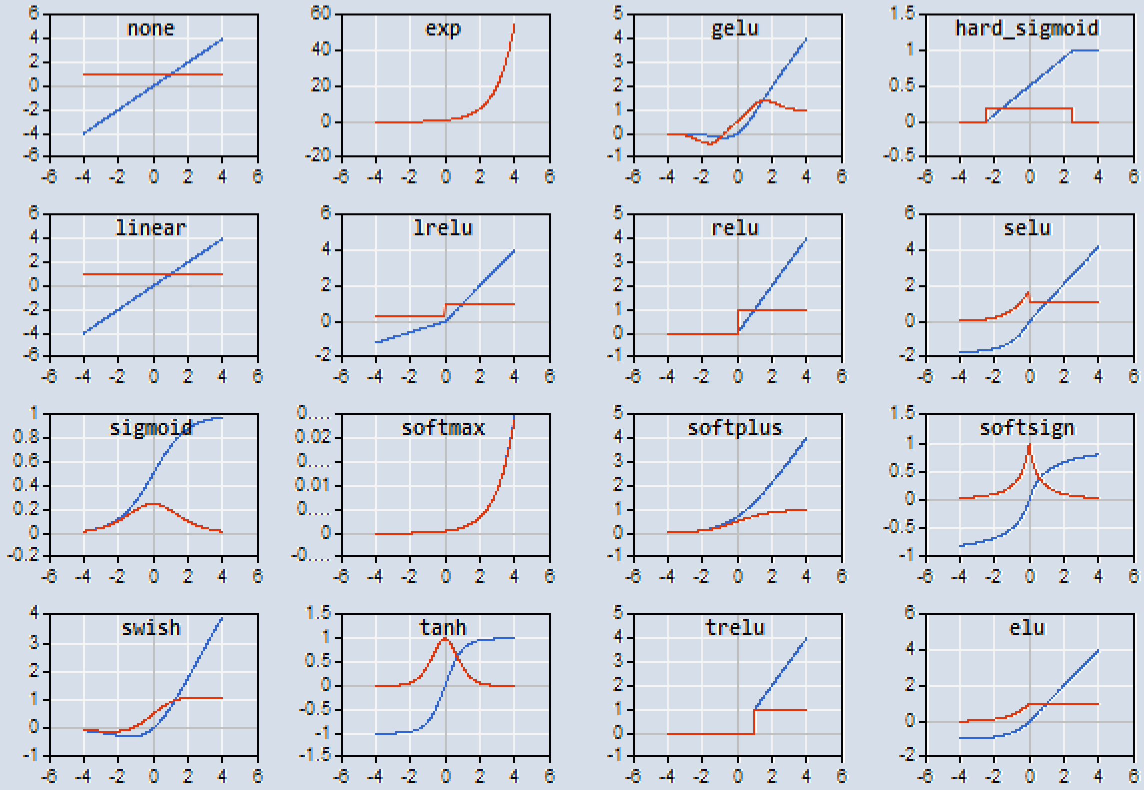
인공신경망 학습



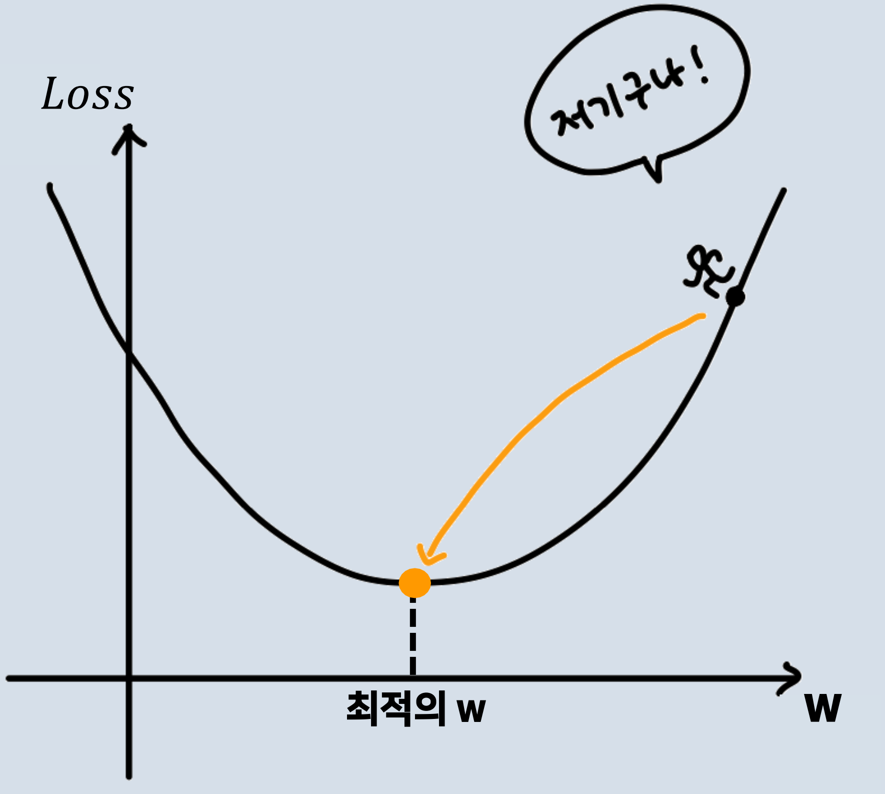
인공신경망의 학습 알고리즘 발전



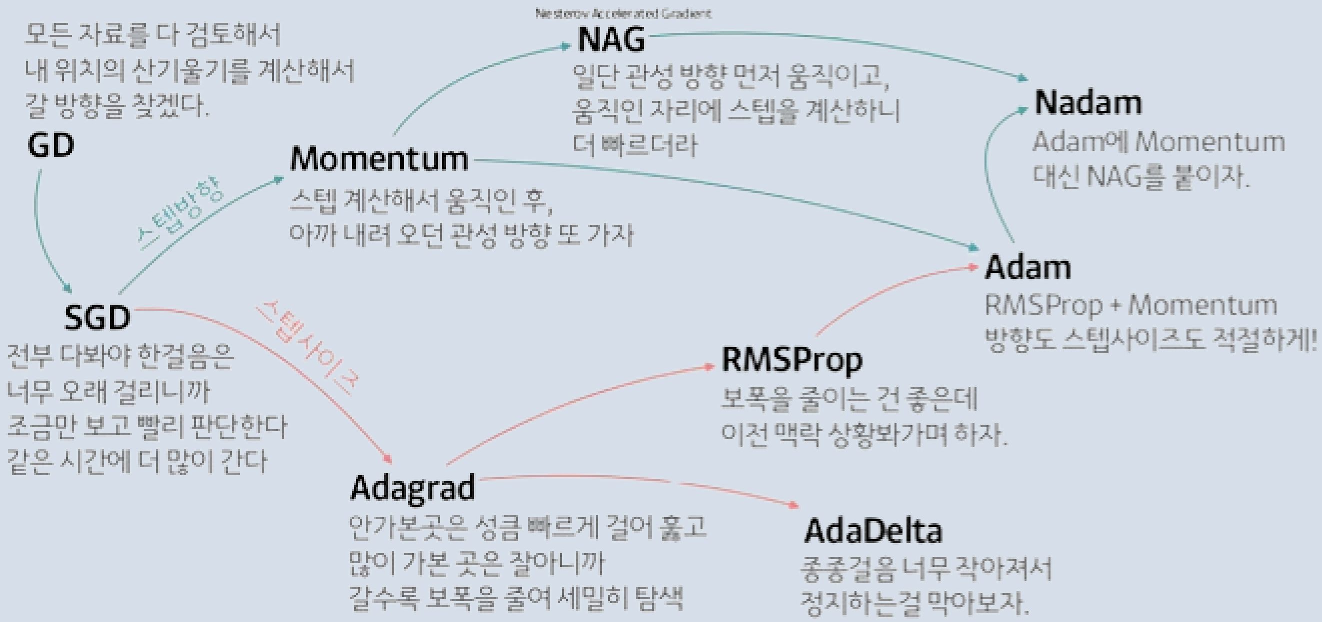
Step function(계단함수)



인공신경망의 학습 알고리즘 발전

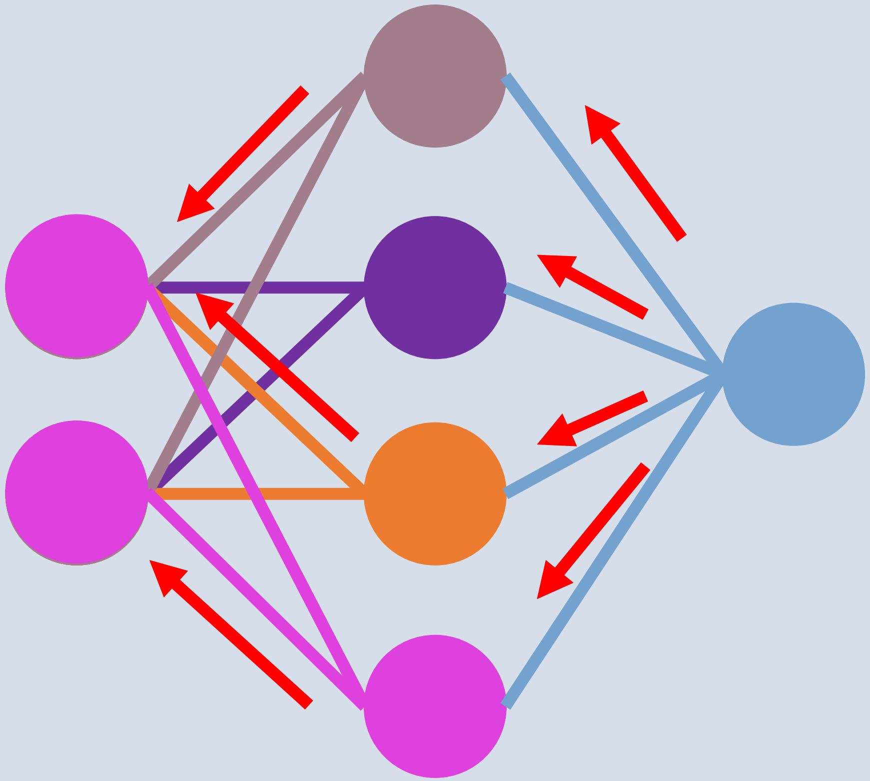


Gradient Descent Algorithm(경사하강법)



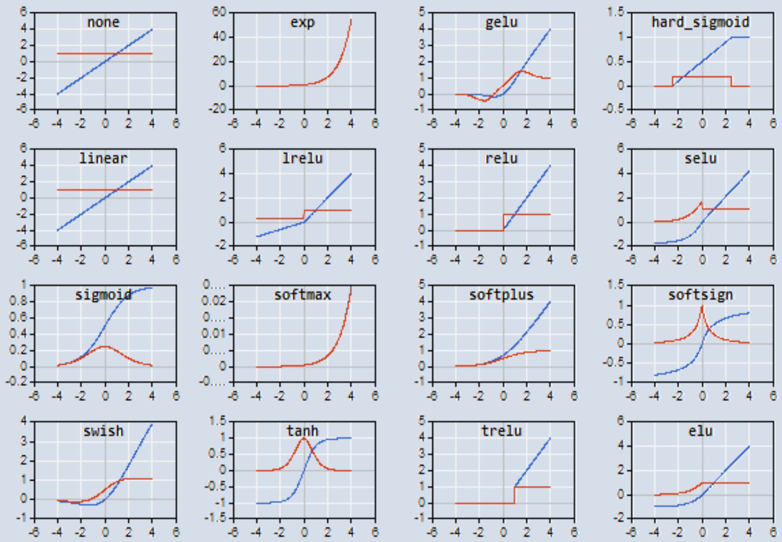
출처 : https://www.slideshare.net/yongho/ss-79607172

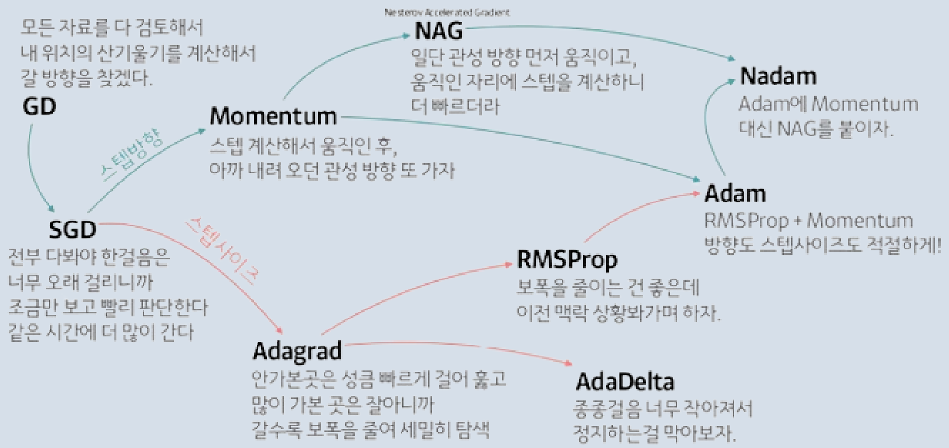
인공신경망의 학습 알고리즘 발전

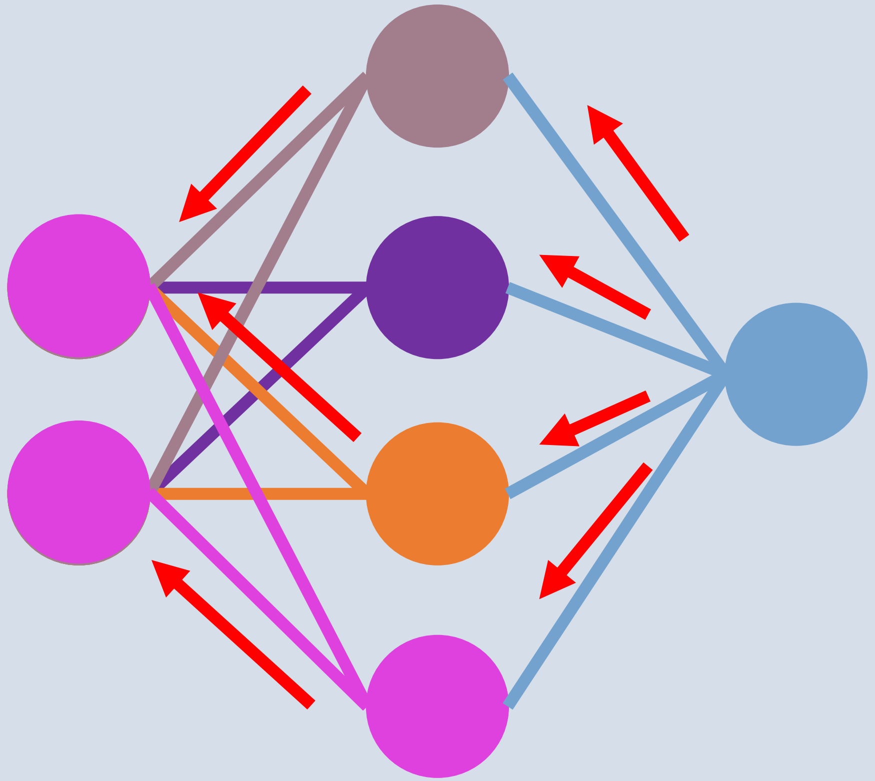


Deep Learning

인공신경망의 학습 알고리즘 발전







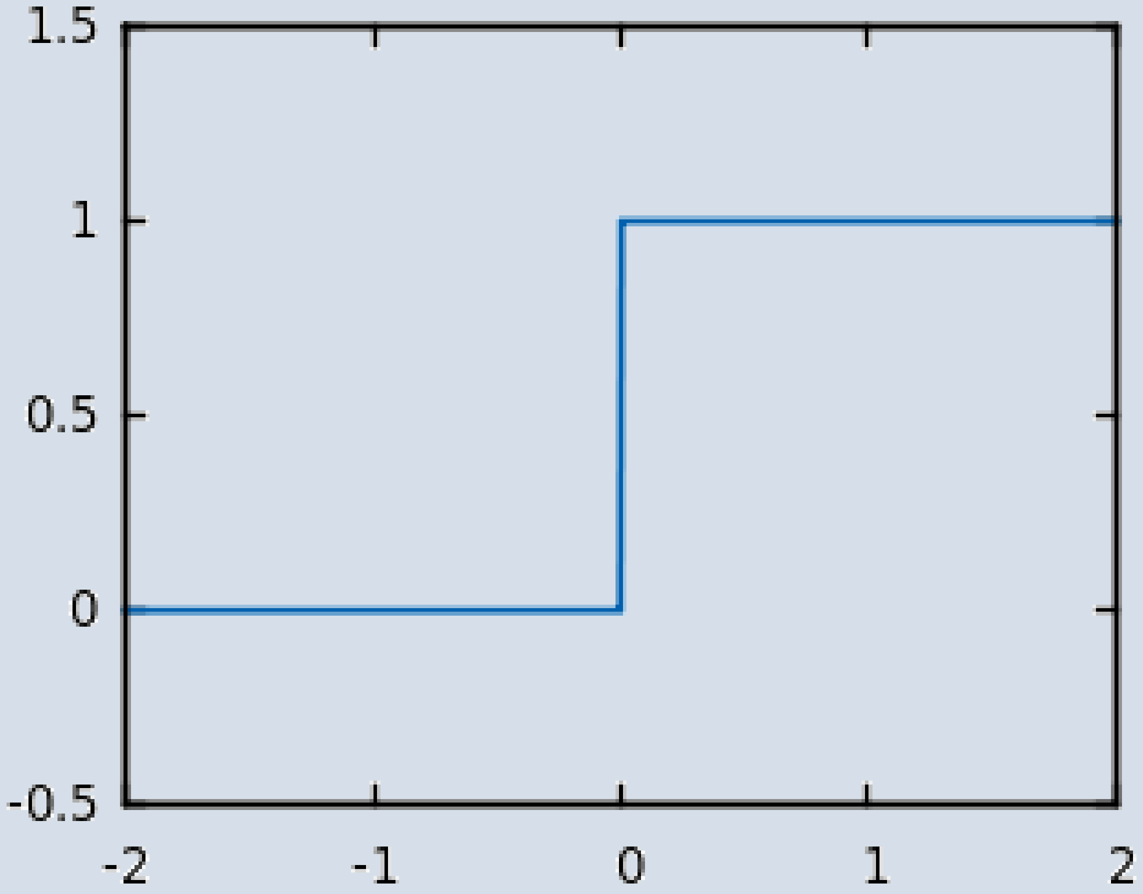
스마트인재개발원
Smart Human Resources Development

활성화 함수 (Activation Function)

활성화 함수(Activation Function)

- 신경망은 한 계층의 신호를 다음 계층으로 그대로 전달하지 않고 활성화 함수를 거친 후에 전달함
- 사람의 신경망 속 뉴런들도 모든 자극을 다 다음 뉴런으로 전달하는 것은 아니고 역치 이상의 자극만 전달하게 됨
- 활성화 함수는 이런 부분까지 사람과 유사하게 구현하여 사람처럼 사고하고 행동하는 인공지능 기술을 실현하기 위해 도입됨
- 또한 선형모델을 기반으로 하는 딥러닝 신경망에서 분류 문제를 해결하기 위해서 비선형 활성화 함수가 필요함

활성화 함수(Activation Function)



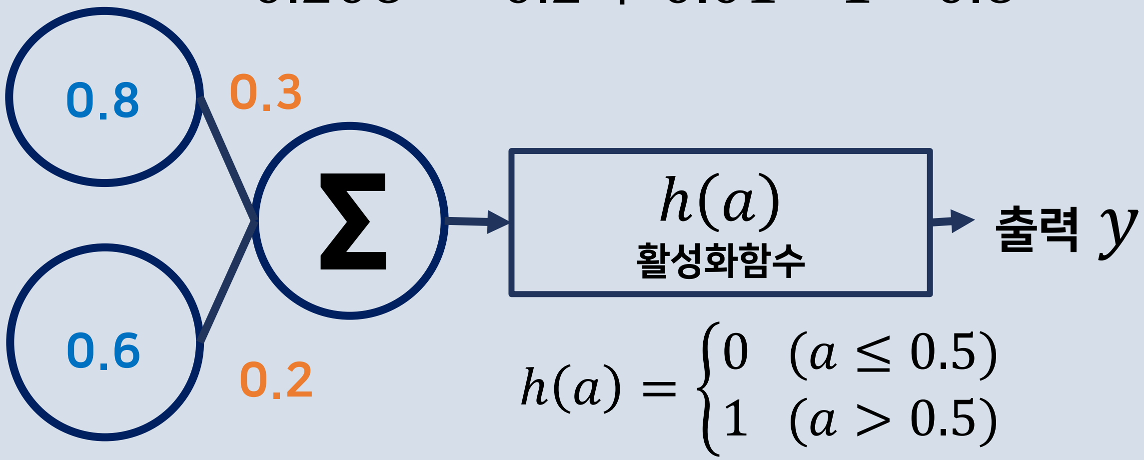
Step function(계단함수)

활성화 함수(Activation Function)

$$w_i := w_i + \alpha * (y - \hat{y}) * x_i$$

$$0.308 := 0.3 + 0.01 * 1 * 0.8$$
$$0.206 := 0.2 + 0.01 * 1 * 0.6$$

x1	x2
0.8	0.6
0.6	0.9
0.1	0.2
0.3	0.1
0.6	0.6
0.4	0.3
0.1	0.2



$$h(a) = \begin{cases} 0 & (a \leq 0.5) \\ 1 & (a > 0.5) \end{cases}$$

0.8 * 0.3 = 0.24
0.6 * 0.2 = 0.12 +

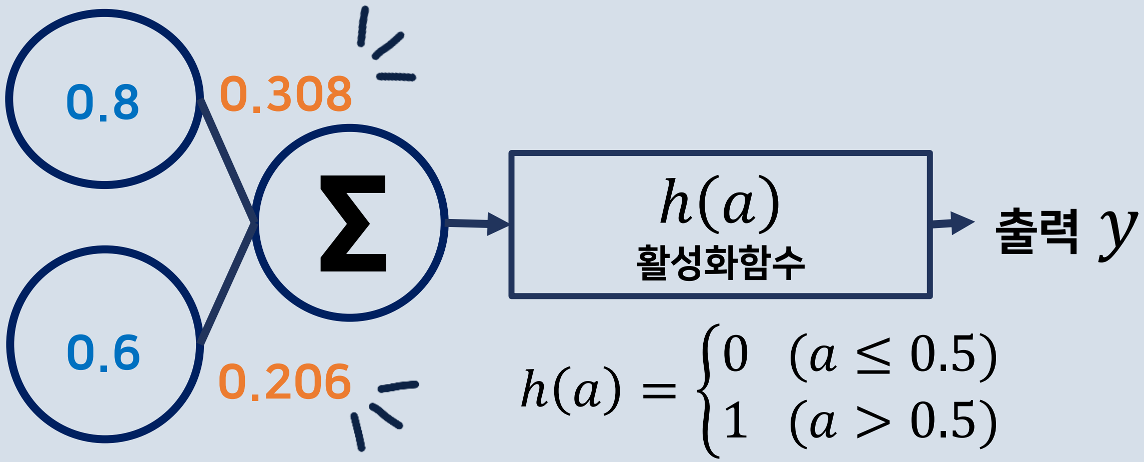
0.36

y	$\hat{y}$
1	0
1	
0	
0	
1	
0	
0	

활성화 함수(Activation Function)

$$w_i := w_i + \alpha * (y - \hat{y}) * x_i$$

x1	x2
0.8	0.6
0.6	0.9
0.1	0.2
0.3	0.1
0.6	0.6
0.4	0.3
0.1	0.2



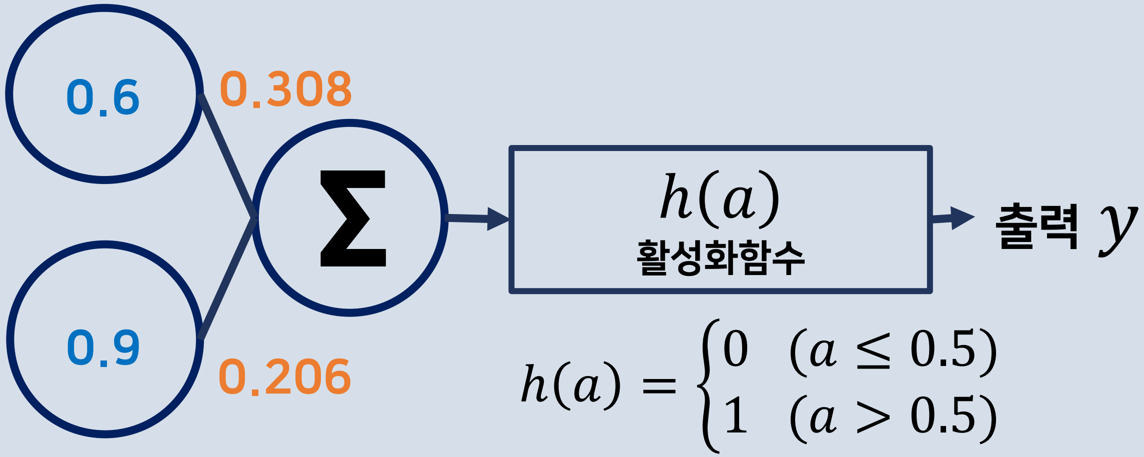
$$h(a) = \begin{cases} 0 & (a \le 0.5) \\ 1 & (a > 0.5) \end{cases}$$

y	$\hat{y}$
1	0
1	
0	
0	
1	
0	
0	

활성화 함수(Activation Function)

$$w_i \coloneqq w_i + \alpha * (y - \hat{y}) * x_i$$

x1	x2
0.8	0.6
0.6	0.9
0.1	0.2
0.3	0.1
0.6	0.6
0.4	0.3
0.1	0.2



$$h(a) = \begin{cases} 0 & (a \leq 0.5) \\ 1 & (a > 0.5) \end{cases}$$

0.6 * 0.308 = 0.1848
0.9 * 0.206 = 0.1854 +

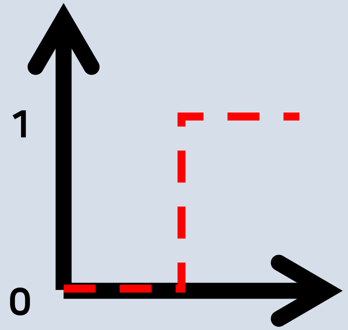
0.3702

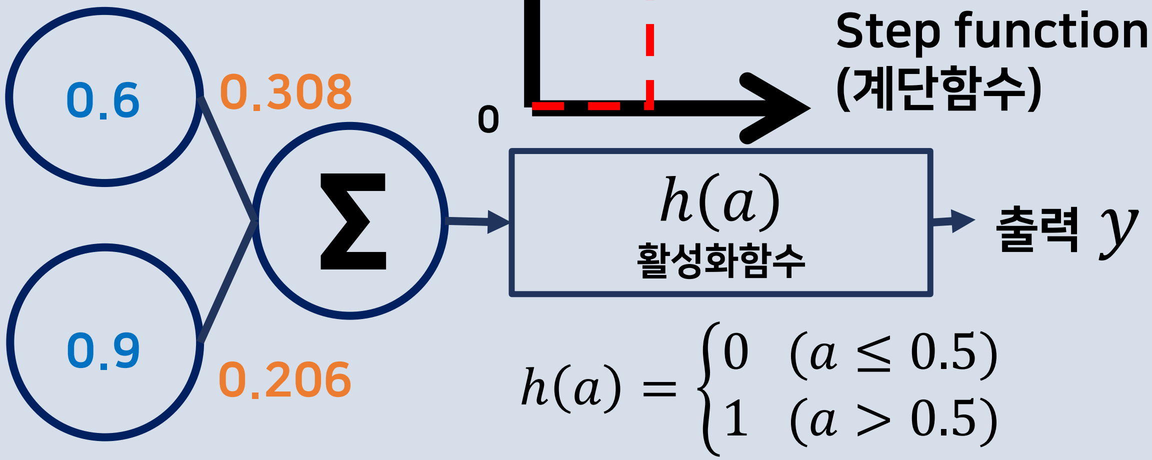
y	$\hat{y}$
1	0
1	
0	
0	
1	
0	
0	

활성화 함수(Activation Function)

$$w_i := w_i + \alpha * (y - \hat{y}) * x_i$$

x1	x2
0.8	0.6
0.6	0.9
0.1	0.2
0.3	0.1
0.6	0.6
0.4	0.3
0.1	0.2





$$h(a) = \begin{cases} 0 & (a \leq 0.5) \\ 1 & (a > 0.5) \end{cases}$$

0.6 * 0.308 = 0.1848
0.9 * 0.206 = 0.1854 +

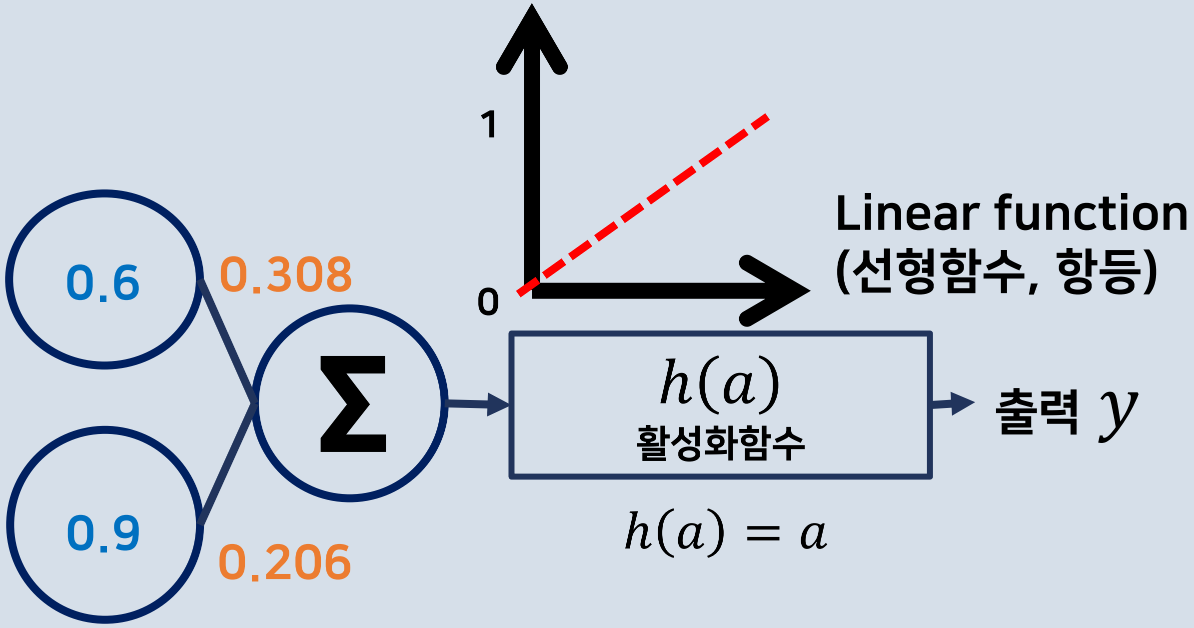
0.3702

y	$\hat{y}$
1	0
1	0
0	
0	
1	
0	
0	

퍼셉트론 학습

x1	x2
0.8	0.6
0.6	0.9
0.1	0.2
0.3	0.1
0.6	0.6
0.4	0.3
0.1	0.2

$$w_i := w_i + \alpha * (y - \hat{y}) * x_i$$

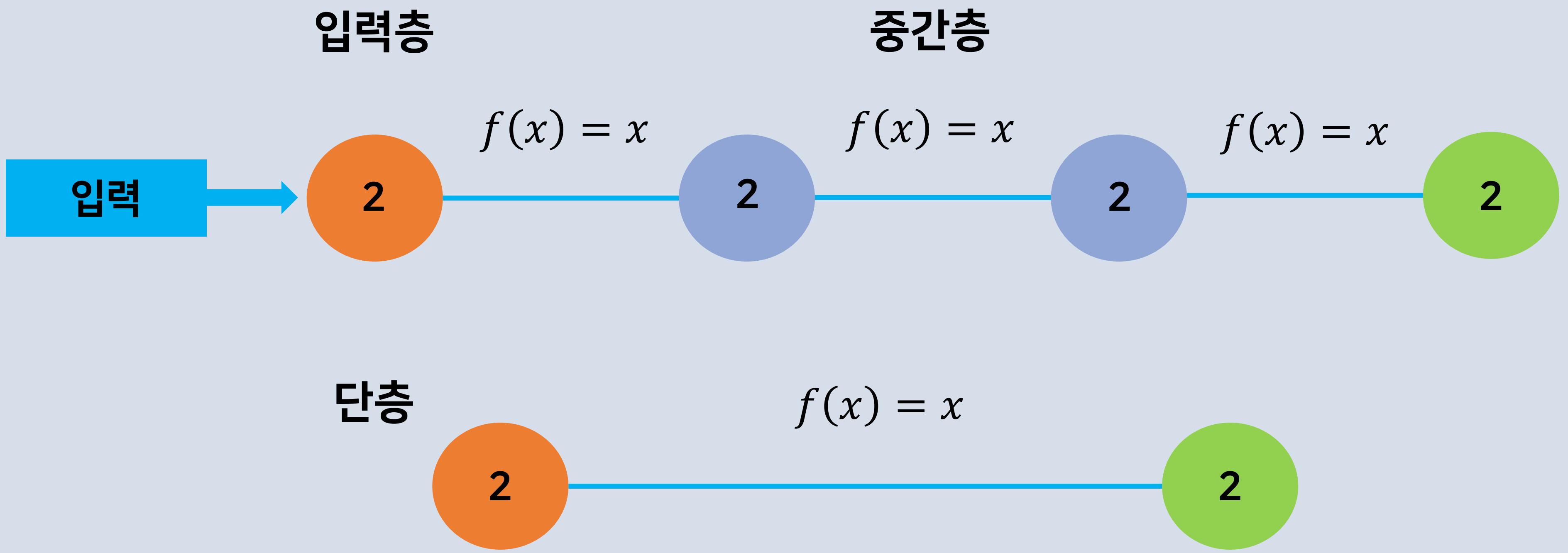


0.6 * 0.308 = 0.1848
0.9 * 0.206 = 0.1854 +

0.3702

y	$\hat{y}$
1	0
1	0
0	
0	
1	
0	
0	

중간층 활성화 함수로 선형함수(linear)를 사용하면 다층 구조의 효과를 살릴 수 없다.

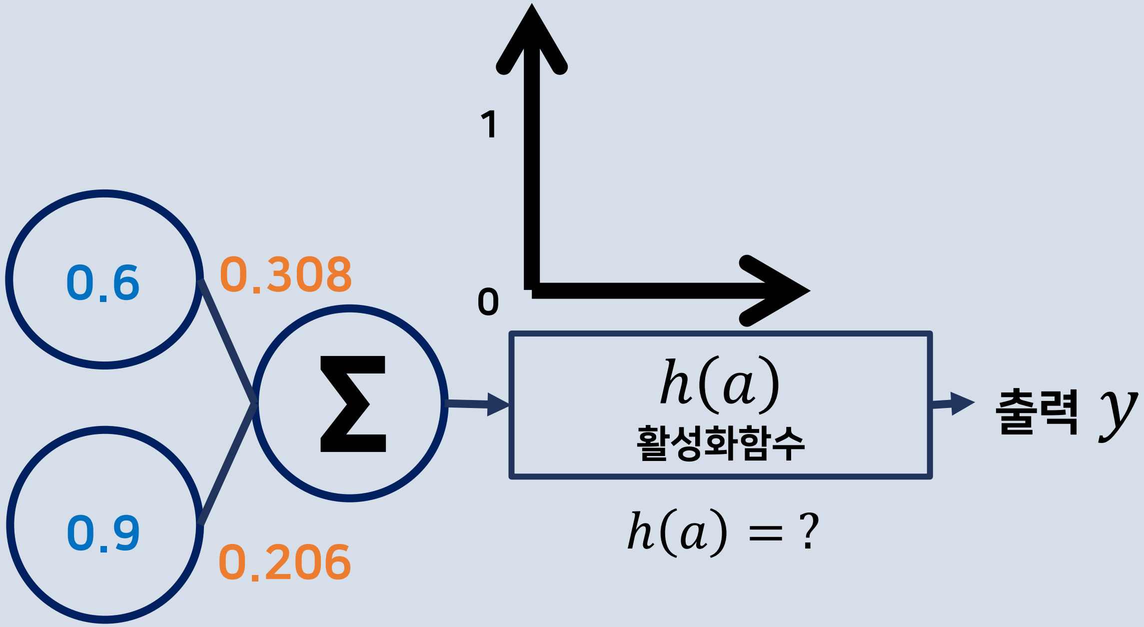


선형함수(linear, 항등함수) 수식은 h(x) = x

활성화 함수(Activation Function)

$$w_i := w_i + \alpha * (y - \hat{y}) * x_i$$

x1	x2
0.8	0.6
0.6	0.9
0.1	0.2
0.3	0.1
0.6	0.6
0.4	0.3
0.1	0.2



0.6 * 0.308 = 0.1848
0.9 * 0.206 = 0.1854 +

0.3702

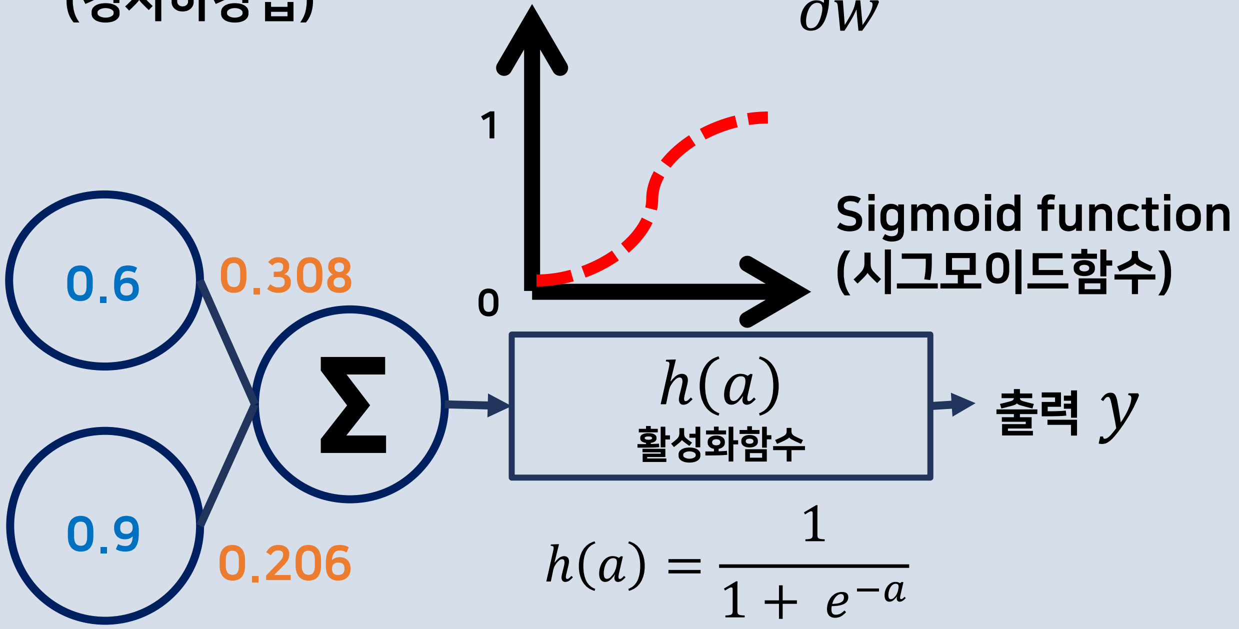
y	$\hat{y}$
1	0
1	0
0	
0	
1	
0	
0	

활성화 함수(Activation Function)

x1	x2
0.8	0.6
0.6	0.9
0.1	0.2
0.3	0.1
0.6	0.6
0.4	0.3
0.1	0.2

Gradient Descent
(경사하강법)

$$w := w - \alpha \frac{\partial MSE(w,b)}{\partial w}$$



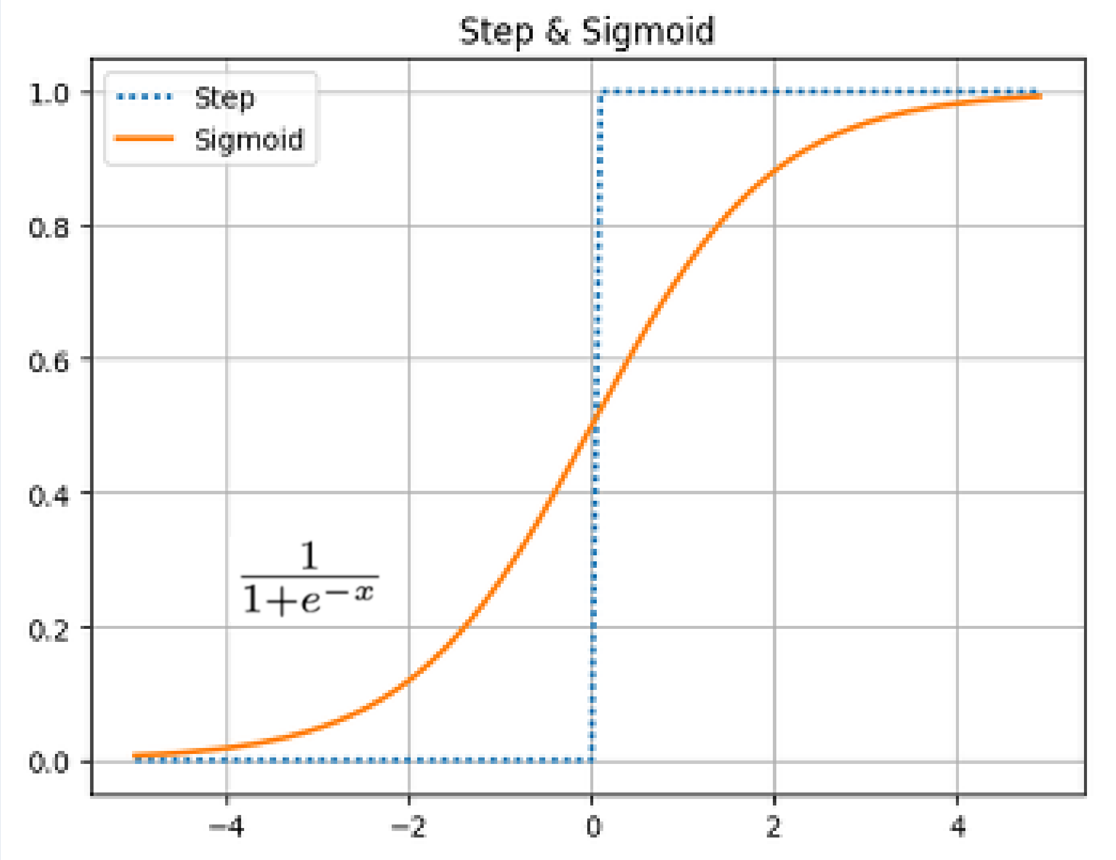
$$h(a) = \frac{1}{1 + e^{-a}}$$

0.6 * 0.308 = 0.1848
0.9 * 0.206 = 0.1854 +

0.3702

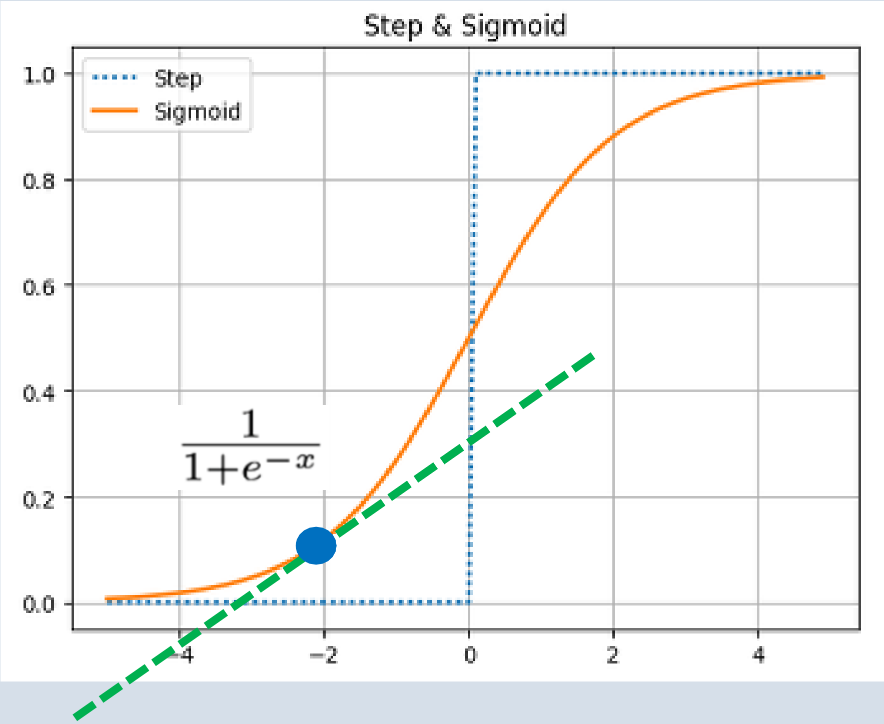
y	$\hat{y}$
1	0
1	0
0	
0	
1	
0	
0	

활성화 함수(Activation Function)

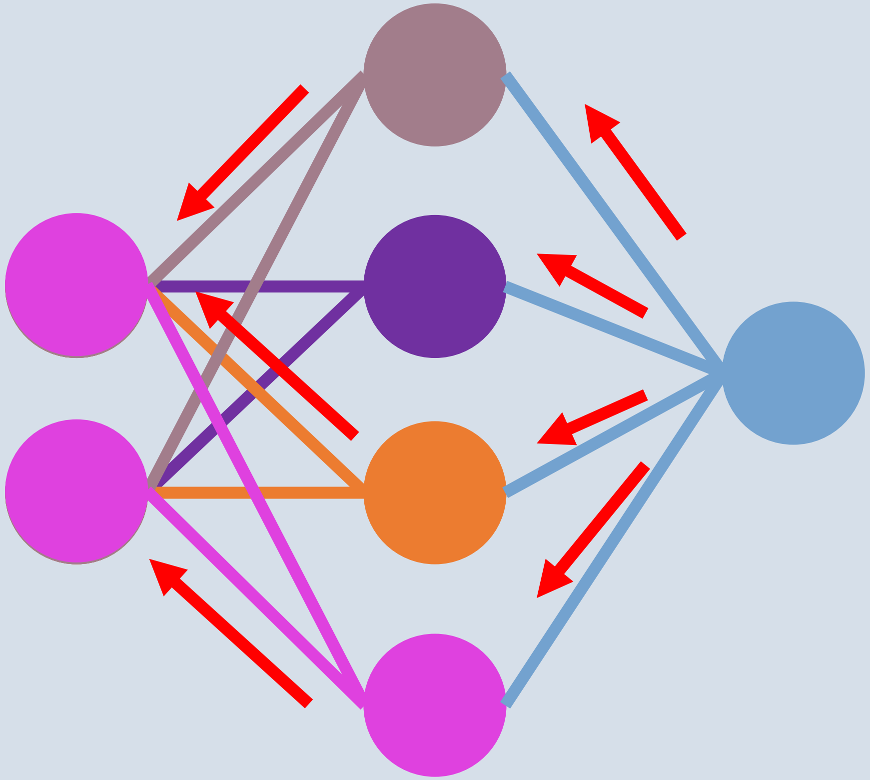


Sigmoid function
(시그모이드함수)

활성화 함수(Activation Function)



Sigmoid function
(시그모이드 함수)

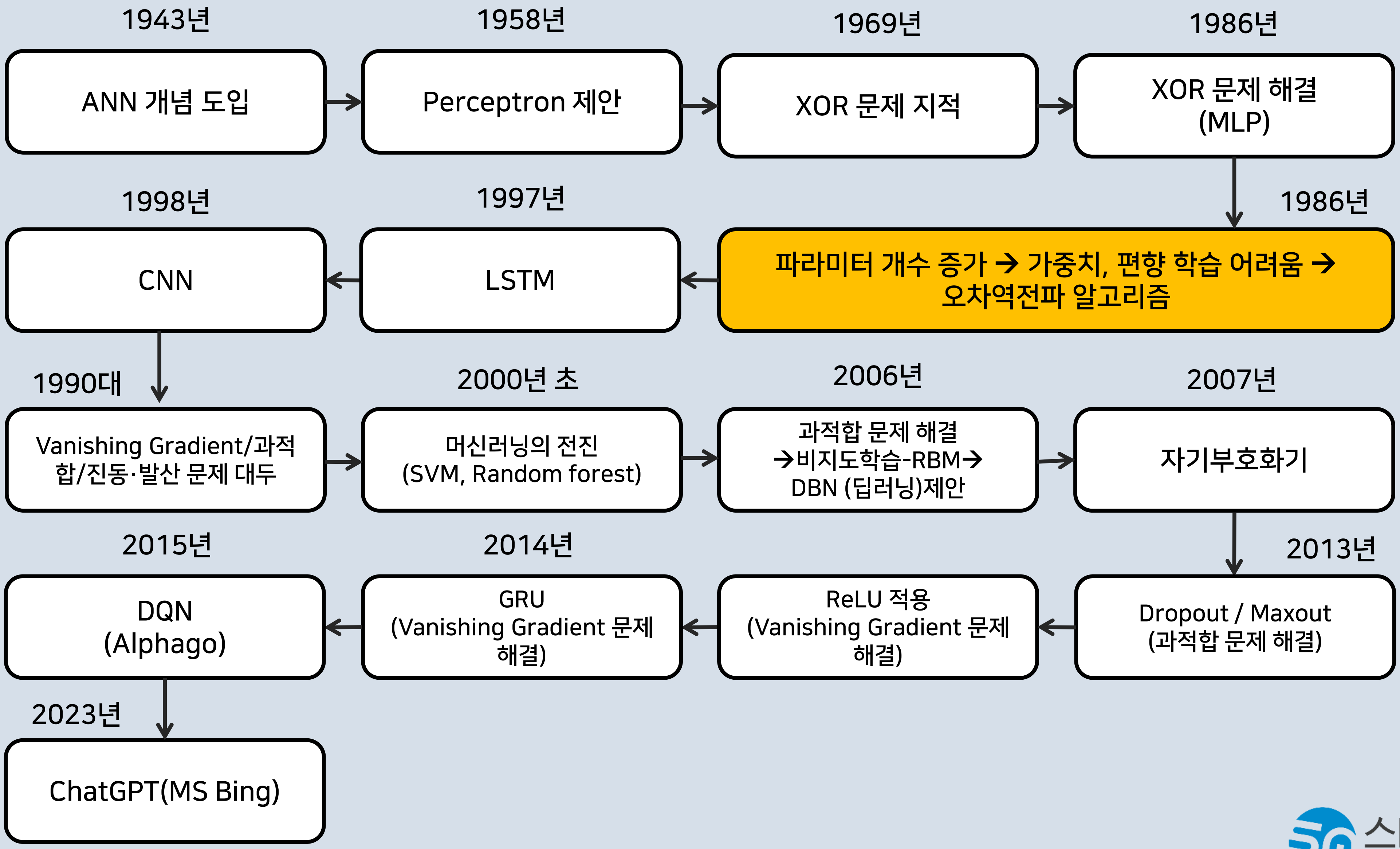


Back Propagation
(오차 역전파 알고리즘)

오차역전파 (Back Propagation)

Deep Learning

딥러닝 역사



스마트인재개발원
Smart Human Resources Development

딥러닝 역사

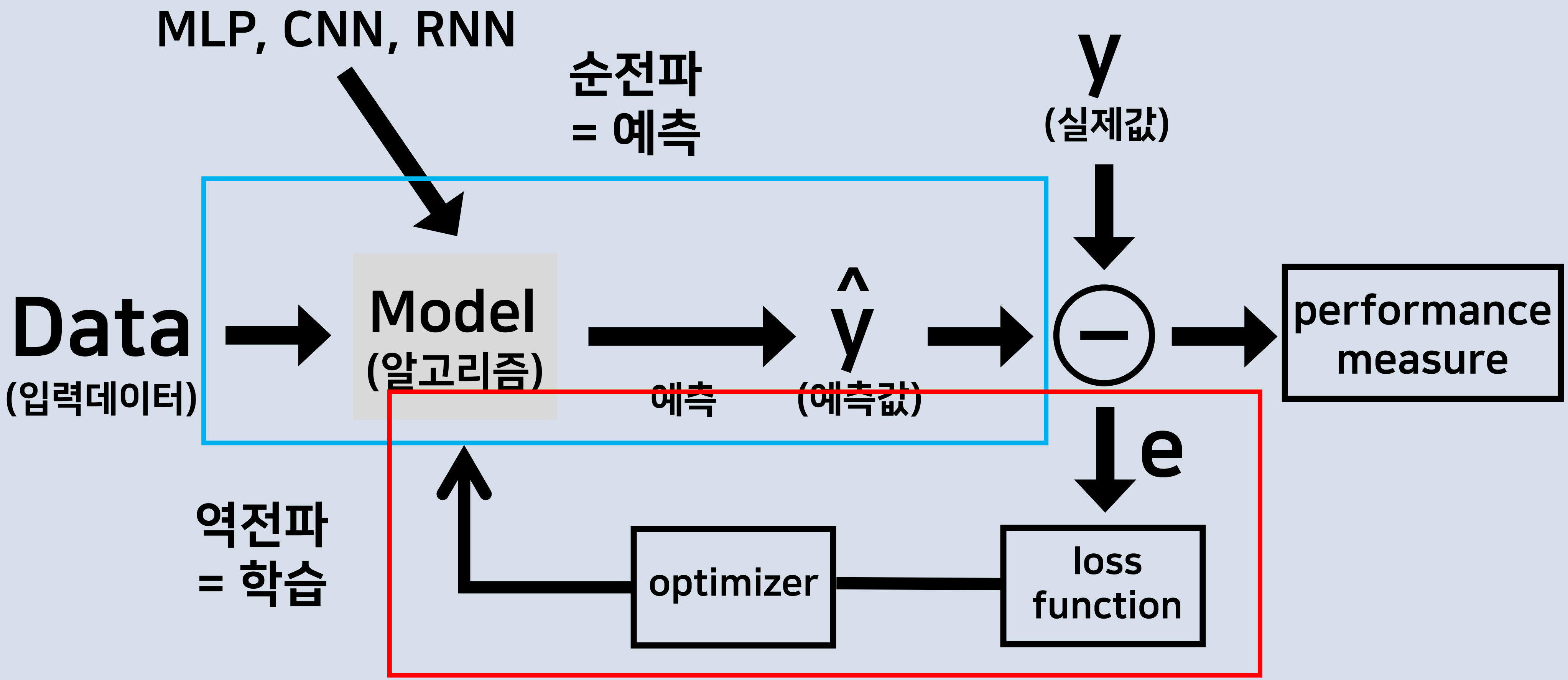
David Rumelhart
수리심리학자

Geoffrey Hinton
실험심리학자
컴퓨터과학자

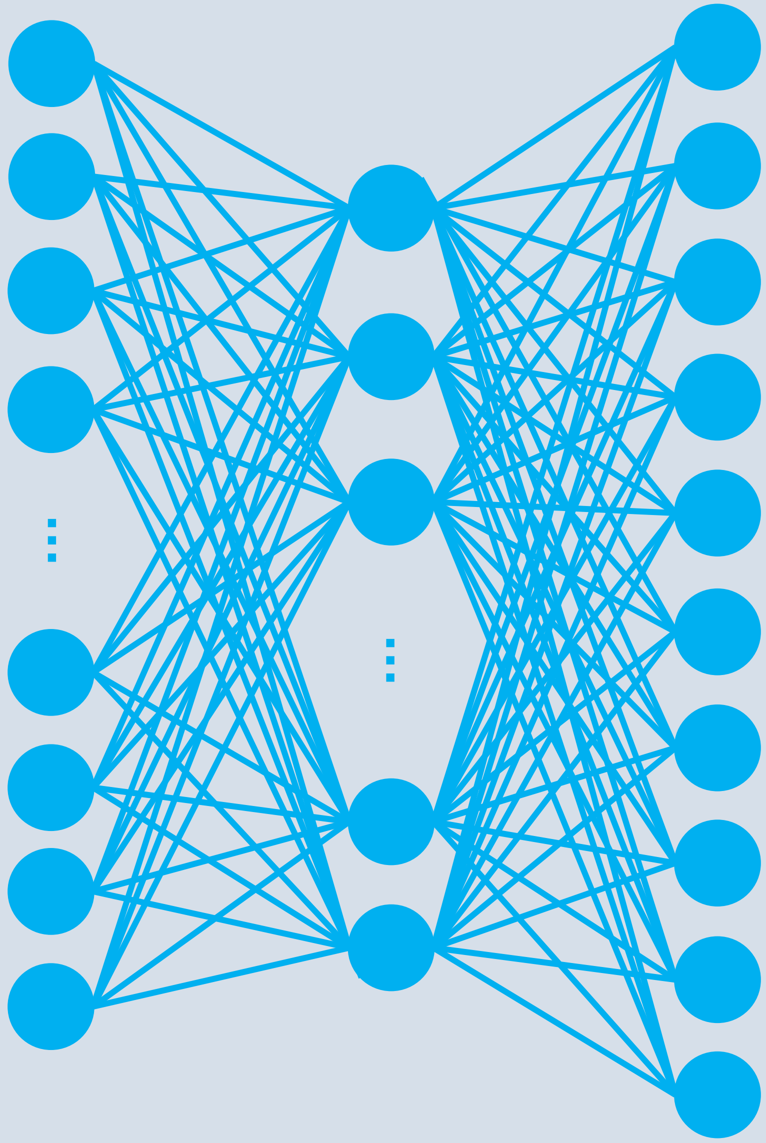
Ronald J. Williams
컴퓨터과학자

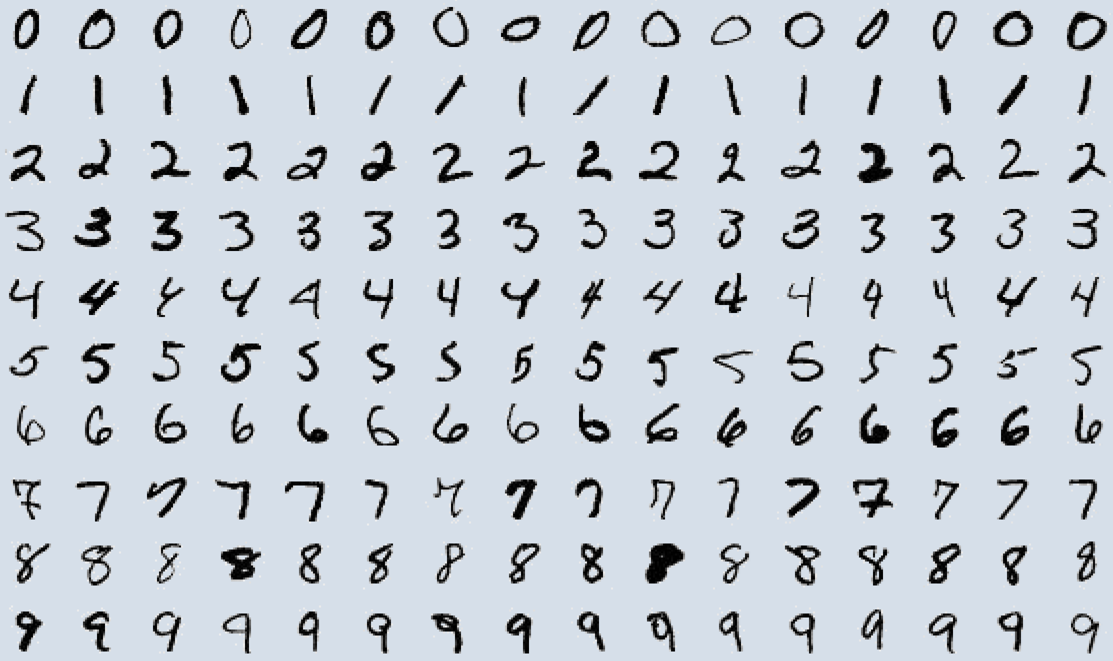
출처
https://www.psychologicalscience.org/observer/david-rumelhart
https://www.frontiersofknowledgeawards-fbbva.es/galardonados/geoffrey-hinton-2/
https://www.ccs.neu.edu/home/rjw/

오차역전파 (Back Propagation)



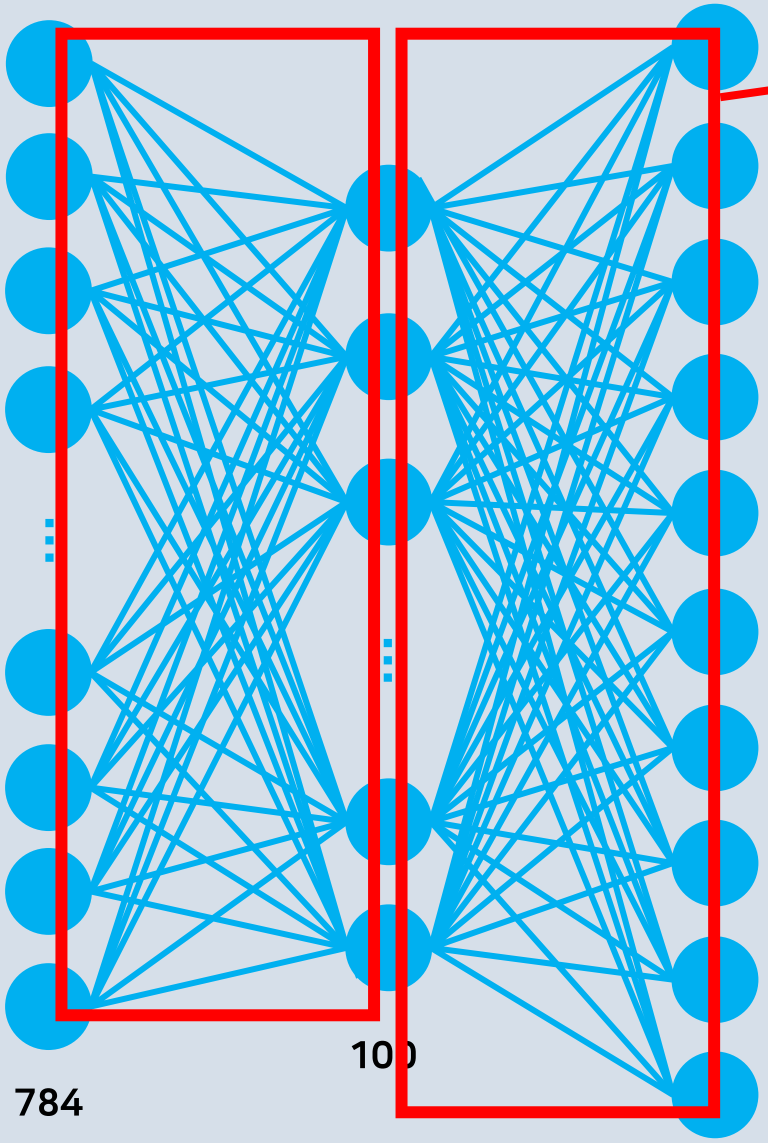
오차역전파 (Back Propagation)





출처
https://paperswithcode.com/dataset/mnist

오차역전파 (Back Propagation)_수치 미분의 한계



연결 가중치의 개수 : 784 * 100 + 100 * 10 = 79,400

편향 개수 : 100 + 10 = 110

한 개의 이미지 손실 계산 : 79,400번의 연산량 (가중치만 고려)

한 개의 파라미터를 업데이트 하기 위한 연산량 :

4,446,400,000 = 79,400 * 56,000

Train_size = 80%

모든 파라미터들를 업데이트 하기 위한 각각의 연산량 :

4,446,400,000 * 79,511 = 353,537,710,400,000

(=79,400 + 110 + 1)

3층 신경망 가중치를 한번 바꾸는데 걸리는 시간 :

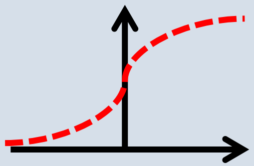
353,537,710,400,000 / 850,000,000

= 약 415,926 초 = 약 115h

오차역전파 (Back Propagation)

Learning rate(학습률, α) = 0.1

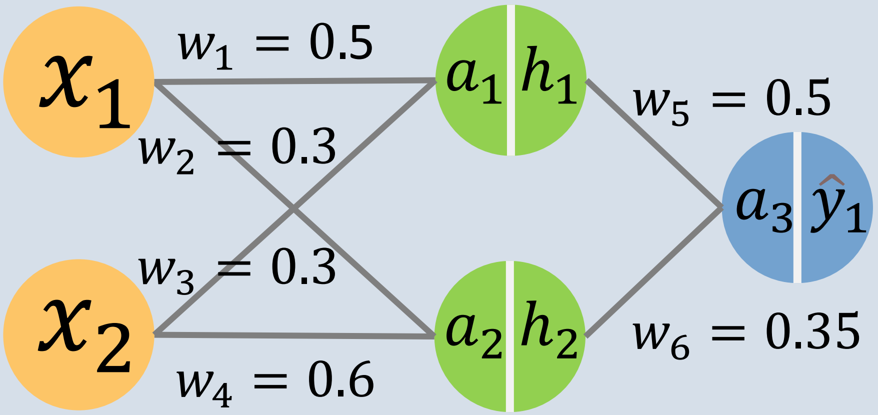
Activation function(활성화 함수) = $sigmoid()$



Loss function(손실 함수) = MSE

$y = 1$

$$E = \frac{1}{N}\sum_{i=1}^{n}(y_1 - \hat{y}_1)^2$$



1단계 : Forward Propagation

2단계 : 오차 계산

3단계 : Back Propagation

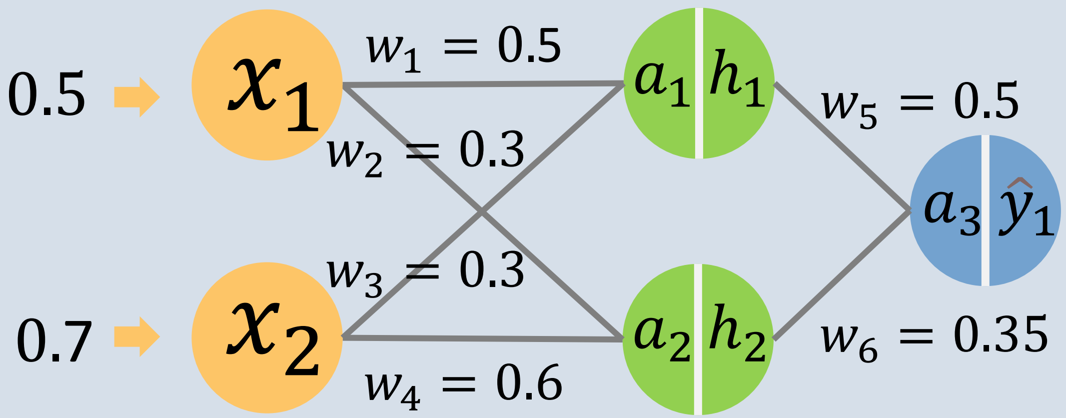
오차역전파 (Back Propagation)

Learning rate(학습률, α) = 0.1

Activation function(활성화 함수) = $sigmoid()$

Loss function(손실 함수) = MSE

$y = 1$



$$a_1 = w_1x_1 + w_3x_2 = 0.5 * 0.5 + 0.3 * 0.7 = 0.46$$

$$a_2 = w_2x_1 + w_4x_2 = 0.3 * 0.5 + 0.6 * 0.7 = 0.57$$

$$h_1 = sigmoid(a_1) \approx 0.613$$

$$h_2 = sigmoid(a_2) \approx 0.638$$

$$a_3 = w_5h_1 + w_6h_2 = 0.5 * 0.613 + 0.35 * 0.638 = 0.5298$$

$$\hat{y}_1 = sigmoid(a_3) \approx 0.629$$

1단계 : Forward Propagation

오차역전파 (Back Propagation)

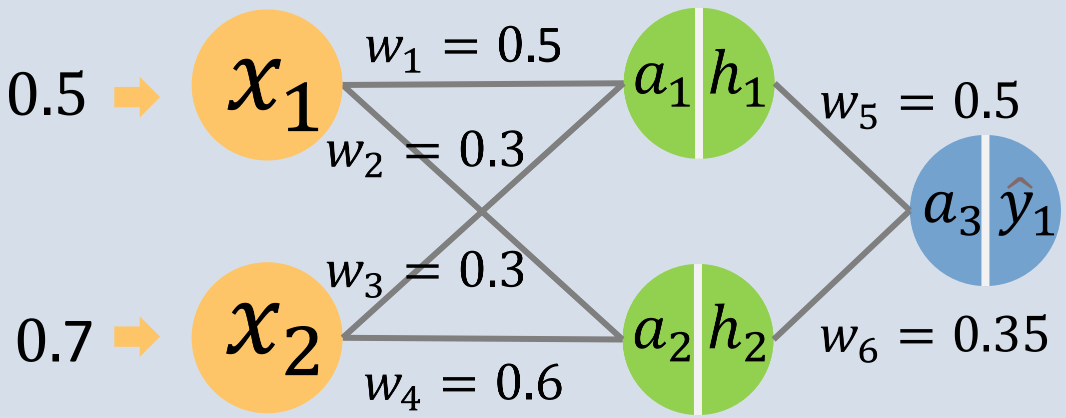
Learning rate(학습률, α) = 0.1

Activation function(활성화 함수) = $sigmoid()$

Loss function(손실 함수) = MSE

$y = 1$

$$E = \frac{1}{N}\sum_{i=1}^{n}(y_1 - \hat{y}_1)^2$$



$a_1 = 0.46$

$a_2 = 0.57$

$h_1 \approx 0.613$

$h_2 \approx 0.638$

$a_3 = 0.5298$

$\hat{y}_1 \approx 0.629$

2단계 : 오차 계산

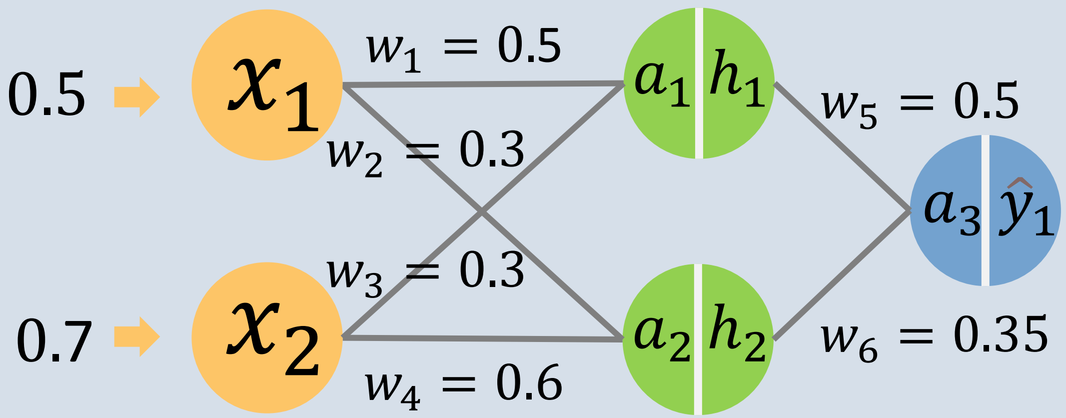
오차역전파 (Back Propagation)

Learning rate(학습률, α) = 0.1

Activation function(활성화 함수) = $sigmoid()$

Loss function(손실 함수) = MSE

$y = 1$



$a_1 = 0.46$

$a_2 = 0.57$

$h_1 \approx 0.613$

$h_2 \approx 0.638$

$a_3 = 0.5298$

$\hat{y}_1 \approx 0.629$

$$\mathrm{E} = \frac{1}{1}\sum_{i=1}^{1}(1 - 0.629)^2$$

$$= (1 - 0.629)^2 \approx 0.1376$$

2단계 : 오차 계산

오차역전파 (Back Propagation)

Learning rate(학습률, α) = 0.1

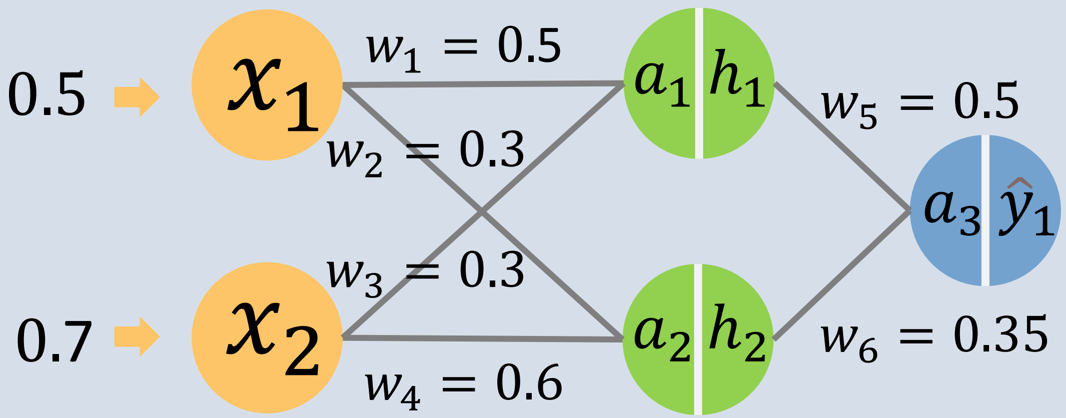
Activation function(활성화 함수) = $sigmoid()$

Loss function(손실 함수) = MSE

Chain rule $\dfrac{\partial E}{\partial w_5} = \dfrac{\partial E}{\partial \hat{y}_1} * \dfrac{\partial \hat{y}_1}{\partial a_3} * \dfrac{\partial a_3}{\partial w_5}$

$$w_5 := w_5 - \alpha \frac{\partial E}{\partial w_5}$$

$y = 1$



$a_1 = 0.46$

$a_2 = 0.57$

$h_1 \approx 0.613$

$h_2 \approx 0.638$

$a_3 = 0.5298$

$\hat{y}_1 \approx 0.629$

$\mathrm{E} \approx 0.1376$

3단계 : Back Propagation

체인룰 (Chain rule)

Chain rule $\frac{\partial E}{\partial w_5} = \frac{\partial E}{\partial \hat{y}_1} * \frac{\partial \hat{y}_1}{\partial a_3} * \frac{\partial a_3}{\partial w_5}$

사과는 오렌지보다 2배 빠르게 익고, → 사과의 익는 속도 = 2 * 오렌지의 속도
오렌지는 바나나보다 2배 빠르게 익고, → 오렌지의 익는 속도 = 2 * 바나나의 속도
바나나는 레몬보다 1.5배 빠르게 익는다고 할 때, → 바나나의 익는 속도 = 1.5 * 레몬의 속도
사과는 레몬보다 몇 배 더 빠르게 익는가?

사과의 익는 속도 = 2 * (2 * (1.5 * 레몬의 속도))

사과의 익는 속도 = 6 * 레몬의 속도

$$\frac{d\text{사과}}{d\text{레몬}} = \frac{d\text{사과}}{d\text{오렌지}} * \frac{d\text{오렌지}}{d\text{바나나}} * \frac{d\text{바나나}}{d\text{레몬}}$$

6 = 2 * 2 * 1.5

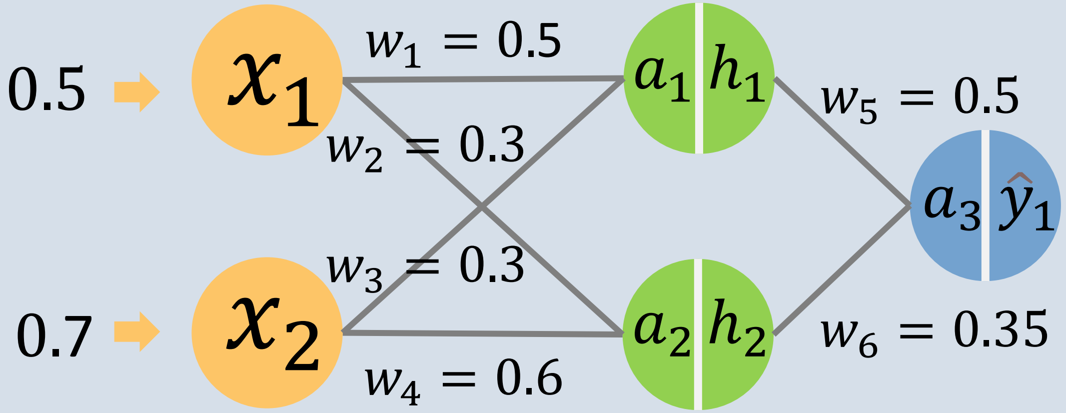
오차역전파 (Back Propagation)

Learning rate(학습률, α) = 0.1

Activation function(활성화 함수) = $sigmoid()$

Loss function(손실 함수) = MSE

$y = 1$



Chain rule $$\frac{\partial E}{\partial w_5} = \boxed{\frac{\partial E}{\partial \hat{y}_1}} * \frac{\partial \hat{y}_1}{\partial a_3} * \frac{\partial a_3}{\partial w_5}$$

$$w_5 := w_5 - \alpha \frac{\partial E}{\partial w_5}$$

$a_1 = 0.46$

$a_2 = 0.57$

$h_1 \approx 0.613$

$h_2 \approx 0.638$

$a_3 = 0.5298$

$\boxed{\hat{y}_1 \approx 0.629}$

$\mathrm{E} \approx 0.1376$

$$\mathrm{E} = \frac{1}{N}\sum_{i=1}^{n}(y_1 - \hat{y}_1)^2$$

$$\mathrm{E} = (y_1 - \hat{y}_1)^2$$

$$\frac{\partial E}{\partial \hat{y}_1} = -2(y_1 - \hat{y}_1)^{2-1}$$

3단계 : Back Propagation

오차역전파 (Back Propagation)

Learning rate(학습률, α) = 0.1

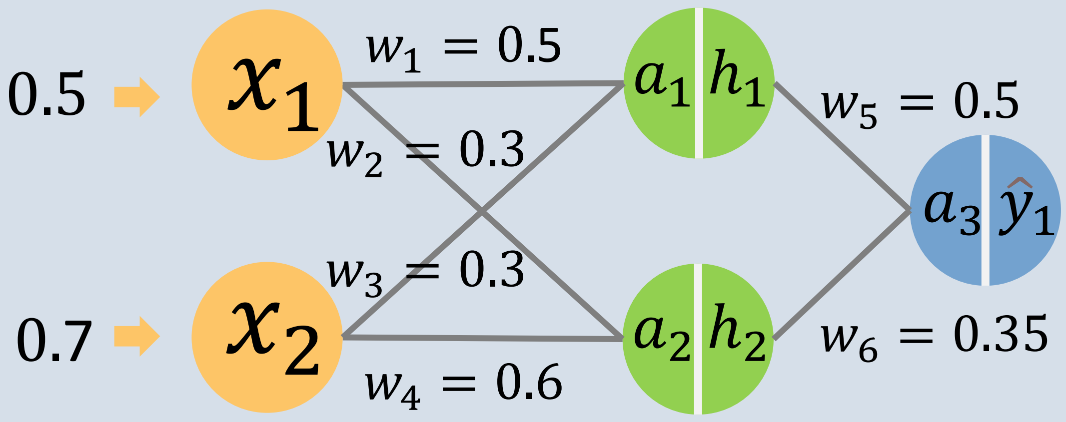
Activation function(활성화 함수) = $sigmoid()$

Loss function(손실 함수) = MSE

Chain rule $\dfrac{\partial E}{\partial w_5} = \boxed{\dfrac{\partial E}{\partial \hat{y}_1}} * \dfrac{\partial \hat{y}_1}{\partial a_3} * \dfrac{\partial a_3}{\partial w_5}$

$$w_5 := w_5 - \alpha \frac{\partial E}{\partial w_5}$$

$y = 1$



$a_1 = 0.46$

$a_2 = 0.57$

$h_1 \approx 0.613$

$h_2 \approx 0.638$

$a_3 = 0.5298$

$\boxed{\hat{y}_1 \approx 0.629}$

$\mathrm{E} \approx 0.1376$

$$\mathrm{E} = \frac{1}{N}\sum_{i=1}^{n}(y_1 - \hat{y}_1)^2$$

$$\mathrm{E} = (y_1 - \hat{y}_1)^2$$

$$\frac{\partial E}{\partial \hat{y}_1} = -2(1 - 0.629)^1 = -0.742$$

3단계 : Back Propagation

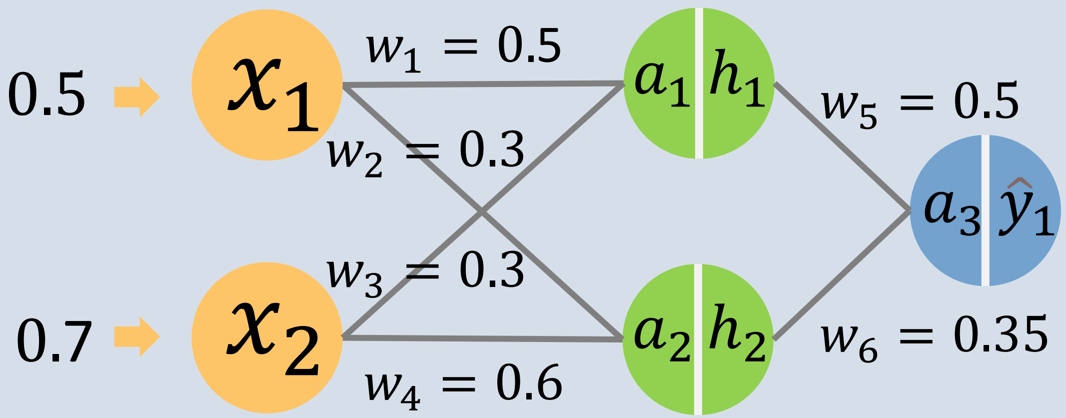
오차역전파 (Back Propagation)

Learning rate(학습률, α) = 0.1

Activation function(활성화 함수) = $sigmoid()$

Loss function(손실 함수) = MSE

$y = 1$



Chain rule $\dfrac{\partial E}{\partial w_5} = -0.742 * \dfrac{\partial \hat{y}_1}{\partial a_3} * \dfrac{\partial a_3}{\partial w_5}$

$$w_5 := w_5 - \alpha \frac{\partial E}{\partial w_5}$$

$a_1 = 0.46$

$a_2 = 0.57$

$h_1 \approx 0.613$

$h_2 \approx 0.638$

$a_3 = 0.5298$

$\boxed{\hat{y}_1 \approx 0.629}$

$\mathrm{E} \approx 0.1376$

$$\mathrm{E} = \frac{1}{N}\sum_{i=1}^{n}(y_1 - \hat{y}_1)^2$$

$$\mathrm{E} = (y_1 - \hat{y}_1)^2$$

$$\frac{\partial E}{\partial \hat{y}_1} = -2(1 - 0.629)^1 = -0.742$$

3단계 : Back Propagation

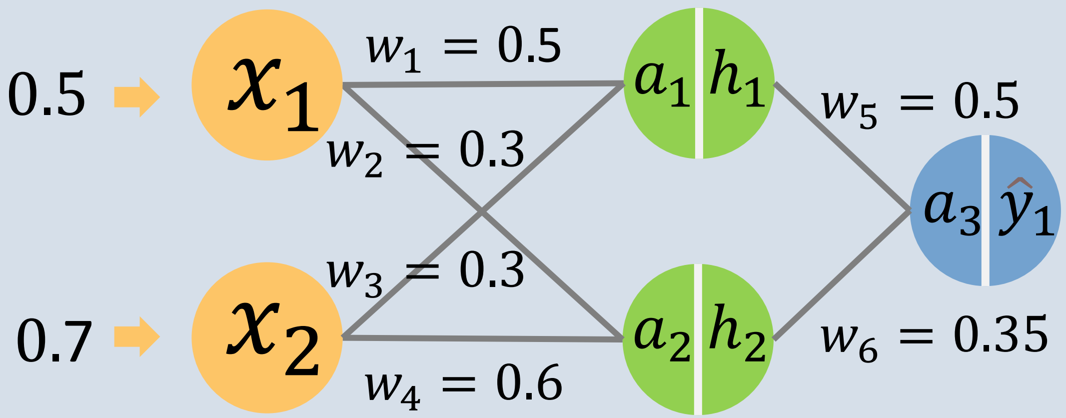
오차역전파 (Back Propagation)

Learning rate(학습률, α) = 0.1

Activation function(활성화 함수) = $sigmoid()$

Loss function(손실 함수) = MSE

$y = 1$



Chain rule $$\frac{\partial E}{\partial w_5} = -0.742 * \boxed{\frac{\partial \hat{y}_1}{\partial a_3}} * \frac{\partial a_3}{\partial w_5}$$

$$w_5 := w_5 - \alpha \frac{\partial E}{\partial w_5}$$

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

$a_1 = 0.46$

$a_2 = 0.57$

$h_1 \approx 0.613$

$h_2 \approx 0.638$

$\boxed{a_3 = 0.5298}$

$\hat{y}_1 \approx 0.629$

$\mathrm{E} \approx 0.1376$

$$\hat{y} = \frac{1}{1 + e^{-a}}$$

$$\frac{\partial \hat{y}}{\partial a} = \hat{y}(1 - \hat{y}) = 0.629 * 0.371 \approx 0.233$$

$$\hat{y}_1 = sigmoid(a_3) \approx 0.629$$

3단계 : Back Propagation

Deep Learning

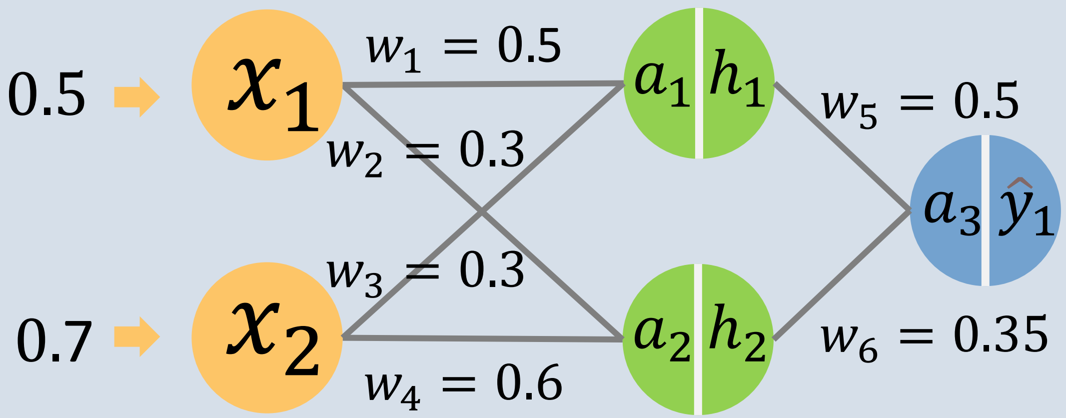
오차역전파 (Back Propagation)

Learning rate(학습률, α) = 0.1

Activation function(활성화 함수) = $sigmoid()$

Loss function(손실 함수) = MSE

$y = 1$



Chain rule $\dfrac{\partial E}{\partial w_5} = -0.742 * 0.233 * \boxed{\dfrac{\partial a_3}{\partial w_5}}$

$$w_5 := w_5 - \alpha \frac{\partial E}{\partial w_5}$$

$a_1 = 0.46$

$a_2 = 0.57$

$\boxed{h_1 \approx 0.613}$

$h_2 \approx 0.638$

$a_3 = 0.5298$

$\hat{y}_1 \approx 0.629$

$E \approx 0.1376$

$$a_3 = h_1 \cancel{w_5} + \cancel{h_2 w_6}$$

$$\frac{\partial a_3}{\partial w_5} = h_1 \approx 0.613$$

3단계 : Back Propagation

오차역전파 (Back Propagation)

Learning rate(학습률, α) = 0.1

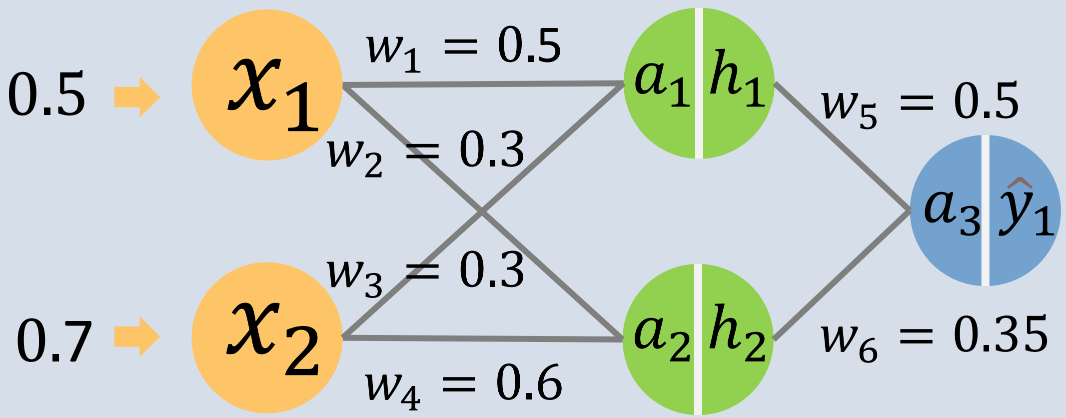
Activation function(활성화 함수) = $sigmoid()$

Loss function(손실 함수) = MSE

Chain rule $\frac{\partial E}{\partial w_5} = -0.742 * 0.233 * 0.613$

$$\approx -0.1059$$

$y = 1$



$a_1 = 0.46$

$a_2 = 0.57$

$h_1 \approx 0.613$

$h_2 \approx 0.638$

$a_3 = 0.5298$

$\hat{y}_1 \approx 0.629$

$$w_5 := w_5 - \alpha \frac{\partial E}{\partial w_5}$$

$\mathrm{E} \approx 0.1376$

3단계 : Back Propagation

오차역전파 (Back Propagation)

Learning rate(학습률, α) = 0.1

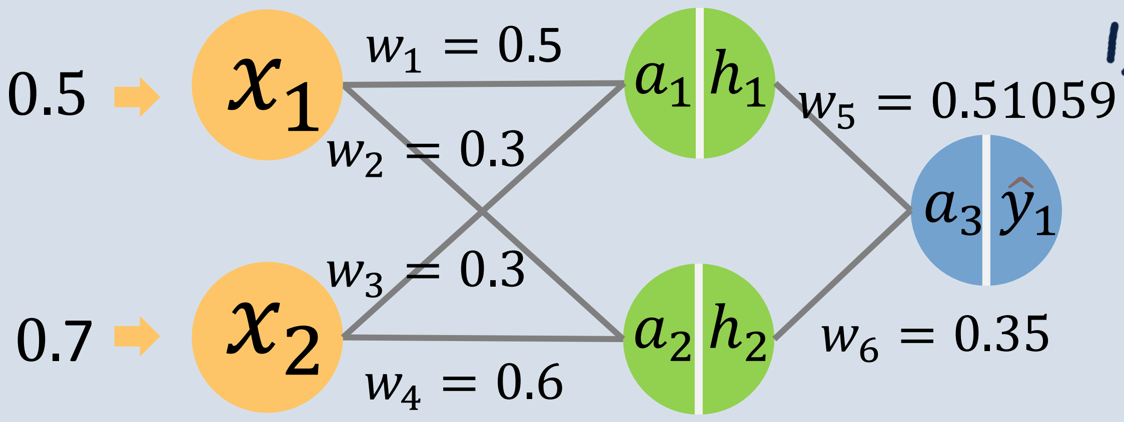
Activation function(활성화 함수) = $sigmoid()$

Loss function(손실 함수) = MSE

Chain rule $\dfrac{\partial E}{\partial w_5} = -0.742 * 0.233 * 0.613$

≈ -0.1059

$y = 1$



$a_1 = 0.46$

$a_2 = 0.57$

$h_1 \approx 0.613$

$h_2 \approx 0.638$

$a_3 = 0.5298$

$\hat{y}_1 \approx 0.629$

$\mathrm{E} \approx 0.1376$

$0.51059 = 0.5 - 0.1(-0.1059)$

3단계 : Back Propagation

오차역전파 (Back Propagation)

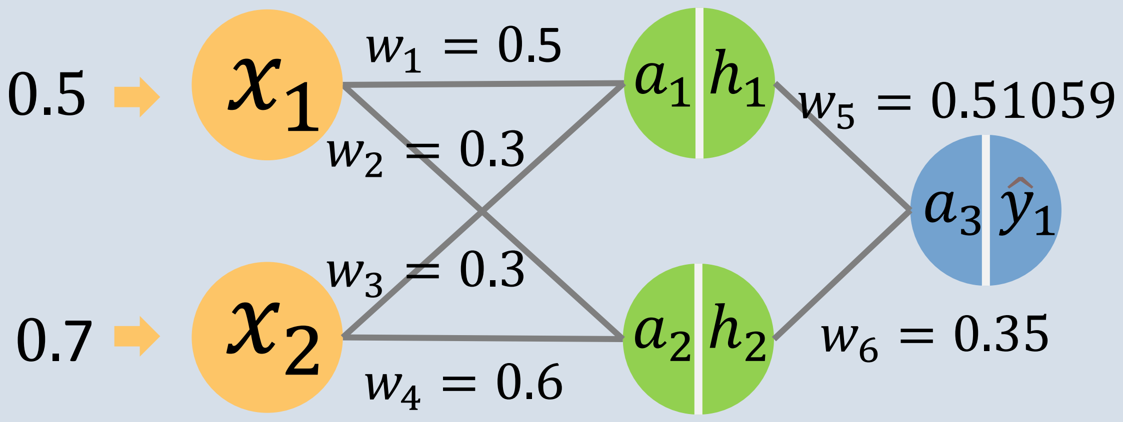
Learning rate(학습률, α) = 0.1

Activation function(활성화 함수) = $sigmoid()$

Loss function(손실 함수) = MSE

Chain rule $$\frac{\partial E}{\partial w_6} = \boxed{\frac{\partial E}{\partial \hat{y}_1} * \frac{\partial \hat{y}_1}{\partial a_3}} * \frac{\partial a_3}{\partial w_6}$$

$y = 1$



$a_1 = 0.46$

$a_2 = 0.57$

$h_1 \approx 0.613$

$h_2 \approx 0.638$

$a_3 = 0.5298$

$\hat{y}_1 \approx 0.629$

$\mathrm{E} \approx 0.1376$

3단계 : Back Propagation

오차역전파 (Back Propagation)

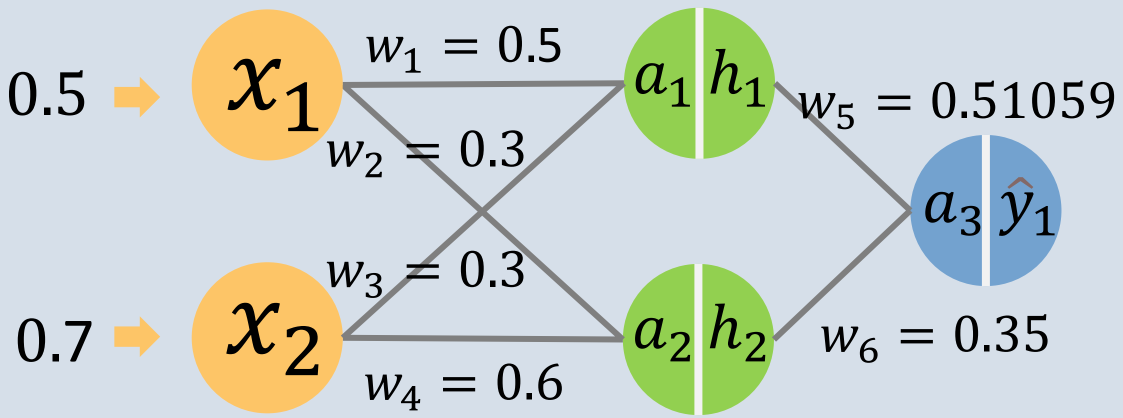
Learning rate(학습률, α) = 0.1

Activation function(활성화 함수) = $sigmoid()$

Loss function(손실 함수) = MSE

Chain rule $$\frac{\partial E}{\partial w_6} = \boxed{-0.742 * 0.233} * \frac{\partial a_3}{\partial w_6}$$

$y = 1$



$a_1 = 0.46$

$a_2 = 0.57$

$h_1 \approx 0.613$

$\boxed{h_2 \approx 0.638}$

$a_3 = 0.5298$

$\hat{y}_1 \approx 0.629$

$\mathrm{E} \approx 0.1376$

$$a_3 = \cancel{h_1 w_5} + h_2 \cancel{w_6}$$

$$\frac{\partial a_3}{\partial w_6} = h_2 \approx 0.638$$

3단계 : Back Propagation

오차역전파 (Back Propagation)

Learning rate(학습률, α) = 0.1

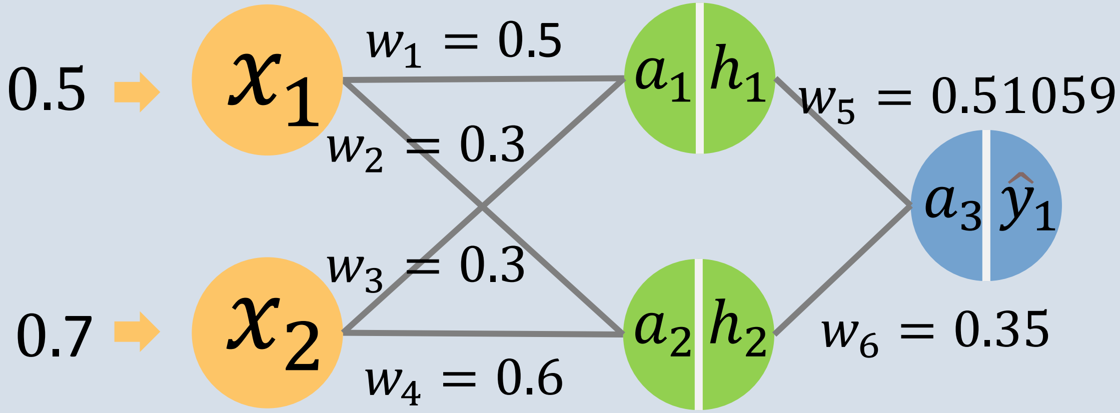
Activation function(활성화 함수) = $sigmoid()$

Loss function(손실 함수) = MSE

Chain rule $\frac{\partial E}{\partial w_6} = -0.742 * 0.233 * 0.638$

≈ -0.1103

$y = 1$



$a_1 = 0.46$

$a_2 = 0.57$

$h_1 \approx 0.613$

$h_2 \approx 0.638$

$a_3 = 0.5298$

$\hat{y}_1 \approx 0.629$

$\mathrm{E} \approx 0.1376$

$$w_6 := w_6 - \alpha \frac{\partial E}{\partial w_6}$$

3단계 : Back Propagation

오차역전파 (Back Propagation)

Deep Learning

Learning rate(학습률, α) = 0.1

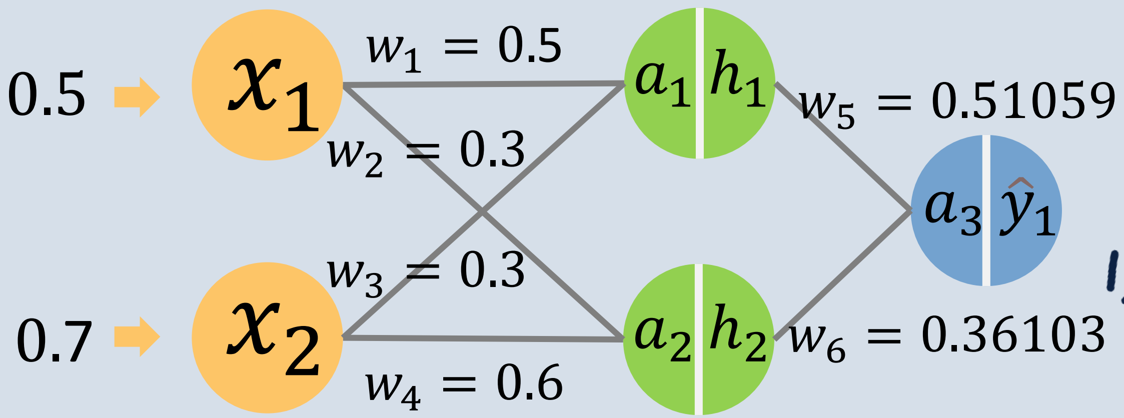
Activation function(활성화 함수) = $sigmoid()$

Loss function(손실 함수) = MSE

Chain rule $\frac{\partial E}{\partial w_6} = -0.742 * 0.233 * 0.638$

≈ -0.1103

$y = 1$



$a_1 = 0.46$

$a_2 = 0.57$

$h_1 \approx 0.613$

$h_2 \approx 0.638$

$a_3 = 0.5298$

$\hat{y}_1 \approx 0.629$

$0.36103 = 0.35 - 0.1(-0.1103)$

$\mathrm{E} \approx 0.1376$

3단계 : Back Propagation

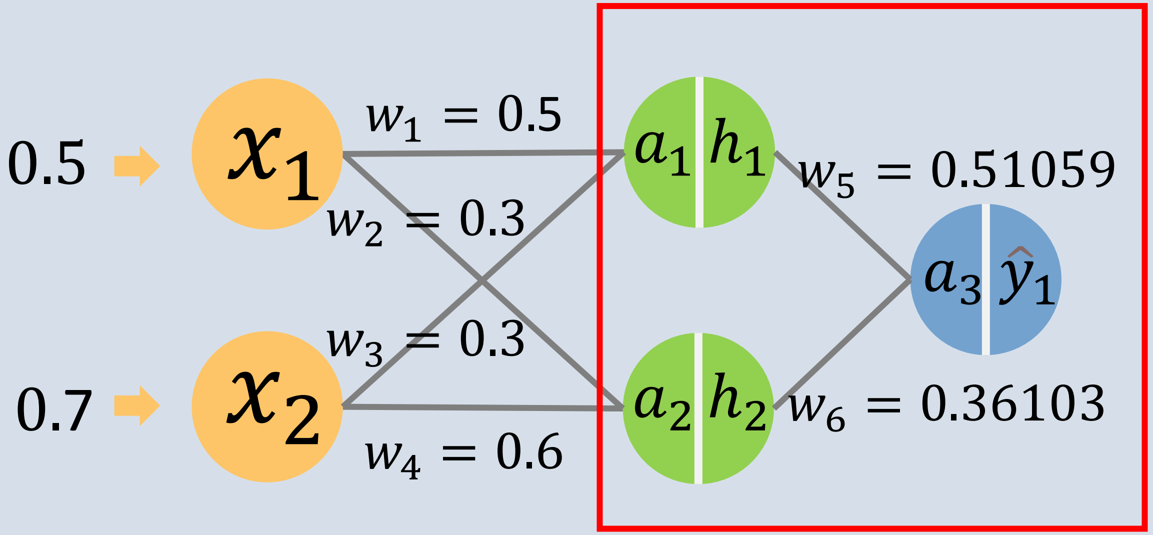
오차역전파 (Back Propagation)

Learning rate(학습률, α) = 0.1

Activation function(활성화 함수) = $sigmoid()$

Loss function(손실 함수) = MSE

$y = 1$



$a_1 = 0.46$

$a_2 = 0.57$

$h_1 \approx 0.613$

$h_2 \approx 0.638$

$a_3 = 0.5298$

$\hat{y}_1 \approx 0.629$

$\mathrm{E} \approx 0.1376$

3단계 : Back Propagation

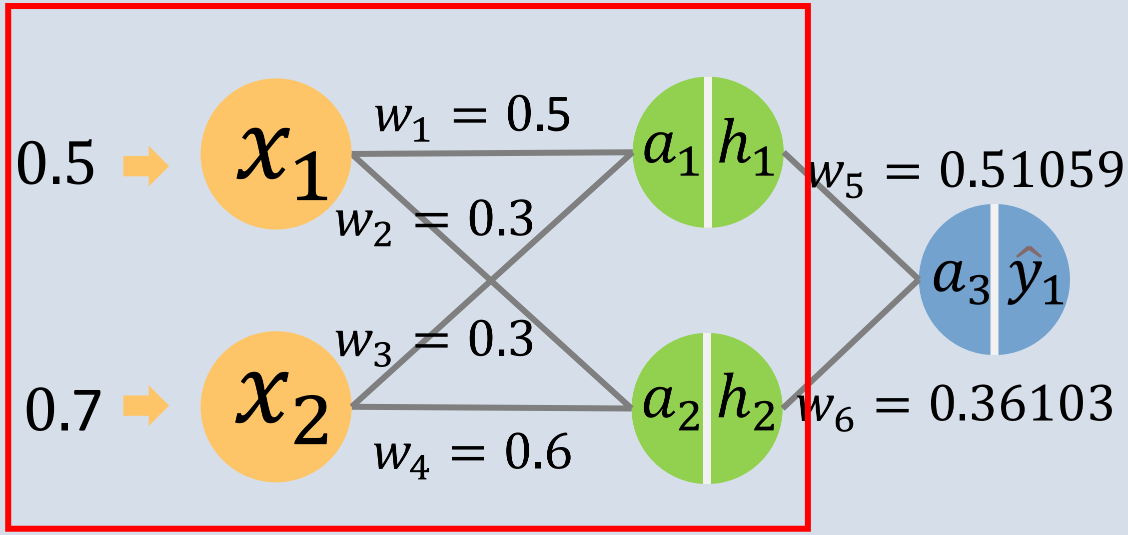
오차역전파 (Back Propagation)

Learning rate(학습률, α) = 0.1

Activation function(활성화 함수) = $sigmoid()$

Loss function(손실 함수) = MSE

$y = 1$



$a_1 = 0.46$

$a_2 = 0.57$

$h_1 \approx 0.613$

$h_2 \approx 0.638$

$a_3 = 0.5298$

$\hat{y}_1 \approx 0.629$

$\mathrm{E} \approx 0.1376$

3단계 : Back Propagation

오차역전파 (Back Propagation)

Learning rate(학습률, α) = 0.1

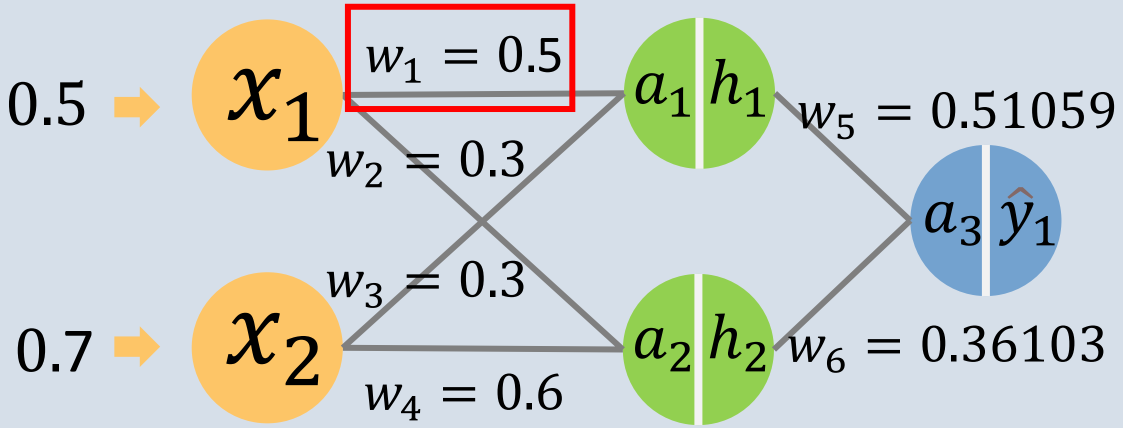
Activation function(활성화 함수) = $sigmoid()$

Loss function(손실 함수) = MSE

$$\frac{\partial E}{\partial w_1} = \boxed{\frac{\partial E}{\partial \hat{y}_1} * \frac{\partial \hat{y}_1}{\partial a_3}} * \frac{\partial a_3}{\partial h_1} * \frac{\partial h_1}{\partial a_1} * \frac{\partial a_1}{\partial w_1}$$

$$-0.742 * 0.233$$

$y = 1$



$a_1 = 0.46$

$a_2 = 0.57$

$h_1 \approx 0.613$

$h_2 \approx 0.638$

$a_3 = 0.5298$

$\hat{y}_1 \approx 0.629$

$\mathrm{E} \approx 0.1376$

3단계 : Back Propagation

오차역전파 (Back Propagation)

Learning rate(학습률, α) = 0.1

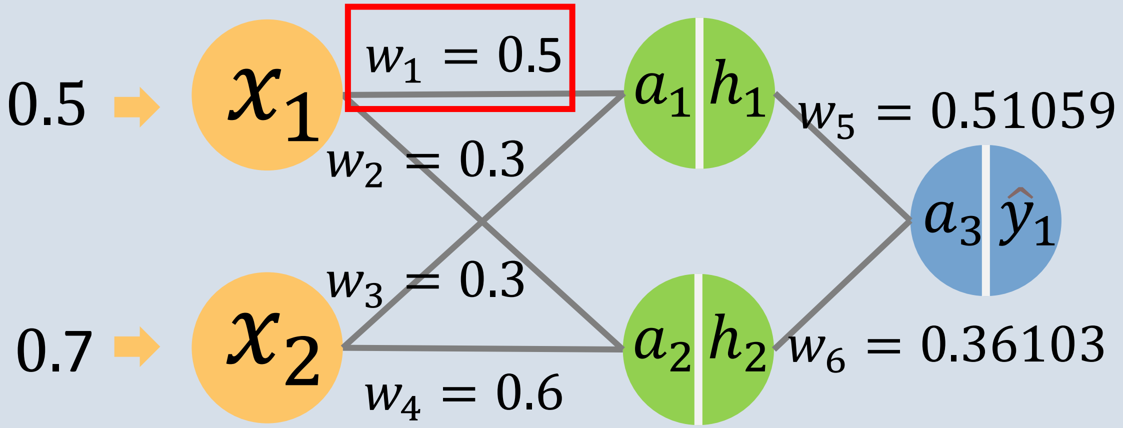
Activation function(활성화 함수) = $sigmoid()$

Loss function(손실 함수) = MSE

$$\frac{\partial E}{\partial w_1} = \frac{\partial E}{\partial \hat{y}_1} * \frac{\partial \hat{y}_1}{\partial a_3} * \boxed{\frac{\partial a_3}{\partial h_1}} * \frac{\partial h_1}{\partial a_1} * \frac{\partial a_1}{\partial w_1}$$

$$-0.742 * 0.233 * 0.5$$

$$y = 1$$



$a_1 = 0.46$

$a_2 = 0.57$

$h_1 \approx 0.613$

$h_2 \approx 0.638$

$a_3 = 0.5298$

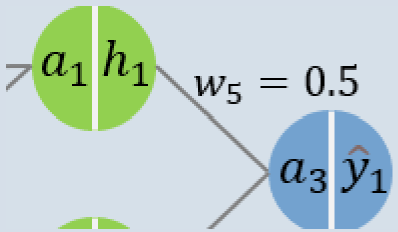
$\hat{y}_1 \approx 0.629$

$\mathrm{E} \approx 0.1376$

$$a_3 = \cancel{h_1} w_5 + \cancel{h_2 w_6}$$

$$\frac{\partial a_3}{\partial h_1} = w_5 = 0.5$$

※수정 되기 전 w_5 의 값



3단계 : Back Propagation

오차역전파 (Back Propagation)

Learning rate(학습률, α) = 0.1

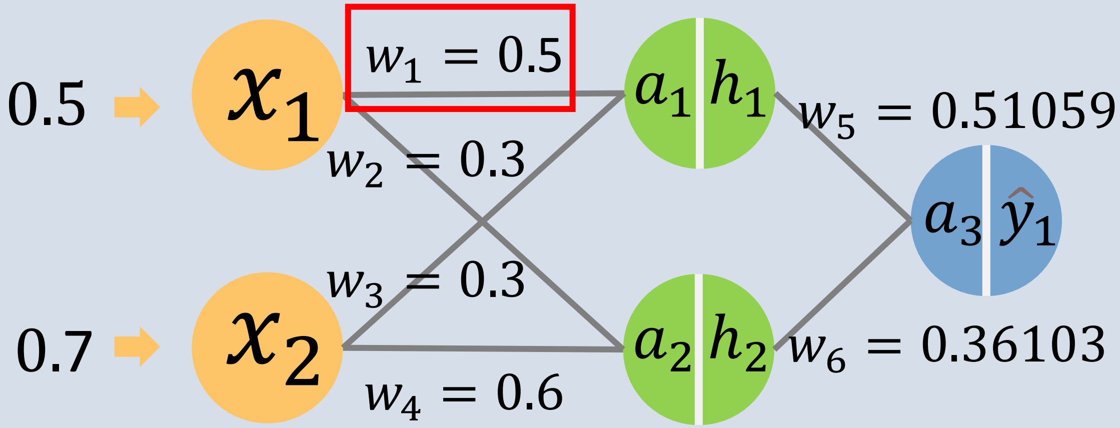
Activation function(활성화 함수) = $sigmoid()$

Loss function(손실 함수) = MSE

$$\frac{\partial E}{\partial w_1} = \frac{\partial E}{\partial \hat{y}_1} * \frac{\partial \hat{y}_1}{\partial a_3} * \frac{\partial a_3}{\partial h_1} * \boxed{\frac{\partial h_1}{\partial a_1}} * \frac{\partial a_1}{\partial w_1}$$

$$-0.742 * 0.233 * 0.5 \ * 0.237$$

$y = 1$



$a_1 = 0.46$

$a_2 = 0.57$

$\boxed{h_1 \approx 0.613}$

$h_2 \approx 0.638$

$a_3 = 0.5298$

$\hat{y}_1 \approx 0.629$

$\mathrm{E} \approx 0.1376$

$$\frac{\partial h_1}{\partial a_1} = h_1(1 - h_1) = 0.613 * (1 - 0.613) \approx 0.237$$

$$h_1 = sigmoid(a_1) \approx 0.613$$

3단계 : Back Propagation

스마트인재개발원
Smart Human Resources Development

오차역전파 (Back Propagation)

Learning rate(학습률, α) = 0.1

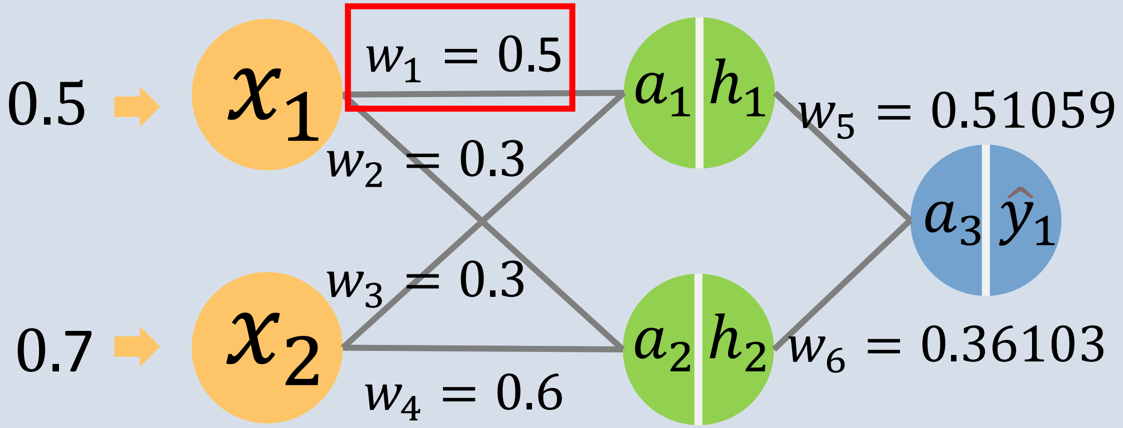
Activation function(활성화 함수) = $sigmoid()$

Loss function(손실 함수) = MSE

$$y = 1$$

$$\frac{\partial E}{\partial w_1} = \frac{\partial E}{\partial \hat{y}_1} * \frac{\partial \hat{y}_1}{\partial a_3} * \frac{\partial a_3}{\partial h_1} * \frac{\partial h_1}{\partial a_1} * \boxed{\frac{\partial a_1}{\partial w_1}}$$

$$-0.742 * 0.233 * 0.5 * 0.237 * 0.5$$



$$a_1 = 0.46$$

$$a_2 = 0.57$$

$$h_1 \approx 0.613$$

$$h_2 \approx 0.638$$

$$a_3 = 0.5298$$

$$\hat{y}_1 \approx 0.629$$

$$\mathrm{E} \approx 0.1376$$

$$a_1 = x_1 \cancel{w_1} + \cancel{x_2 w_3}$$

$$\frac{\partial a_1}{\partial w_1} = x_1 = 0.5$$

3단계 : Back Propagation

오차역전파 (Back Propagation)

Learning rate(학습률, α) = 0.1

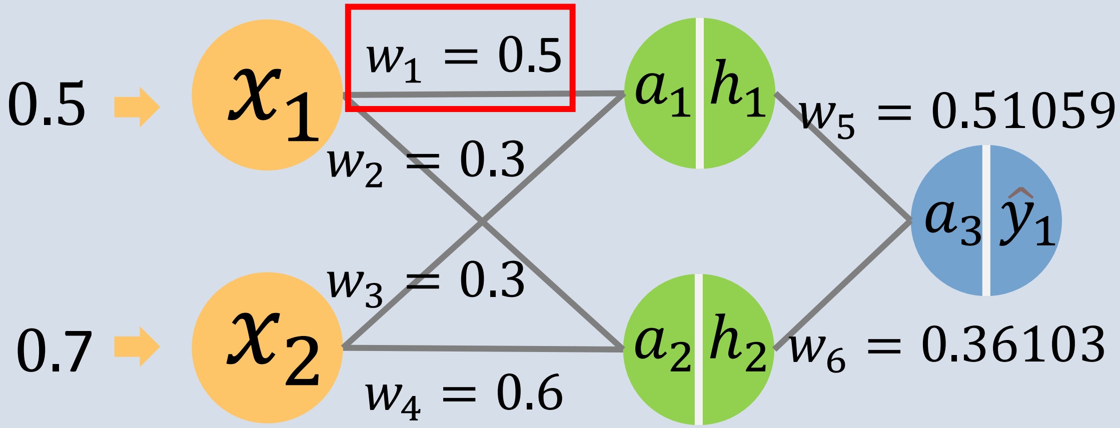
Activation function(활성화 함수) = $sigmoid()$

Loss function(손실 함수) = MSE

$$\frac{\partial E}{\partial w_1} = \frac{\partial E}{\partial \hat{y}_1} * \frac{\partial \hat{y}_1}{\partial a_3} * \frac{\partial a_3}{\partial h_1} * \frac{\partial h_1}{\partial a_1} * \frac{\partial a_1}{\partial w_1}$$

$$-0.742 * 0.233 * 0.5 * 0.237 * 0.5 \approx -0.0102$$

$y = 1$



$a_1 = 0.46$

$a_2 = 0.57$

$h_1 \approx 0.613$

$h_2 \approx 0.638$

$a_3 = 0.5298$

$\hat{y}_1 \approx 0.629$

$$w_1 := w_1 - \alpha \frac{\partial E}{\partial w_1}$$

$\mathrm{E} \approx 0.1376$

3단계 : Back Propagation

오차역전파 (Back Propagation)

Learning rate(학습률, α) = 0.1

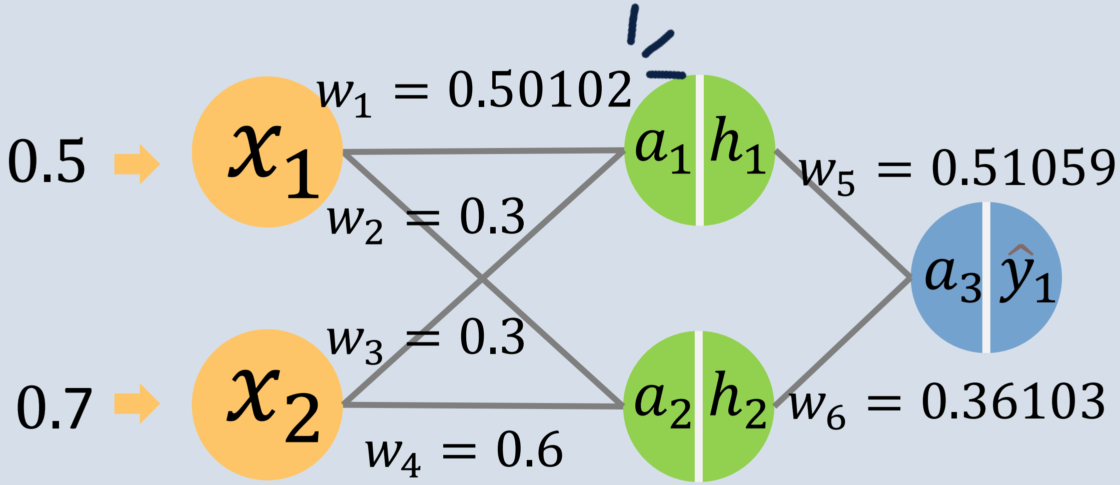
Activation function(활성화 함수) = $sigmoid()$

Loss function(손실 함수) = MSE

$$\frac{\partial E}{\partial w_1} = \frac{\partial E}{\partial \hat{y}_1} * \frac{\partial \hat{y}_1}{\partial a_3} * \frac{\partial a_3}{\partial h_1} * \frac{\partial h_1}{\partial a_1} * \frac{\partial a_1}{\partial w_1}$$

$$-0.742 * 0.233 * 0.5 * 0.237 * 0.5 \approx -0.0102$$

$y = 1$



$a_1 = 0.46$

$a_2 = 0.57$

$h_1 \approx 0.613$

$h_2 \approx 0.638$

$a_3 = 0.5298$

$\hat{y}_1 \approx 0.629$

$$0.50102 = 0.5 - 0.1 * (-0.0102)$$

$\mathrm{E} \approx 0.1376$

3단계 : Back Propagation

오차역전파 (Back Propagation)

Learning rate(학습률, α) = 0.1

Activation function(활성화 함수) = $sigmoid()$

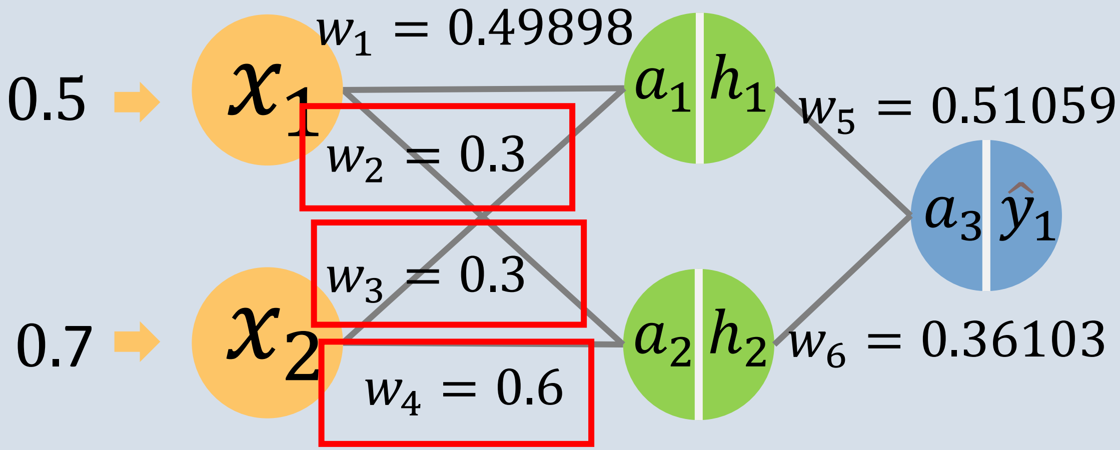
Loss function(손실 함수) = MSE

$$\frac{\partial E}{\partial w_2} = \frac{\partial E}{\partial \hat{y}_1} * \frac{\partial \hat{y}_1}{\partial a_3} * \frac{\partial a_3}{\partial h_2} * \frac{\partial h_2}{\partial a_2} * \frac{\partial a_2}{\partial w_2} \approx -0.00698$$

$$\frac{\partial E}{\partial w_3} = \frac{\partial E}{\partial \hat{y}_1} * \frac{\partial \hat{y}_1}{\partial a_3} * \frac{\partial a_3}{\partial h_1} * \frac{\partial h_1}{\partial a_1} * \frac{\partial a_1}{\partial w_3} \approx -0.01433$$

$$\frac{\partial E}{\partial w_4} = \frac{\partial E}{\partial \hat{y}_1} * \frac{\partial \hat{y}_1}{\partial a_3} * \frac{\partial a_3}{\partial h_2} * \frac{\partial h_2}{\partial a_2} * \frac{\partial a_2}{\partial w_4} \approx -0.00978$$

$y = 1$



$a_1 = 0.46$

$a_2 = 0.57$

$h_1 \approx 0.613$

$h_2 \approx 0.638$

$a_3 = 0.5298$

$\hat{y}_1 \approx 0.629$

$\mathrm{E} \approx 0.1376$

3단계 : Back Propagation

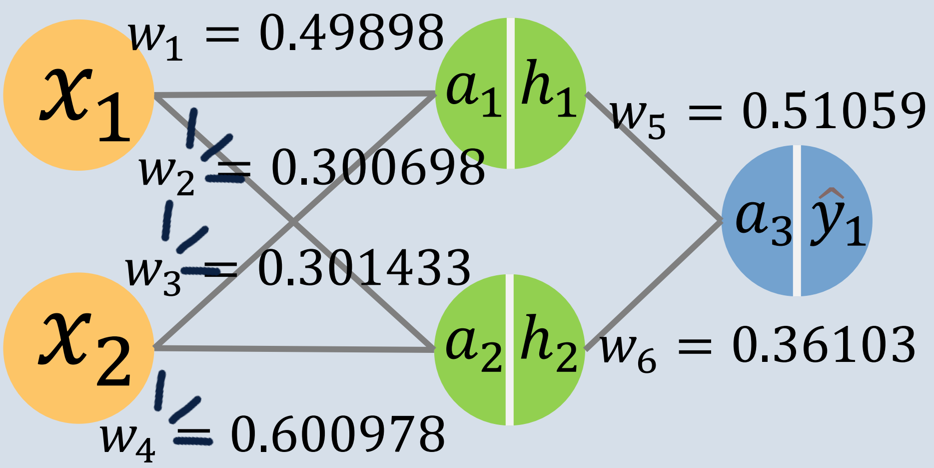
오차역전파 (Back Propagation)

Learning rate(학습률, α) = 0.1

Activation function(활성화 함수) = $sigmoid()$

Loss function(손실 함수) = MSE

$y = 1$



이전오차 $\mathrm{E} \approx 0.1376$

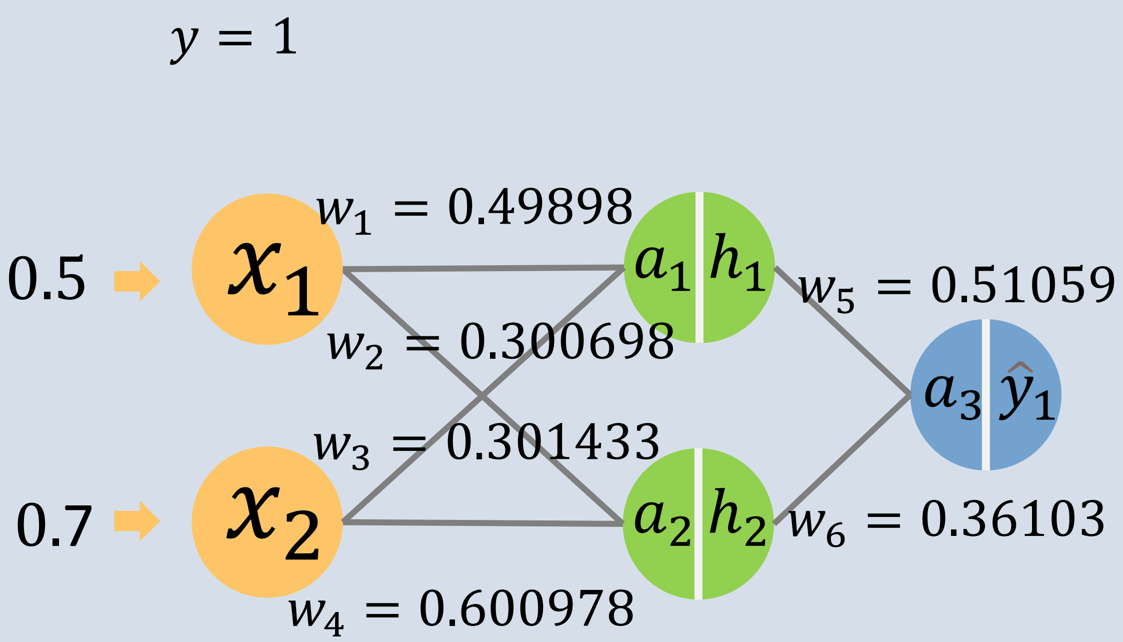
3단계 : Back Propagation

오차역전파 (Back Propagation)

Learning rate(학습률, α) = 0.1

Activation function(활성화 함수) = $sigmoid()$

Loss function(손실 함수) = MSE



$$a_1 = w_1x_1 + w_3x_2 \approx 0.4604$$

$$a_2 = w_2x_1 + w_4x_2 \approx 0.5710$$

$$h_1 = sigmoid(a_1) \approx 0.613089$$

$$h_2 = sigmoid(a_2) \approx 0.639466$$

$$a_3 = w_5h_1 + w_6h_2 \approx 0.543653$$

$$\hat{y}_1 = sigmoid(a_3) \approx 0.6327$$

$$\mathrm{E} = \frac{1}{1}\sum_{i=1}^{1}(y_1 - \hat{y}_1)^2 \approx 0.1349$$

이전오차 $\mathrm{E} \approx 0.1376$

3단계 : Back Propagation

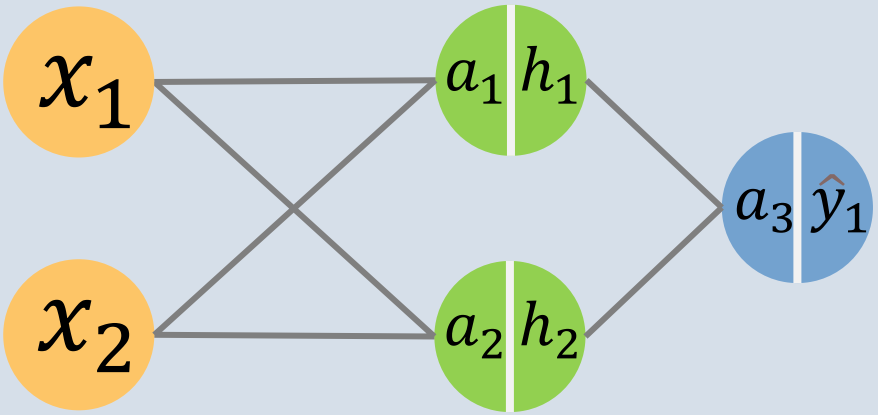
오차역전파 (Back Propagation)

Learning rate(학습률, α) = 0.1

Activation function(활성화 함수) = $sigmoid()$

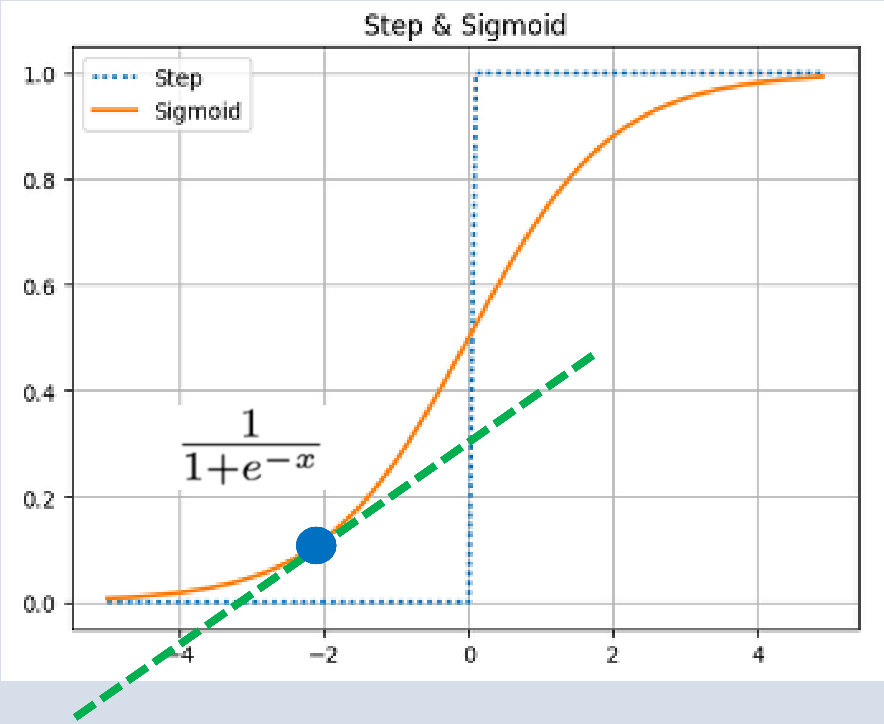
Loss function(손실 함수) = MSE

$y = 1$

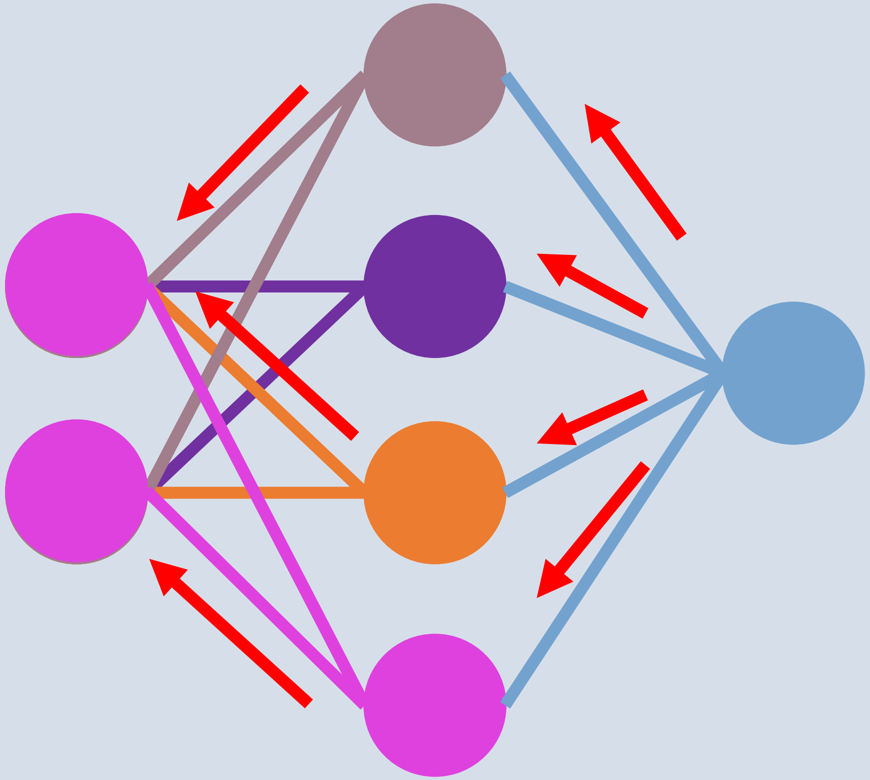


1단계 : Forward Propagation
2단계 : 오차 계산
3단계 : Back Propagation

활성화 함수(Activation Function)

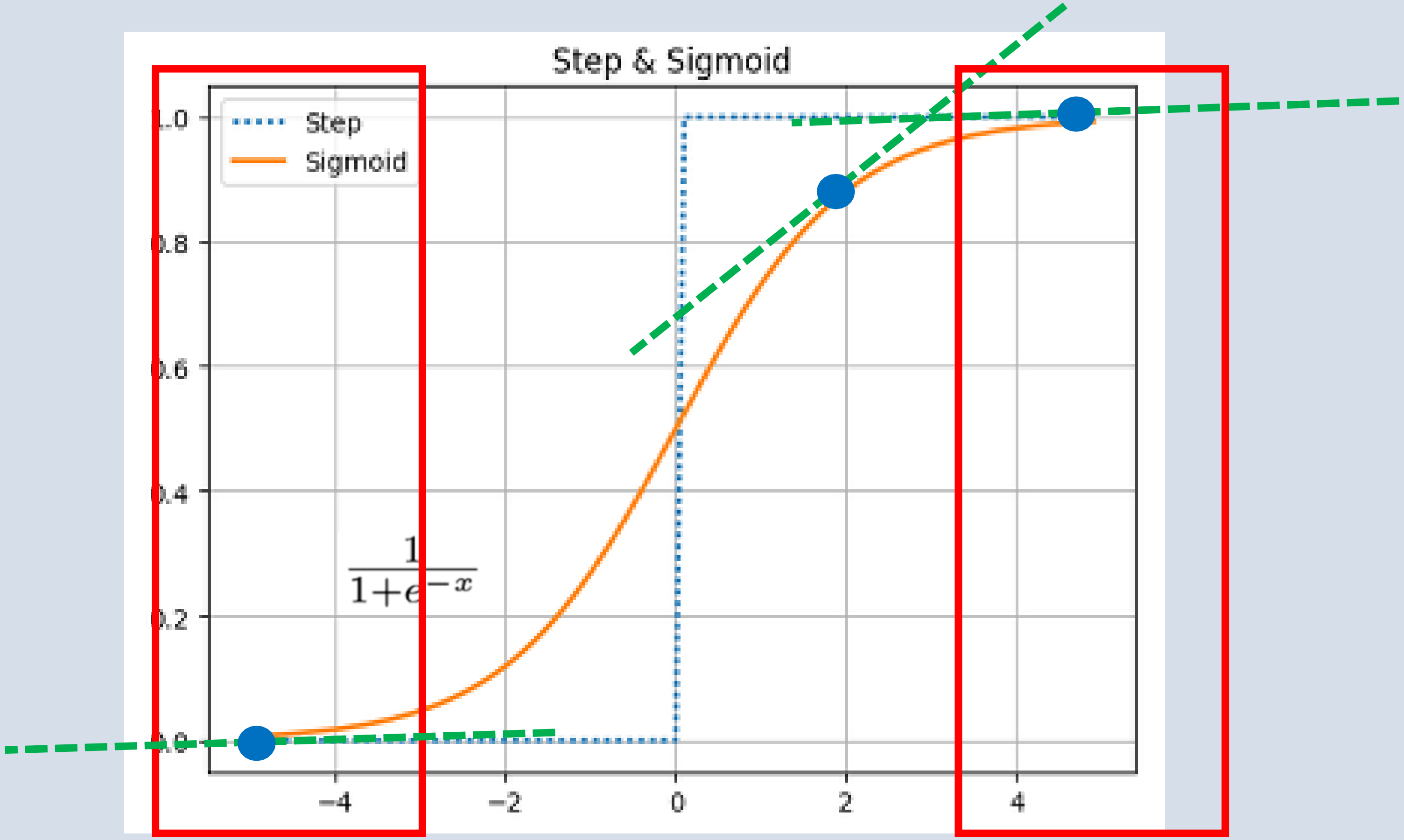


Sigmoid function
(시그모이드 함수)



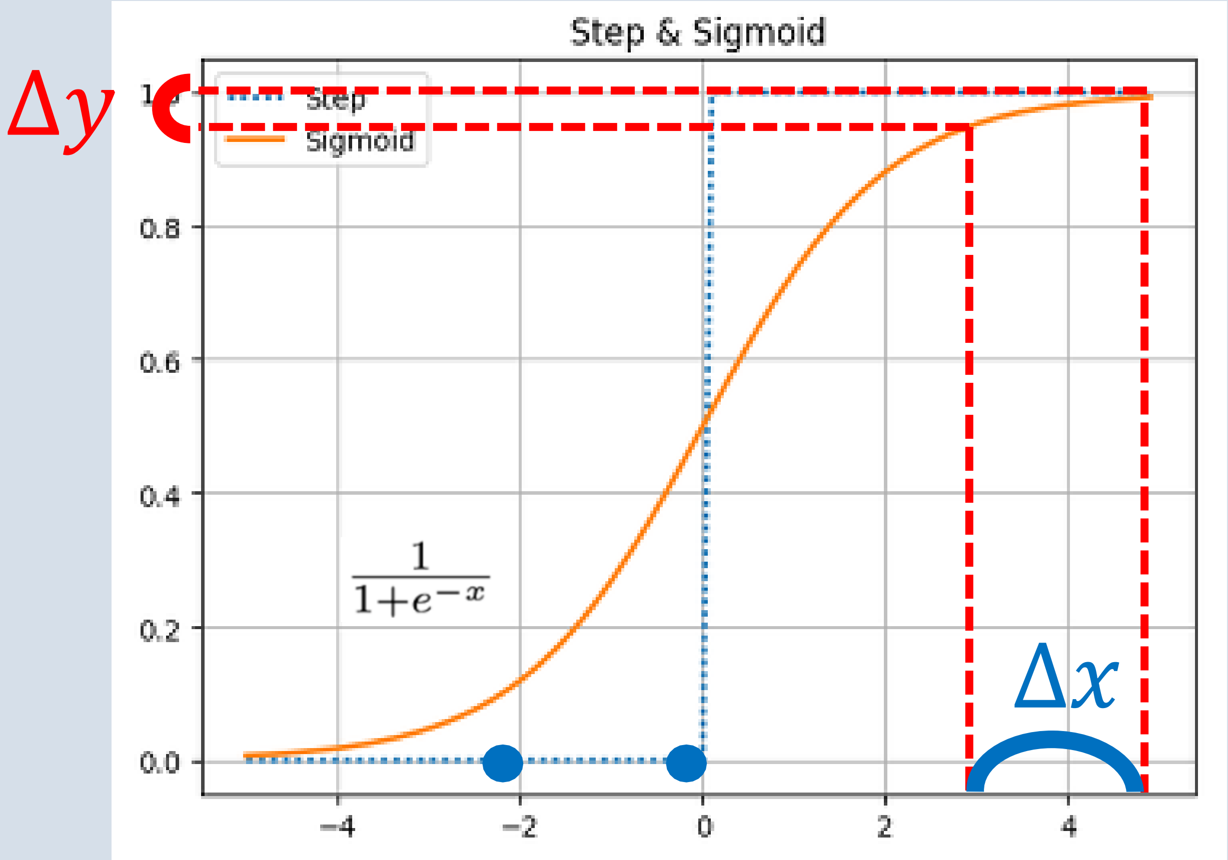
Back Propagation
(오차 역전파 알고리즘)

활성화 함수(Activation Function)



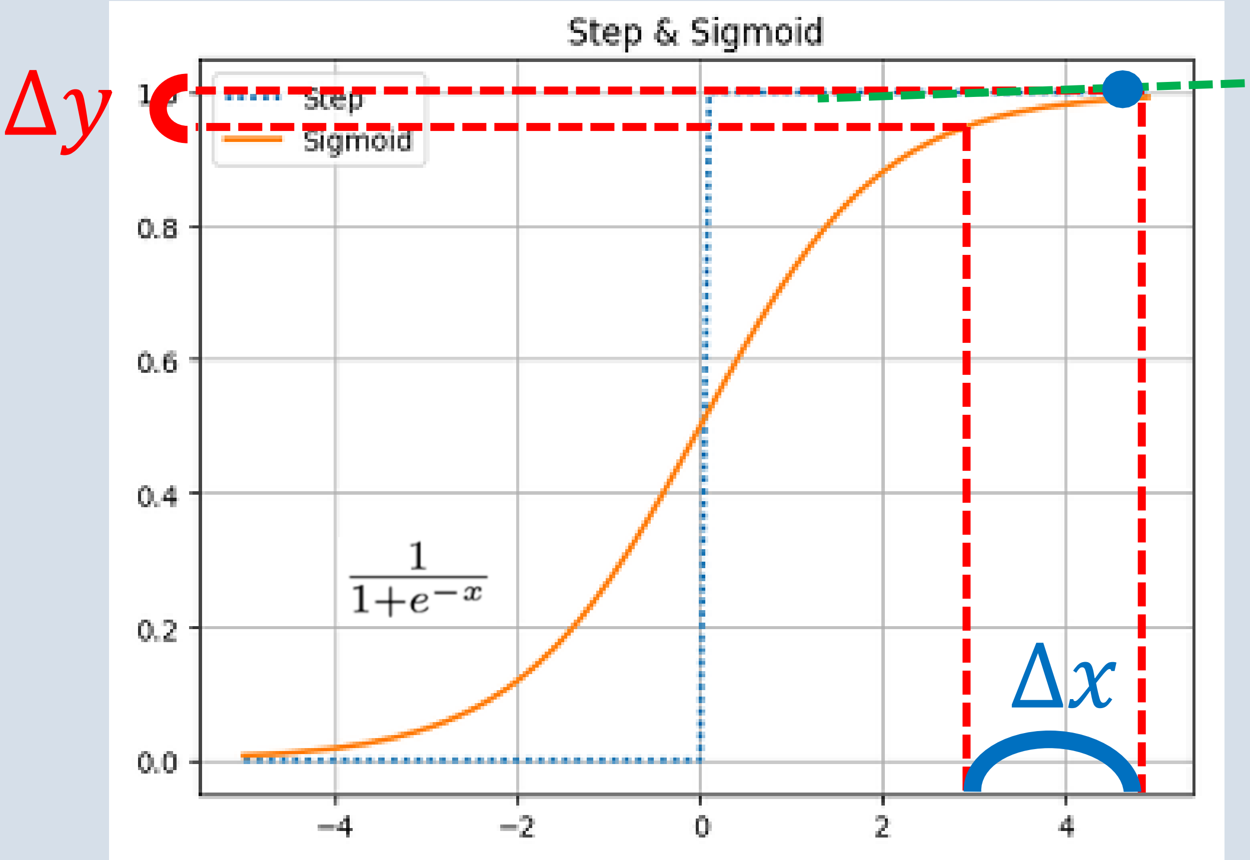
Sigmoid function
(시그모이드함수)

활성화 함수(Activation Function)

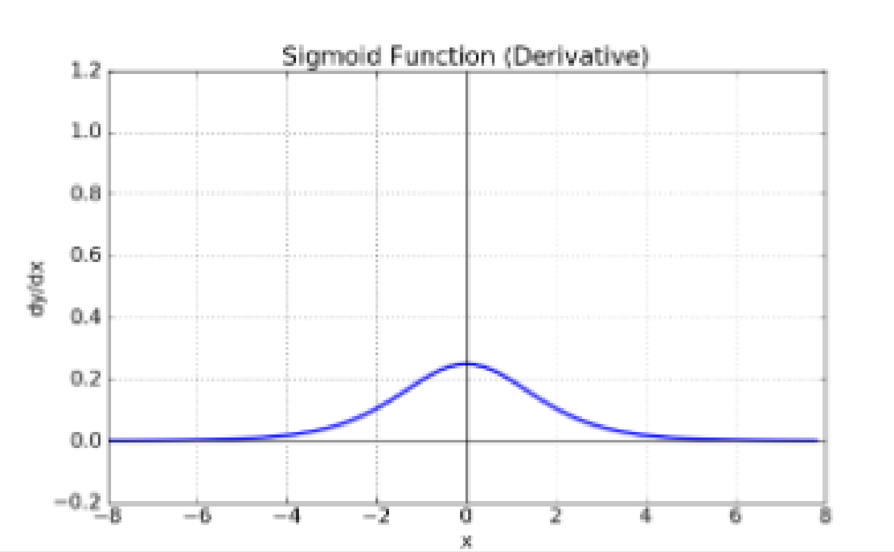


Sigmoid function
(시그모이드함수)

활성화 함수(Activation Function)



Chain rule $\frac{\partial E}{\partial w_5} = \frac{\partial E}{\partial \hat{y}_1} * \frac{\partial \hat{y}_1}{\partial a_3} * \frac{\partial a_3}{\partial w_5}$



Sigmoid function (시그모이드함수)

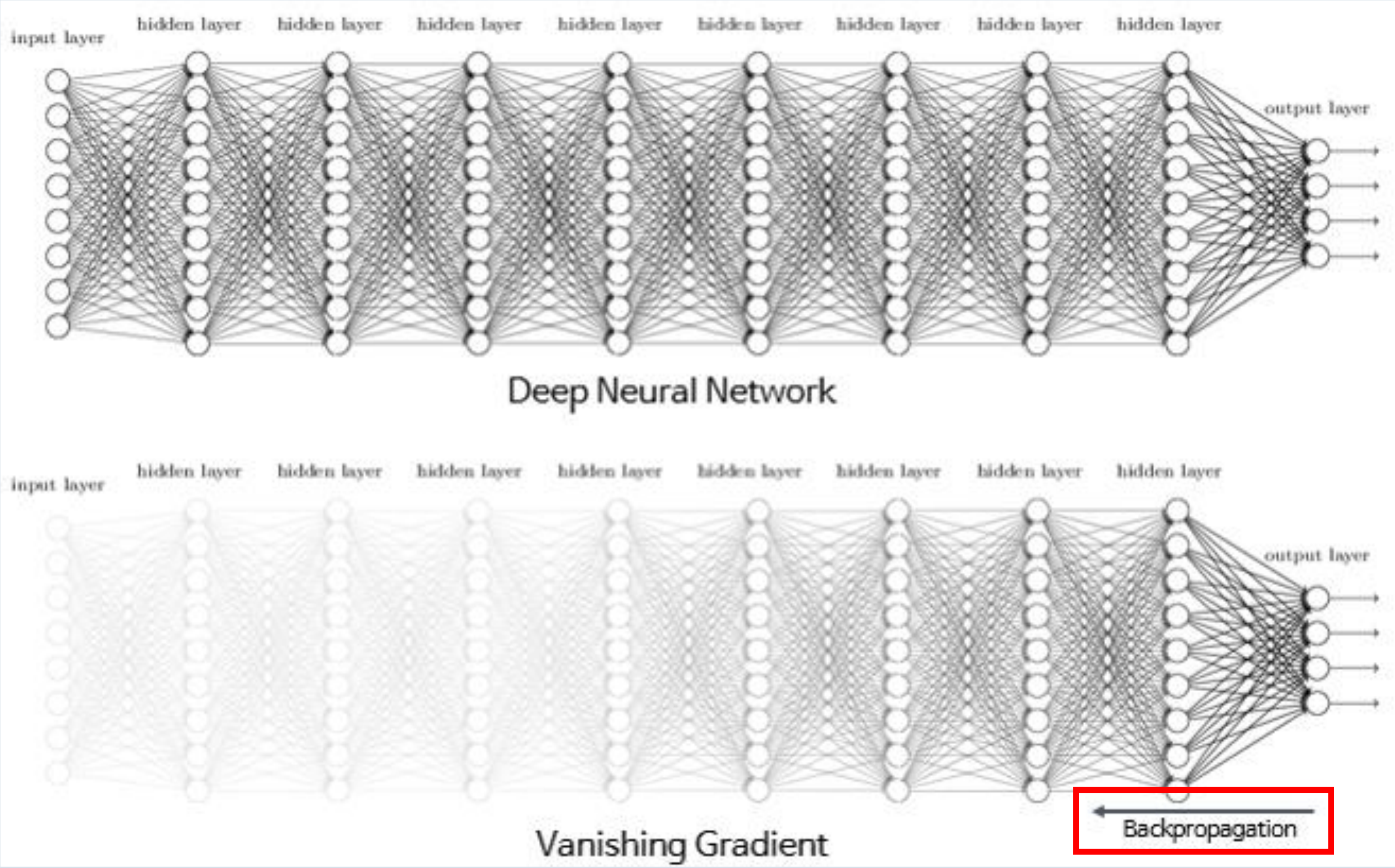
Sigmoid 도함수

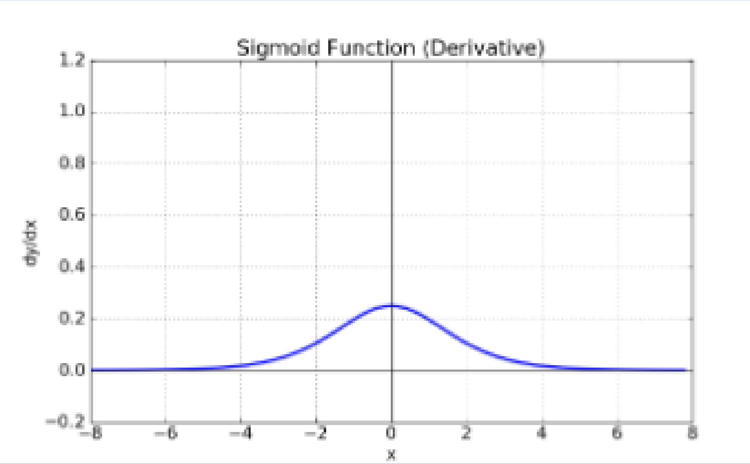
활성화 함수(Activation Function)

Sigmoid 함수의 문제점

- 기울기 소실 문제(Vanishing Gradient Problem)

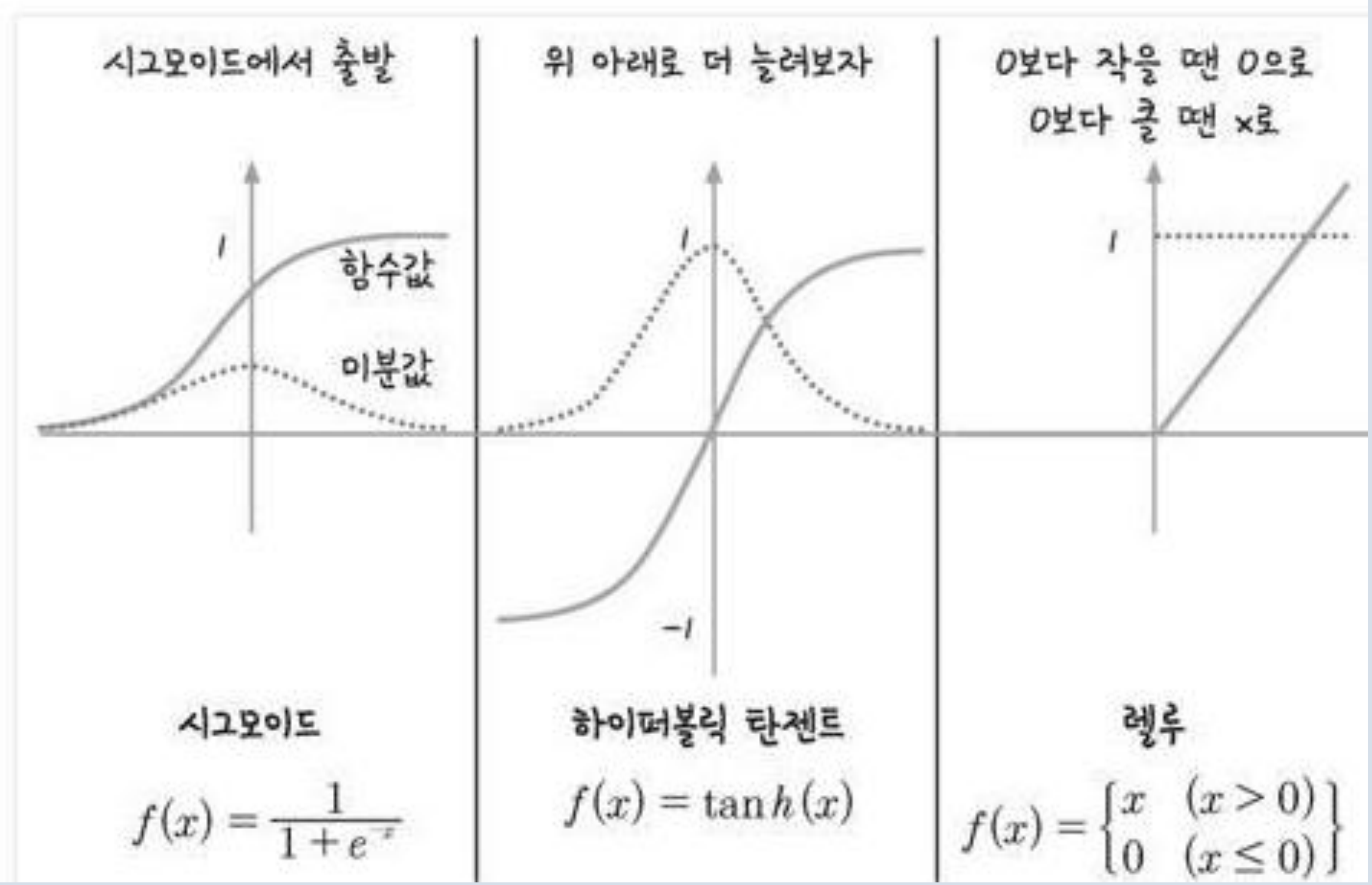
Chain rule $$\frac{\partial E}{\partial w_5} = \frac{\partial E}{\partial \hat{y}_1} * \frac{\partial \hat{y}_1}{\partial a_3} * \frac{\partial a_3}{\partial w_5}$$



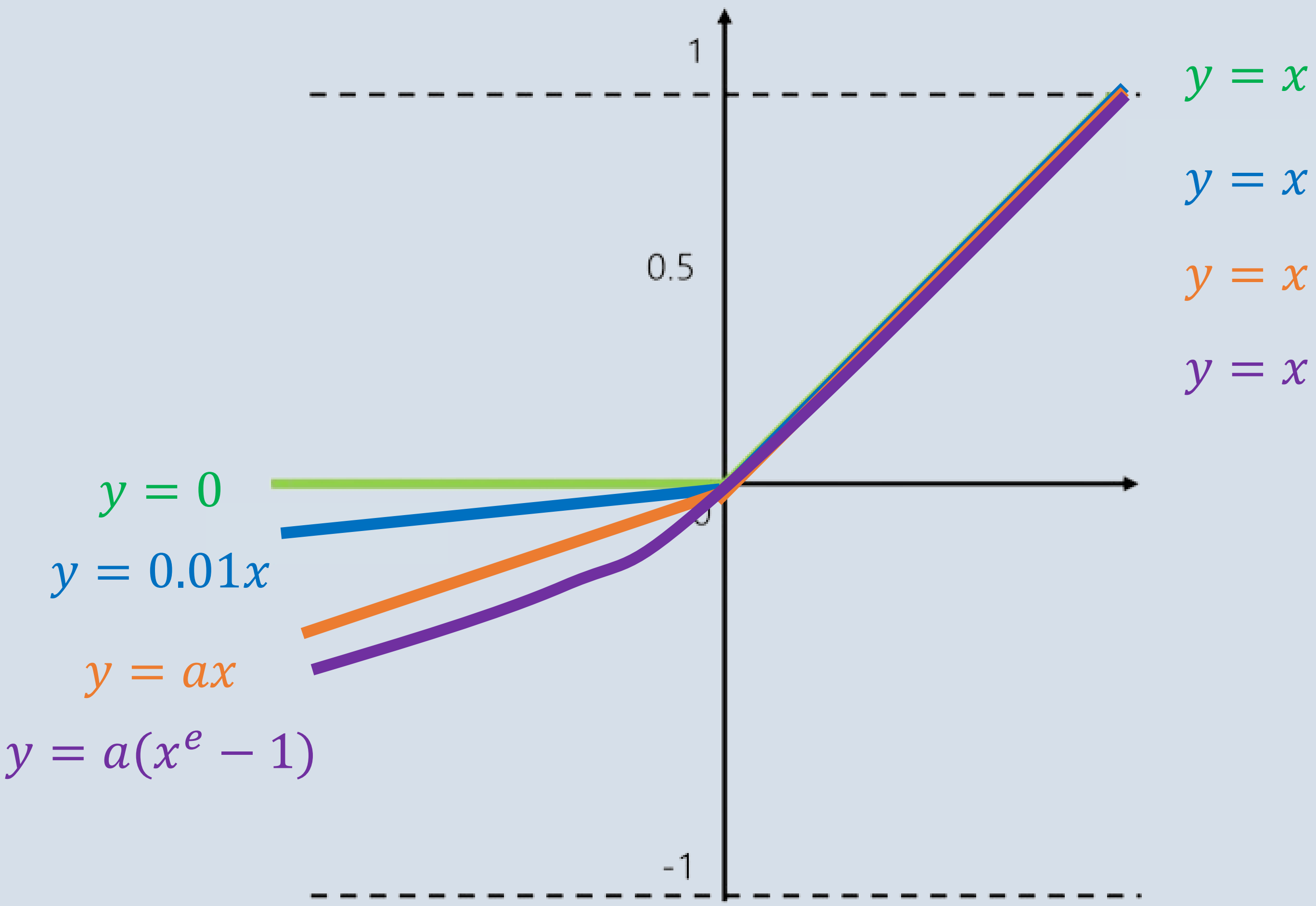


Sigmoid 도함수

활성화 함수(Activation Function)



활성화 함수(Activation Function)



ReLU

Leaky ReLU

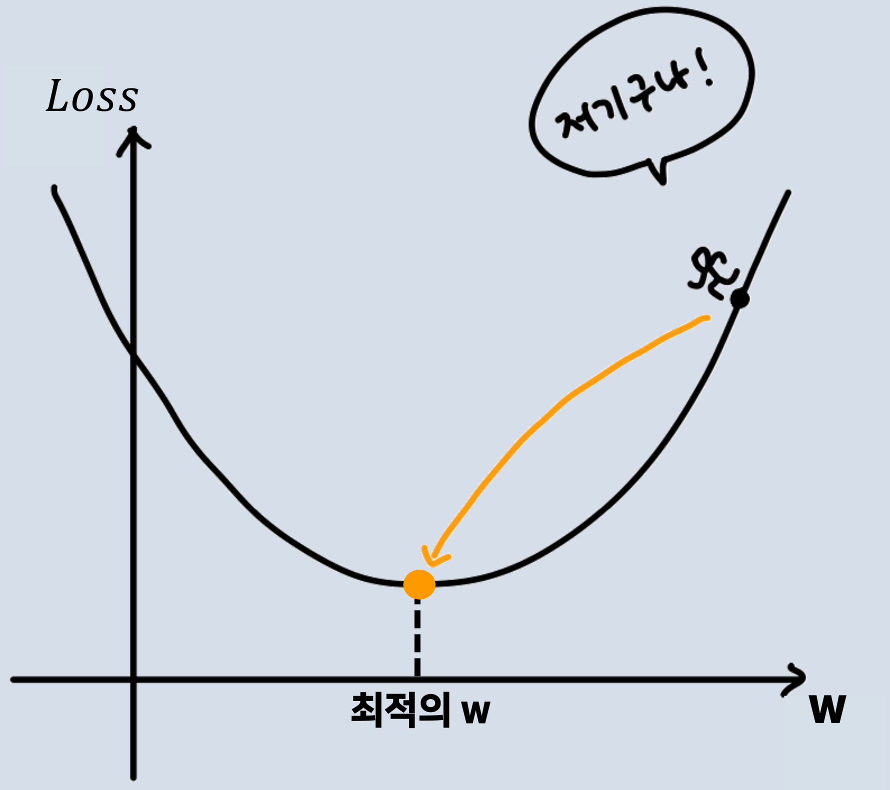
PReLU

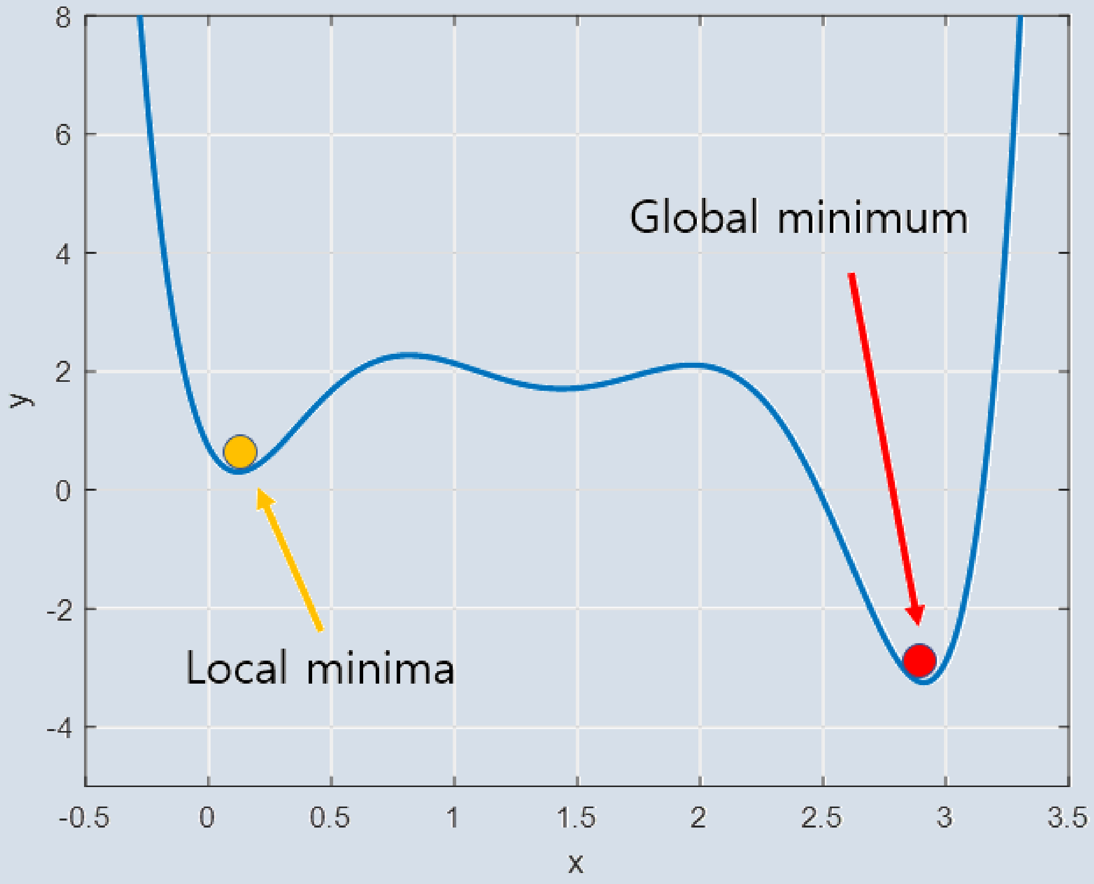
ELU

최적화 함수 (Optimizer)

최적화 함수(Optimizer)

경사하강법(Gradient Descent Algorithm)

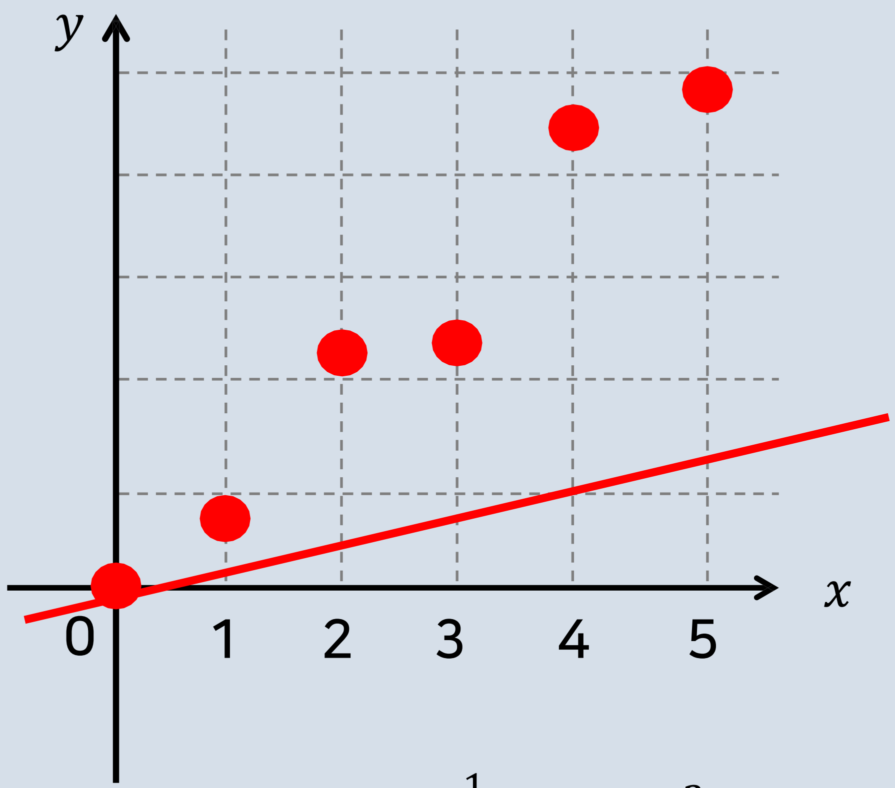




Batch Gradient Descent(배치 경사하강법)

$\alpha = 0.01$

$\hat{y} = 0.2x + 0$



x	y	$\hat{y}$
0	0	0
1	0.8	0.2
2	2.2	0.4
3	2.3	0.6
4	4.3	0.8
5	4.9	1

$$w := w - \alpha \frac{\partial L}{\partial w}$$

$$w := w - \alpha \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial w}$$

$$w := w - \alpha \left(\frac{1}{n} * (-2) \sum_{i=1}^{n} (y_i - \hat{y}_i) \right) \frac{\partial \hat{y}}{\partial w}$$

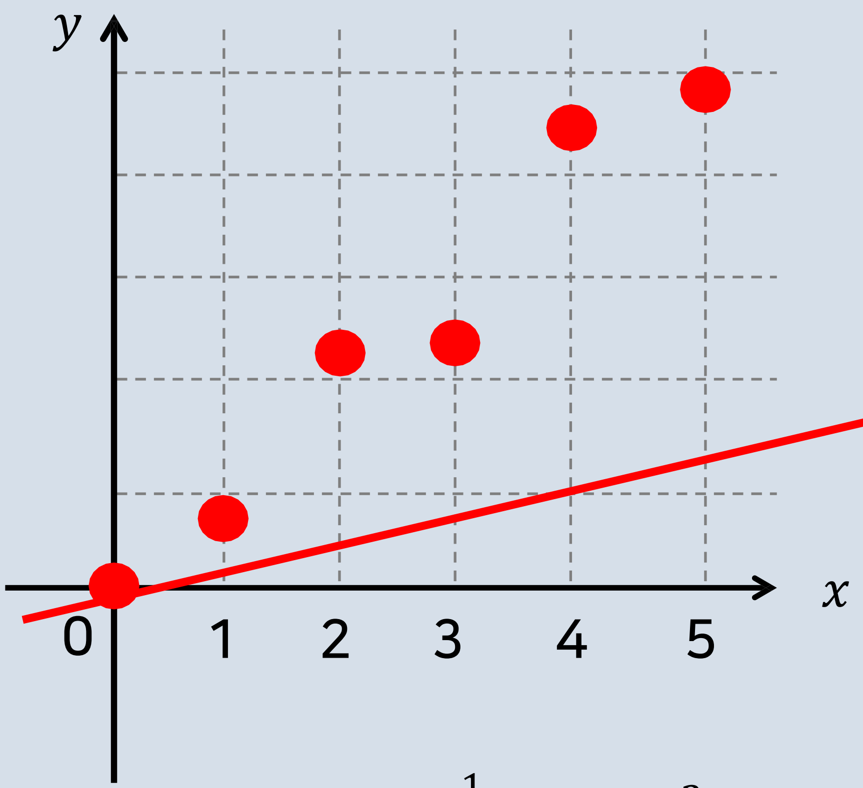
$$MSE(L) = \frac{1}{N} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

$$= \frac{1}{6}\{(0-0)^2 + (0.8-0.2)^2 + (2.2-0.4)^2 + (2.3-0.6)^2 + (4.3-0.8)^2 + (4.9-1)^2\}$$
$$= 5.66$$

Batch Gradient Descent(배치 경사하강법)

$\alpha = 0.01$

$\hat{y} = 0.2x + 0$



x	y	$\hat{y}$
0	0	0
1	0.8	0.2
2	2.2	0.4
3	2.3	0.6
4	4.3	0.8
5	4.9	1

$$w := w - \alpha \frac{\partial L}{\partial w}$$

$$w := w - \alpha \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial w}$$

$$w := w - \alpha \left(\frac{1}{n} * (-2) \sum_{i=1}^{n} (y_i - \hat{y_i}) * x_i \right)$$

$$MSE(L) = \frac{1}{N} \sum_{i=1}^{n} (y_i - \hat{y_i})^2$$

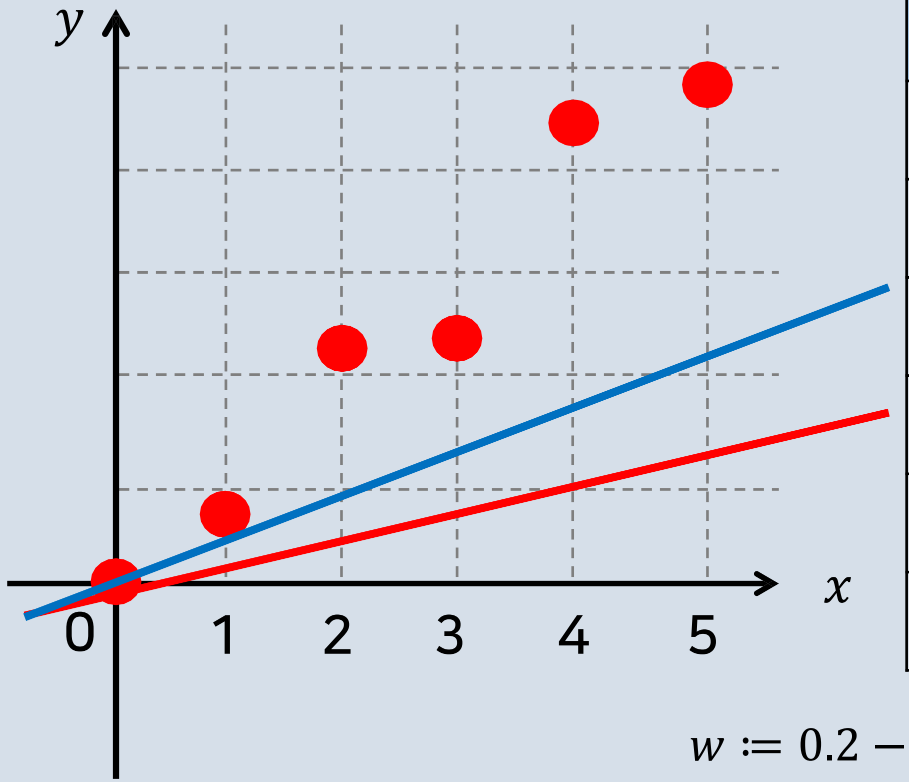
$$= \frac{1}{6}\{(0-0)^2 + (0.8-0.2)^2 + (2.2-0.4)^2 + (2.3-0.6)^2 + (4.3-0.8)^2 + (4.9-1)^2\}$$
$$= 5.66$$

Batch Gradient Descent(배치 경사하강법)

$\alpha = 0.01$

$\hat{y} = 0.2x + 0$

$$MSE(L) = \frac{1}{N}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$



x	y	$\hat{y}$
0	0	0
1	0.8	0.2
2	2.2	0.4
3	2.3	0.6
4	4.3	0.8
5	4.9	1

$$w := w - \alpha\frac{\partial L}{\partial w}$$

$$w := w - \alpha\frac{\partial L}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial w}$$

$$w := w - \alpha\left(\frac{1}{n} * (-2)\sum_{i=1}^{n}(y_i - \hat{y}_i) * x_i\right)$$

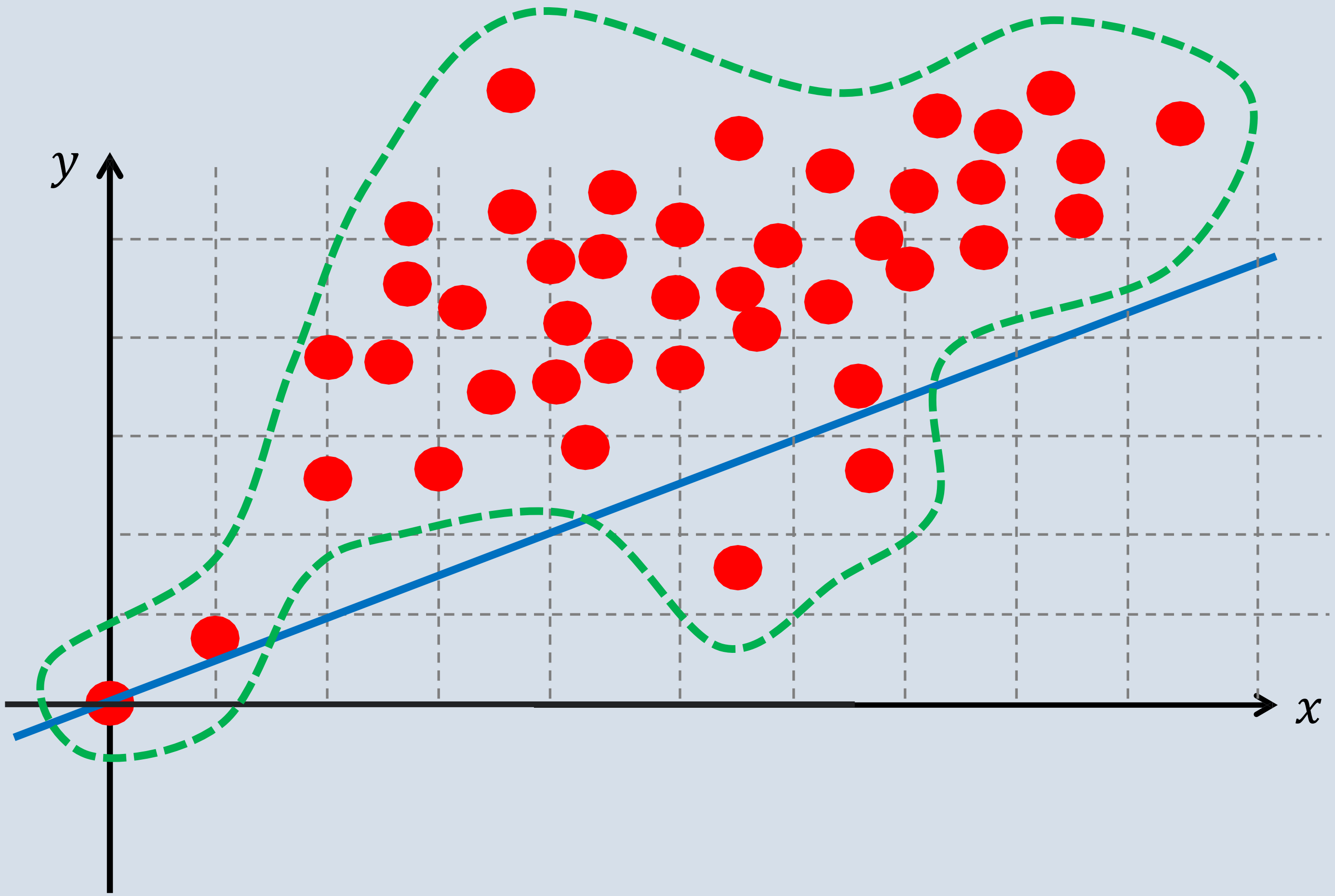
new $w = 0.343$

$$w := 0.2 - 0.01\left(\frac{1}{6} * (-2) * (0 * 0 + 0.6 * 1 + 1.8 * 2 + 1.7 * 3 + 3.5 * 4 + 3.9 * 5)\right)$$

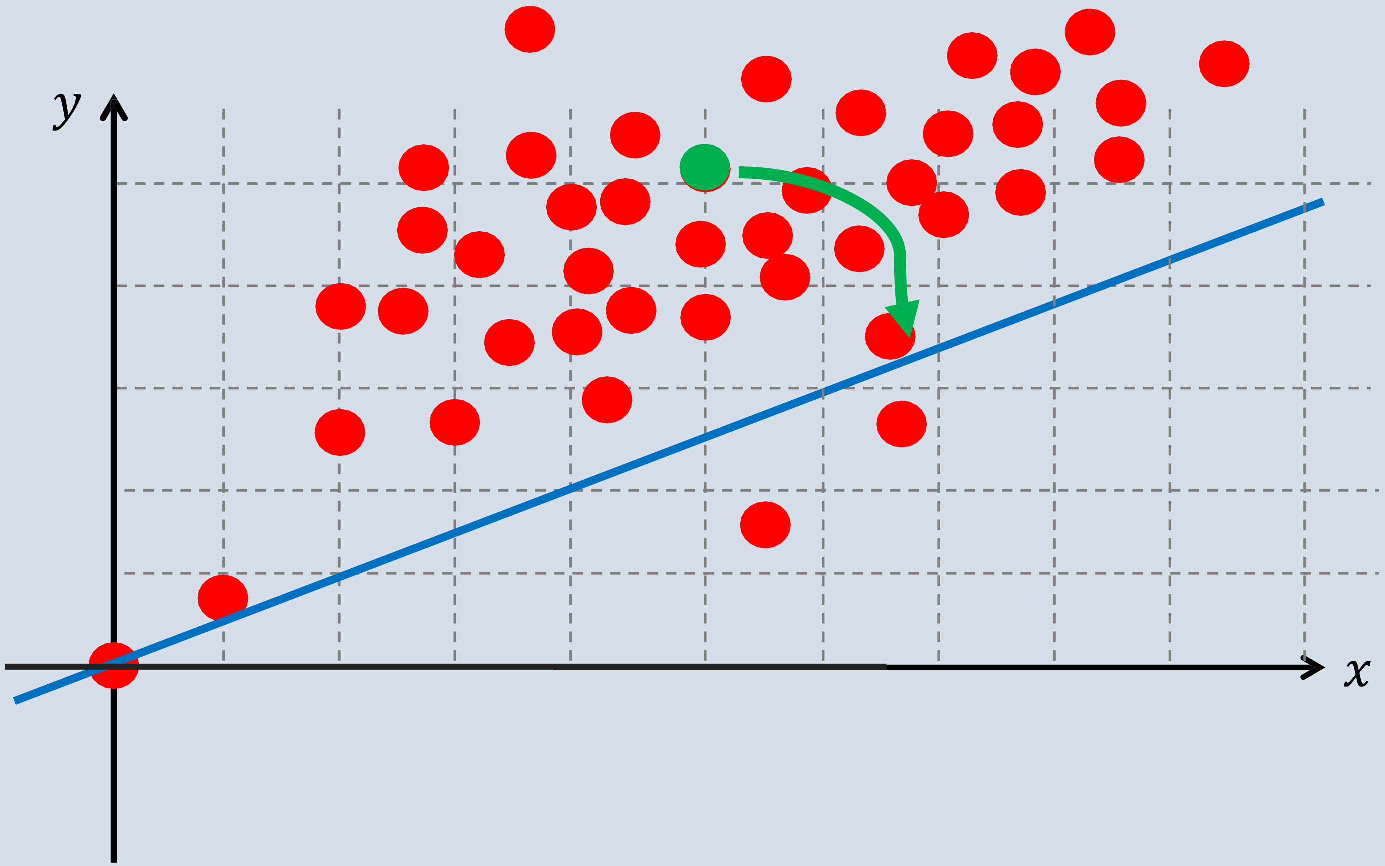
$$= \frac{1}{6}\{(0-0)^2 + (0.8-0.2)^2 + (2.2-0.4)^2 + (2.3-0.6)^2 + (4.3-0.8)^2 + (4.9-1)^2\}$$
$$= 5.66$$

스마트인재개발원 Smart Human Resources Development

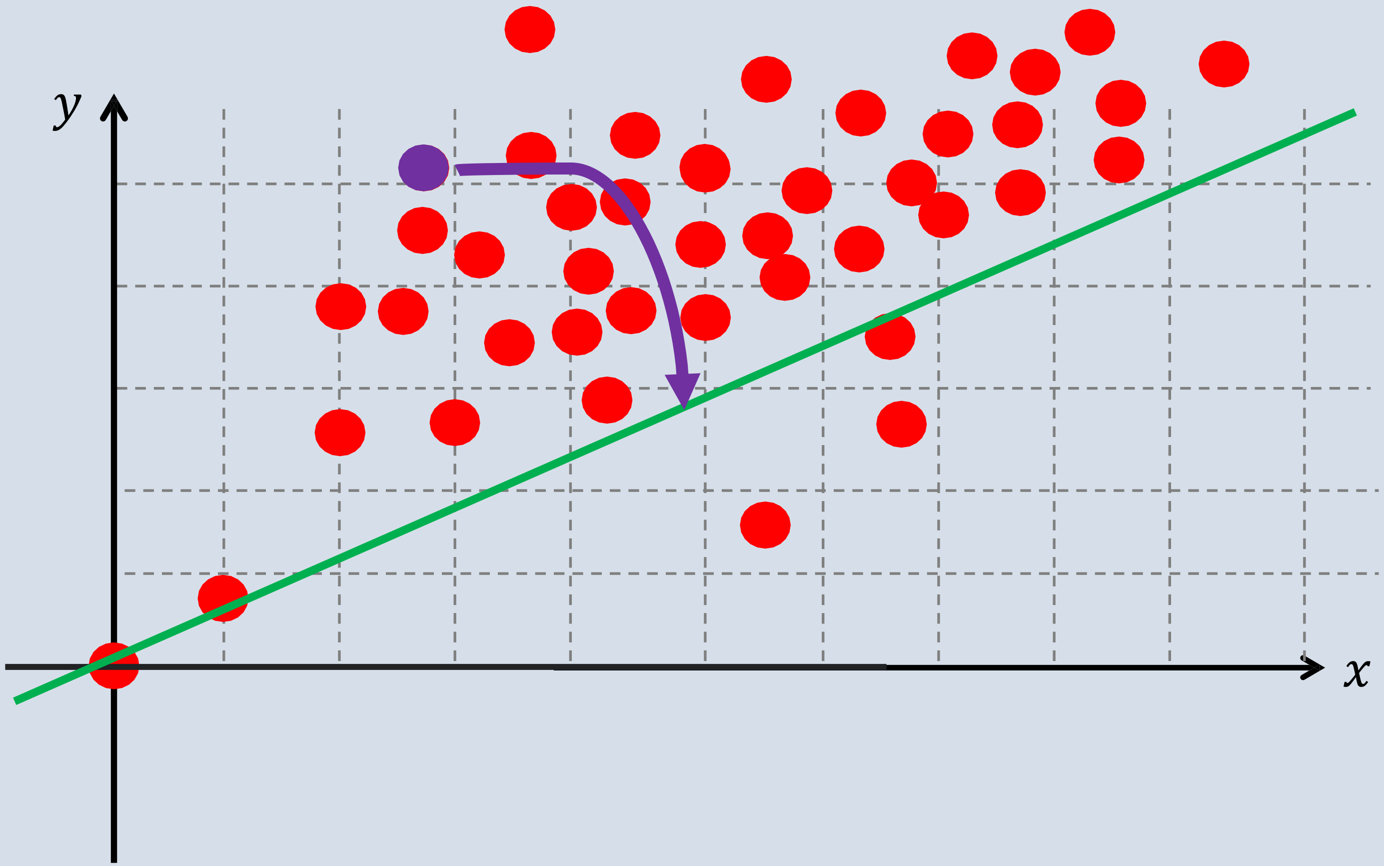
Batch Gradient Descent(배치 경사하강법)



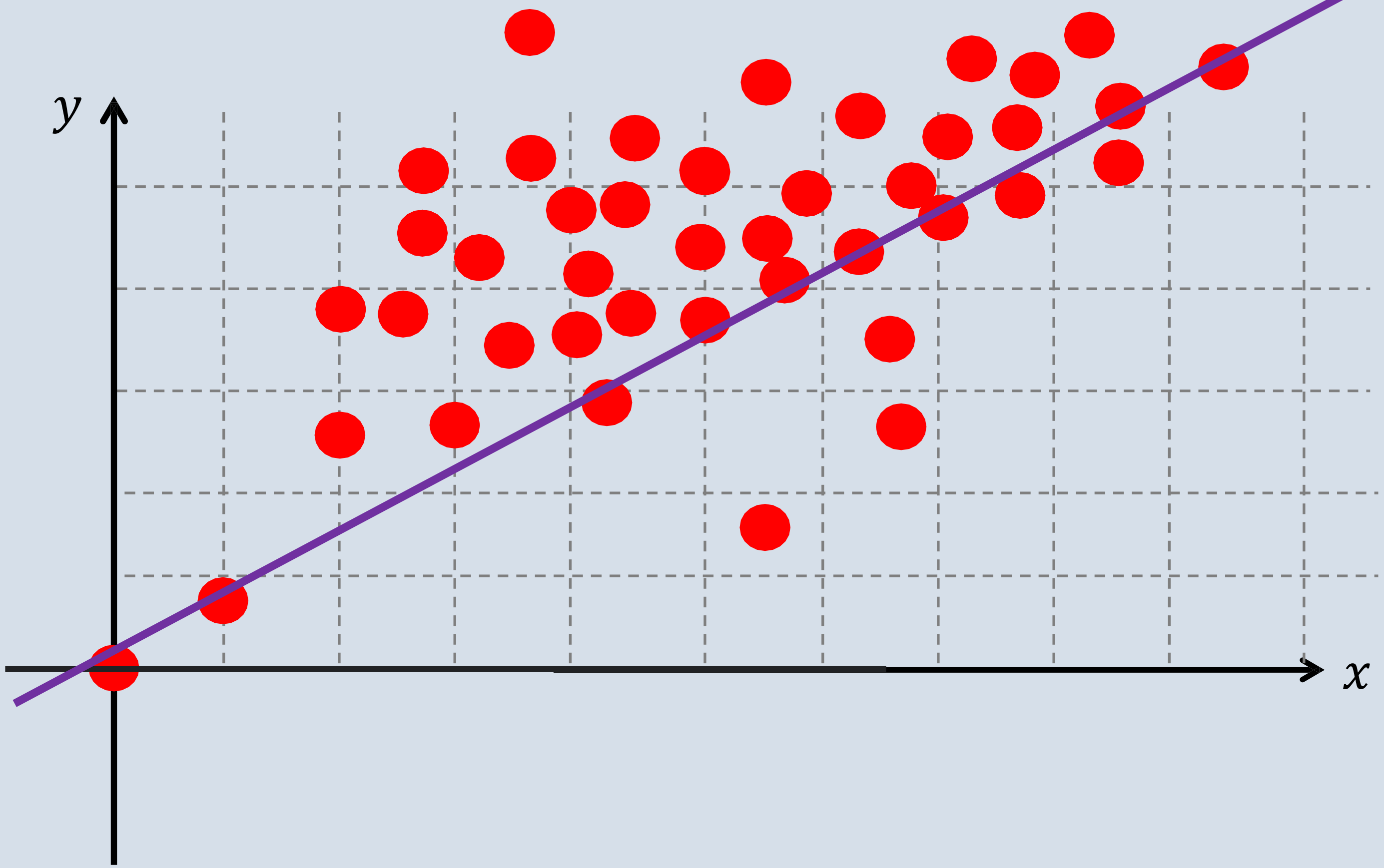
Stochastic Gradient Descent(확률적 경사하강법)



Stochastic Gradient Descent(확률적 경사하강법)



Stochastic Gradient Descent(확률적 경사하강법)

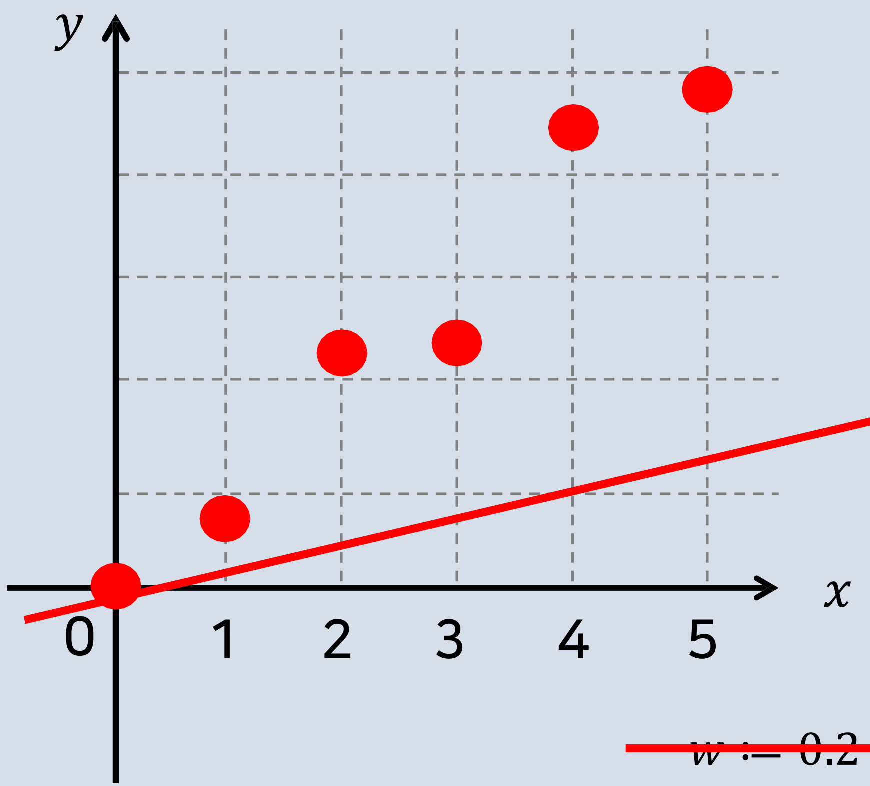


Stochastic Gradient Descent(확률적 경사하강법)

$\alpha = 0.01$

$\hat{y} = 0.2x + 0$

$$MSE(L) = \frac{1}{N}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$



x	y	$\hat{y}$
0	0	0
1	0.8	0.2
2	2.2	0.4
3	2.3	0.6
4	4.3	0.8
5	4.9	1

$$w := w - \alpha\frac{\partial L}{\partial w}$$

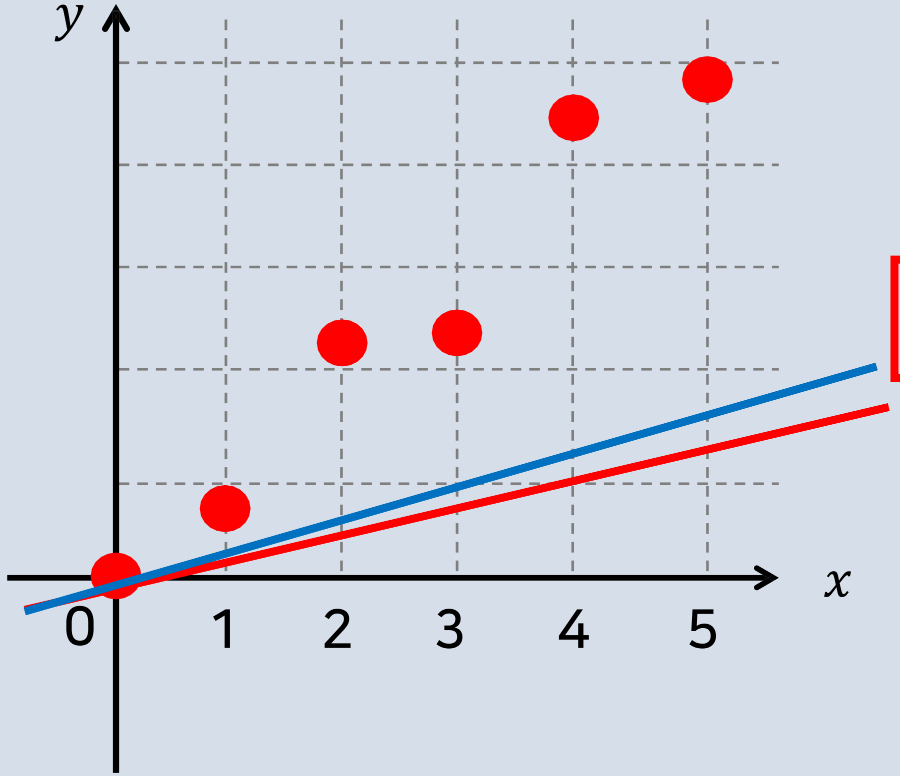
$$w := w - \alpha\frac{\partial L}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial w}$$

$$w := w - \alpha\left(\frac{1}{n} * (-2)\sum_{i=1}^{n}(y_i - \hat{y}_i) * x_i\right)$$

$$\cancel{w := 0.2 - 0.01\left(\frac{1}{6} * (-2) * (0 * 0 + 0.6 * 1 + 1.8 * 2 + 1.7 * 3 + 3.5 * 4 + 3.9 * 5)\right)}$$

Stochastic Gradient Descent(확률적 경사하강법)

$\alpha = 0.01$



x	y	$\hat{y}$
0	0	0
1	0.8	0.2
2	2.2	0.4
3	2.3	0.6
4	4.3	0.8
5	4.9	1

$$\hat{y} = 0.2x + 0 \qquad MSE(L) = \frac{1}{N}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

$$w := w - \alpha\frac{\partial L}{\partial w}$$

$$w := w - \alpha\frac{\partial L}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial w}$$

$$w := w - \alpha\left(\frac{1}{n} * (-2)\sum_{i=1}^{n}(y_i - \hat{y}_i) * x_i\right)$$

$$\text{new } w = 0.272$$

$$w := 0.2 - 0.01\left(\frac{1}{1} * (-2) * (2.2 * 0.4) * 2\right)$$

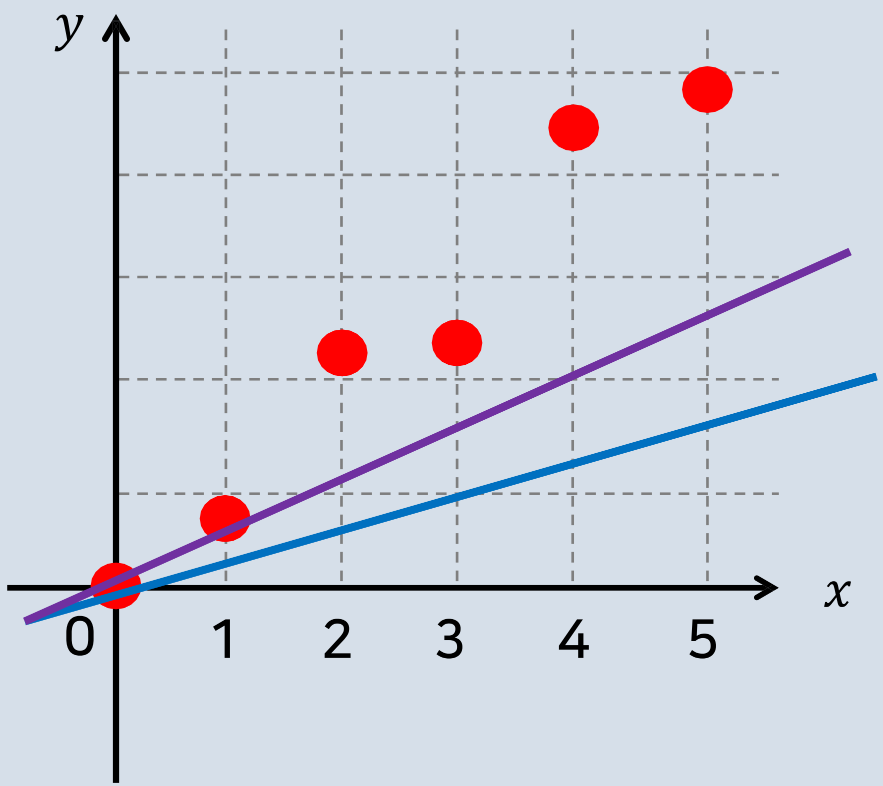
Deep Learning

Stochastic Gradient Descent(확률적 경사하강법)

$\alpha = 0.01$

$\hat{y} = 0.2x + 0$

$$MSE(L) = \frac{1}{N}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$



x	y	$\hat{y}$
0	0	0
1	0.8	0.2
2	2.2	0.4
3	2.3	0.6
4	4.3	0.8
5	4.9	1

$$w := w - \alpha\frac{\partial L}{\partial w}$$

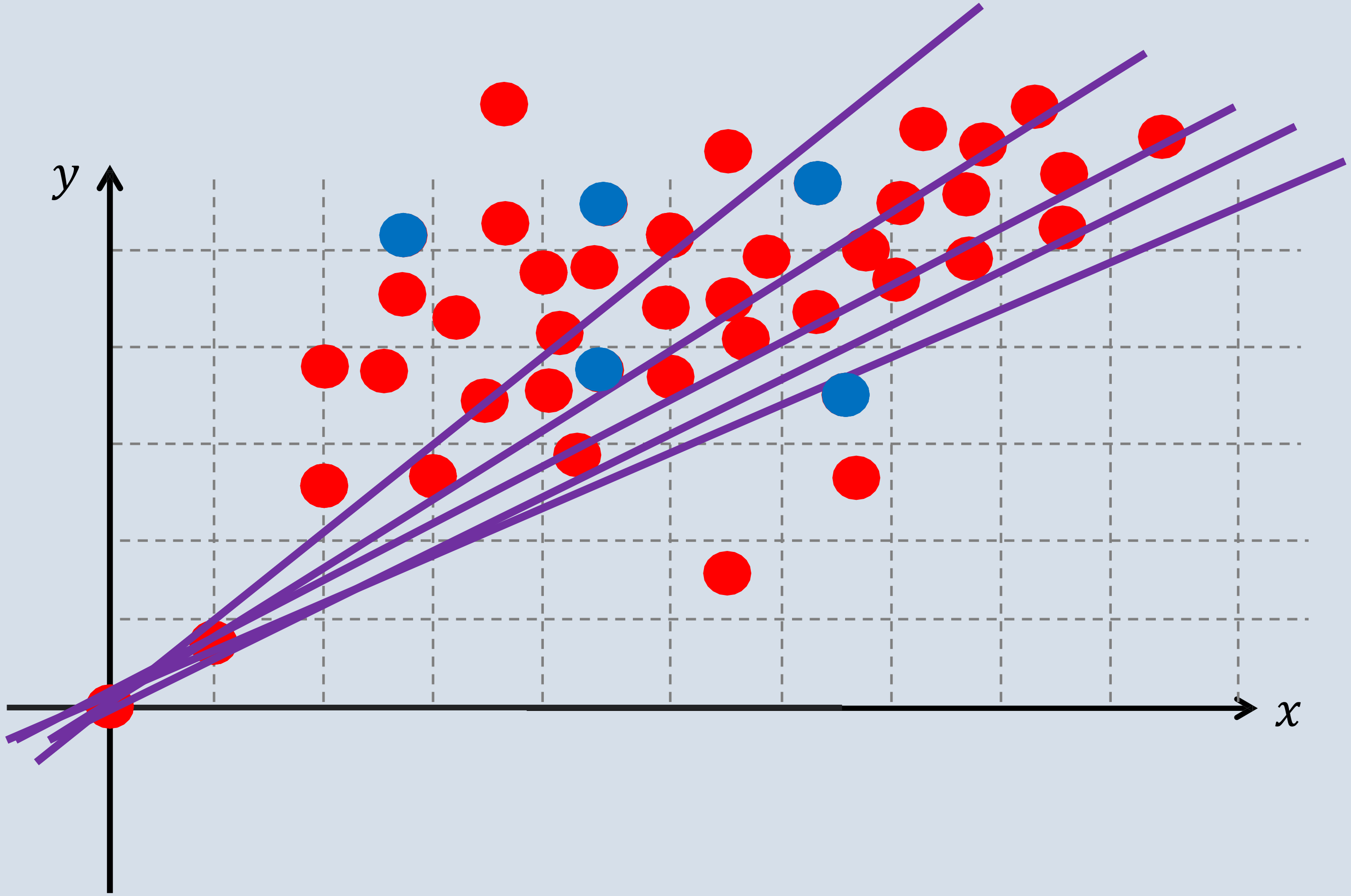
$$w := w - \alpha\frac{\partial L}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial w}$$

$$w := w - \alpha\left(\frac{1}{n} * (-2)\sum_{i=1}^{n}(y_i - \hat{y}_i) * x_i\right)$$

$$\text{new } w = 0.552$$

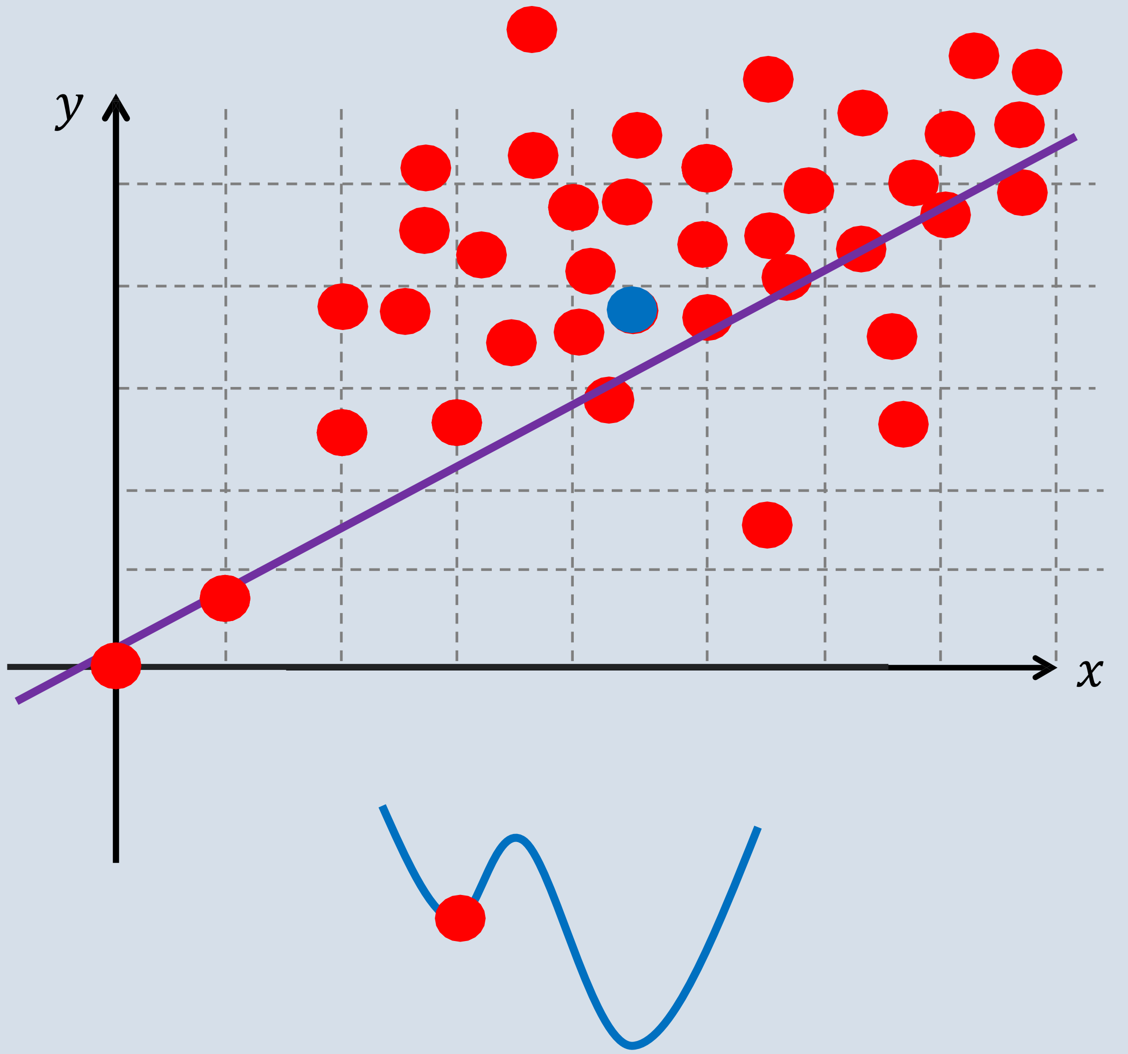
$$w := 0.272 - 0.01\left(\frac{1}{1} * (-2) * (4.3 * 0.8) * 4\right)$$

Stochastic Gradient Descent(확률적 경사하강법)

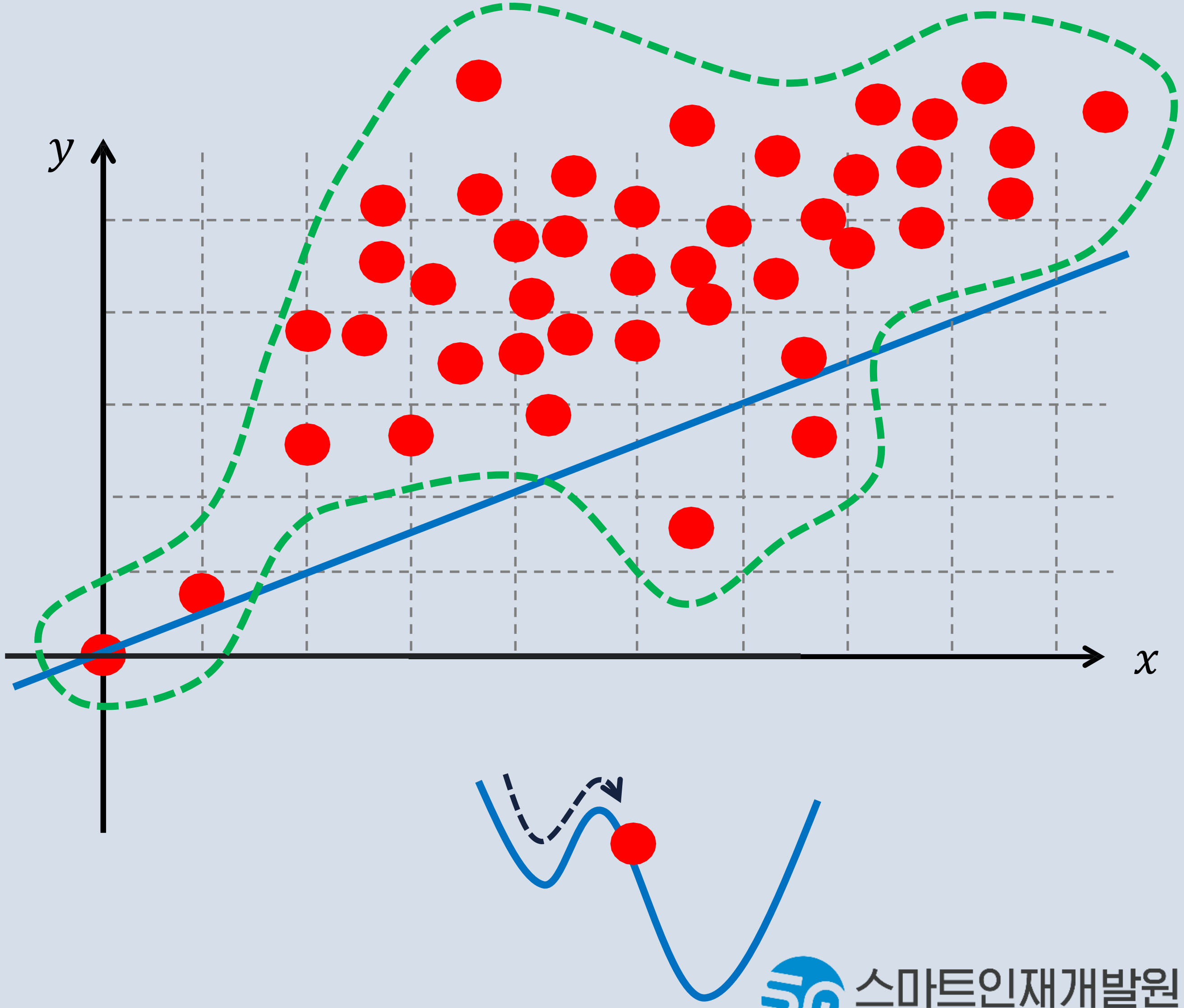


Mini-batch Gradient Descent(미니배치 경사하강법)

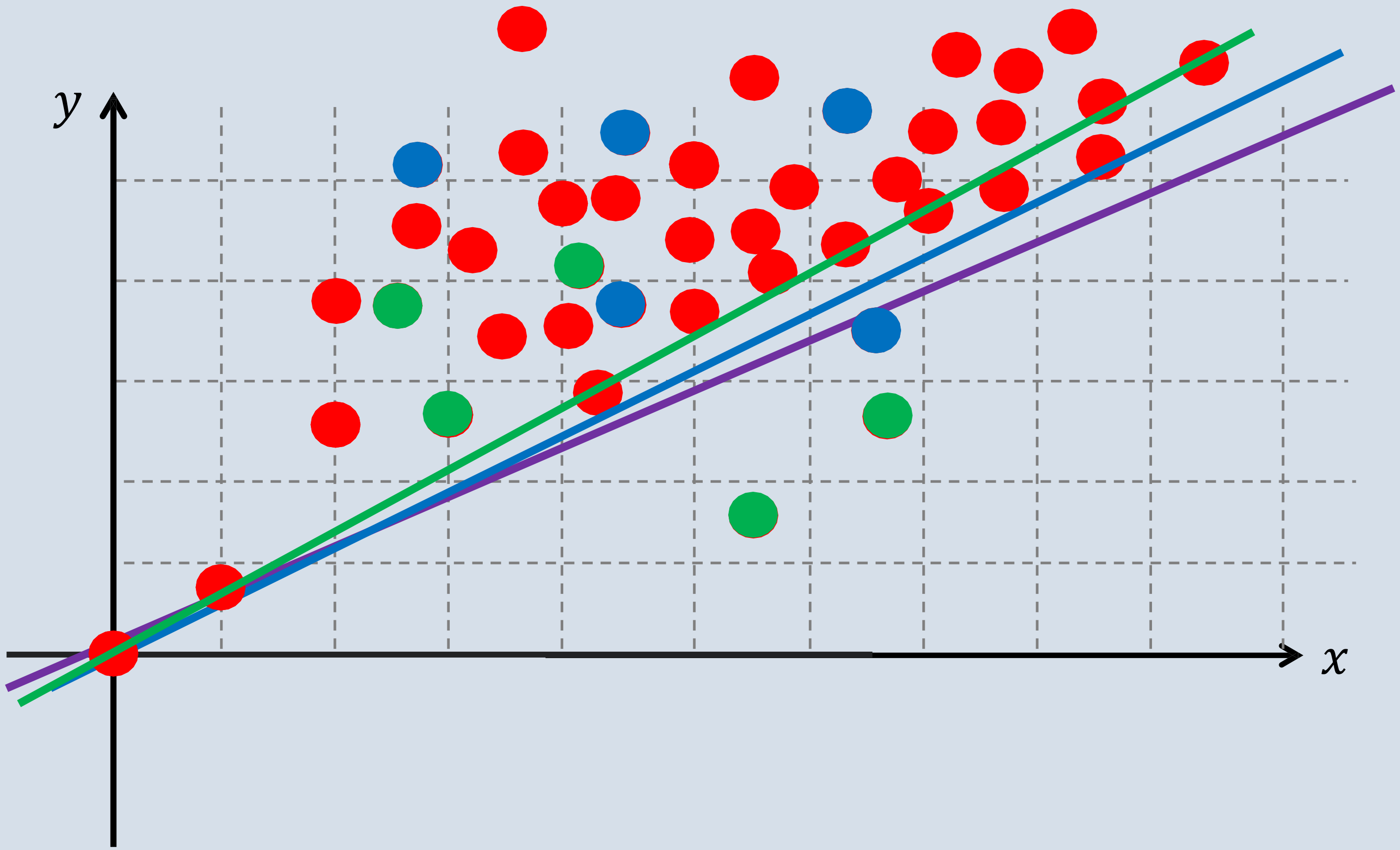
확률적 경사하강법(Stochastic Gradient Descent)

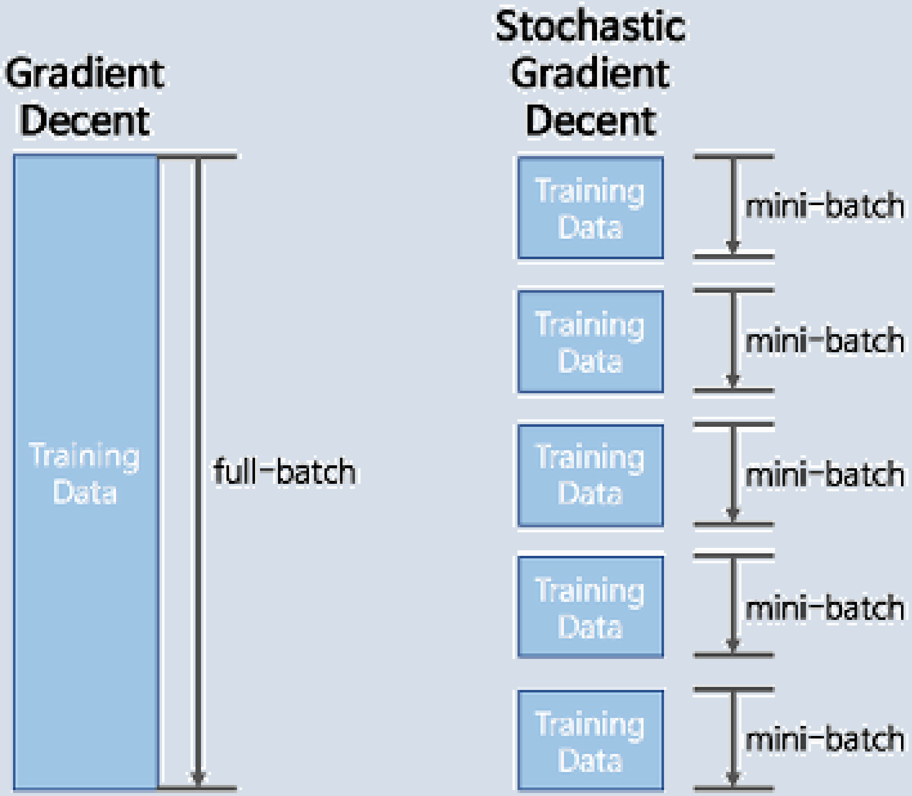


배치 경사하강법(Batch Gradient Descent)

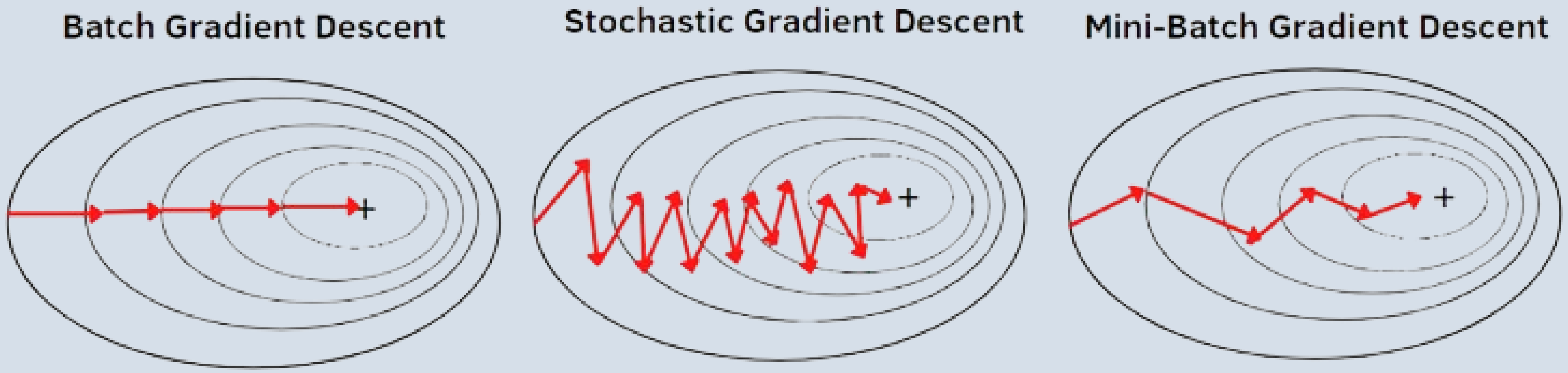


Mini-batch Gradient Descent(미니배치 경사하강법)





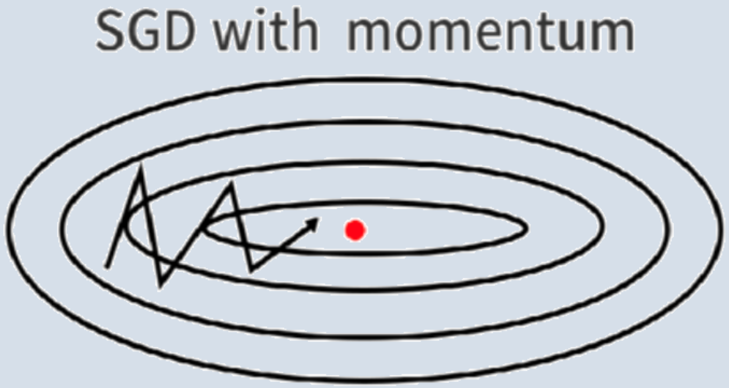
최적화 함수(Optimizer) 종류



출처 : https://ml-explained.com

미니배치 경사하강법
(Mini-Batch Gradient Descent)

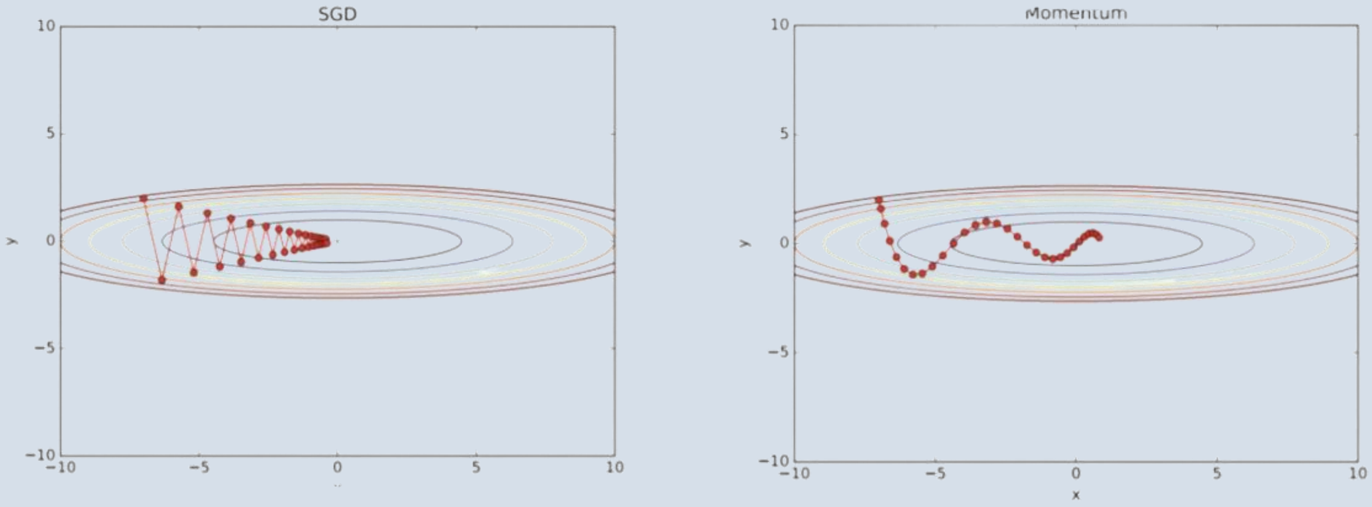
확률적으로 선택된 일부 데이터들을
이용해 업데이트

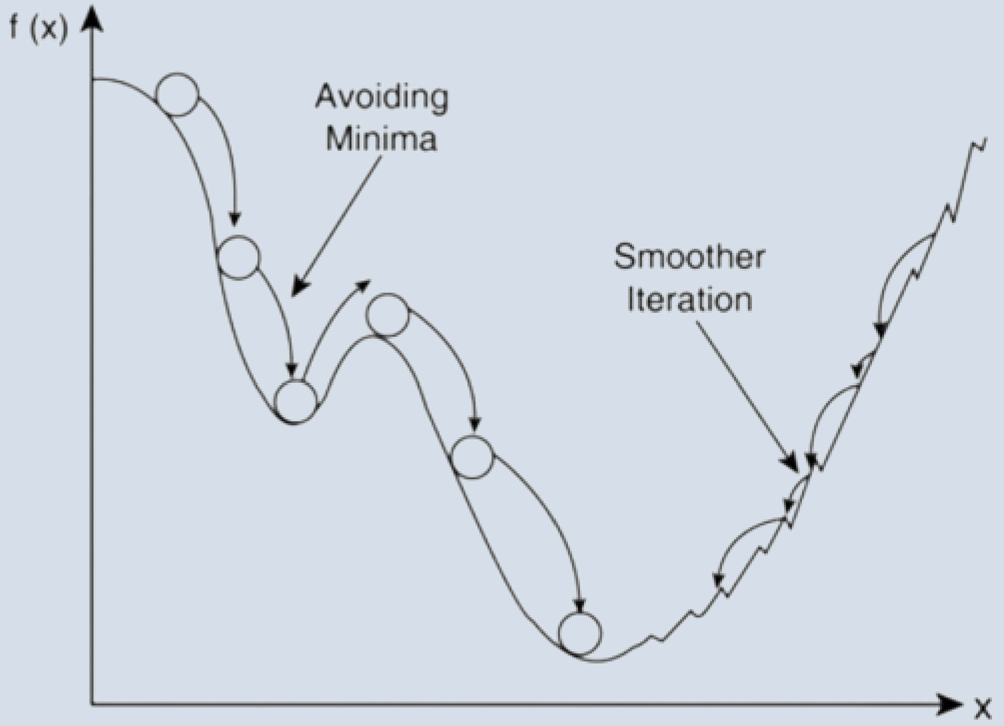


모멘텀
(Momentum)

경사 하강법에 **관성**을 적용해 업데이트
현재 batch뿐만 아니라 **이전 batch 데이터의 학습 결과**도 반영

최적화 함수(Optimizer) 종류





모멘텀
(Momentum)

- **가중치를 업데이트 하기 전, 이전 방향을 참고하여 업데이트**

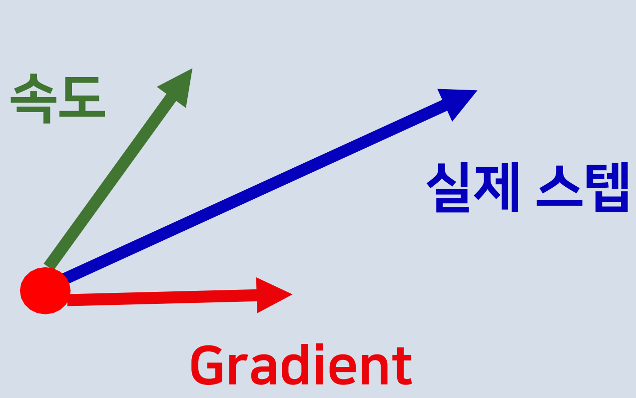
v : 속도 벡터

β : 모멘텀 계수(일반적으로 0.9로 설정)

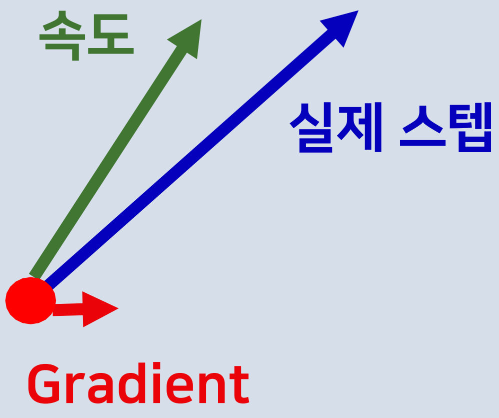
$$v := \beta v + (1-\beta)\frac{\partial L}{\partial w}$$

$$w := w - \alpha v$$

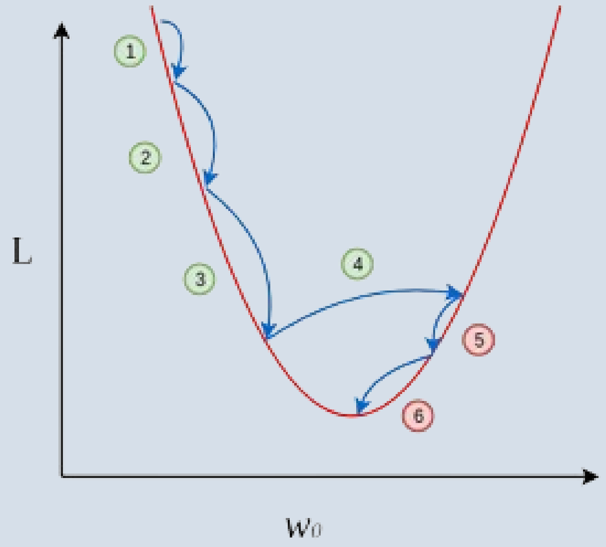
최적화 함수(Optimizer) 종류

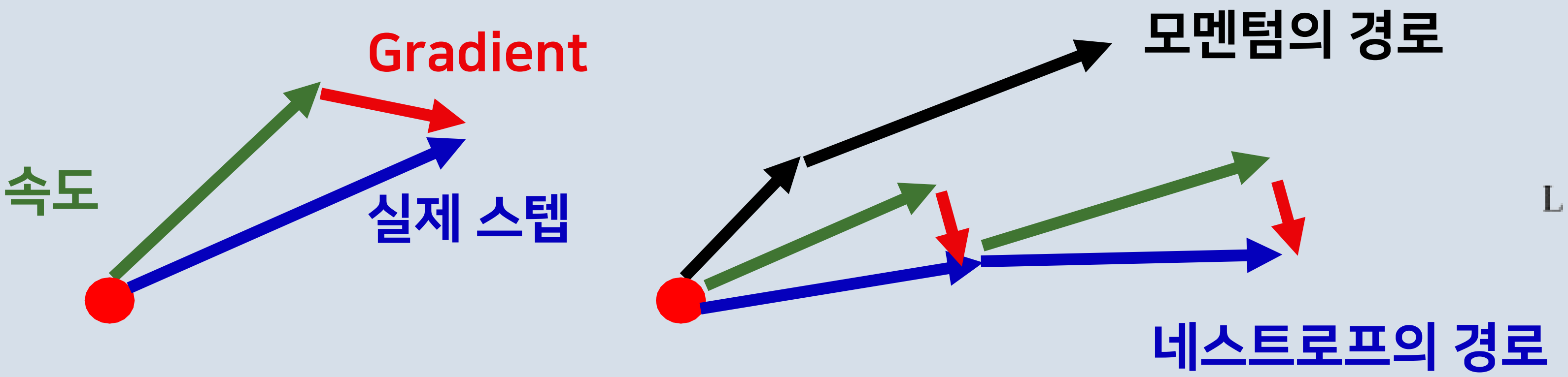


모멘텀
(Momentum)

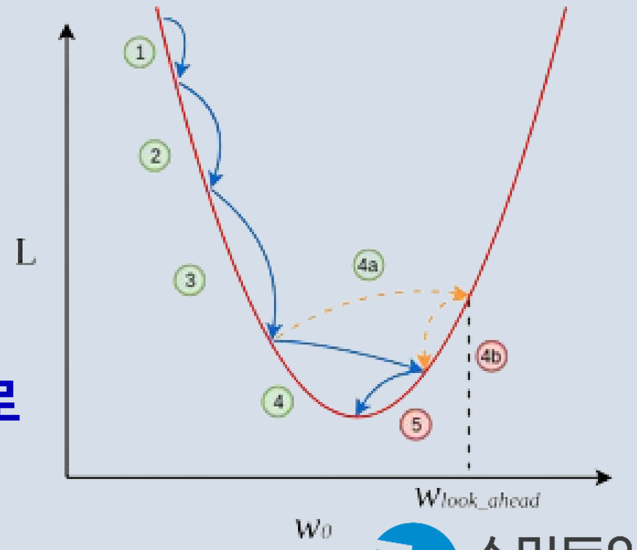


오버슈팅
(Over shooting)





NAG
(Nesterov Accelerated Gradient)



NAG
(Nesterov Accelerated Gradient)

- w, b값 업데이트 시 모멘텀 방식으로 먼저 더한 다음 계산
- 미리 해당 방향으로 이동한다고 가정하고 기울기를 계산해본 뒤 실제 업데이트 반영
- 불필요한 이동을 줄일 수 있음

$$v := \beta v + \alpha \frac{\partial L(w-\beta v)}{\partial w}$$

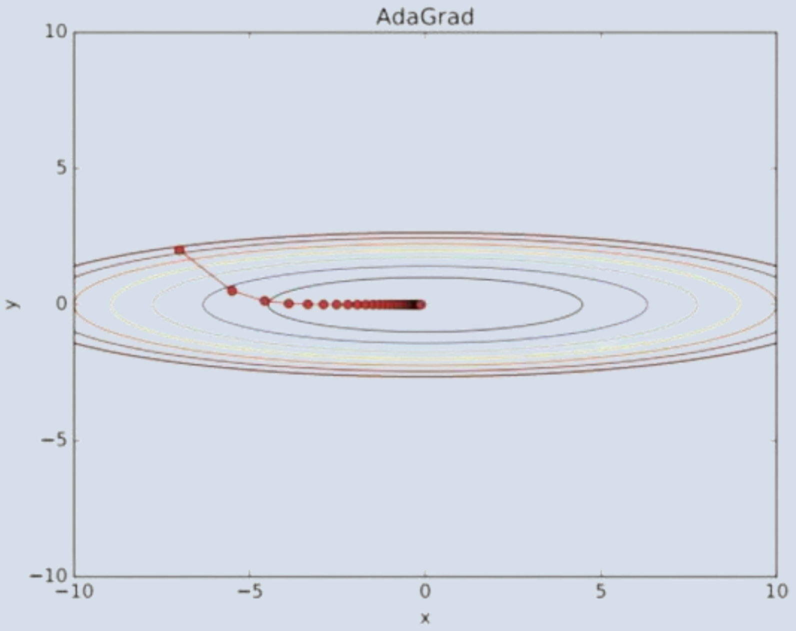
$$w := w - v$$

v : 속도 벡터

β : 모멘텀 계수(일반적으로 0.9로 설정)

$\frac{\partial L(w-\beta v)}{\partial w}$: 모멘텀에 의해 예측 된 위치에서 계산한 Gradient

최적화 함수(Optimizer) 종류



AdaGrad

(Adaptive Gradient)

- 학습을 진행하면서 학습률을 점차 줄여가는 방법
- 처음에는 크게 학습하다가 조금씩 작게 학습

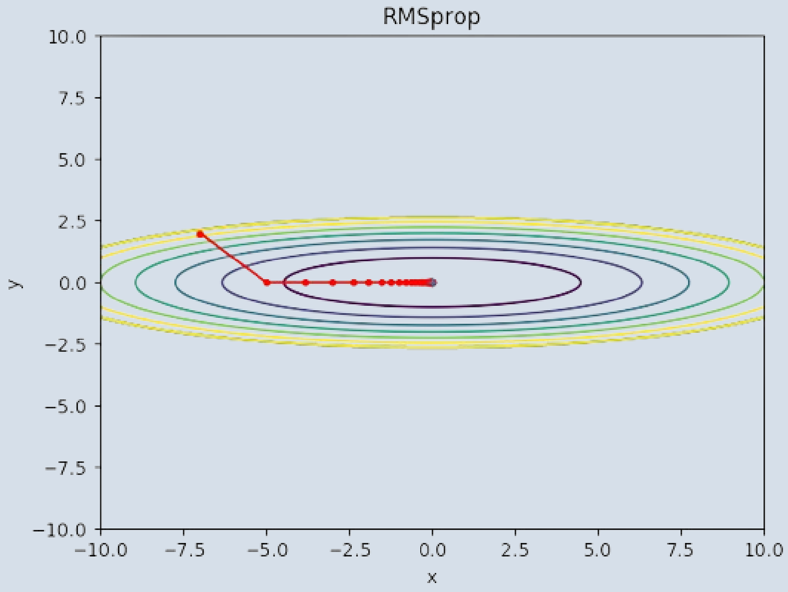
$$G_t = G_{t-1} + \left(\frac{\partial L}{\partial w_i}\right)^2$$

$$w := w - \frac{\alpha}{\sqrt{G_t + \epsilon}} \frac{\partial L}{\partial w}$$

G_t: 각 파라미터의 Gradient 제곱의 누적 합

ϵ: 수치적 안정성을 위한 작은 값(일반적으로 10^{-8})

최적화 함수(Optimizer) 종류



$$E[g^2]_t = E[g^2]_{t-1} + (1-\beta)\left(\frac{\partial L}{\partial w_i}\right)^2$$

$$w := w - \frac{\alpha}{\sqrt{E[g^2]_t + \epsilon}} \frac{\partial L}{\partial w}$$

RMSProp
(Root Mean Square Propagation)

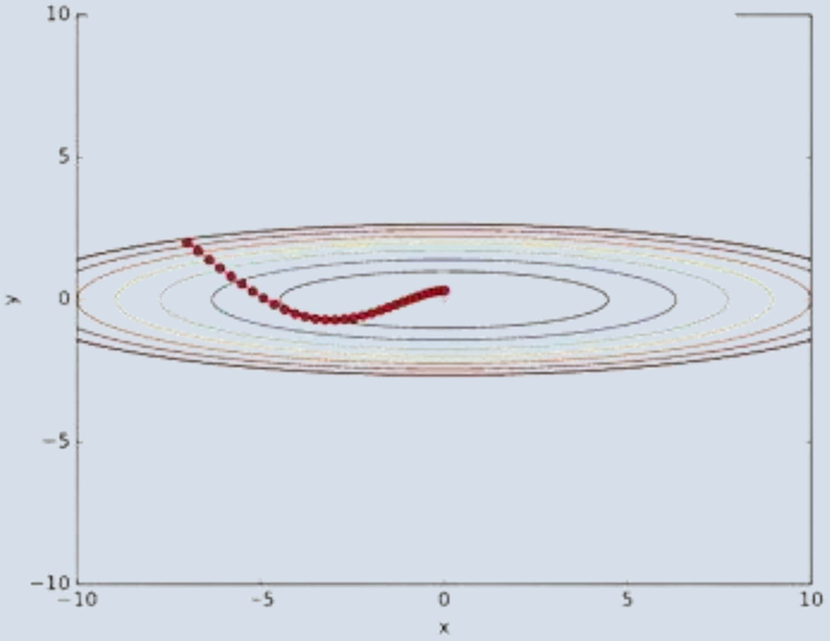
$E[g^2]_t$: Gradient 제곱의 지수 이동 평균

β : 지수 이동 평균의 감쇠 계수(일반적으로 0.9)

ϵ: 수치적 안정성을 위한 작은 값(일반적으로 10^{-8})

- Adagrad와 동일하게 학습을 진행하면서 학습률을 점차 줄여가는 방법
- 최소값을 찾기전 학습이 멈추는 Adagrad의 단점을 지수이동 평균을 도입해서 해결
- 지수 이동 평균 : 최근 학습한 수치의 영향력은 높이고 과거 학습한 수치의 영향력은 낮추는 방식

최적화 함수(Optimizer) 종류



Adam
(Adaptive Moment Estimation)

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)\frac{\partial L}{\partial w}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2)\left(\frac{\partial L}{\partial w}\right)^2$$

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}$$

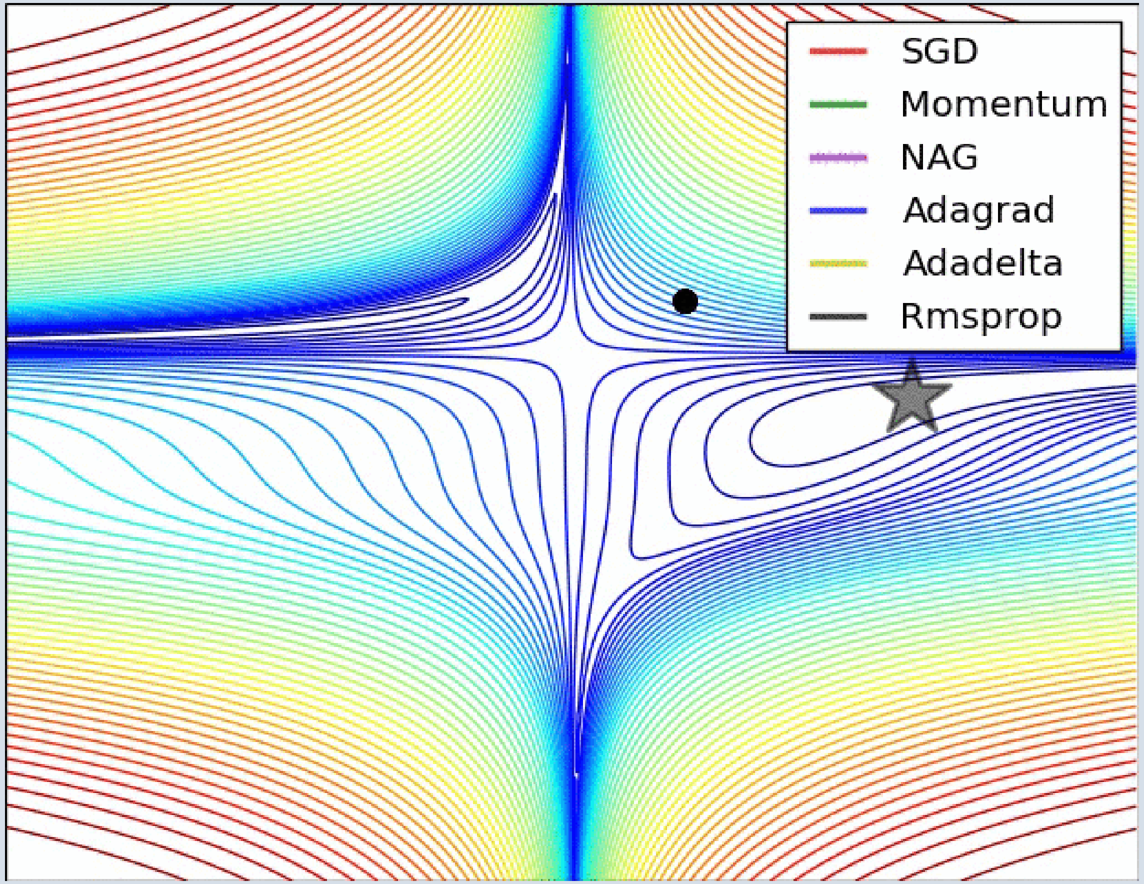
$$\hat{v}_t = \frac{v_t}{1-\beta_2^t}$$

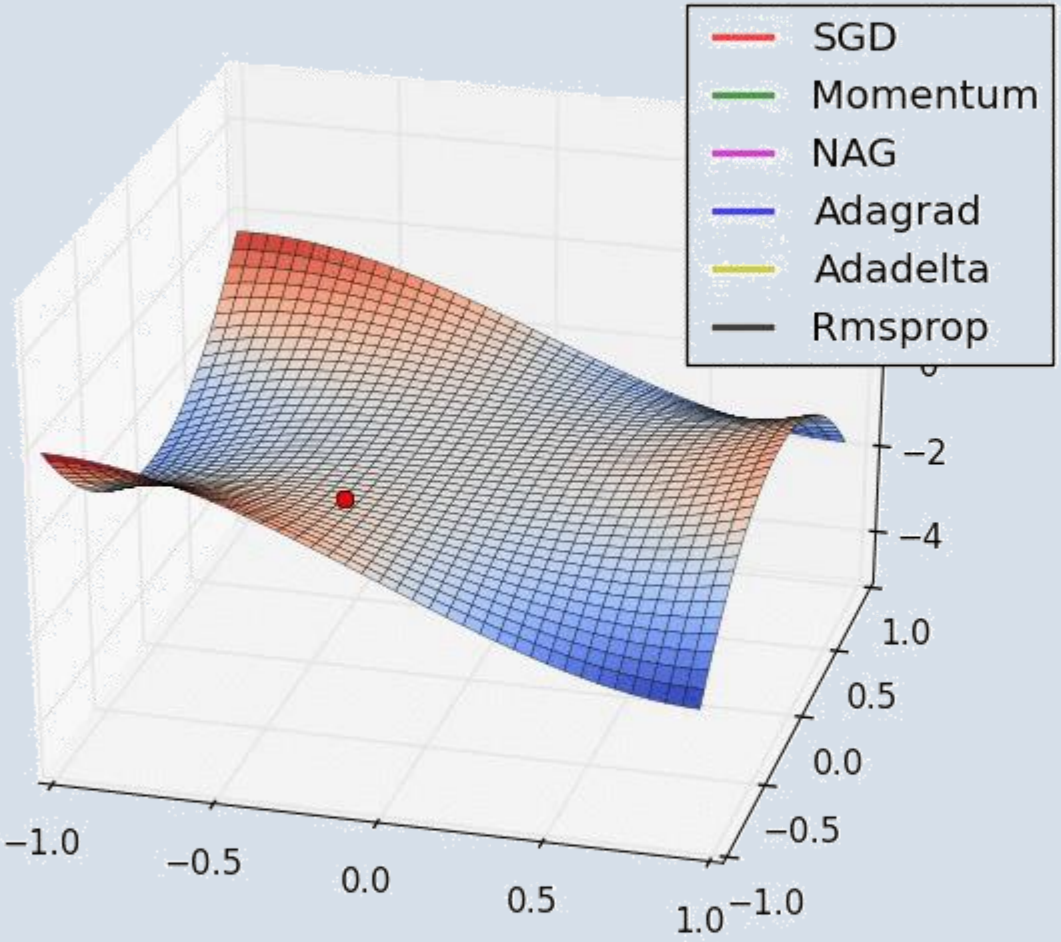
$$w := w - \alpha\frac{\hat{m}_t}{\sqrt{\hat{v}_t+\epsilon}}$$

m_t: Gradient 지수 이동 평균 (1차 모멘트)
v_t: Gradient 제곱의 지수 이동 평균 (2차 모멘트)
β_1 : 1차 모멘트의 감쇠 계수(일반적으로 0.9로 설정)
β_2 : 1차 모멘트의 감쇠 계수(일반적으로 0.999로 설정)
$\hat{m}_t$: 편향 보정된 1차 모멘트
$\hat{v}_t$: 편향 보정된 2차 모멘트
ϵ: 수치적 안정성을 위한 작은 값(일반적으로 10^{-8})

- 관성 방향으로 움직이는 Momentum과 보폭을 조절하며 움직이는 AdaGrad의 특성을 하나로 합친 최적화 함수.
- 현재 보편적으로 사용하는 최적화 함수이며, 성능적인 측면에서 가장 나은 최적화 함수라 할 수 있음

최적화 함수(Optimizer) 종류



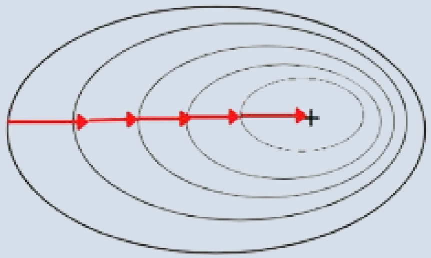


출처 : https://cs231n.github.io/neural-networks-3/

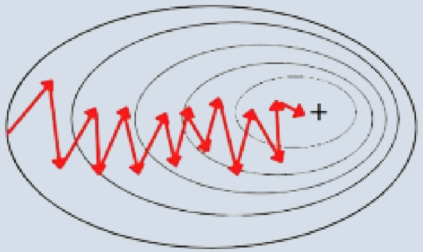
Deep Learning

최적화 함수(Optimizer) 종류

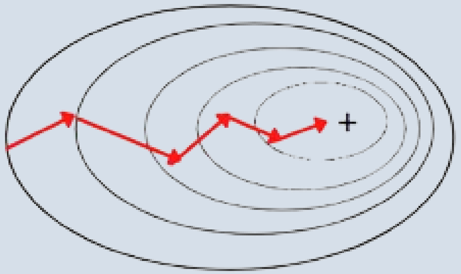
Batch Gradient Descent



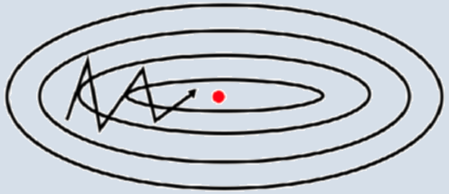
Stochastic Gradient Descent

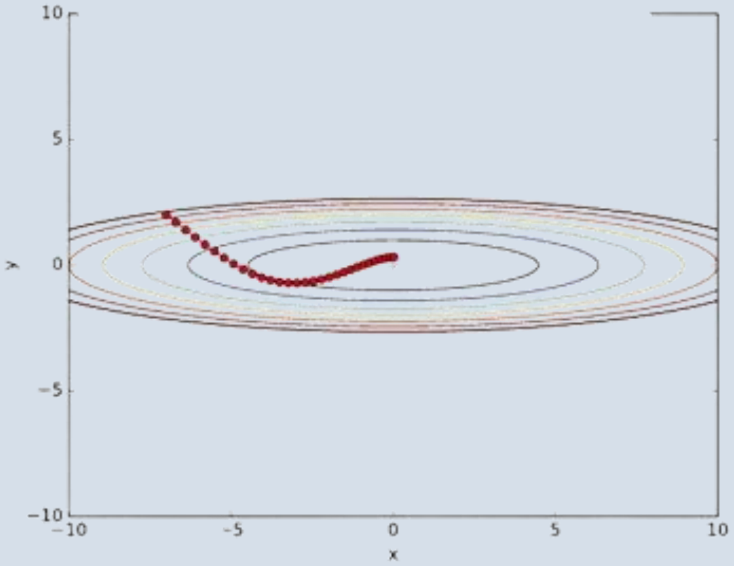


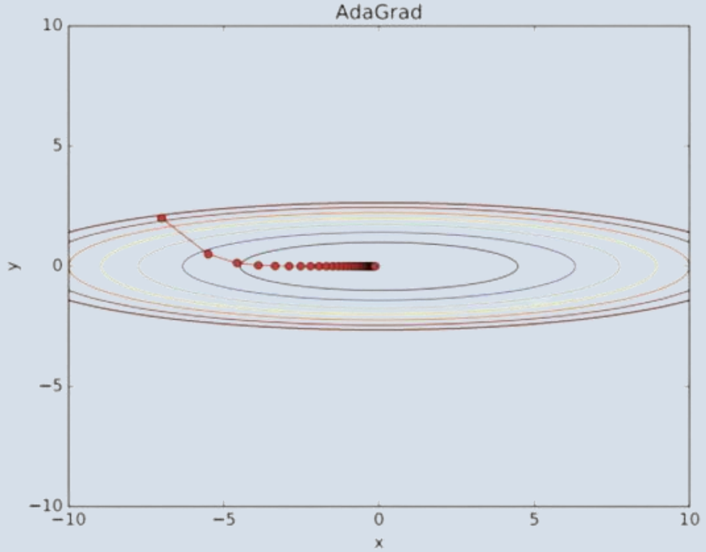
Mini-Batch Gradient Descent

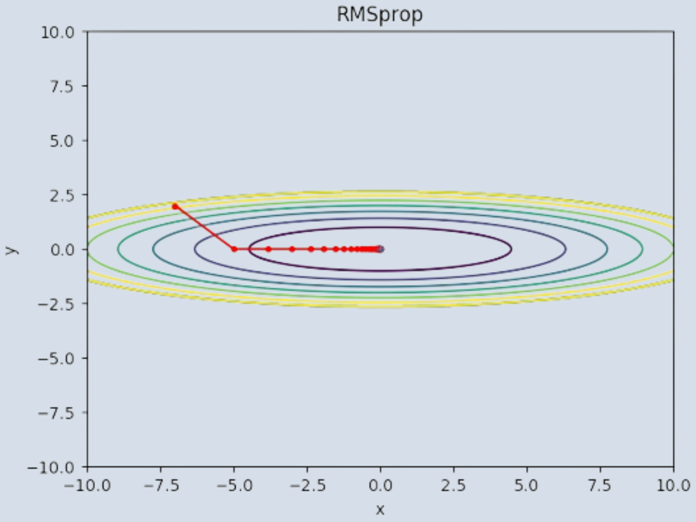


SGD with momentum

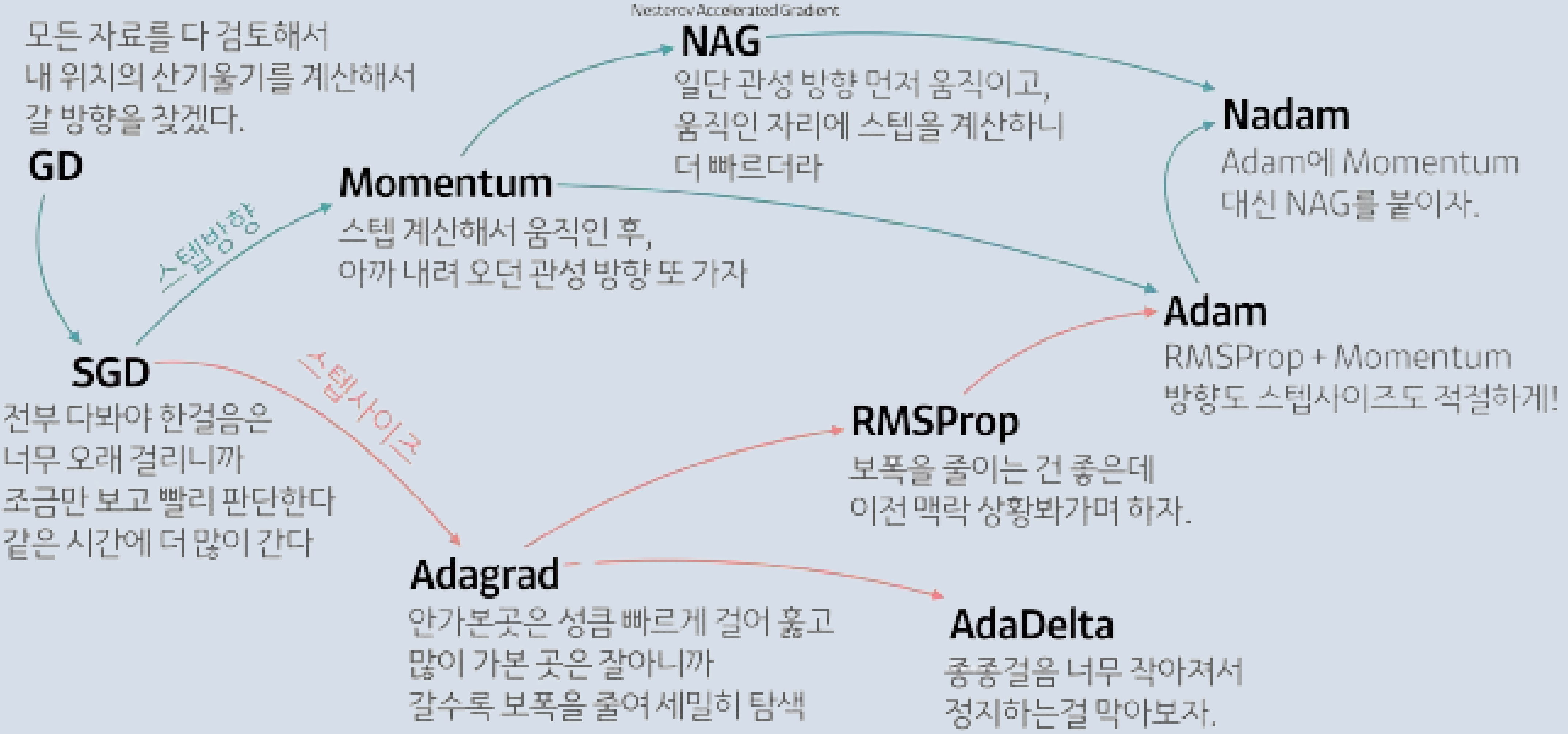








최적화 함수(Optimizer) 종류



출처 : https://www.slideshare.net/slideshow/ss-79607172/79607172

최적화 함수(Optimizer) 종류

Keras

```
from tensorflow.keras import optimizers

opti = optimizers.SGD(learning_rate=0.01, momentum=0.9)

model.compile(loss='mse', optimizer=opti, metrics=['acc'])
```

Momentum

```
from tensorflow.keras import optimizers

opti = optimizers.SGD(learning_rate=0.01, momentum=0.9, nesterov=True)

model.compile(loss='mse', optimizer=opti, metrics=['acc'])
```

NAG

```
model.compile(loss ="mse", optimizer="Adam", metrics=["acc"])
```

Adam

Adagrad, RMSprop, Adam 등은 이름으로 지정 가능

최적화 함수(Optimizer) 종류

$$w := w - \alpha \frac{\partial L}{\partial w}$$

Batch Gradient Descent

- 모든 데이터 포인트에 대해 손실 함수의 Gradient 계산 후 가중치 업데이트
- 업데이트가 안정적이며, 각 스텝에서의 변화가 크지 않음
- 전체 데이터셋을 로드해야 하므로, 메모리 사용량이 크고, 속도가 느림

$$w := w - \alpha \frac{\partial L_i}{\partial w}$$

i : 데이터셋의 각 샘플

Stochastic Gradient Descent

- 각 데이터 포인트에 대해 개별적으로 Gradient 계산 후 가중치 업데이트
- 업데이트 스탭이 빠르며, 메모리 사용량이 적음
- 업데이트가 불안정할 수 있고, 각 스텝의 변화가 큼

최적화 함수(Optimizer) 종류

$$w := w - \alpha \frac{\partial L_{mini-batch}}{\partial w}$$

Mini-Batch Gradient Descent

$L_{mini-batch}$: 미니 배치에 대한 손실 함수

$\frac{\partial L_{mini-batch}}{\partial w}$: w에 대한 $L_{mini-batch}$의 Gradient

- 데이터셋을 여러 작은 배치로 나누어 각 배치마다 Gradient 계산 후 가중치 업데이트
- 확률적 장점 + 배치 장점
- 배치 크기 설정이 중요하며, 배치 크기에 따라 Gradient 변동성이 다름

$$v := \beta v + (1 - \beta) \frac{\partial L}{\partial w}$$

$$w := w - \alpha v$$

Momentum

v : 속도 벡터(Momentum term)

β : 모멘텀 계수(일반적으로 0.9로 설정)

- 이전 Gradient에 기반한 업데이트 방향에 일정한 관성을 부여하여, 가중치가 최적값에 더 빠르고 안정적으로 수렴하도록 함
- 속도 벡터(v)를 활용하여, Gradient 방향과 크기에 대한 과거 정보를 포함
- 모멘텀 계수(β) 조정 필요
- 추가적인 속도 벡터(v)를 관리해야 함

최적화 함수(Optimizer) 종류

$$v := \beta v + \alpha \frac{\partial L(w-\beta v)}{\partial w}$$

$$w := w - v$$

v : 속도 벡터(Momentum term)

β : 모멘텀 계수(일반적으로 0.9로 설정)

$\frac{\partial L(w-\beta v)}{\partial w}$: 모멘텀에 의해 예측 된 위치에서 계산한 Gradient

Nesterov Accelerated Gradient(NAG)

- 모멘텀에 의해 예측된 위치에서 Gradient 계산(모멘텀을 적용한 후의 위치에서 Gradient를 계산하여 더 빠르고 정확하게 수렴)
- 모멘텀 상위 버전
- 여전히 학습률(α)과 모멘텀 계수(β) 조정 필요하며, 더 복잡해짐

$$G_t = G_{t-1} + \left(\frac{\partial L}{\partial w_i}\right)^2$$

$$w := w - \frac{\alpha}{\sqrt{G_t+\epsilon}} \frac{\partial L}{\partial w}$$

G_t: 각 파라미터의 Gradient 제곱의 누적 합

ϵ: 수치적 안정성을 위한 작은 값(일반적으로 10^{-8})

AdaGrad

- Gradient 크기에 따라 학습률을 조절하여, 자주 업데이트 되는 파라미터의 학습률을 줄이고, 적게 업데이트 되는 학습률을 유지
- 자주 업데이트 되는 파라미터의 변화가 안정화됨
- 감쇠(시간이 지나면서 학습률이 너무 작아져, 최적화가 정체될 수도 있음)

최적화 함수(Optimizer) 종류

$$E[g^2]_t = \beta E[g^2]_{t-1} + (1-\beta)\left(\frac{\partial L}{\partial w_i}\right)^2$$

$$\Delta w_t = -\frac{\sqrt{E[\Delta w^2]_{t-1}+\epsilon}}{\sqrt{E[g^2]_t+\epsilon}}\frac{\partial L}{\partial w}$$

$$E[\Delta w^2]_t = \beta E[\Delta w^2]_{t-1} + (1-\beta)(\Delta w_t)^2$$

$$w := w + \Delta w_t$$

AdaDelta

- Adagrad의 변형으로 학습률을 적응적으로 조정하여 학습률이 작아지는 문제(감쇠)를 해결
- 지수 이동 평균의 감쇠 계수(β)의 적절한 조절 필요

$E[g^2]_t$: Gradient 제곱의 지수 이동 평균

$E[\Delta w^2]_t$: 업데이트 값 Δw 제곱의 지수 이동 평균

β : 지수 이동 평균의 감쇠 계수(일반적으로 0.9)

ϵ: 수치적 안정성을 위한 작은 값(일반적으로 10^{-8})

$$E[g^2]_t = E[g^2]_{t-1} + (1-\beta)\left(\frac{\partial L}{\partial w_i}\right)^2$$

$$w := w - \frac{\alpha}{\sqrt{E[g^2]_t+\epsilon}}\frac{\partial L}{\partial w}$$

RMSProp
(Root Mean Square Propagation)

$E[g^2]_t$: Gradient 제곱의 지수 이동 평균

β : 지수 이동 평균의 감쇠 계수(일반적으로 0.9)

ϵ: 수치적 안정성을 위한 작은 값(일반적으로 10^{-8})

- Adagrad의 문제를 해결하기 위해, Gradient의 제곱을 지수 이동 평균으로 누적하여, 각 파라미터 학습률을 조정함
- 안정성과 효율성이 높음
- 지수 이동 평균의 감쇠 계수(β)의 조절 필요

최적화 함수(Optimizer) 종류

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)\frac{\partial L}{\partial w}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2)\frac{\partial L}{\partial w}^2$$

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t}$$

$$w := w - \alpha\frac{\hat{m}_t}{\sqrt{\hat{v}_t}+\epsilon}$$

- RMSProp + Momentum
- 각 파라미터에 대해 적응형 학습률을 사용하며, 모멘텀을 통해 수렴속도를 높임
- 안정성과 효율성이 크며, 편향 보정이 포함됨
- 여러 하이퍼 파라미터(β_1, β_2, α)의 적절한 조정 필요

m_t: Gradient 지수 이동 평균 (1차 모멘트)

v_t: Gradient 제곱의 지수 이동 평균 (2차 모멘트)

β_1 : 1차 모멘트의 감쇠 계수(일반적으로 0.9로 설정)

β_2 : 1차 모멘트의 감쇠 계수(일반적으로 0.999로 설정)

$\hat{m}_t$: 편향 보정된 1차 모멘트

$\hat{v}_t$: 편향 보정된 2차 모멘트

ϵ: 수치적 안정성을 위한 작은 값(일반적으로 10^{-8})

Adam(Adaptive Moment Estimation)