Chapter 3

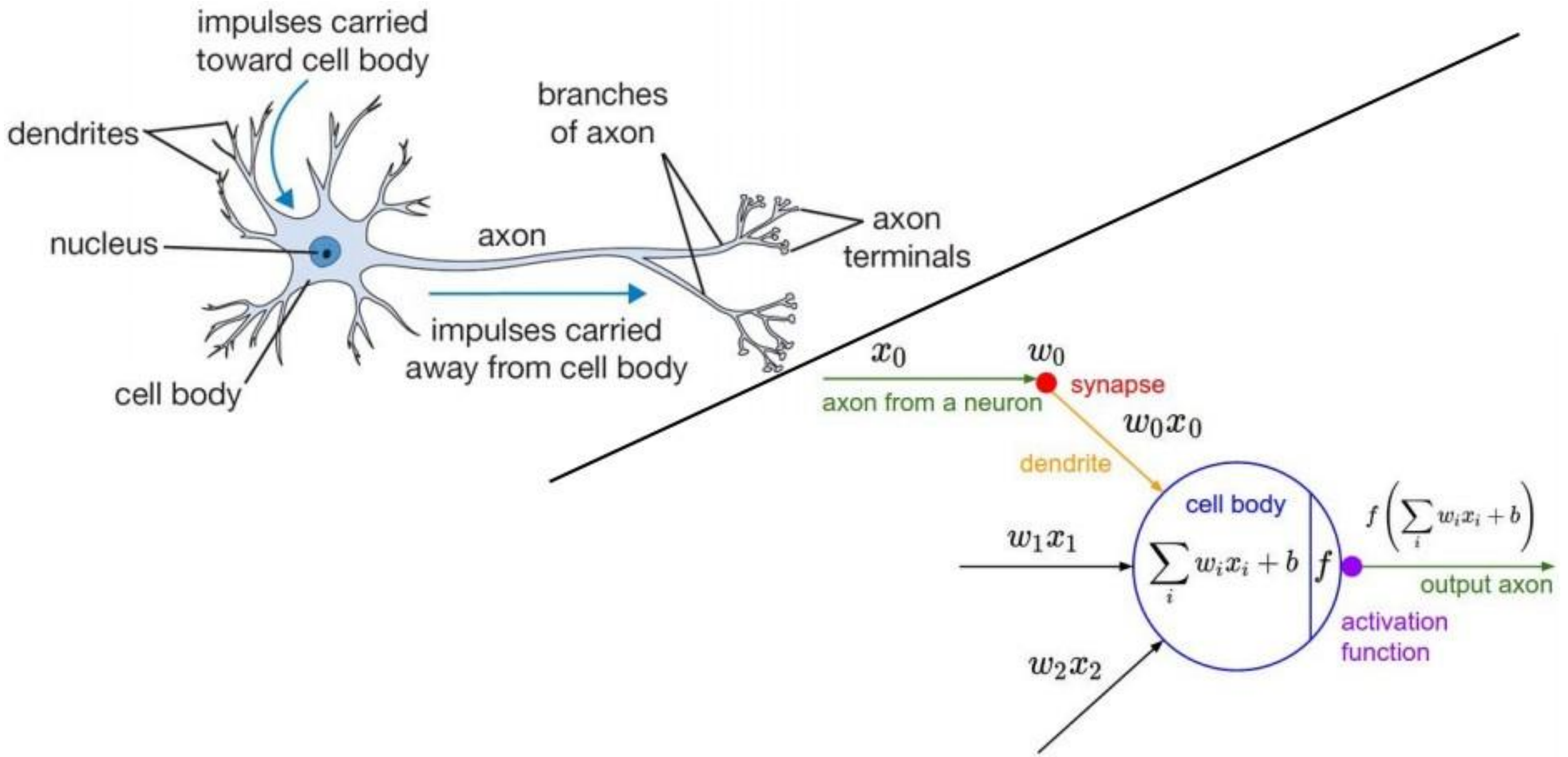
Feed Forward Neural Net

김지환

서강대학교 컴퓨터공학과

Table of contents

3.1 Perceptron - 신경망

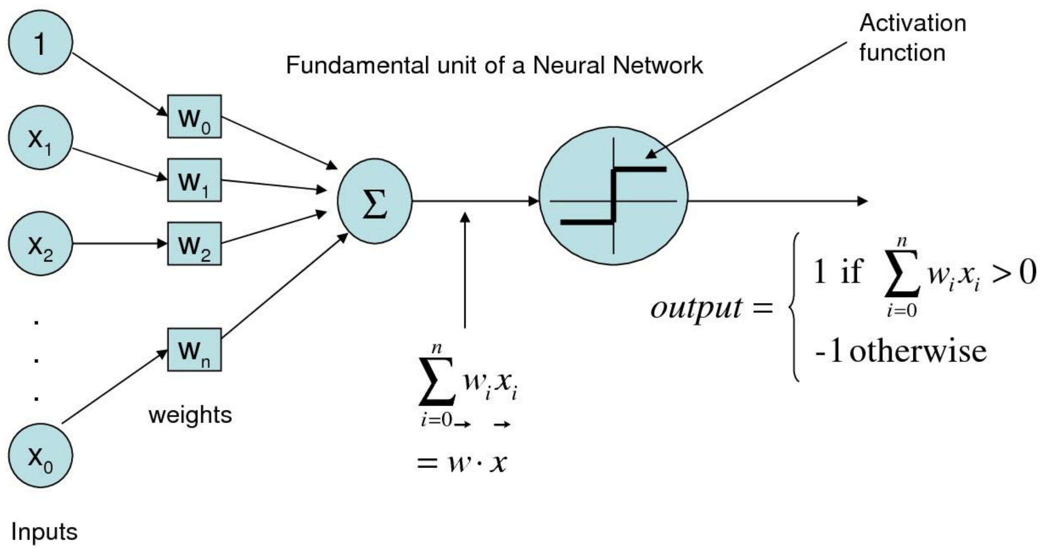


http ://cs231n.stanford.edu/slides/winter1516_lecture4.pdf

3.1 Perceptron

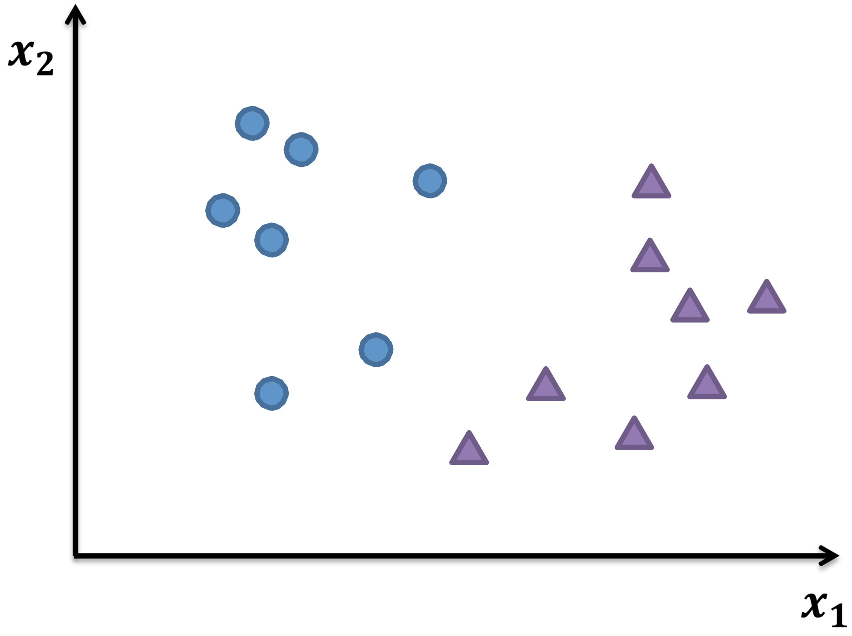
Artificial Neural Networks

The Perceptron



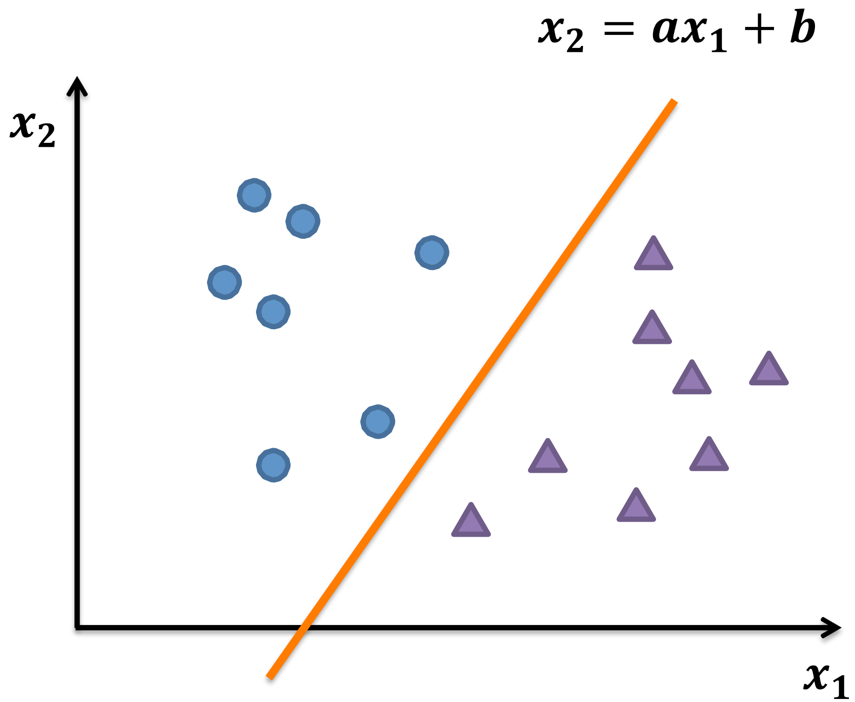
3.1 Perceptron

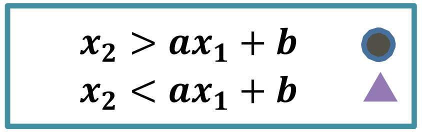
- 동그라미와 세모를 분리하려면 어떻게 해야 할까?



3.1 Perceptron

- y = ax + b 형태의 직선을 이용할 수 있다.

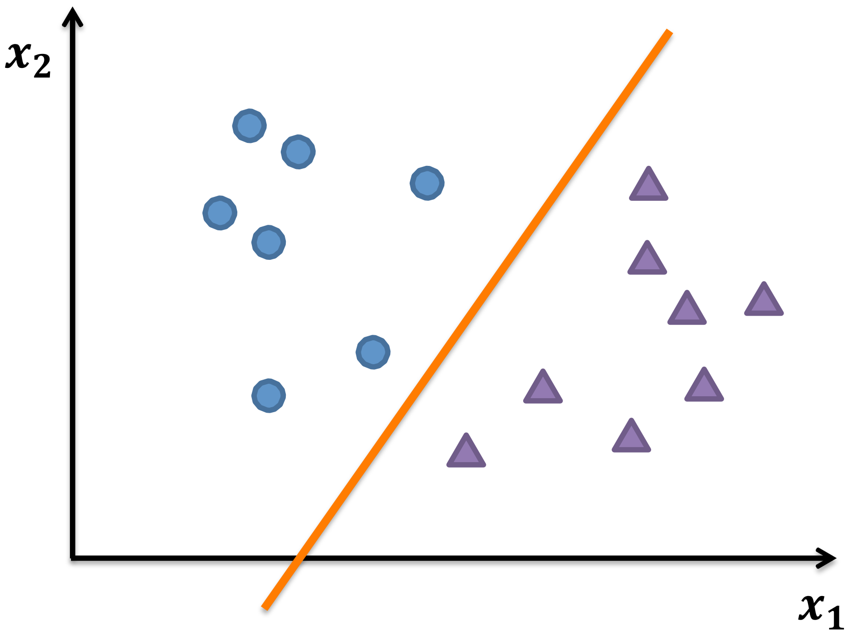




3.1 Perceptron

- 일반화해서 표현해 보자.

$$f(x_1, x_2) = w_1x_1 + w_2x_2 + b = 0$$



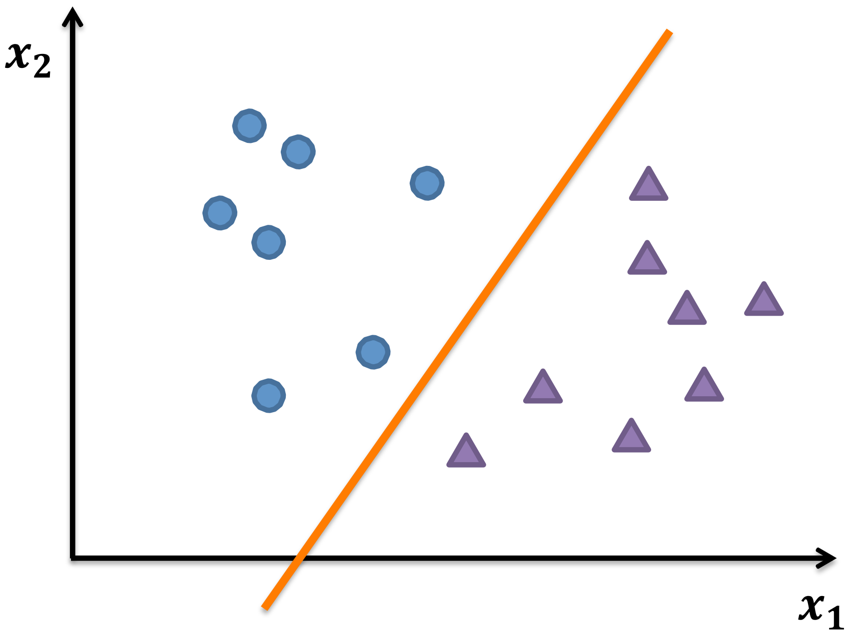
$f(x_1, x_2) > 0$

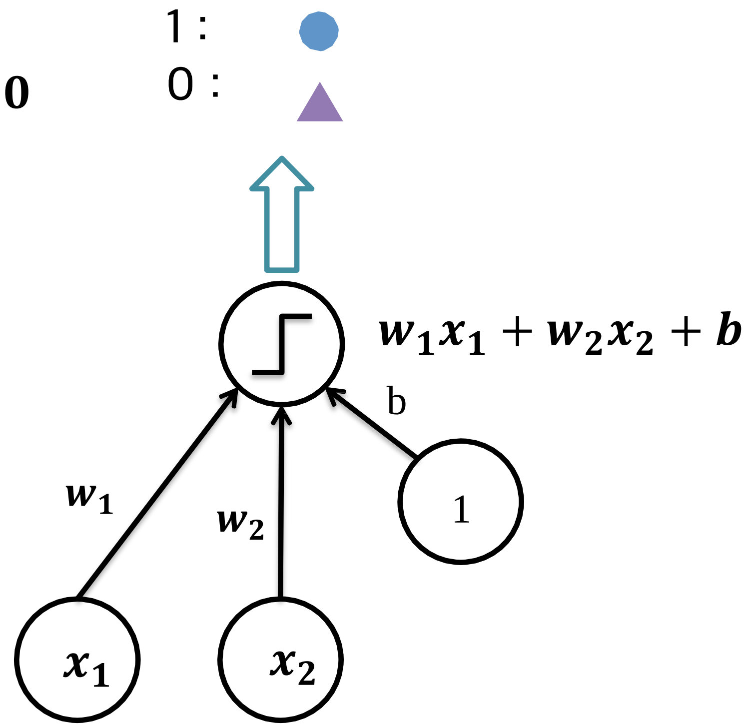
$f(x_1, x_2) < 0$

3.1 Perceptron

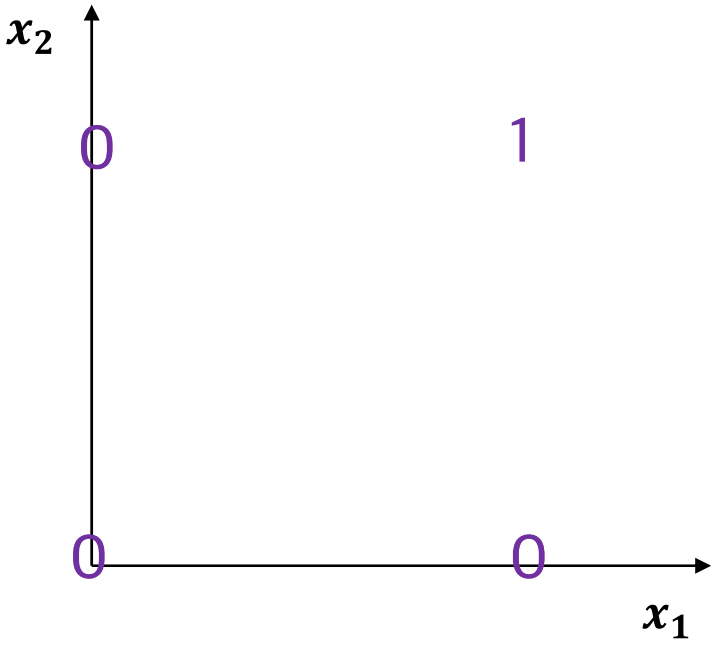
- Generalization

$$f(x_1, x_2) = w_1x_1 + w_2x_2 + b = 0$$



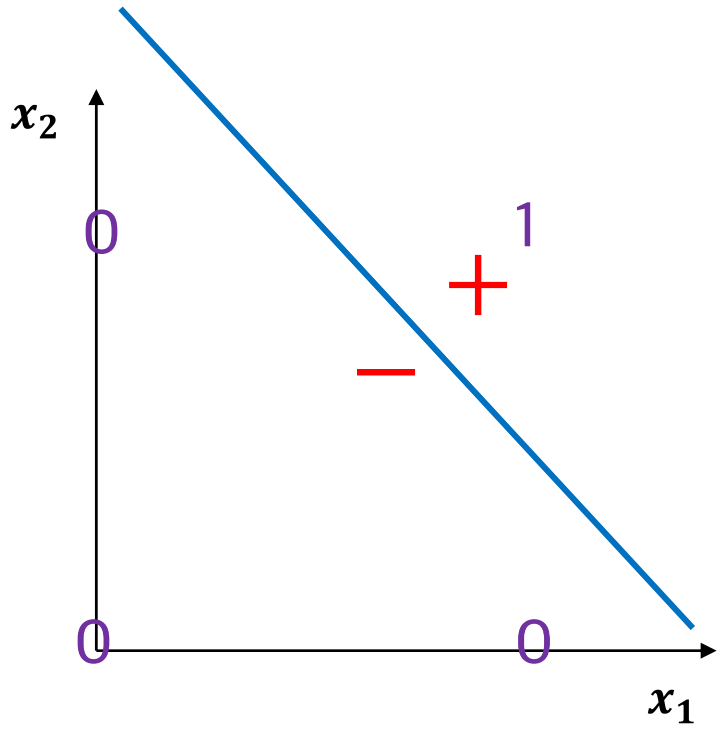


3.1 Perceptron – AND 분류



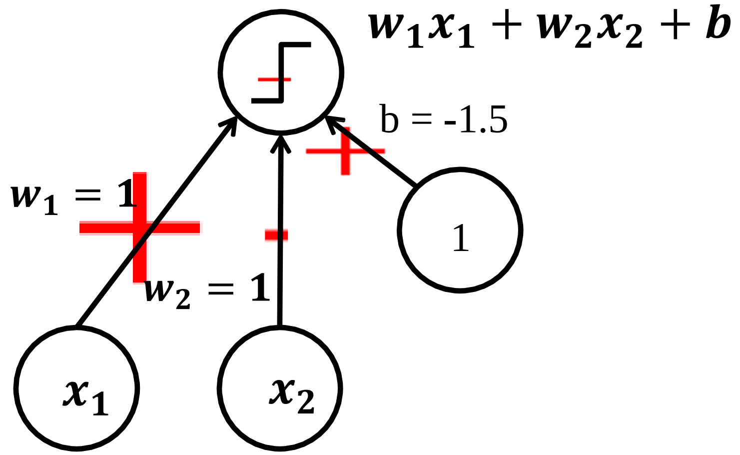
Input		Output
x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

3.1 Perceptron – AND 분류

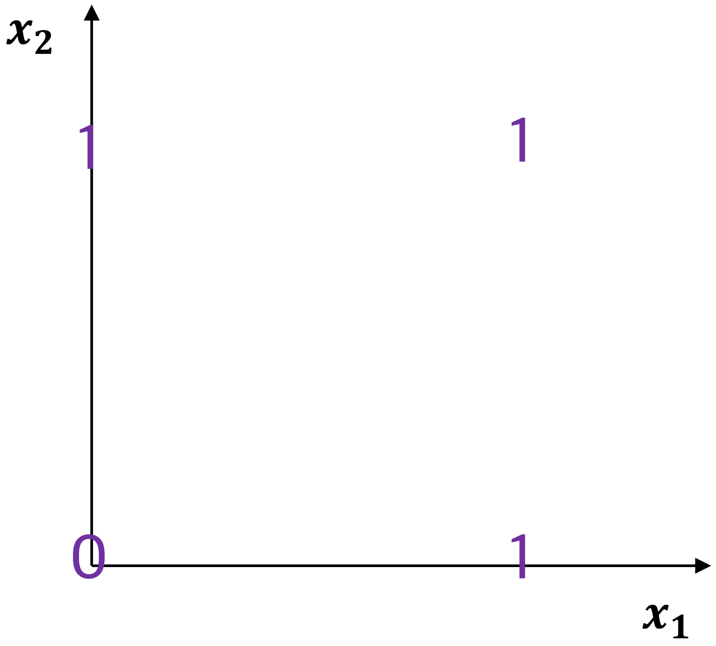


$$f(x_1, x_2) = x_1 + x_2 - 1.5$$

Input		Output
x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

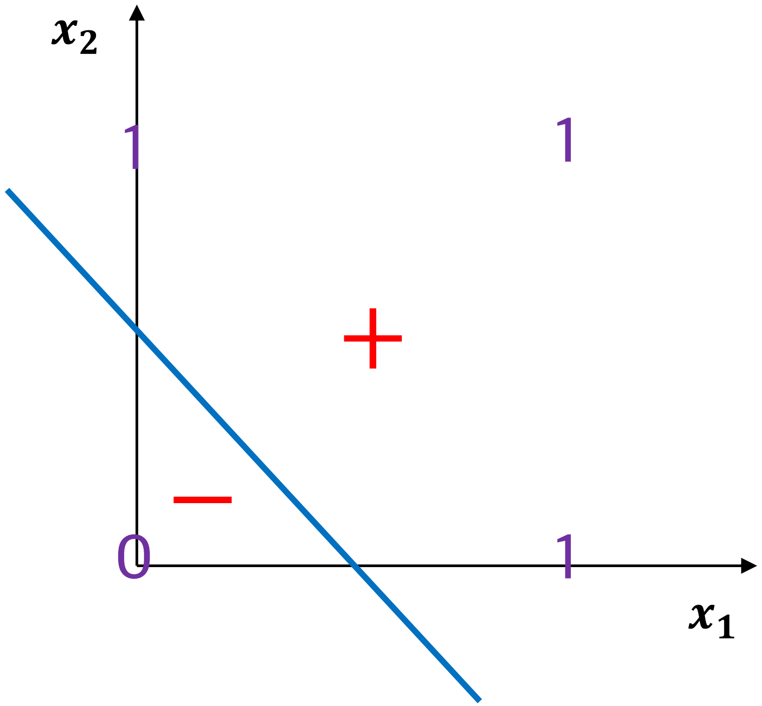


3.1 Perceptron – OR 분류



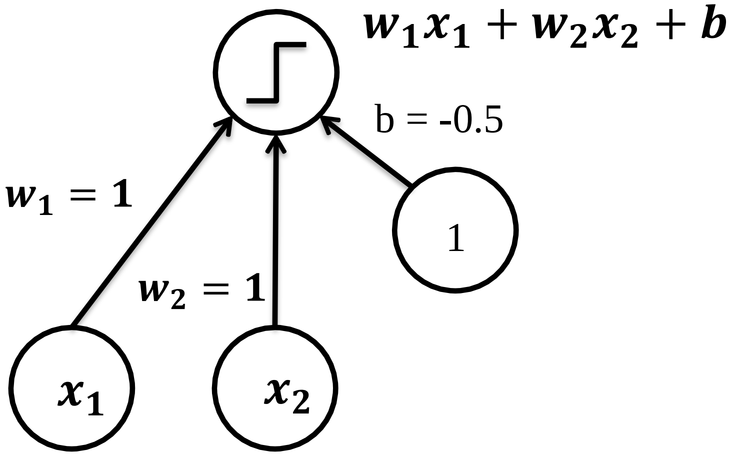
Input		Output
x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1

3.1 Perceptron – OR 분류

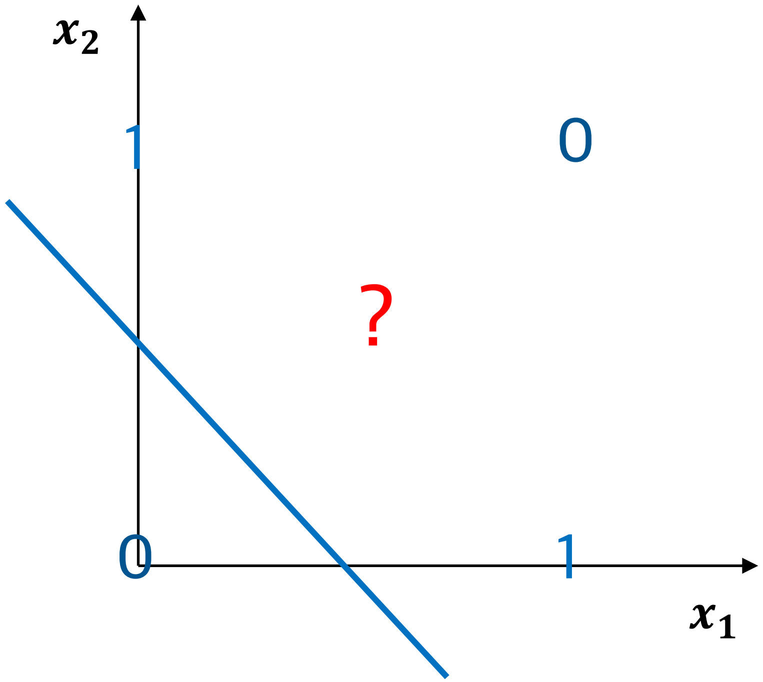


$$f(x_1, x_2) = x_1 + x_2 - 0.5$$

Input		Output
x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1

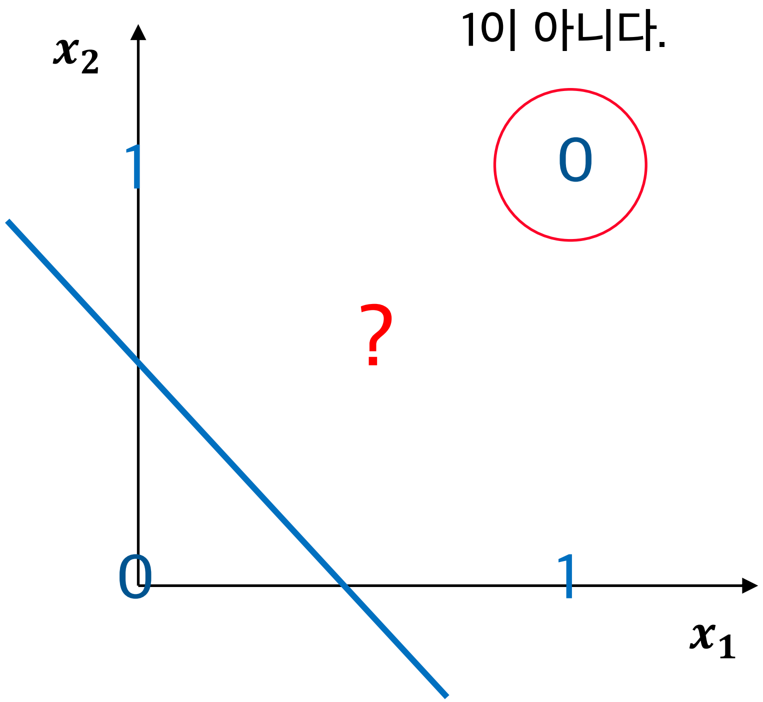


3.1 Perceptron – XOR 분류



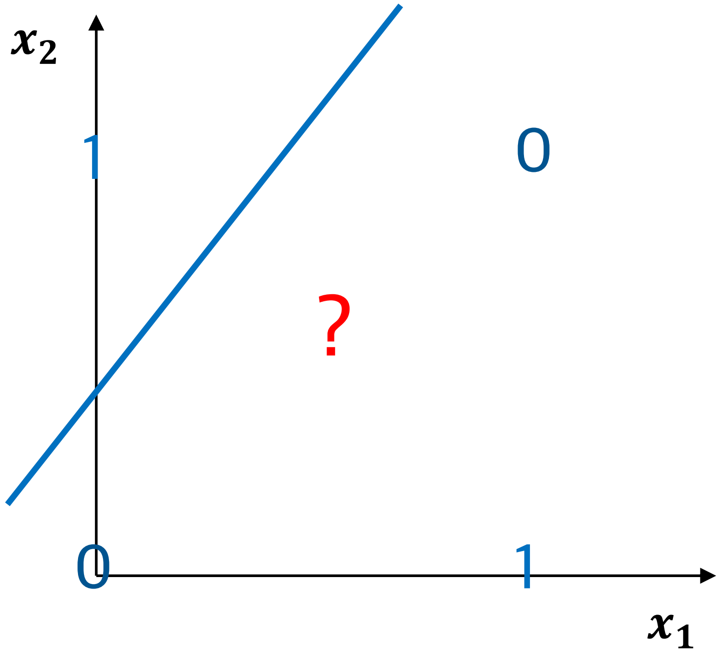
Input		Output
x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

3.1 Perceptron – XOR 분류



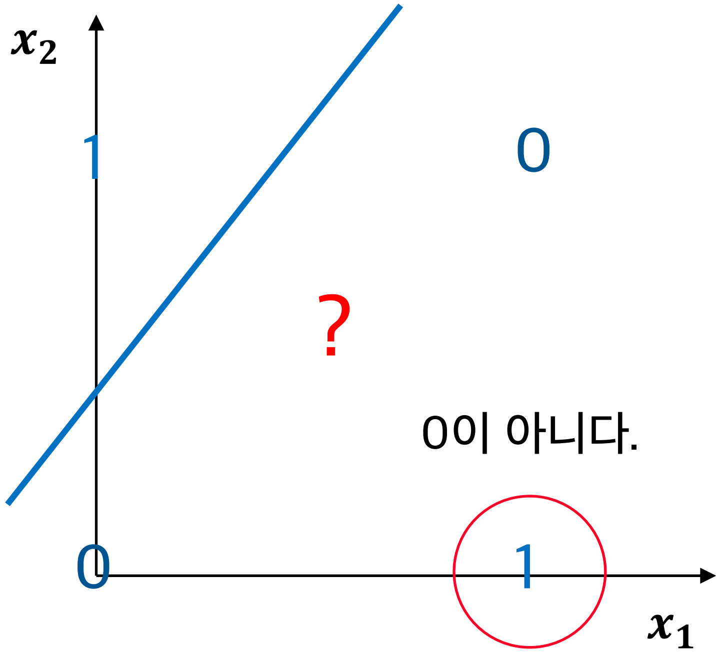
Input		Output
x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

3.1 Perceptron – XOR 분류



Input		Output
x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

3.1 Perceptron – XOR 분류

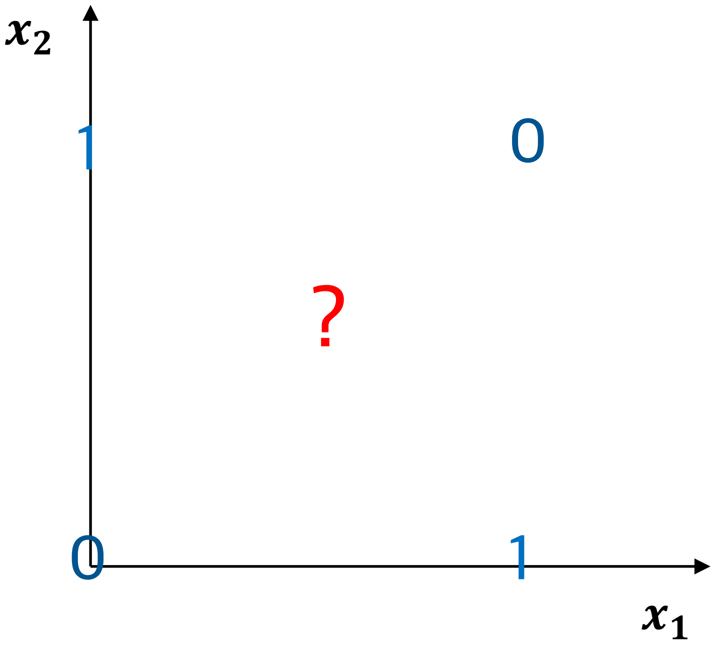


Input		Output
x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

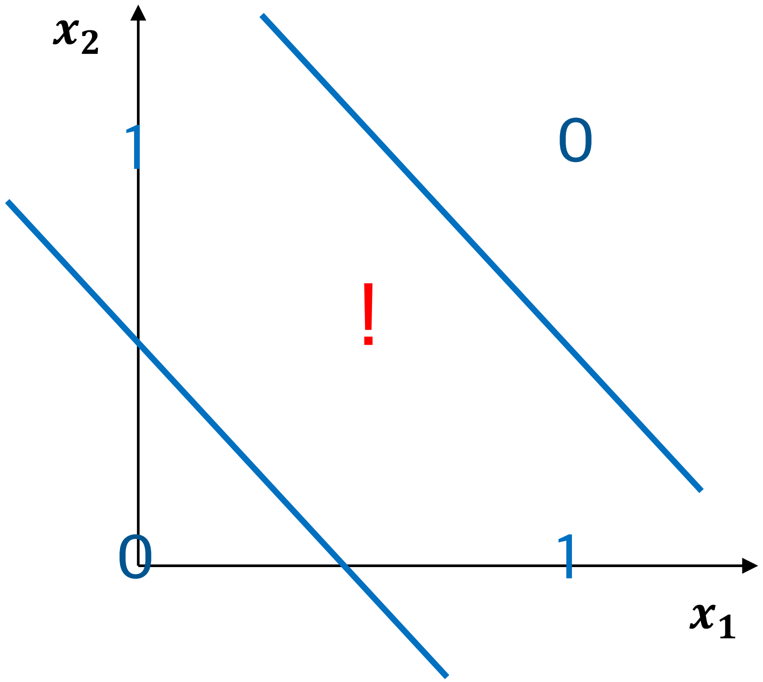
- Perceptron은 XOR을 분류하지 못한다…

3.1 Perceptron – XOR 분류

- 그러면, 어떤 모델을 이용하여 직선 두 개를 나타낼 수 있을까?



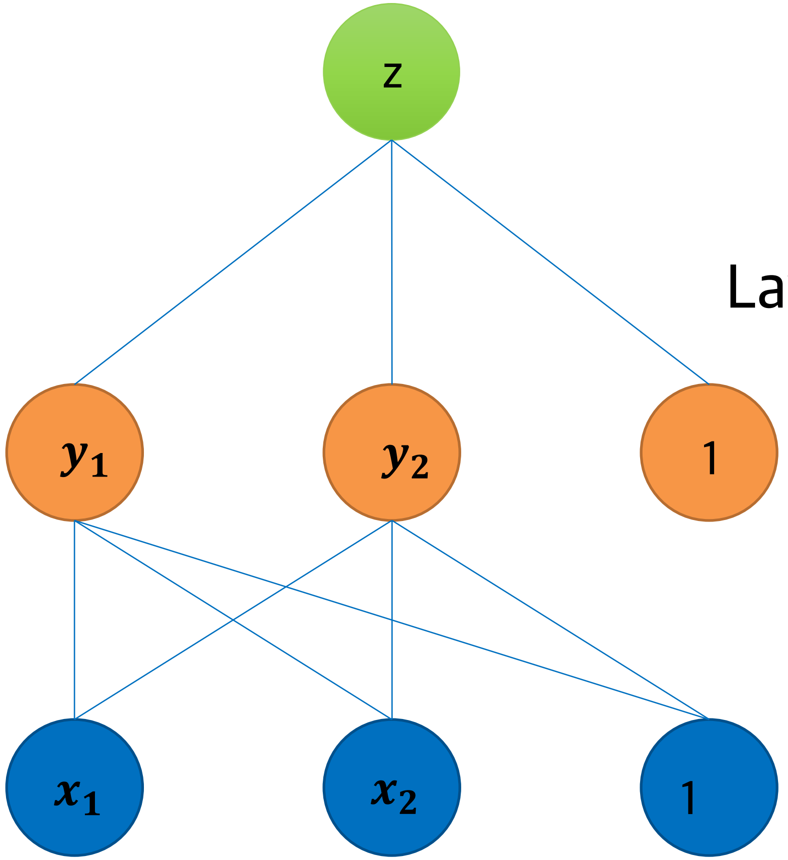
3.1 Perceptron – XOR 분류



Input		Output
x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

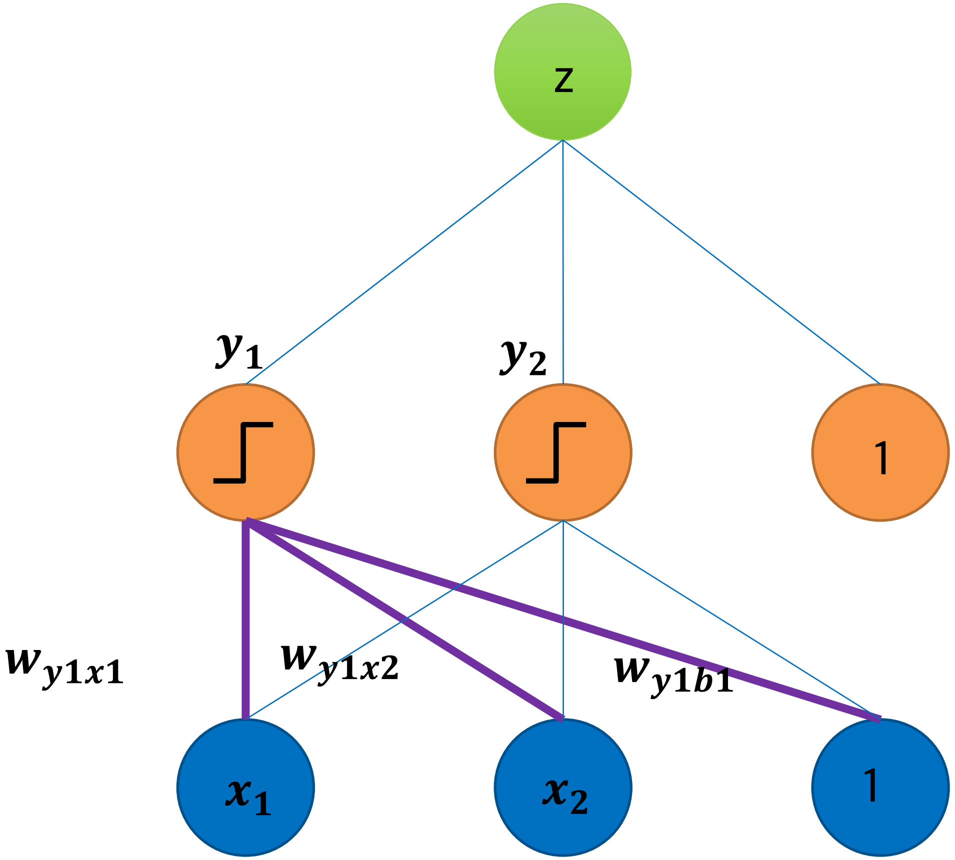
- 직선 두 개를 이용하여 XOR 분류를 해결 할 수 있다.

3.2.1 Multi-layer Perceptron

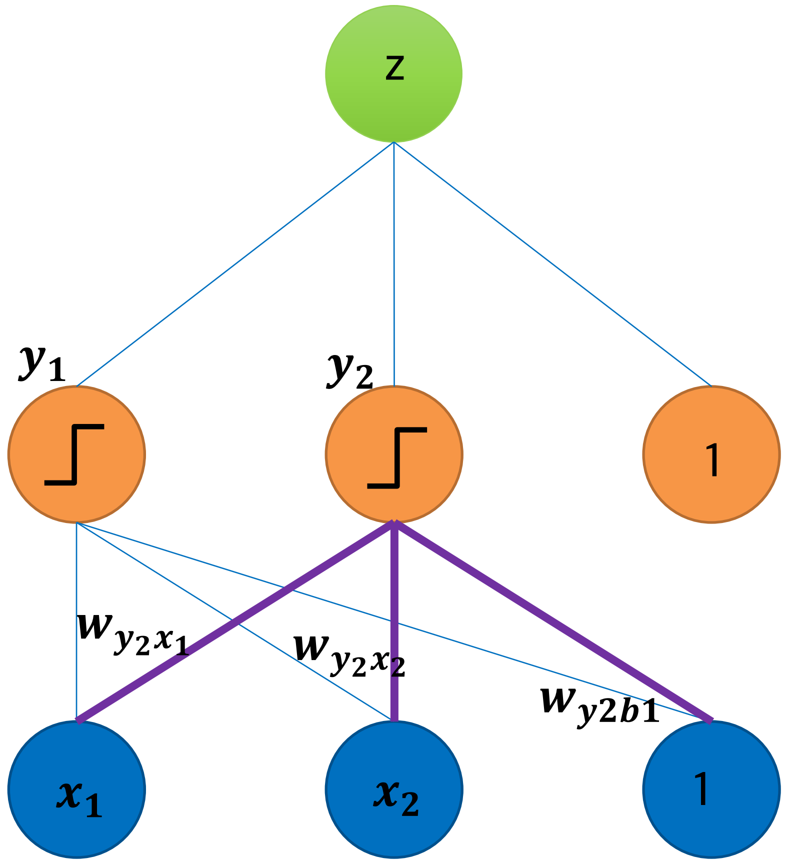


Layer를 추가하면 해결 가능

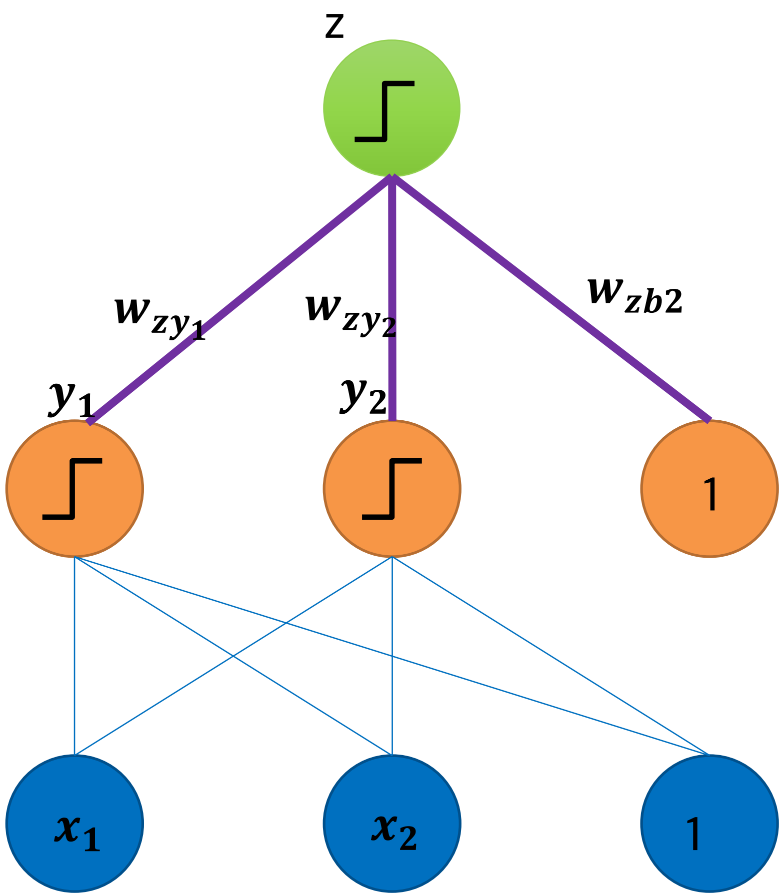
3.2.1 Multi-layer Perceptron



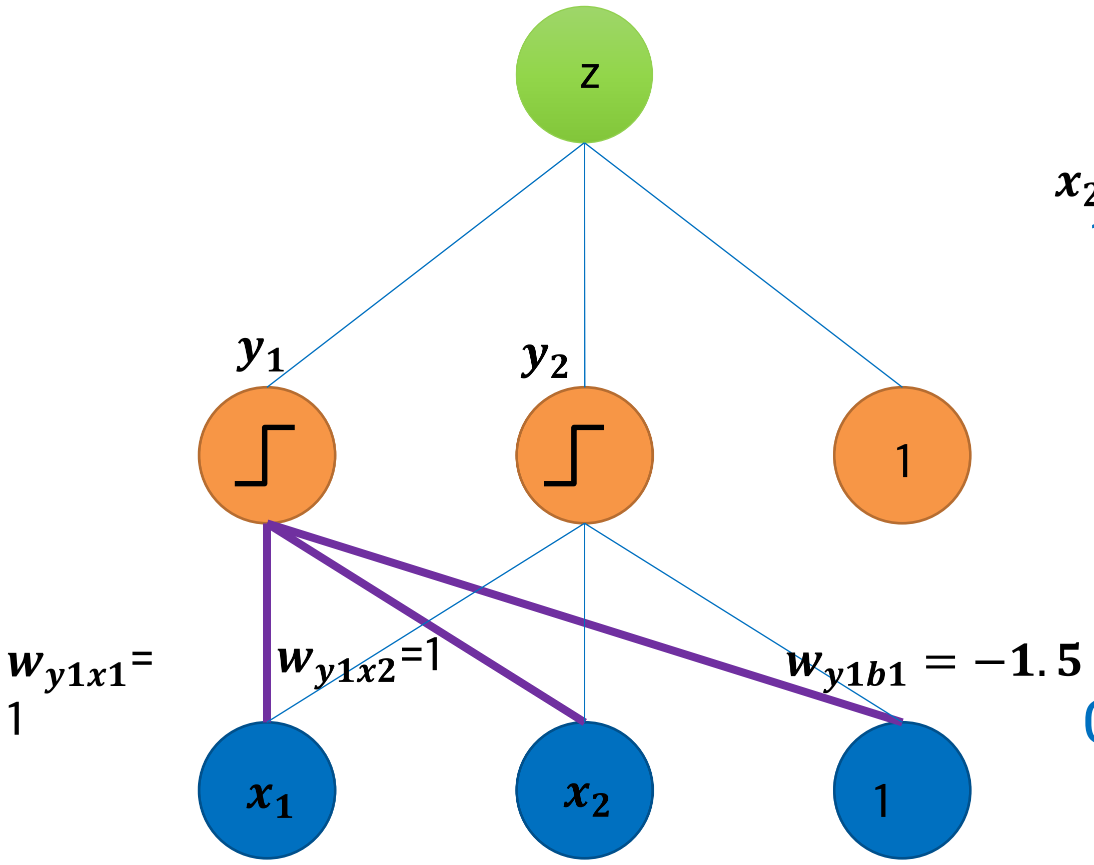
3.2.1 Multi-layer Perceptron

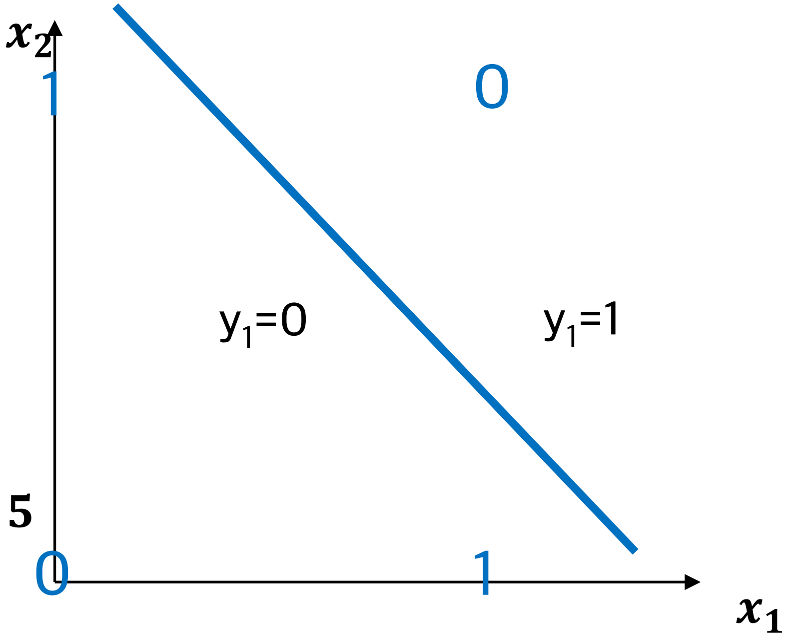


3.2.1 Multi-layer Perceptron

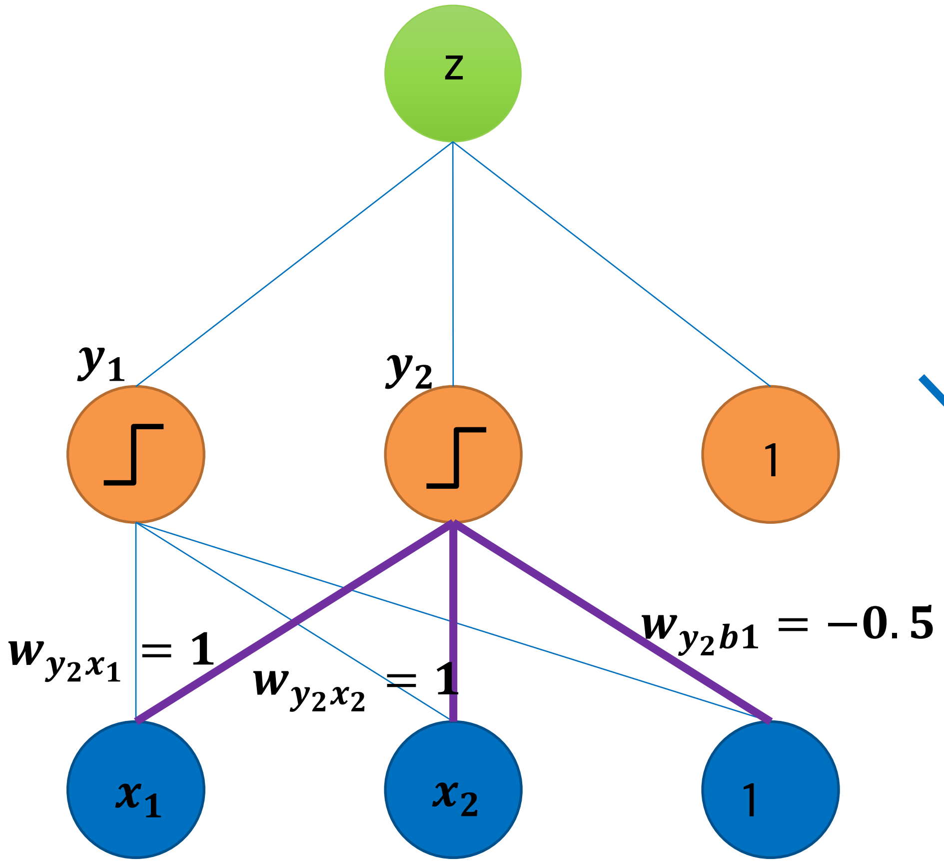


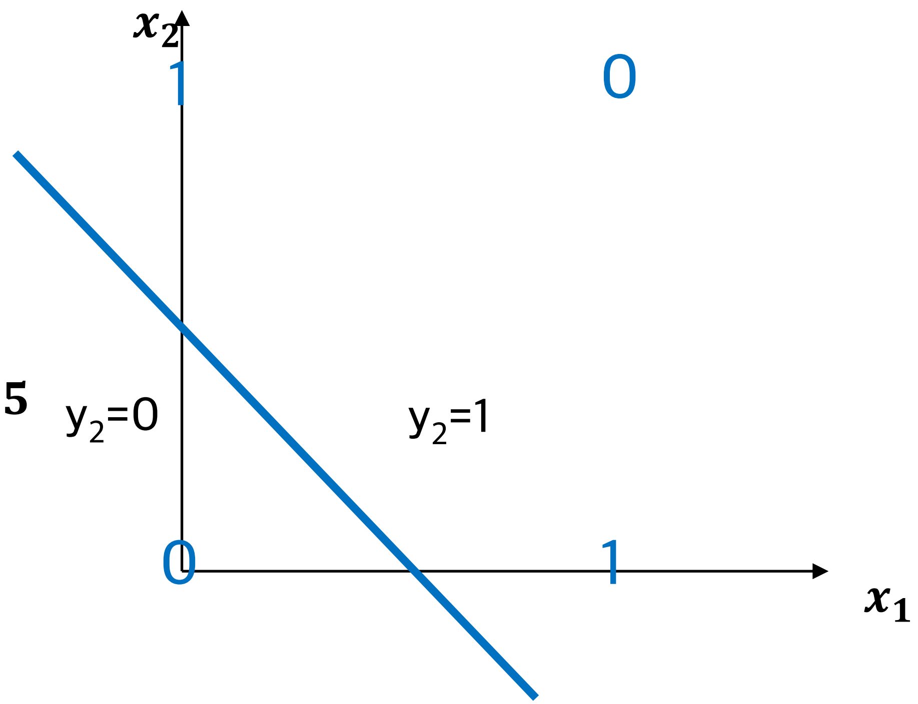
3.2.1 Multi-layer Perceptron



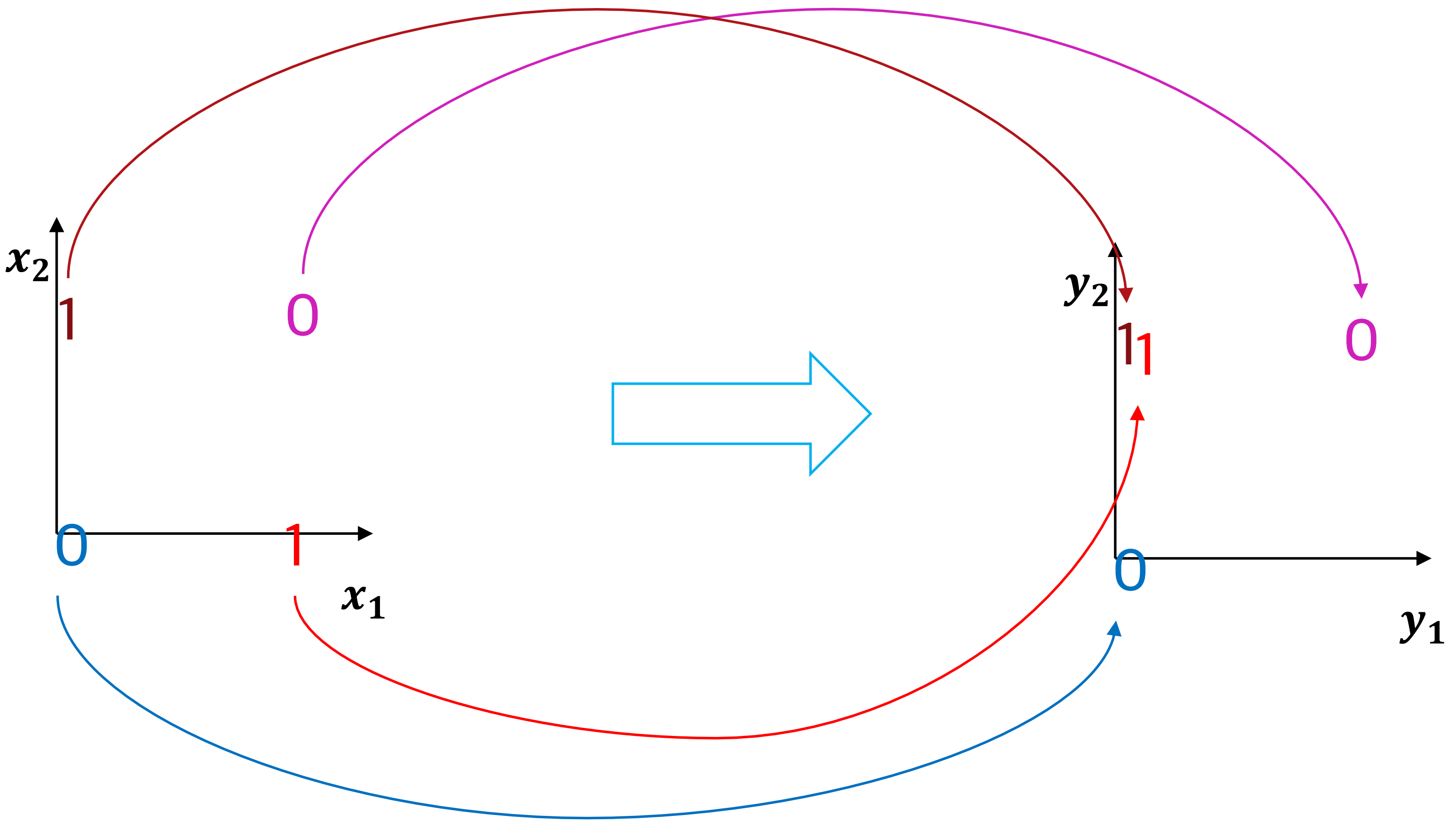


3.2.1 Multi-layer Perceptron

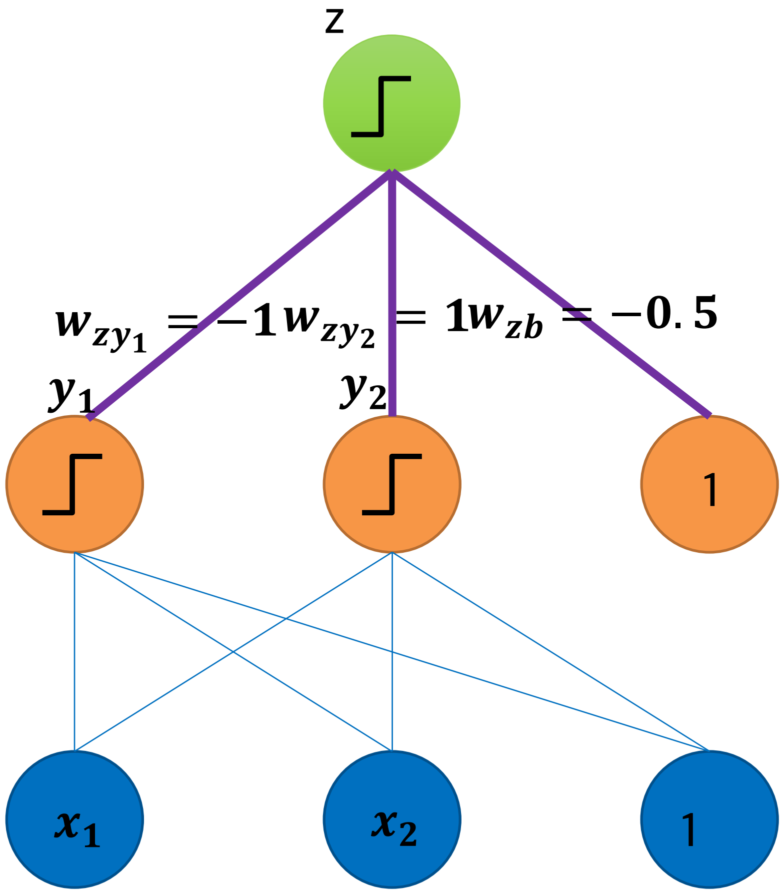


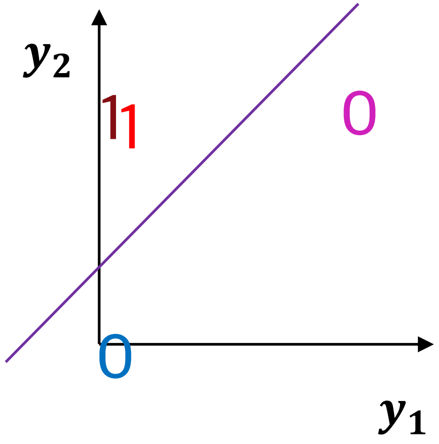


3.2.1 Multi-layer Perceptron

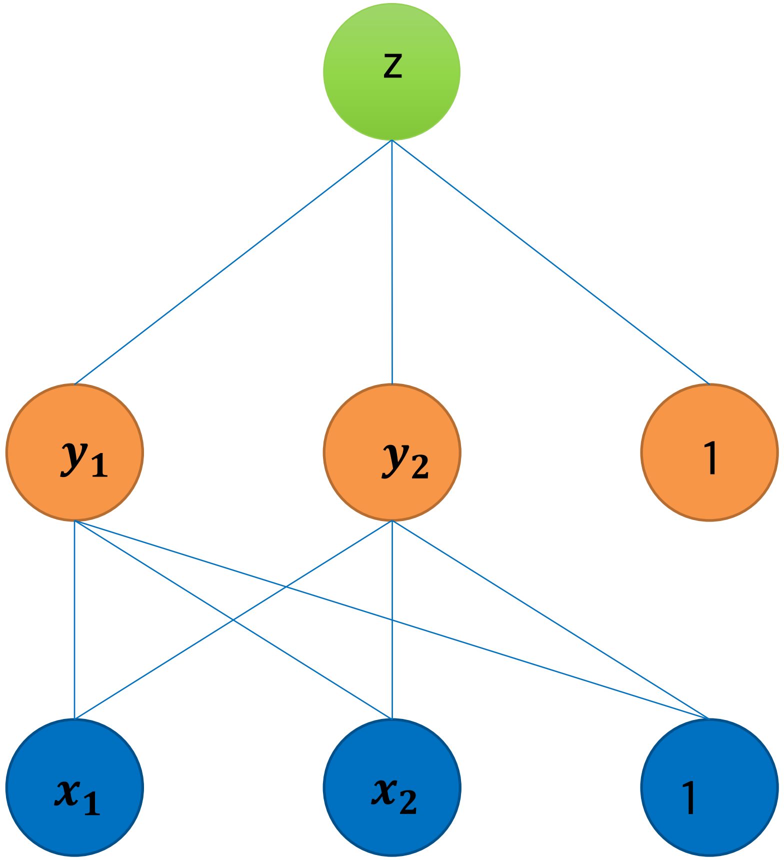


3.2.1 Multi-layer Perceptron



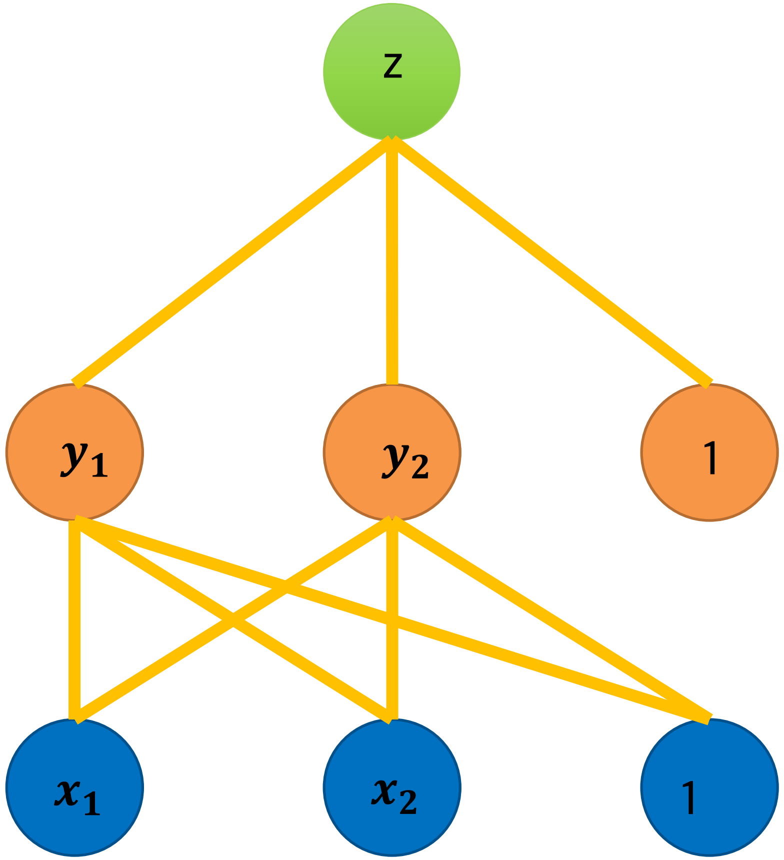


3.2.2 Multi-layer Perceptron - 학습



MLP 학습이란?

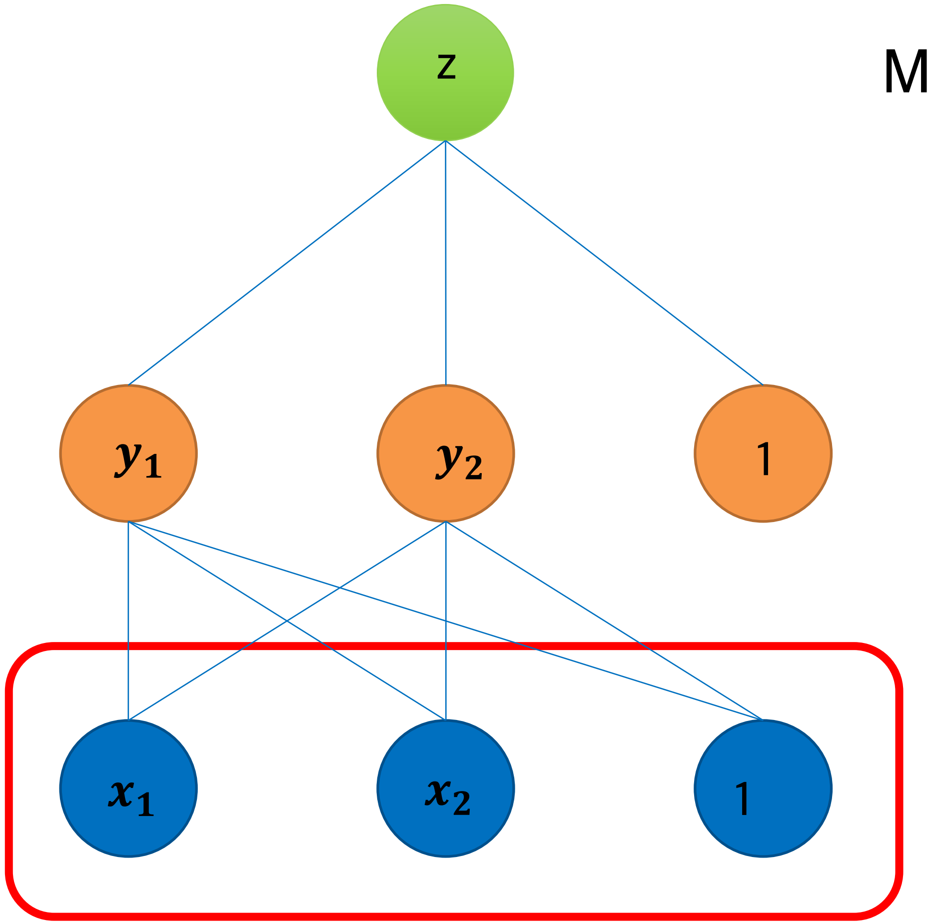
3.2.2 Multi-layer Perceptron - 학습



MLP 학습이란?

학습 자료를 이용하여
*W*를 추정하는 것

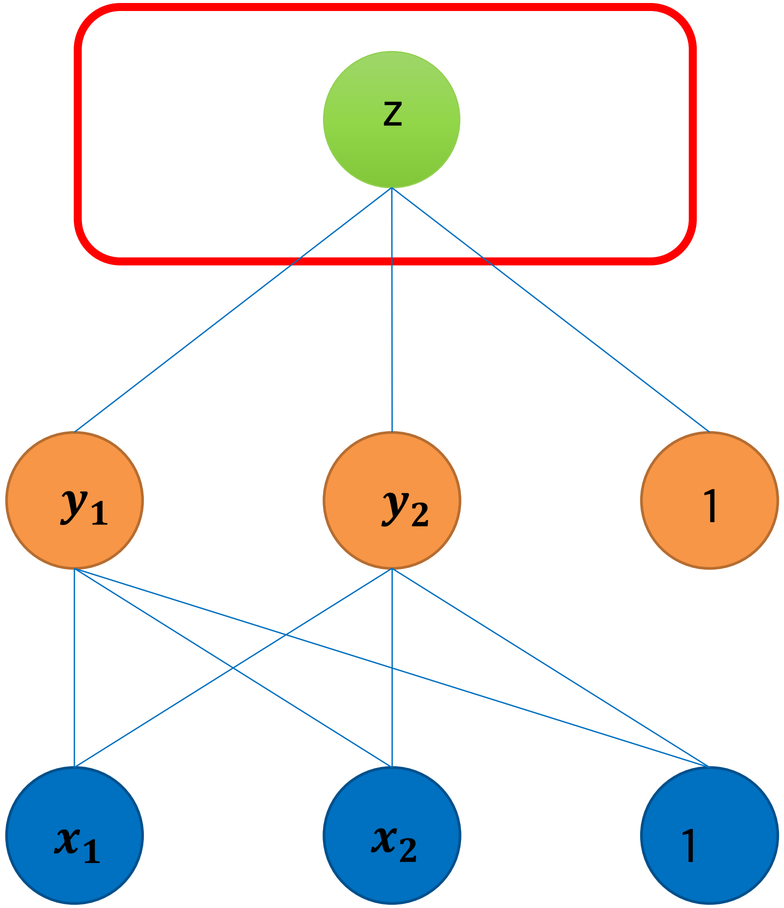
3.2.2 Multi-layer Perceptron - 학습



MLP 학습을 위해 해주어야 하는 일

1. Input layer의 node 수 결정
2. Output layer의 node 수 결정
3. Hidden layer의 node 수 결정
4. 학습 algorithm을 이용한 weight 추정

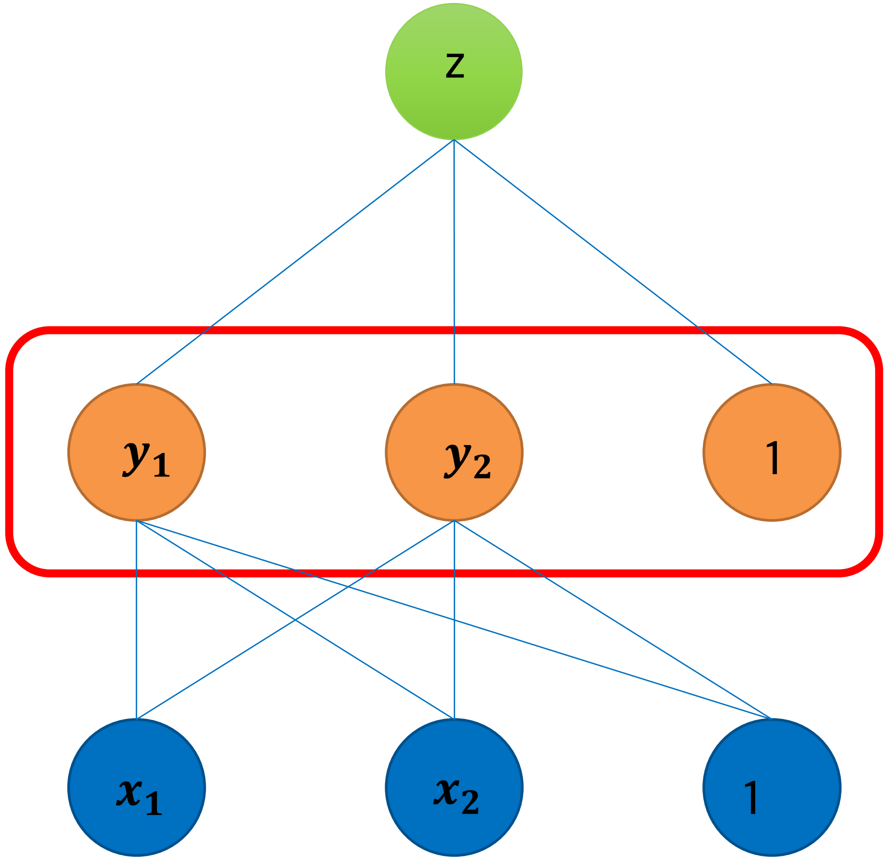
3.2.2 Multi-layer Perceptron - 학습



MLP 학습을 위해 해주어야 하는 일

1. Input layer의 node 수 결정
2. Output layer의 node 수 결정
3. Hidden layer의 node 수 결정
4. 학습 algorithm을 이용한 weight 추정

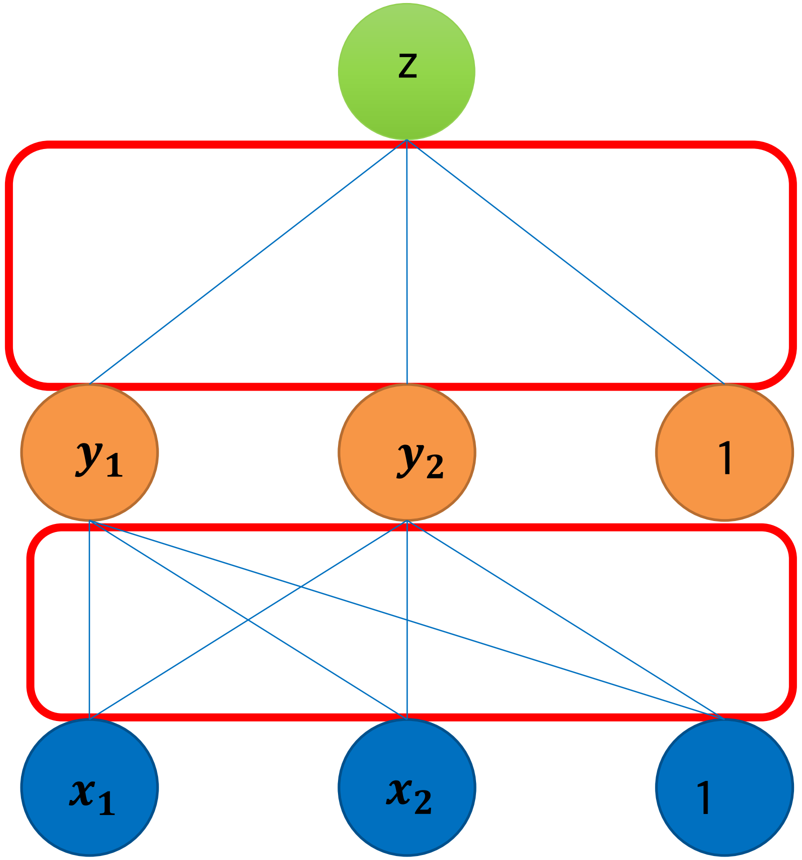
3.2.2 Multi-layer Perceptron - 학습



MLP 학습을 위해 해주어야 하는 일

1. Input layer의 node 수 결정
2. Output layer의 node 수 결정
3. Hidden layer의 node 수 결정
4. 학습 algorithm을 이용한 weight 추정

3.2.2 Multi-layer Perceptron - 학습



MLP 학습을 위해 해주어야 하는 일

1. Input layer의 node 수 결정
2. Output layer의 node 수 결정
3. Hidden layer의 node 수 결정
4. 학습 algorithm을 이용한 weight 추정

3.2.2 Multi-layer Perceptron - 학습

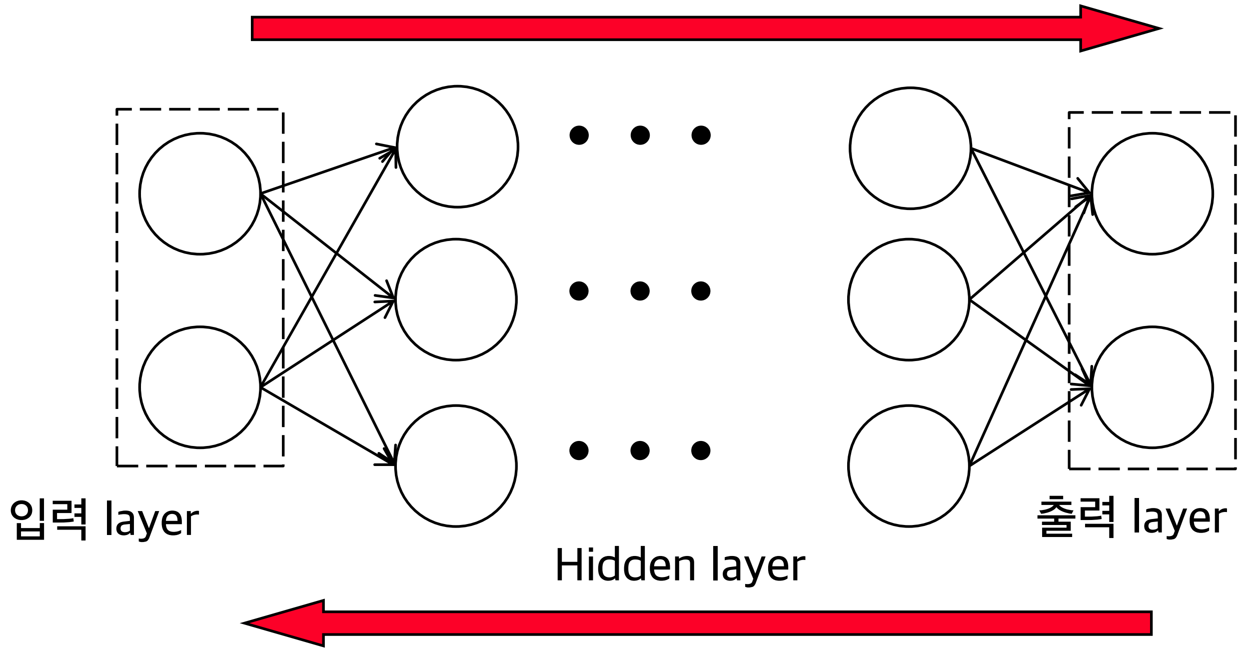
- MLP 학습을 위해 해주어야 하는 일
  - Input layer의 node 수 결정
    - Domain에 따라 결정
  - Output layer의 node 수 결정
    - Domain에 따라 결정한다
  - Hidden layer의 node 수 결정
    - 실험을 통해 결정
  - 학습 algorithm을 이용한 weight 추정
    - Back-propagation알고리즘을 이용하여 레이블 된 학습 자료에 최적 weight를 추정한다

3.2.2 Back-propagation algorithm

- 학습은 back propagation 알고리즘으로 수행됨

Forward 알고리즘

$$y = f\left(\sum_i x_i w_{ij} + b\right)$$

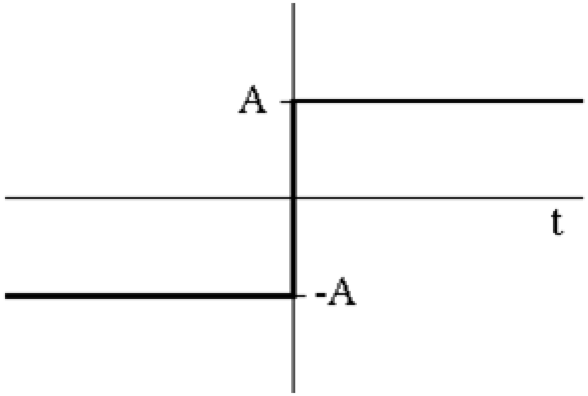


Back propagation 알고리즘

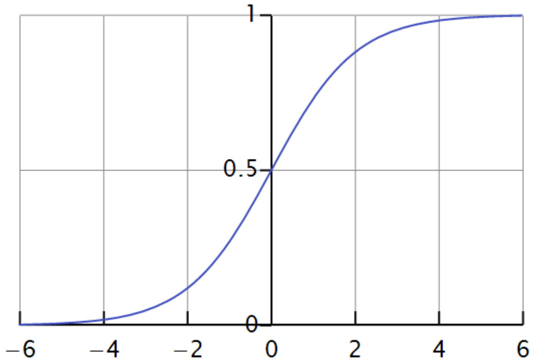
$$w_{ij} = w_{ij} - \eta \frac{\partial E}{\partial w_{ij}}$$

3.2.2 Back-propagation algorithm

- Activation 함수로는 sigmoid를 이용한다.



$f(k)$: Logistic sigmoid function

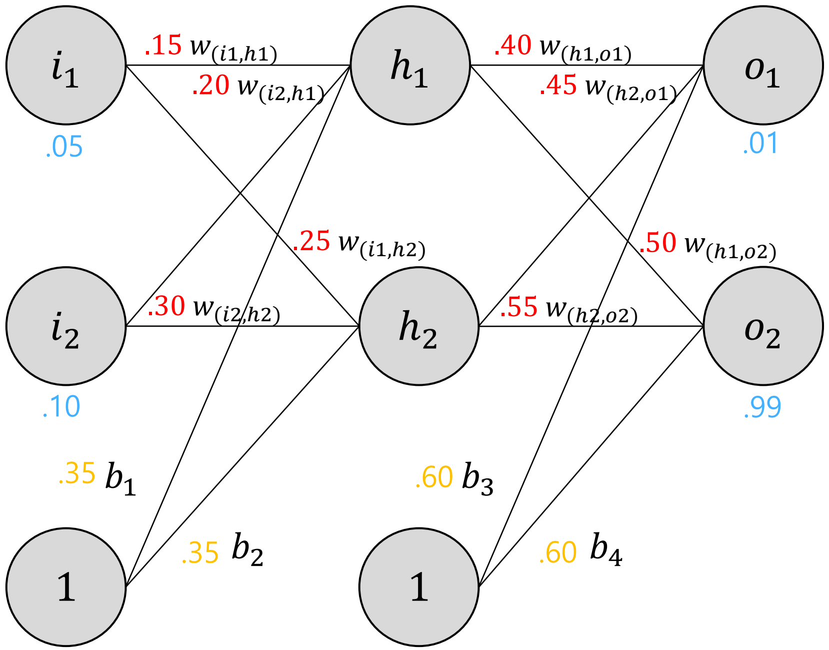


$$f(k) = \frac{1}{1 + e^{-k}}$$

왼쪽 함수는 미분이 되지 않는다.

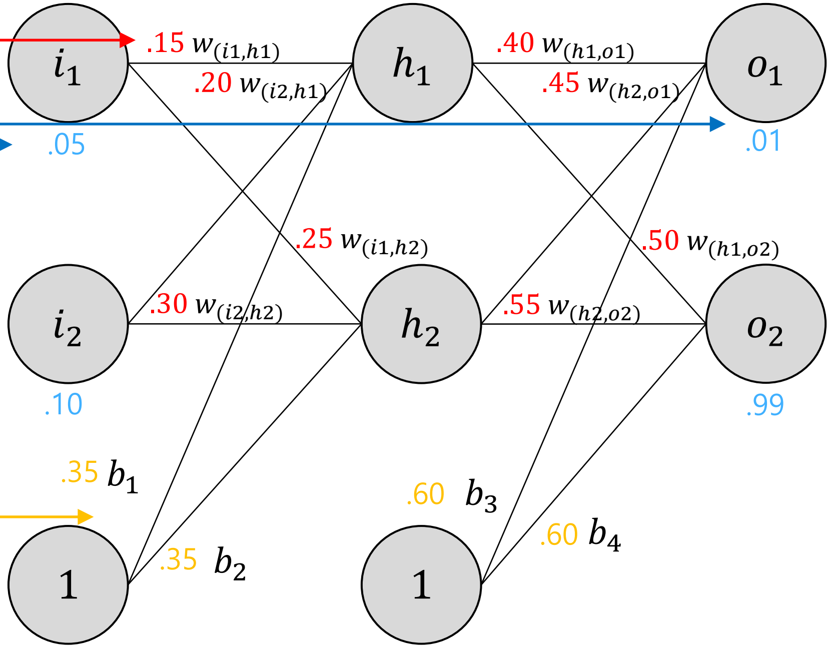
3.2.2 Back-propagation algorithm

- 두개의 input, 두개의 hidden neurons, 두개의 output neurons으로 구성된 기본 neural network 구조



3.2.2 Back-propagation algorithm

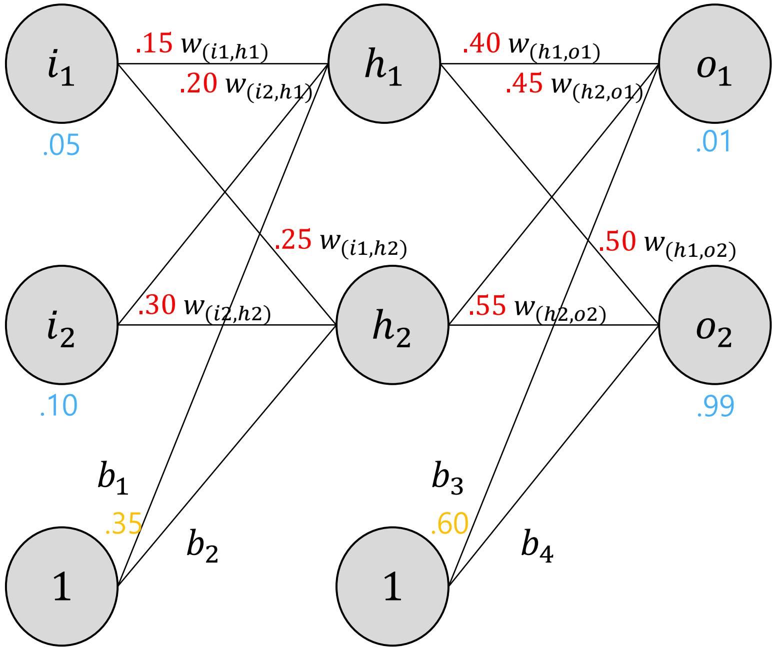
- 예제
  - 학습을 위한
    - input
    - output
  - 초기
    - weight
    - biases



- Backpropagation
  - weight들을 최적화하여, 임의의 input/output에 대하여 올바른 답을 낼 수 있도록 함

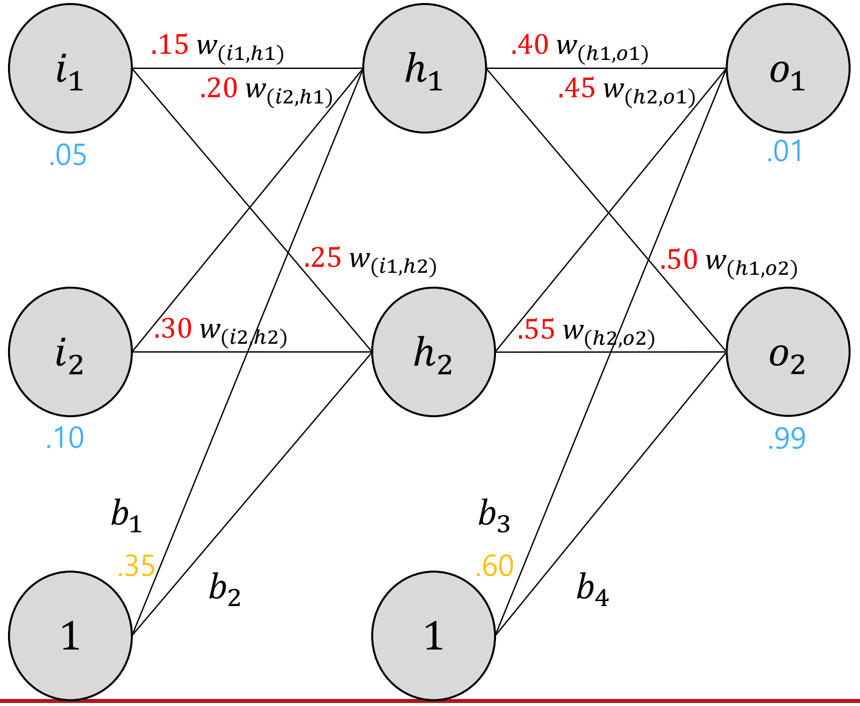
3.2.2 Back-propagation algorithm

- The forward Pass
  - h_1뉴론 input
    - $net_{h_1} = w_{(i_1,h_1)} * i_1 + w_{(i_1,h_2)} * i_2 + b_1 * 1$
    - $net_{h_1} = 0.15 * 0.05 + 0.2 * 0.1 + 0.35 * 1 = 0.3775$
  - logistic function을 거친 h_1 output
    - $out_{h_1} = \frac{1}{1+e^{-net_{h_1}}} = \frac{1}{1+e^{-0.3775}} = 0.593269992$
  - h_2에 대하여 같은 과정을 반복
    - $out_{h_1} = 0.596884378$



3.2.2 Back-propagation algorithm

- The forward Pass (Cont.)
  - h1과 h2의 방법으로 o1을 진행
    - $net_{o_1} = w_{(h1,o1)} * out_{h_1} + w_{(h2,o1)} * out_{h_2} + b3 * 1$
    - $net_{o_1} = 0.4 * 0.593269992 + 0.45 * 0.596884378 + 0.6 * 1 = 1.105905967$
    - $out_{o_1} = \frac{1}{1+e^{-net_{o_1}}} = \frac{1}{1+e^{-1.105905967}} = 0.75136507$
  - o1의 방법으로 o2를 반복
    - $out_{o_2} = 0.772928465$



3.2.2 Back-propagation algorithm

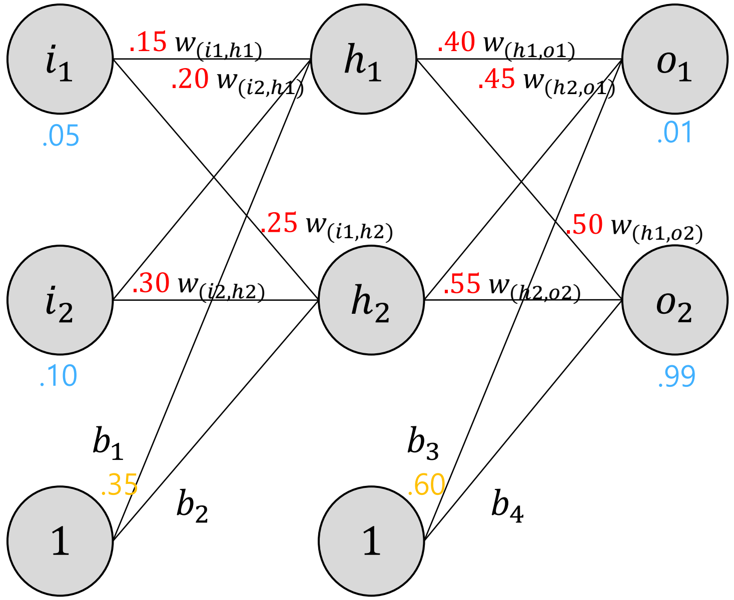
- Calculating the Total Error
  - 앞서 계산된 뉴론의 output을 squared error function의 input으로 에러를 계산

$$E_{total} = \sum \frac{1}{2}(target - output)^2$$

    - 수식의 계수 ½은 미분의 편의성을 위함. 후 약분 됨.

  - 예시
    - o_1의 target output = 0.01
    - o_1 의 neural net output = 0.75136507
    - 따라서, error는

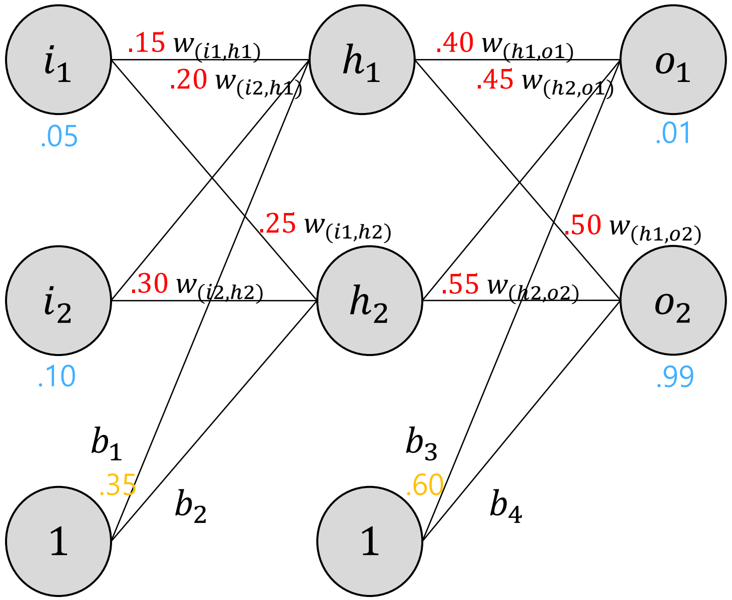
$$\begin{aligned} E_{o_1} &= \frac{1}{2}(target_{o1} - out_{o_1})^2 \\ &= \frac{1}{2}(0.01 - 0.75136507)^2 \\ &= 0.274811083 \end{aligned}$$



3.2.2 Back-propagation algorithm

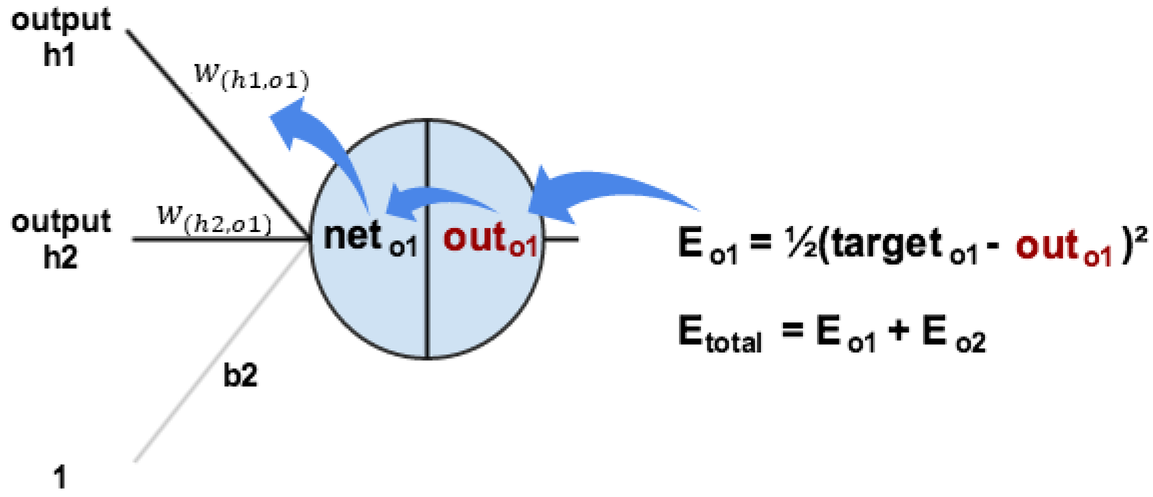
- The Backwards Pass
  - Backpropagation의 목표인 weight update를 진행

- Output Layer
  - The backwards pass
  - 예를 들어, $w_{(h_1,o_1)}$를 update한다면
    - total error에 어느정도 영향을 주는지 알기 위해 편미분을 진행 (Chain Rule적용)
    - $\frac{\partial E_{total}}{\partial w_{(h_1,o_1)}} = \frac{\partial E_{total}}{\partial out_{o_1}} * \frac{\partial out_{o_1}}{\partial net_{o_1}} * \frac{\partial net_{o_1}}{\partial w_{(h_1,o_1)}}$



3.2.2 Back-propagation algorithm

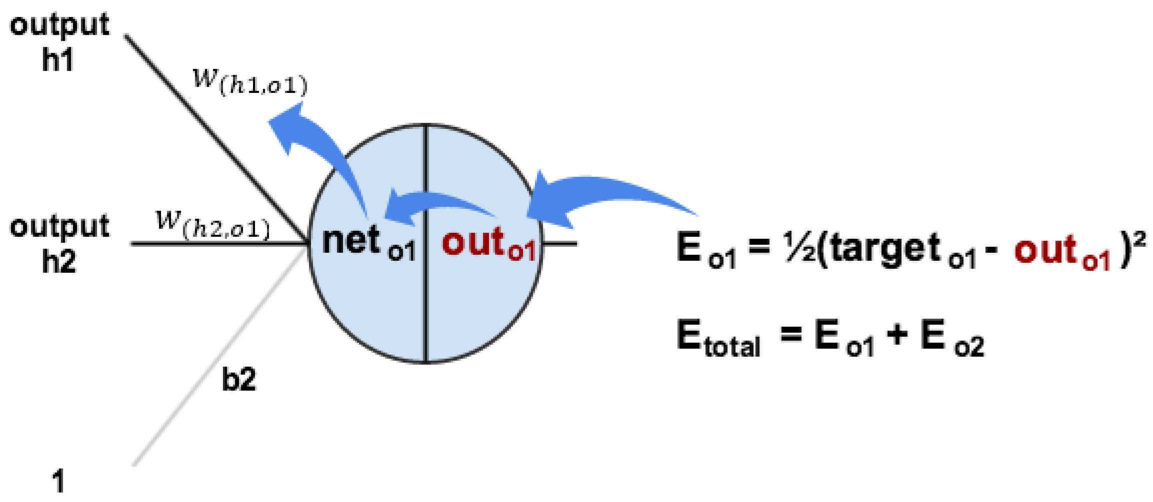
- Output Layer (Cont.)
  - 위 내용을 시각화 아래와 같다
  - 각 부분별로 계산을 진행
  - $\frac{\partial E_{total}}{\partial w_{(h_1,o_1)}} * \frac{\partial out_{o_1}}{\partial net_{o_1}} * \frac{\partial E_{total}}{\partial out_{o_1}} = \frac{\partial E_{total}}{\partial w_{(h_1,o_1)}}$



https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/

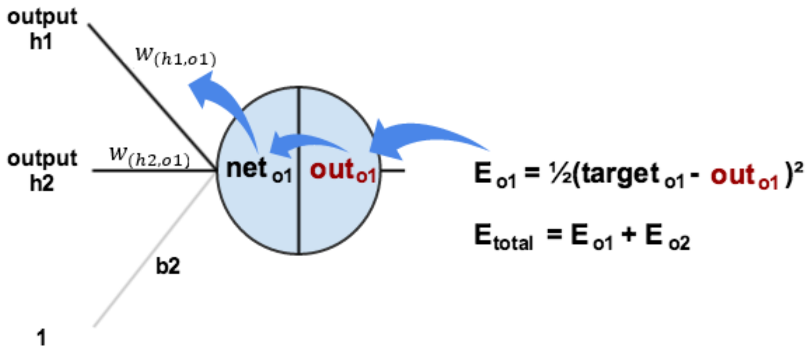
3.2.2 Back-propagation algorithm

- Output Layer (Cont.)
  - 먼저, output(o1)의 변화에 대한 total error의 변화를 계산
    - $E_{total} = \frac{1}{2}(target_{o_1} - out_{o_1})^2 + \frac{1}{2}(target_{o_2} - out_{o_2})^2$
    - $\frac{\partial E_{total}}{\partial out_{o_1}} = 2 * \frac{1}{2}(target_{o_1} - out_{o_1})^{2-1} * -1 + 0$
    - $\frac{\partial E_{total}}{\partial out_{o_1}} = -(target_{o_1} - out_{o_1}) = -(0.01 - 0.75136507) = 0.74136507$



3.2.2 Back-propagation algorithm

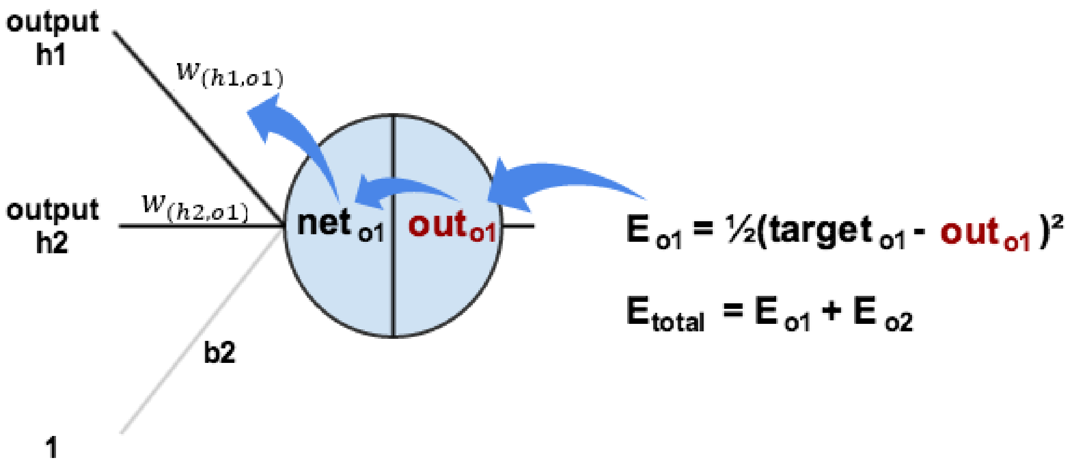
- Output Layer (Cont.)
  - total net input의 변화에 대한 output(o_1)의 변화를 계산
    - logistic function의 편미분 결과는 out(1-out)임
    - $out_{o_1} = \frac{1}{1+e^{-net_{o_1}}}$
    - $\frac{\partial out_{o_1}}{\partial net_{o_1}} = out_{o_1}(1 - out_{o_1}) = 0.75136507(1 - 0.75136507) = 0.186815602$
  - 그리고, $w_{(h_1,o_1)}$의 변화에 대한 total net input of o_1의 변화를 계산
    - $net_{o_1} = w_{(h_1,o_1)} * out_{h_1} + w_{(h_2,o_1)} * out_{h_2} + b_2 * 1$
    - $\frac{\partial net_{o_1}}{\partial w_{(h_1,o_1)}} = 1 * out_{h_1} * w_{(h_1,o_1)}^{(1-1)} + 0 + 0 = out_{h_1} = 0.593269992$



3.2.2 Back-propagation algorithm

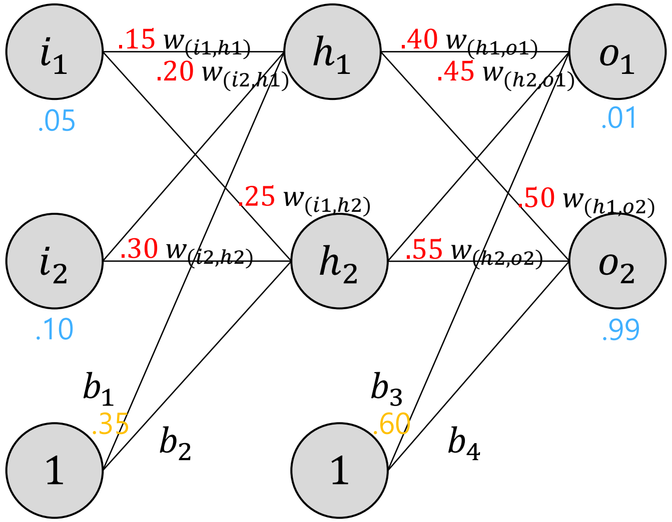
■ Output Layer (Cont.)

- 위 결과 식들을 종합
  - $w_{(h_1,o_1)}$ 의 변화에 대한 total Error
    * $\frac{\partial E_{total}}{\partial w_{(h_1,o_1)}} = \frac{\partial E_{total}}{\partial out_{o_1}} * \frac{\partial out_{o_1}}{\partial net_{o_1}} * \frac{\partial net_{o_1}}{\partial w_{(h_1,o_1)}}$
    * $\frac{\partial E_{total}}{\partial w_{(h_1,o_1)}} = 0.74136507 * 0.186815602 * 0.593269992$
      $= 0.082167041$



3.2.2 Back-propagation algorithm

- Output Layer (Cont.)
  - Error를 감소시키기 위해, 위 식에서 얻은 값을 현재 weight에서 빼준다.
    이때 learning rate(eta)을 곱한 뒤 뺀다.
    - $w^+_{(h_1,o_1)} = w_{(h_1,o_1)} - \eta * \frac{\partial E_{total}}{\partial w_{(h_1,o_1)}} = 0.4 - 0.5 * 0.082167041 = 0.35891648$
  - 동일한 프로세스를
    $w^+_{(h_2,o_1)} \sim w^+_{(h_2,o_2)}$ 에 대하여 반복
    - $w^+_{(h_2,o_1)} = 0.408666186$
    - $w^+_{(h_1,o_2)} = 0.511301270$
    - $w^+_{(h_2,o_2)} = 0.561370121$

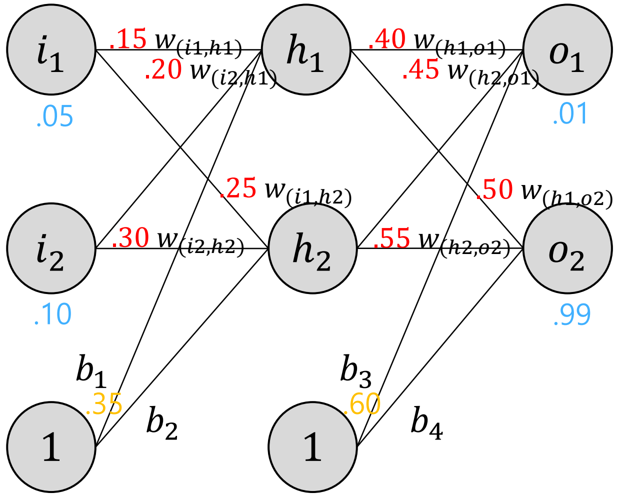


  - weight 값 **실제 update**는 hidden layer에 대해서도 새로운 weight을 **모두 구한 후 update**를 진행

3.2.2 Back-propagation algorithm

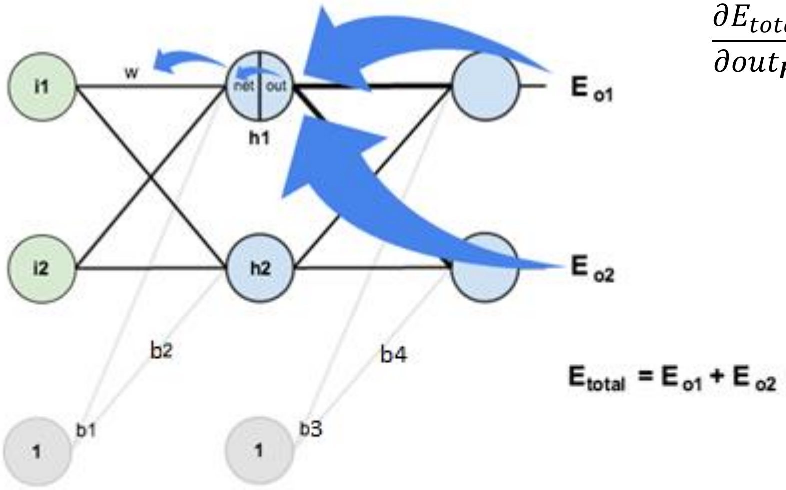
- Hidden Layer
  - The backwards pass (Cont.)
    - $w_{(i_1,h_1)} \sim w_{(i_2,h_2)}$ 에 대해서 값을 계산
  - output layer에서 진행했던 방식과 유사한 방식
    - 차이점 : 여러 개의 output neurons의 변화량 사용
      - h1의 out부분이 o1,o2에 영향을 줌

$$\frac{\partial E_{total}}{\partial w_{(i_1,h_1)}} = \frac{\partial E_{total}}{\partial out_{h_1}} * \frac{\partial out_{h_1}}{\partial net_{h_1}} * \frac{\partial net_{h_1}}{\partial w_{(i_1,h_1)}}$$



$$\frac{\partial E_{total}}{\partial w_{(i_1,h_1)}} = \boxed{\frac{\partial E_{total}}{\partial out_{h_1}}} * \frac{\partial out_{h_1}}{\partial net_{h_1}} * \frac{\partial net_{h_1}}{\partial w_{(i_1,h_1)}}$$

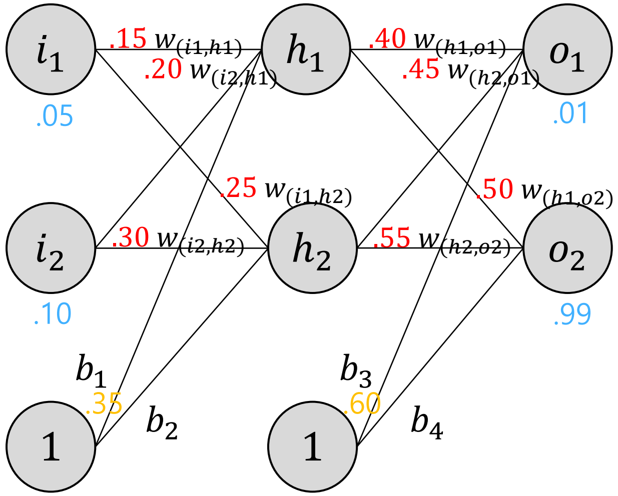
$$\frac{\partial E_{total}}{\partial out_{h_1}} = \frac{\partial E_{o_1}}{\partial out_{h_1}} + \frac{\partial E_{o_2}}{\partial out_{h_1}}$$



https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/

3.2.2 Back-propagation algorithm

- Hidden Layer (Cont.)
  - $\frac{\partial E_{o1}}{\partial out_{h1}}$ 계산시 앞서 계산한 $\frac{\partial E_{o1}}{\partial net_{h1}}$ 을 사용
    - $\frac{\partial E_{o_1}}{\partial out_{h_1}} = \frac{\partial E_{o_1}}{\partial net_{o_1}} * \frac{\partial net_{o_1}}{\partial out_{h_1}}$
    - $\frac{\partial E_{o_1}}{\partial net_{o_1}} = \frac{\partial E_{o_1}}{\partial out_{o_1}} * \frac{\partial out_{o_1}}{\partial net_{o_1}} = 0.74136507 * 0.186815602 = 0.138498562$
  - $\frac{\partial net_{o_1}}{\partial out_{h_1}}$ 와 $w_{(h_1,o_1)}$이 같으므로
    - $net_{o_1} = w_{(h_1,o_1)} * out_{h_1} + w_{(h_2,o_1)} * out_{h_2} + b_3 * 1$
    - $\frac{\partial net_{o_1}}{\partial out_{h_1}} = w_{(h_1,o_1)} = 0.40$
  - 위 식들을 합쳐주면
    - $\frac{\partial E_{o_1}}{\partial out_{h_1}} = \frac{\partial E_{o_1}}{\partial net_{o_1}} * \frac{\partial net_{o_1}}{\partial out_{h_1}} = 0.138498562 * 0.40 = 0.055399425$



3.2.2 Back-propagation algorithm

- Hidden Layer (Cont.)
  - 같은 방식으로 $\frac{\partial E_{o_2}}{\partial out_{h_1}}$ 를 계산
    - $\frac{\partial E_{o_2}}{\partial out_{h_1}} = -0.019049119$
  - 따라서 $\frac{\partial E_{total}}{\partial out_{h_1}}$ 계산 가능
    - $\frac{\partial E_{total}}{\partial out_{h_1}} = \frac{\partial E_{o_1}}{\partial out_{h_1}} + \frac{\partial E_{o_2}}{\partial out_{h_1}} = 0.055399425 + -0.019049119 = 0.036350306$
  - $\frac{\partial E_{total}}{\partial out_{h_1}}$ 를 얻었으므로, 각 weight에 대해$\frac{\partial out_{h_1}}{\partial net_{h_1}}$와 $\frac{\partial net_{h_1}}{\partial w}$ 를 계산
    - $out_{h_1} = \frac{1}{1+e^{-net_{h_1}}}$
    - $\frac{\partial out_{h_1}}{\partial net_{h_1}} = out_{h_1}\left(1 - out_{h_1}\right) = 0.59326999(1 - 0.59326999) = 0.241300709$
  - output neuron에서 했던 방식을 적용하여, $w_{(i_1,h_1)}$ 에 대한 total net input to h1의 편미분을 계산
    - $net_{h_1} = w_{(i_1,h_1)} * i_1 + w_{(i_2,h_1)} * i_2 + b_1 * 1$
    - $\frac{\partial net_{h_1}}{\partial w_{(i_1,h_1)}} = i_1 = 0.05$

3.2.2 Back-propagation algorithm

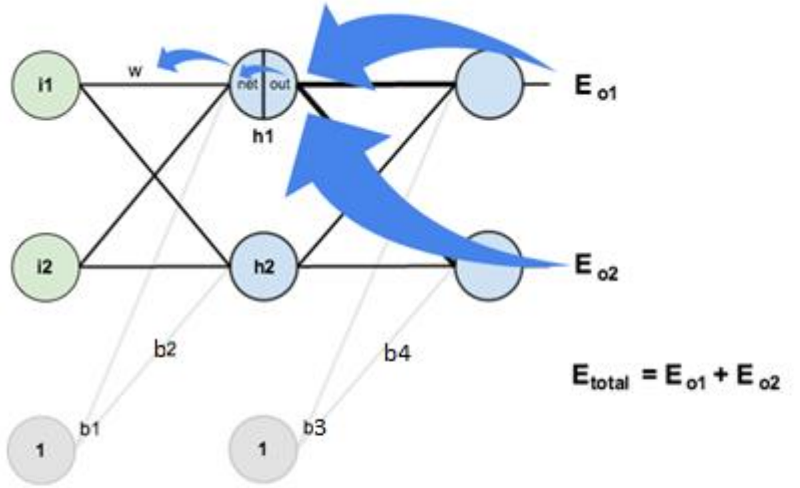
Hidden Layer (Cont.)

- 위 식을 모두 합치면 $\frac{\partial E_{total}}{\partial w_{(i_1,h_1)}} = \frac{\partial E_{total}}{\partial out_{h_1}} * \frac{\partial out_{h_1}}{\partial net_{h_1}} * \frac{\partial net_{h_1}}{\partial w_{(i_1,h_1)}}$
  - $\frac{\partial E_{total}}{\partial w_{(i_1,h_1)}} = 0.036350306 * 0.241300709 * 0.05 = 0.000438568$

$$\frac{\partial E_{total}}{\partial w_{(i_1,h_1)}} = \boxed{\frac{\partial E_{total}}{\partial out_{h_1}}} * \frac{\partial out_{h_1}}{\partial net_{h_1}} * \frac{\partial net_{h_1}}{\partial w_{(i_1,h_1)}}$$

$$\frac{\partial E_{total}}{\partial out_{h_1}} = \frac{\partial E_{o_1}}{\partial out_{h_1}} + \frac{\partial E_{o_2}}{\partial out_{h_1}}$$

- $w_{(i_1,h_1)}$ 을 update
  - $w^{+}_{(i_1,h_1)} = w_{(i_1,h_1)} - \eta \frac{\partial E_{total}}{\partial w_{(i_1,h_1)}} = 0.15 - 0.5 * 0.000438568$
    $= 0.149780716$



- 같은 방법으로 $w^{+}_{(i_2,h_1)} \sim w^{+}_{(i_2,h_2)}$ 를 반복
  - $w^{+}_{(i_2,h_1)} = 0.19956143$
  - $w^{+}_{(i_1,h_2)} = 0.24975114$
  - $w^{+}_{(i_2,h_2)} = 0.29950229$

3.2.2 Back-propagation algorithm

- 학습 결과 예제
  - 1st error = 0.298371109
  - 2nd error = 0.291027924
    - …
    - …
  - 10000th error = 0.000035085
    - 이때, 두 output neurons
      - 0.015912196 (vs 0.01 target)
      - 0.984065734 (vs 0.99 target)