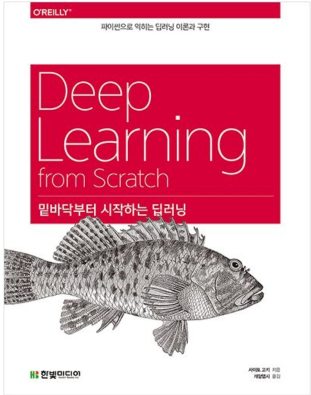
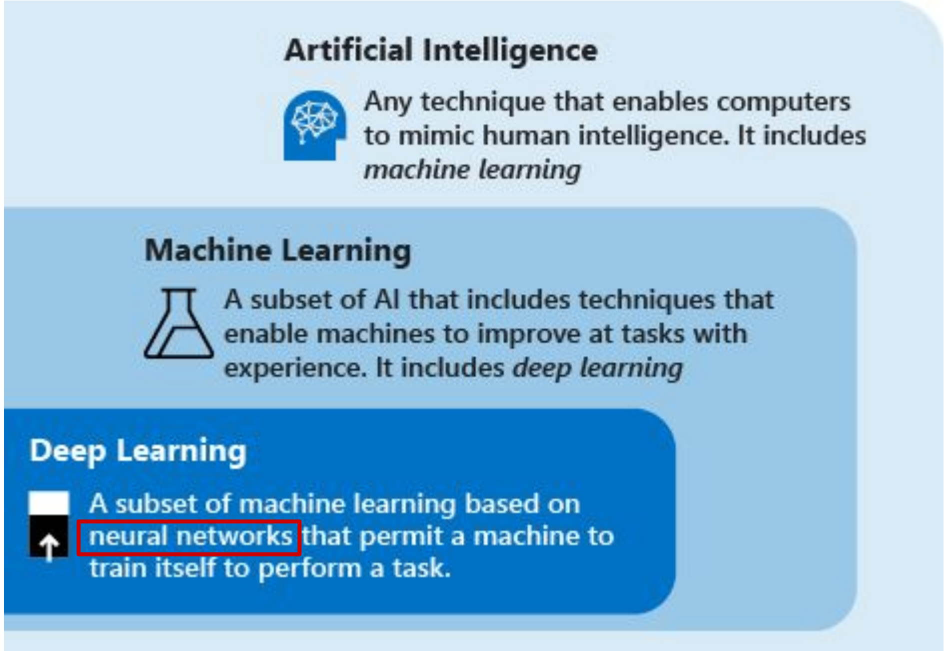
딥 러닝(Deep learning)



딥러닝 개념


Artificial Intelligence
Any technique that enables computers to mimic human intelligence. It includes *machine learning*
Machine Learning
A subset of AI that includes techniques that enable machines to improve at tasks with experience. It includes *deep learning*
Deep Learning
A subset of machine learning based on neural networks that permit a machine to train itself to perform a task.

인공 신경망의 역사

1957년
단층 신경망(perceptron)

1969년
Marvin Minsky의 "Perceptron"
에서 XOR 문제를 풀지 못함

1986년
다층 신경망(MLP),
오류 역전파(back propagation)

1990년
Big data - 수 만의 Neuron,
수 백만의 connection

2006년
Geoffrey Hinton
"A fast learning Algorithm for
deep beliefs Nets

2012년
AlexNet in ImageNet

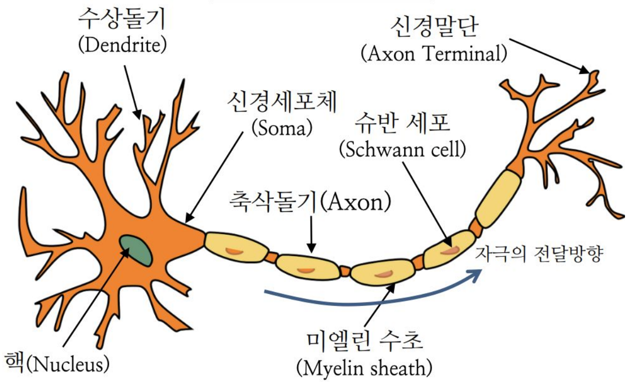
2016년
AlphaGo

혁신적 알고리즘 제안 + 컴퓨팅 파워 발전 + 데이터 폭증

인공 신경망(Artificial Neural Network)

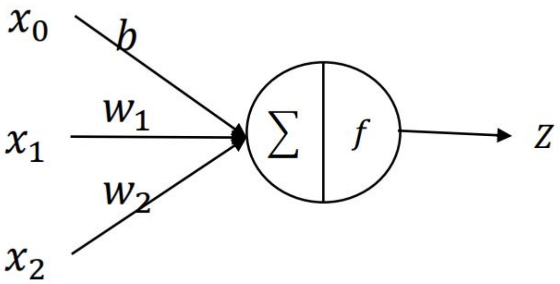
인간의 뉴런

- 시냅스(synapse)를 통해 뉴런간 신호를 전달
- 각 뉴런은 수상돌기(dendrite)를 통해 입력 신호를 받음
- 입력 신호가 특정크기(threshold) 이상인 경우에만 활성화 되어 축삭돌기(axon)을 통해 다음 뉴런으로 전달



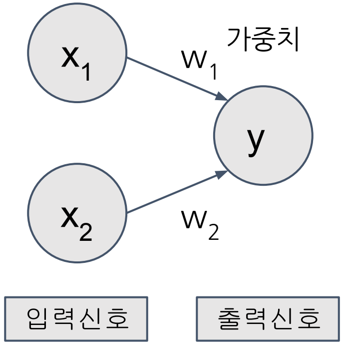
인공 뉴런

- 각 뉴런은 가중치가 있는 입력 신호를 받음
- 입력 신호는 모두 더한 후 활성화 함수(activation function)을 적용함
- 활성화 함수의 값이 특정 값 이상인 경우에만 다음 노드의 입력값으로 전달



퍼셉트론(perceptron)

- 다수의 신호를 입력으로 받아 하나의 신호를 출력
- 입력신호가 뉴런에 보내질 때 각각 가중치가 곱해짐 (입력값을 조절)
- 신호의 총합이 정해진 한계 즉 임계값 를 넘어설 때만 1을 출력



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

퍼셉트론 구현

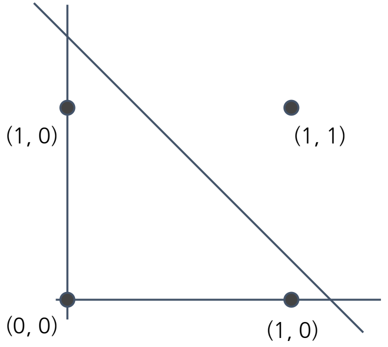
- AND Gate

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

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

b(bias, 편향) 도입

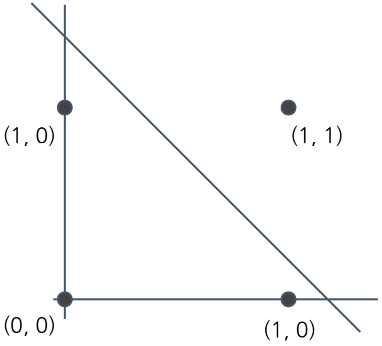
$$y = \begin{cases} 0 & (b + w_1x_1 + w_2x_2 \le 0) \\ 1 & (b + w_1x_1 + w_2x_2 > 0) \end{cases}$$



퍼셉트론 구현

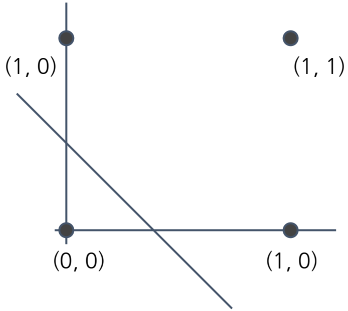
- NAND Gate

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



- OR Gate

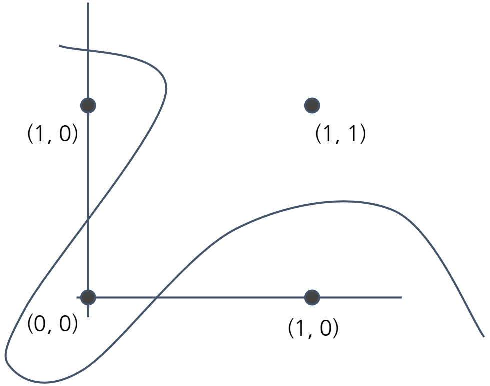
x_1	x_2	y
0	0	0
1	0	1
0	1	1
1	1	1



XOR

- XOR Gate

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

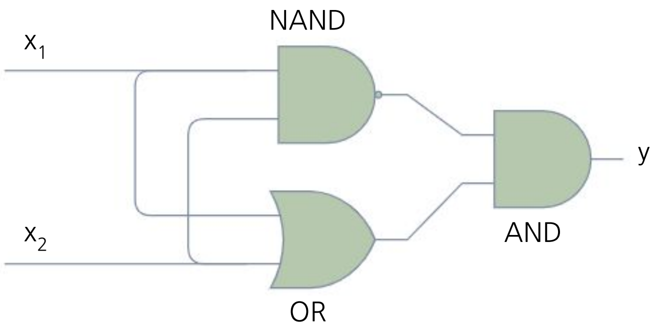


➡️ 1969년 Marvin Minsky(founder of the MIT AI Lab, 1969)의 "Perceptron" 에서 XOR 문제를 풀지 못함

"We need to use MLP, No one on earth had found a viable way to train MLPs good enough to learn such simple functions"

다층 퍼셉트론

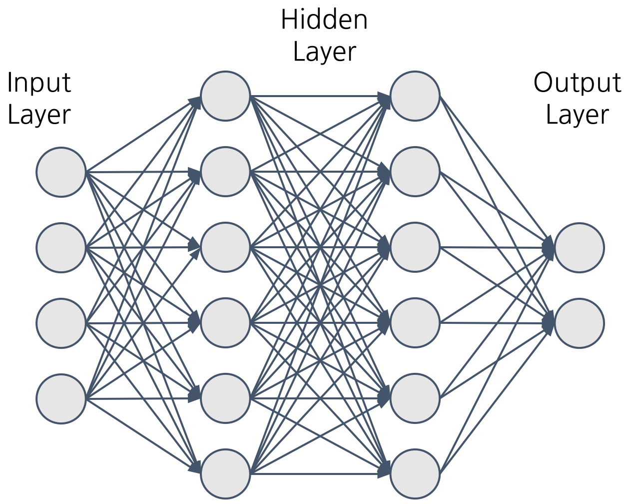
- XOR 문제는 층을 쌓아 다층 퍼셉트론으로 해결



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

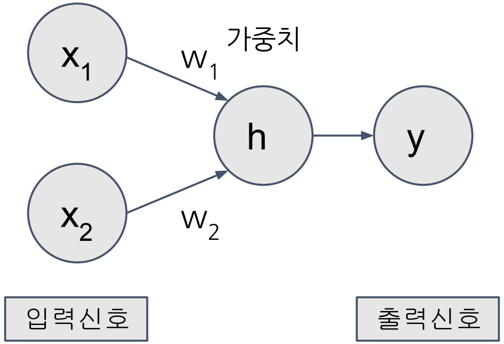
신경망

- Multi layer perceptron (MLP) - 입력층, 은닉층, 출력층으로 이루어짐
  - 입력층 : 입력 개수
  - 은닉층 : 정해진 규칙은 없으나 어느 정도 깊어야 좋다
  - 출력층 : 각 task에 맞게



활성화 함수(Activation function)

- 입력 신호의 총합을 출력 신호로 변환 하는 함수



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

⇩

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

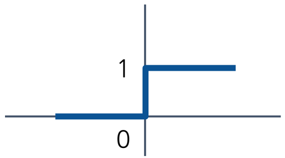
$$h(x) = \begin{cases} 0 & (x \le 0) \\ 1 & (x > 0) \end{cases}$$

활성화 함수(Activation function)

- discrete ➡️ continuous

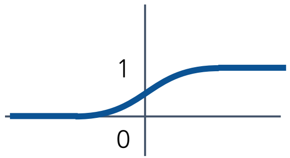
$$y = h(b + w_1 x_1 + w_2 x_2)$$

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



시그모이드 함수

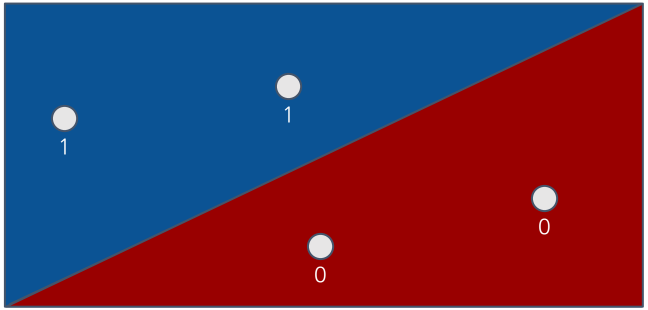
$$h(x) = \frac{1}{1 + e^{-x}}$$



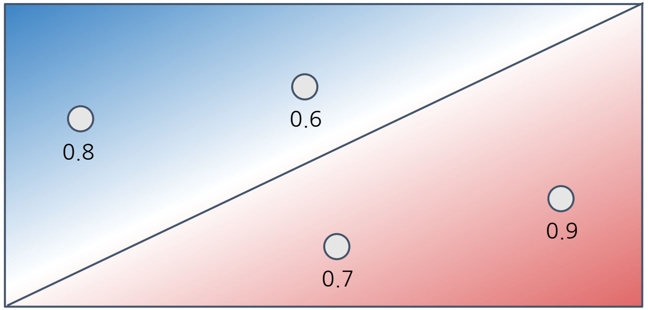
활성화 함수(Activation function)

- 시그모이드 함수 : 얼마나 확실한가? 확률로 표시할 수 있다.

스텝 함수



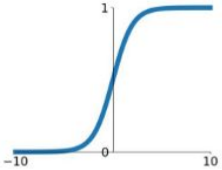
시그모이드 함수



활성화 함수(Activation function)

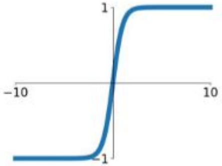
Sigmoid

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



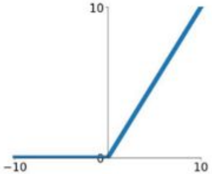
tanh

$$\tanh(x)$$



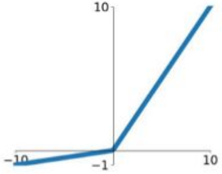
ReLU

$$\max(0, x)$$



Leaky ReLU

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

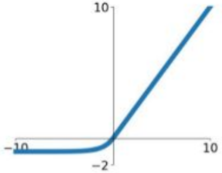


Maxout

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

ELU

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

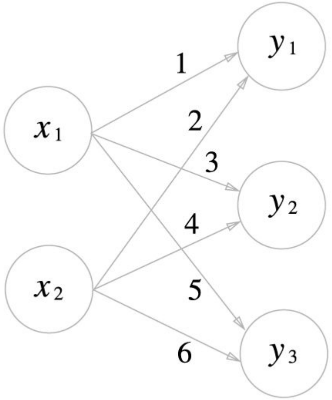


Sigmoid, ReLU

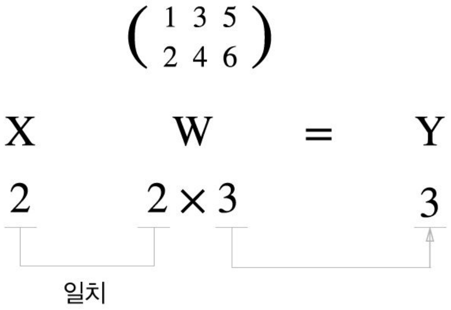
실습

행렬 곱

실습

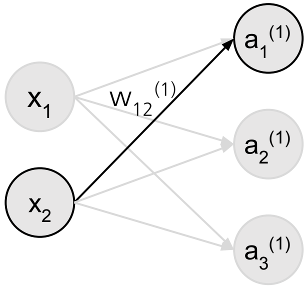


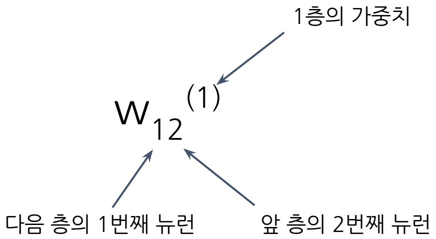
x_1
x_2
y_1
y_2
y_3
1
2
3
4
5
6



$\begin{pmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{pmatrix}$
X W = Y
2 2 × 3 3
일치

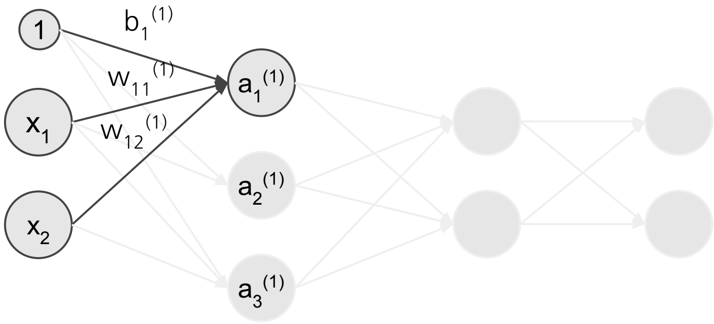
신경망 표기법





□ 다양한 표기 방법이 존재

수식으로 표현



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

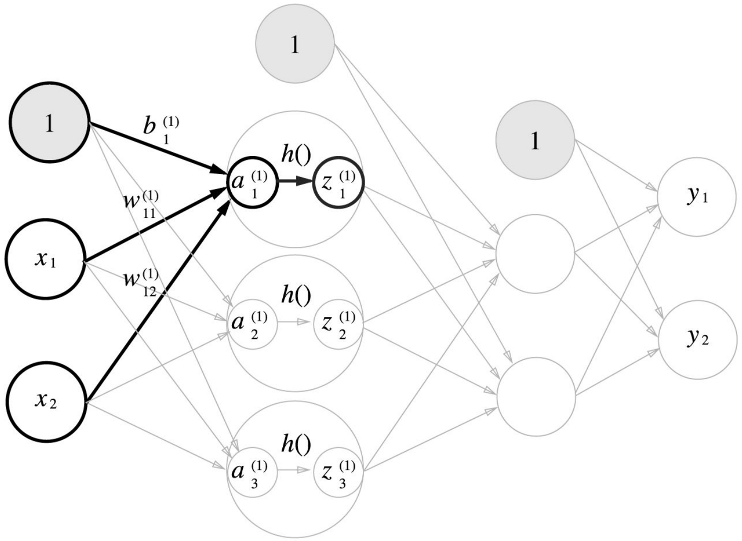
$$a_2^{(1)} = w_{21}^{(1)} x_1 + w_{22}^{(1)} x_2 + b_2^{(1)}$$

$$a_3^{(1)} = w_{31}^{(1)} x_1 + w_{32}^{(1)} x_2 + b_3^{(1)}$$

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

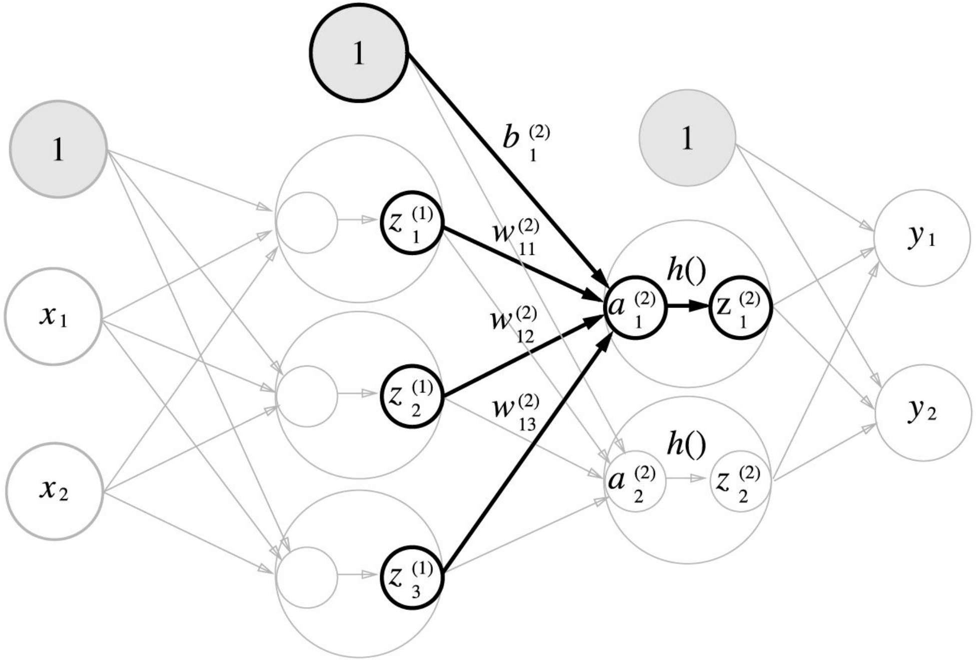
$$\left\{ \begin{aligned} &\mathbf{A}^{(1)} = \left(a_1^{(1)} \ \ a_2^{(1)} \ \ a_3^{(1)}\right), \ \mathbf{X} = (x_1 \ \ x_2) \\ &\mathbf{B}^{(1)} = \left(b_1^{(1)} \ \ b_2^{(1)} \ \ b_3^{(1)}\right) \\ &\mathbf{W}^{(1)} = \begin{pmatrix} w_{11}^{(1)} & w_{21}^{(1)} & w_{31}^{(1)} \\ w_{12}^{(1)} & w_{22}^{(1)} & w_{32}^{(1)} \end{pmatrix} \end{aligned} \right.$$

MultiLayer Perceptron - layer 1



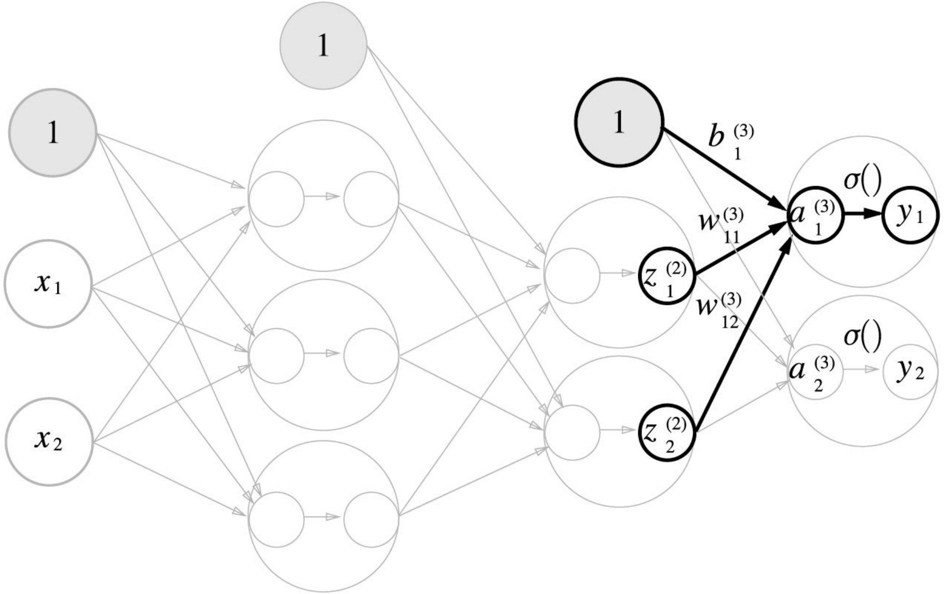
실습

MultiLayer Perceptron - layer 2



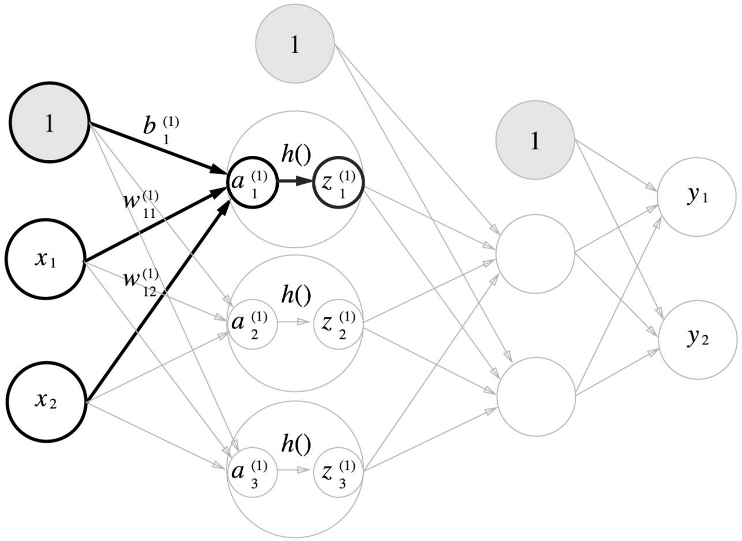
실습

MultiLayer Perceptron - layer 3



실습

MultiLayer Perceptron

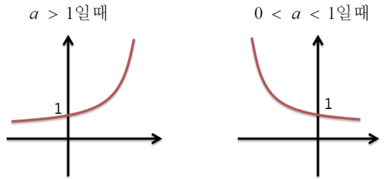


MLP

실습

소프트맥스(softmax)

- 지수 함수(exponential function) : 모든 output은 양수



- 클래스가 여러개인 경우, 즉 multiple classification 에서 output
- softmax 함수의 출력값은 0~1 사이값 + 출력의 총합은 1 ➡️ 확률로 해석이 가능

클래스	레이블	출력	레이블	출력	소프트맥스
A	0	$\frac{0}{0+1+2}$	0	$\frac{1}{1+0-1}$	$\frac{e^0}{e^{-1}+e^0+e^1}$
B	1	$\frac{1}{0+1+2}$	1	$\frac{0}{1+0-1}$	$\frac{e^1}{e^{-1}+e^0+e^1}$
C	2	$\frac{2}{0+1+2}$	-1	$\frac{-1}{1+0-1}$	$\frac{e^{-1}}{e^{-1}+e^0+e^1}$

$$\mathrm{Softmax}(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$$

소프트맥스

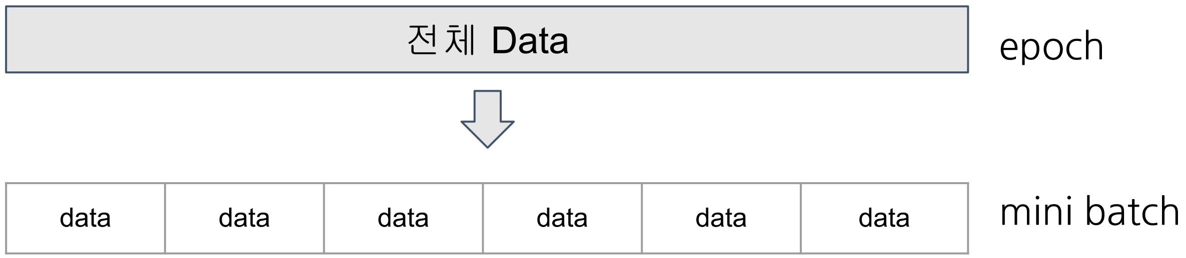
실습

MNIST 예측

실습

에폭 (epoch)과 배치 (batch)

- 학습데이터가 많은 경우 한꺼번에 데이터를 학습 할 수 없기 때문에 데이터를 나누는 작업을 한다. 즉 데이터 전체에 대해서 학습하는 것을 epoch이라고 하고 각 epoch에서는 배치(묶음 데이터)로 나누어 학습한다.



Batch

실습