

# 2022-1 교육세션 Deep Learning(MLP)

발제자: 19기 정은서



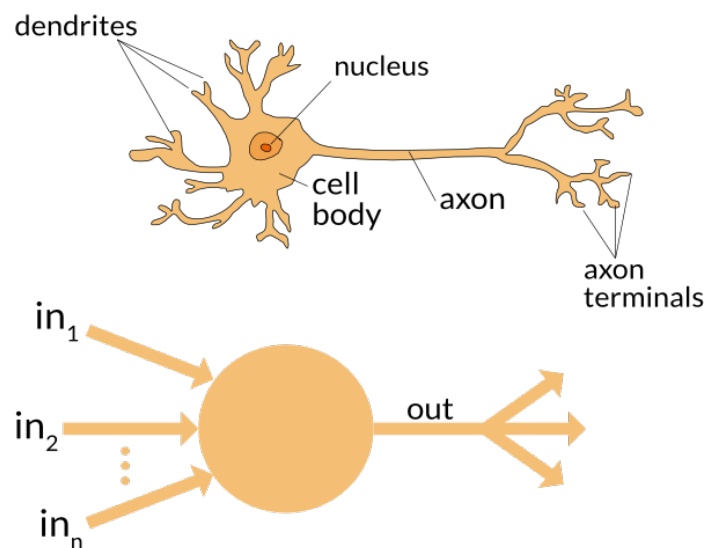
# 목차

1. Perceptron
2. SingleLayer Perceptron
3. MultiLayer Perceptron
4. Deep Learning
5. 실습

# 01 Perceptron

## 인공신경망

- 기계학습과 인지과학에서 생물학의 신경망(동물의 중추신경계중 특히 뇌)에서 영감을 얻은 통계학적 학습 알고리즘

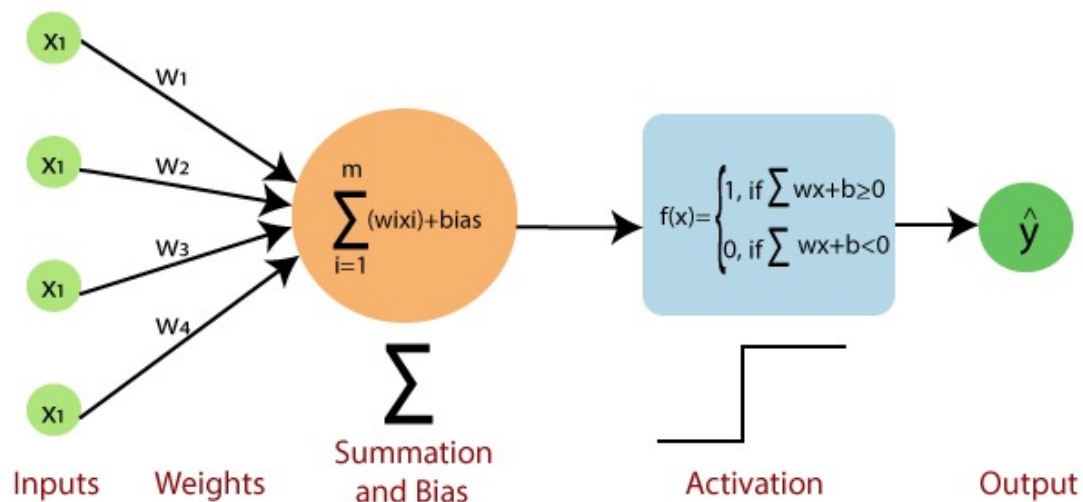


# 01 Perceptron

## 퍼셉트론

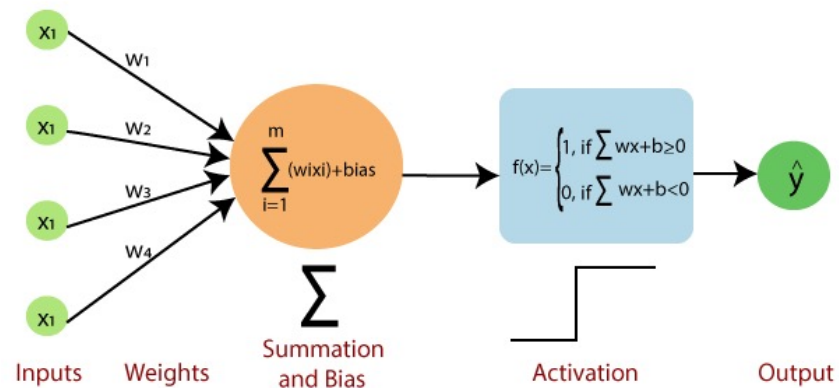
: 프랑크 로젠블라트가 1957년에 제안한 초기형태의 인공신경망

- 입력값-가중치-가중합-계단함수(활성화함수)-출력값

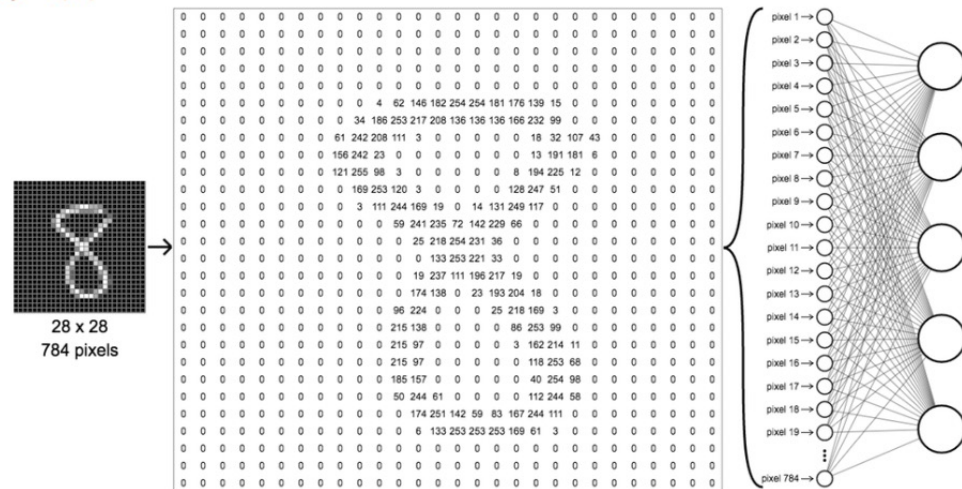


# 01 Perceptron

## Inputs



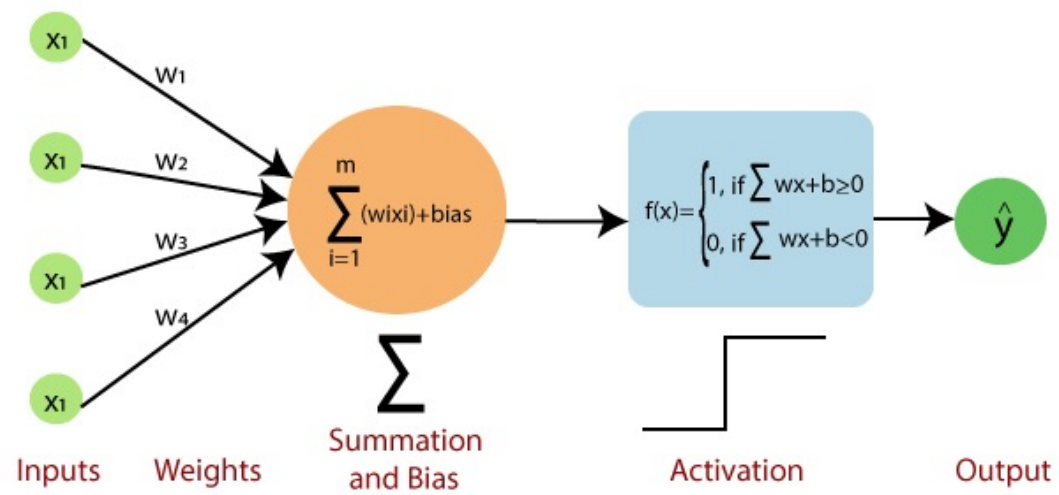
입력값(Inputs)-예시



# 01 Perceptron

## Weights

- 각 입력값에 대해 가중치 존재
- 가중치의 값이 클수록 해당 입력 값이 중요하다는 의미!



# 01 Perceptron

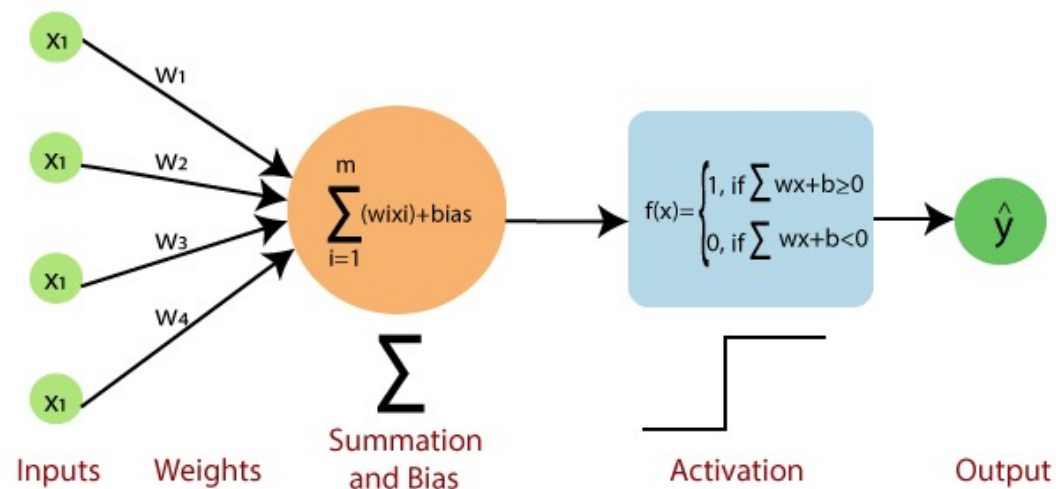
## Weighted Sum

- 가중합(net): 각 입력과 그에 해당하는 가중치의 곱의 합
- 편향: "퍼셉트론이 얼마나 쉽게 활성화되는가?"를 정하는 매개변수

$$w_0 = 0.3, w_1 = 0.4, w_2 = 0.1$$

$$x_0 = -1, x_1 = 0, x_2 = 0$$

$$net = w_1x_1 + w_2x_2 + w_0x_0 = -0.3$$

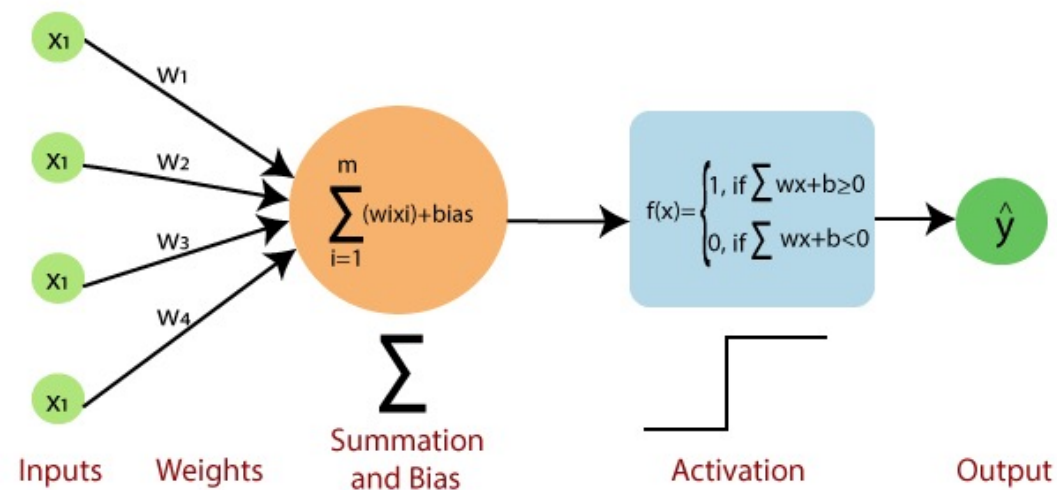


# 01 Perceptron

활성화 함수, 계단함수, threshold, bias

- 활성화 함수(Activation Func.): net값에 따라 뉴런의 활성화 여부를 결정하는 함수
- 계단함수: 가중합(net)이 임계값(threshold)보다 크면 1, 크지 않으면 0을 출력하는 일종의 활성화 함수
- 편향: "퍼셉트론이 얼마나 쉽게 활성화 되는가?"를 정하는 매개변수

$$f(net) = \begin{cases} 1, & net \geq threshold \\ 0, & net < threshold \end{cases}$$





# 01 Perceptron

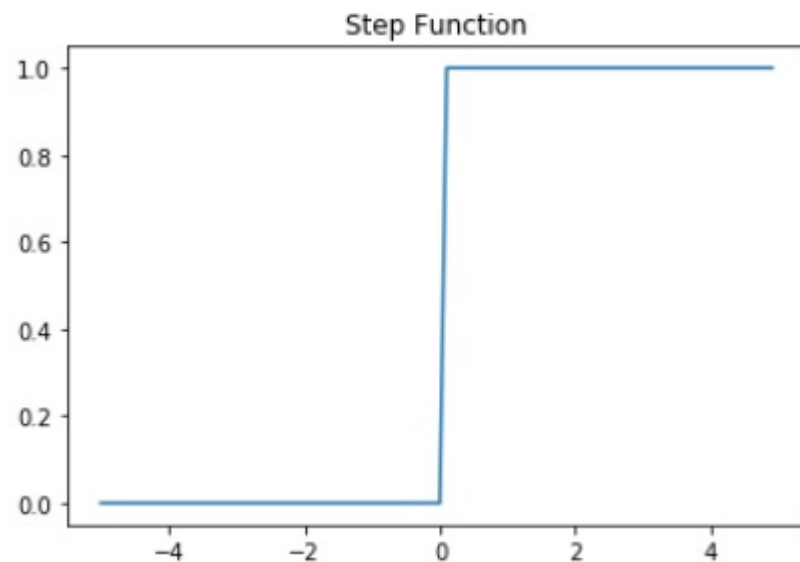
활성화 함수, 계단함수, threshold, bias

- 활성화 함수(Activation Func.): net값에 따라 뉴런의 활성화 여부를 결정하는 함수
- 계단함수: 가중합(net)이 임계값(threshold)보다 크면 1, 크지 않으면 0을 출력하는 일종의 활성화 함수
- 편향: "퍼셉트론이 얼마나 쉽게 활성화되는가?"를 정하는 매개변수

$$\text{if } \sum_i^n W_i x_i \geq \theta \rightarrow y = 1$$

$$\text{if } \sum_i^n W_i x_i < \theta \rightarrow y = 0$$

$$f(\sum_i^n W_i x_i + b) \geq 100 \rightarrow y = 1$$



# 01 Perceptron

활성화 함수, 계단함수, threshold, bias

- 계단함수: Net값이 임계값( $\theta$ )이상이면 1, 미만이면 0
- 임계값: 뉴런이 활성화되기 위한 최소값
- 편향: "퍼셉트론이 얼마나 쉽게 활성화되는가?"를 정하는 매개변수

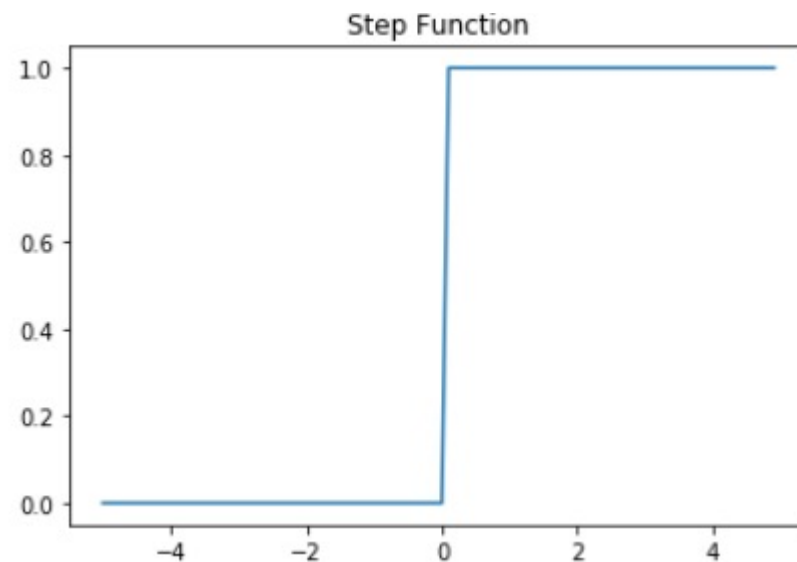
$$\text{if } \sum_i^n W_i x_i \geq \theta \rightarrow y = 1$$

$$\text{if } \sum_i^n W_i x_i < \theta \rightarrow y = 0$$

$$f\left(\sum_i^n W_i x_i + b\right) \geq 100 \rightarrow y = 1$$

$b = 70$

Net값이 30만 넘어도 100을 넘기니 활성화!



# 01 Perceptron

활성화 함수, 계단함수, threshold, bias

- 계단함수: Net값이 임계값( $\theta$ )이상이면 1, 미만이면 0
- 임계값: 뉴런이 활성화되기 위한 최소값
- 편향: "퍼셉트론이 얼마나 쉽게 활성화 되는가?"를 정하는 매개변수

$$\text{if } \sum_i^n W_i x_i \geq \theta \rightarrow y = 1$$

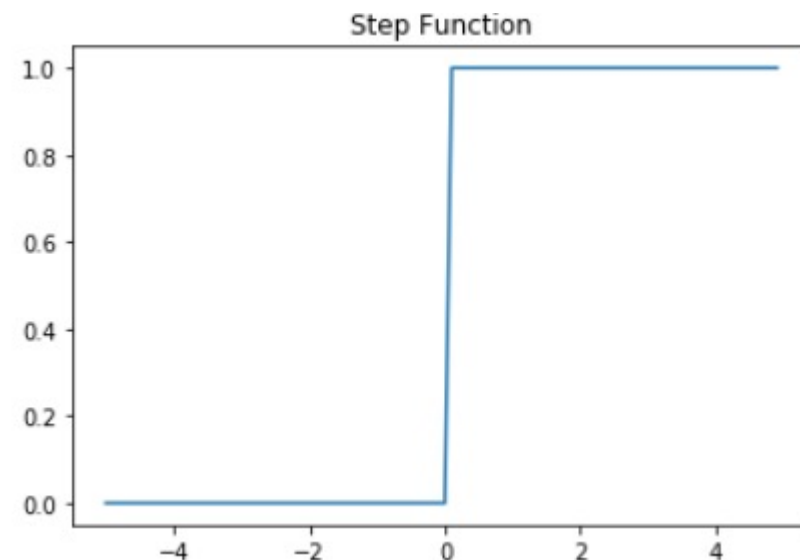
$$\text{if } \sum_i^n W_i x_i < \theta \rightarrow y = 0$$

$$f\left(\sum_i^n W_i x_i + b\right) \geq 100 \rightarrow y = 1$$

$b = 20$

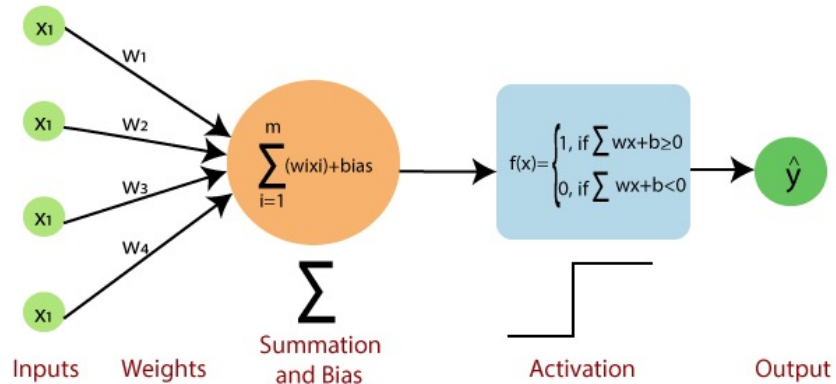
뉴런 활성화를 위해서는 80 넘는 net값이 필요!

편향이 커질수록 활성화 함수의 입력값에 대한 의존도(중요도)가 낮아진다고 볼 수 있다



# 01 Perceptron

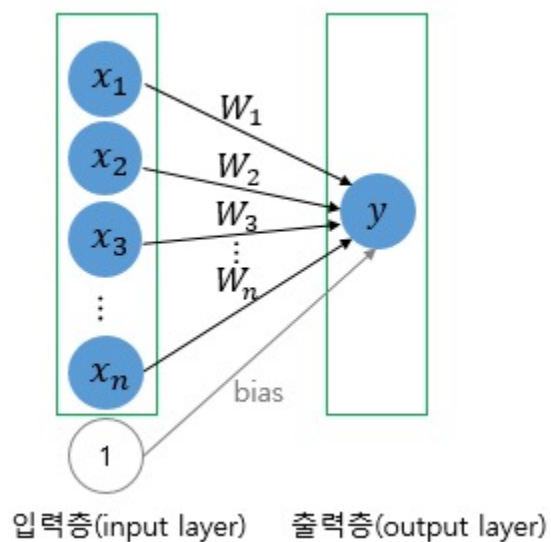
## Output



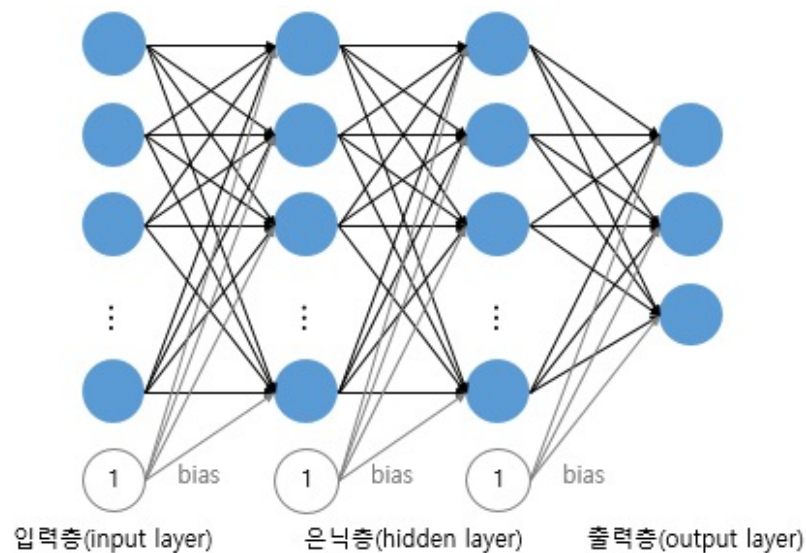
- 하나의 output, 예측값
- 계단함수를 사용하는 퍼셉트론에서는 0 or 1의 값을 갖는다
- 예측값은 목표값과 일치 or 불일치

# 01 Perceptron

## 퍼셉트론의 종류



[단층 퍼셉트론]

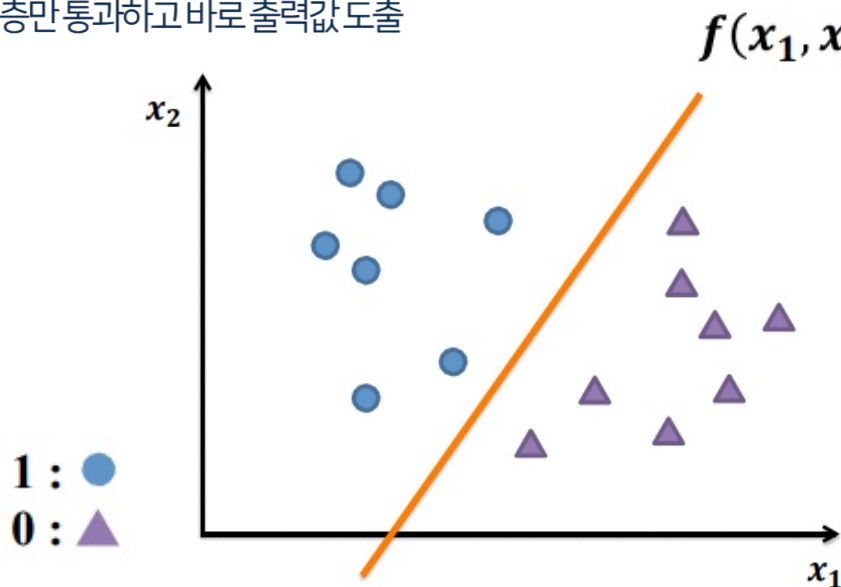


[다층 퍼셉트론]

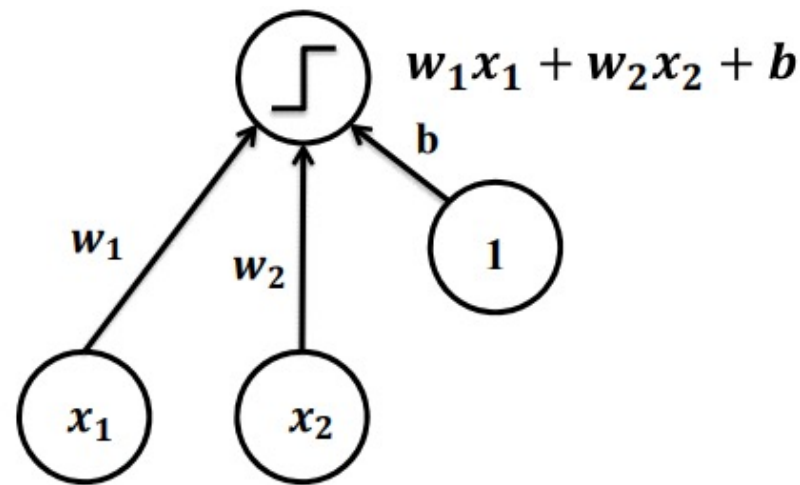
# 02 Single Layer Perceptron

## 단층 퍼셉트론

- Capable of: 선형적으로 분리되는 데이터에 대한 학습
- 하나의 층만 통과하고 바로 출력값 도출



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



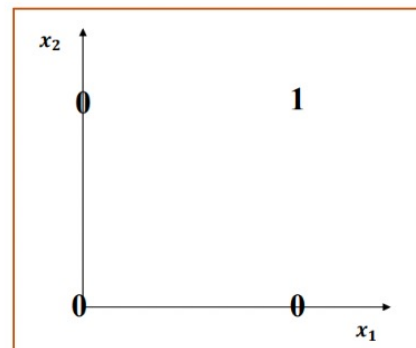
# 02 Single Layer Perceptron

단층 퍼셉트론 예시

- 단층 퍼셉트론으로 AND 게이트 구현해보기

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

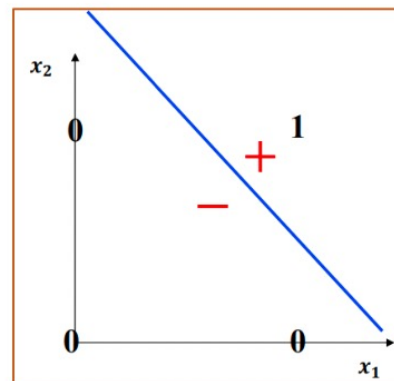
✓ AND 게이트 좌표평면에 시각화



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

✓ AND 게이트를 만족시키는 가중치와 편향 값 찾기

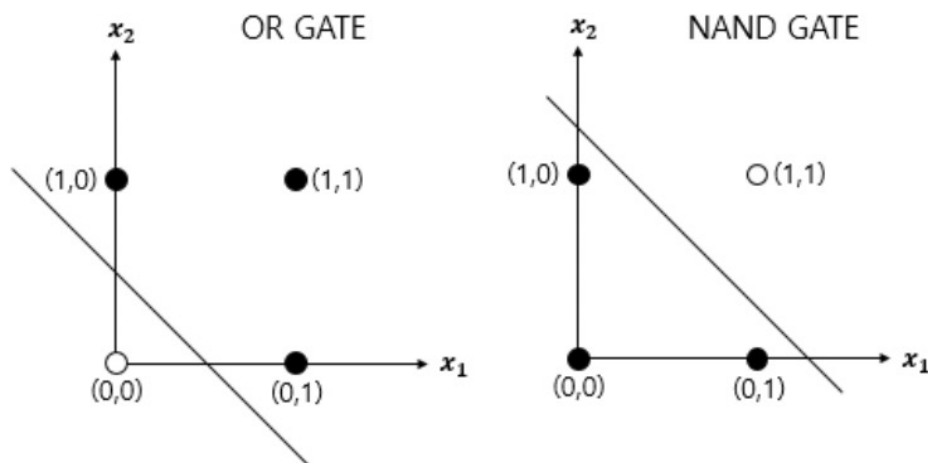
Ex)  $w_1 = 0.5, w_2 = 0.5, b = -0.7$



# 02 Single Layer Perceptron

단층 퍼셉트론 예시

- OR, NAND 게이트도 마찬가지로 단층 퍼셉트론으로 구현가능



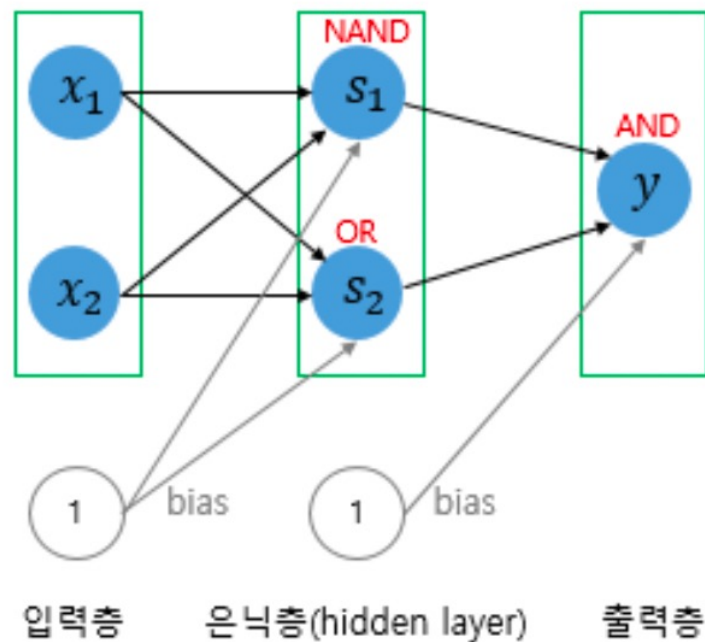
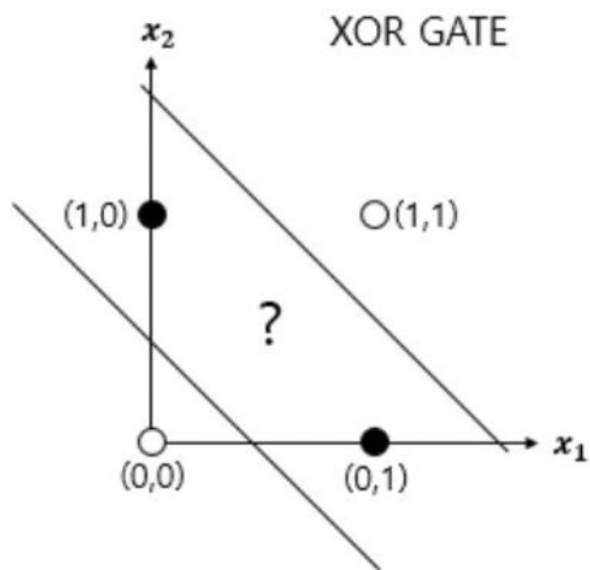
```
def OR_gate(x1, x2):  
    w1=0.6  
    w2=0.6  
    b=-0.5  
    result = x1*w1 + x2*w2 + b  
  
    if result <= 0:  
        return 0  
    else: return 1
```



# 02 Single Layer Perceptron

단층 퍼셉트론 한계

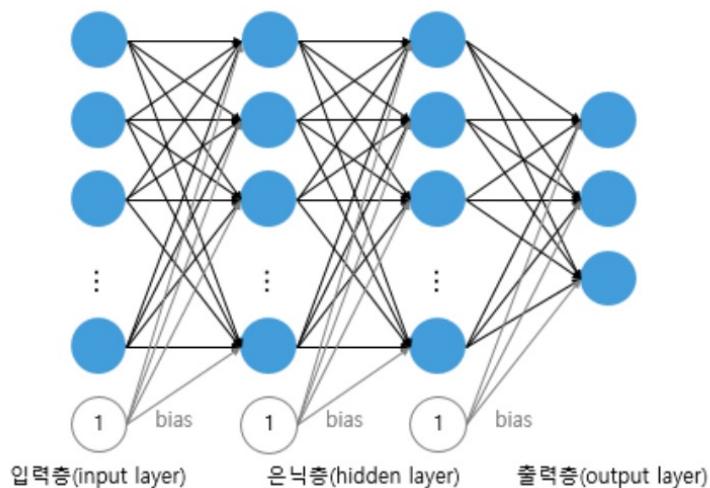
- Single Layer Perceptron: XOR 게이트를 구현하지 못한다
- 비선형적으로 분류되는 데이터에 대해서는 학습이 불가하다



# 03 Multi Layer Perceptron

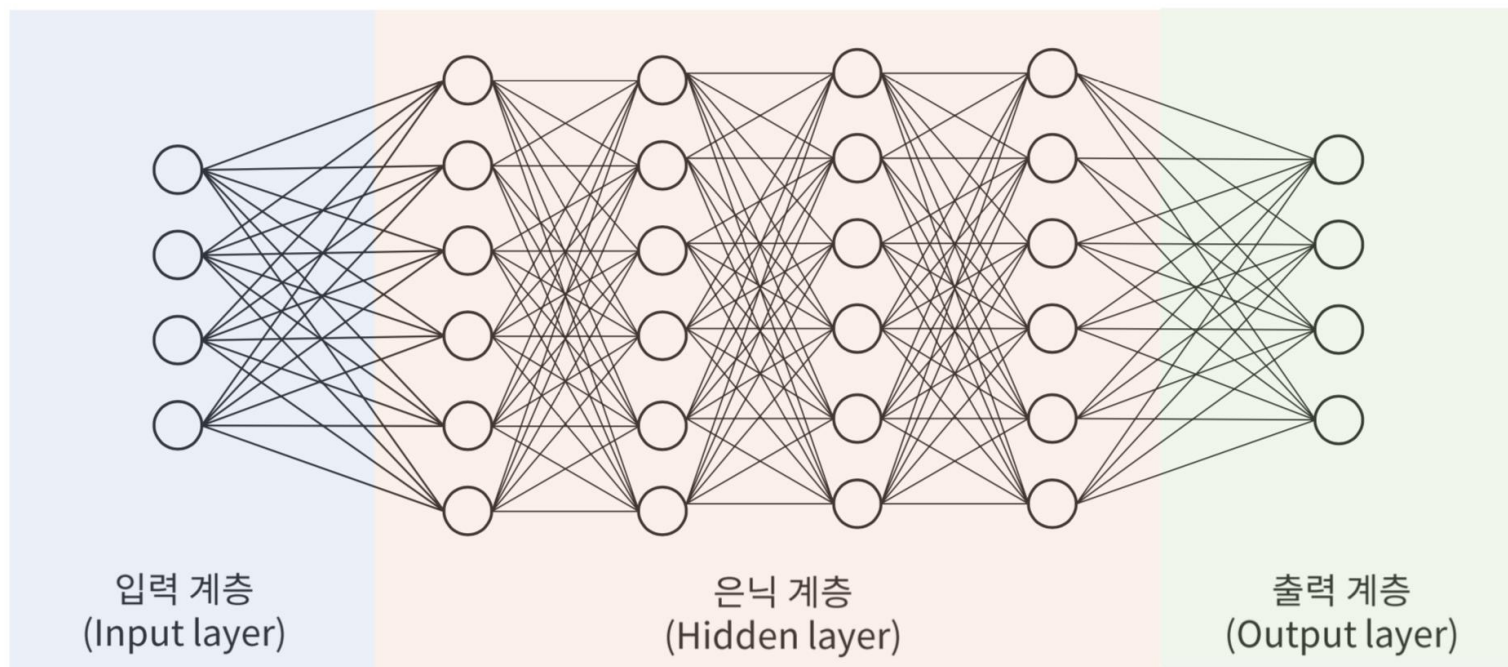
## 다층 퍼셉트론(MLP)

- MLP: 입력층과 출력층 사이에 한 개 이상의 중간층(은닉층)이 존재하는 인공 신경망
- Feedforward Network: 뒤에 이어지는 층으로의 전방향 연결(순환, 역행 X)
- 비선형적으로 분류되는 데이터에 대해서도 학습이 가능하다



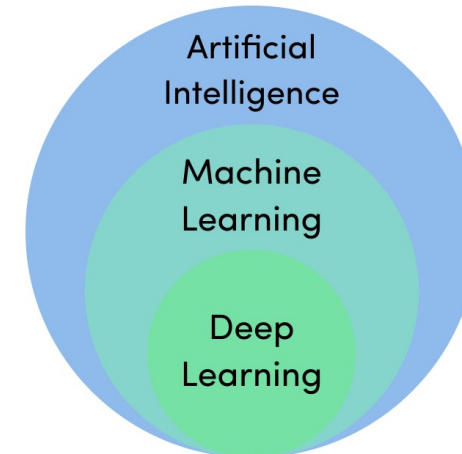
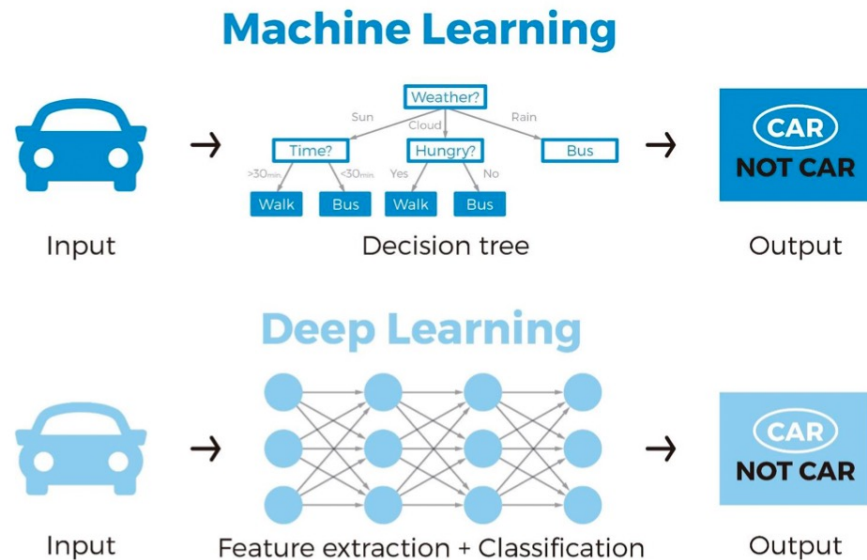
# 04 Deep Learning

DNN: Deep Neural Network



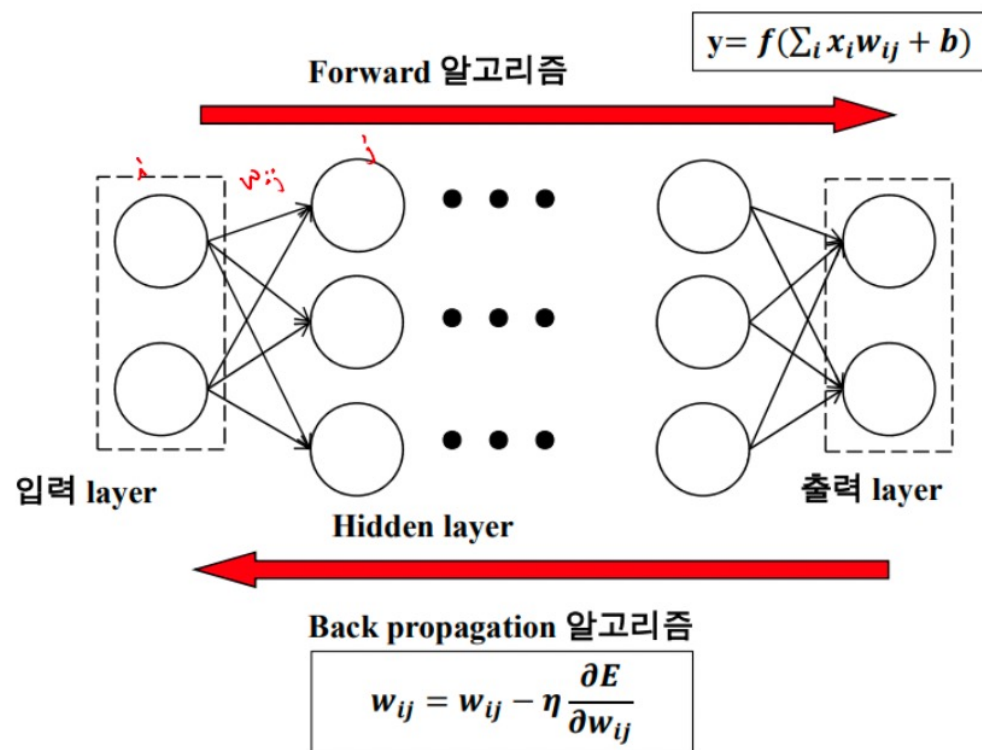
# 04 Deep Learning

## Deep Learning



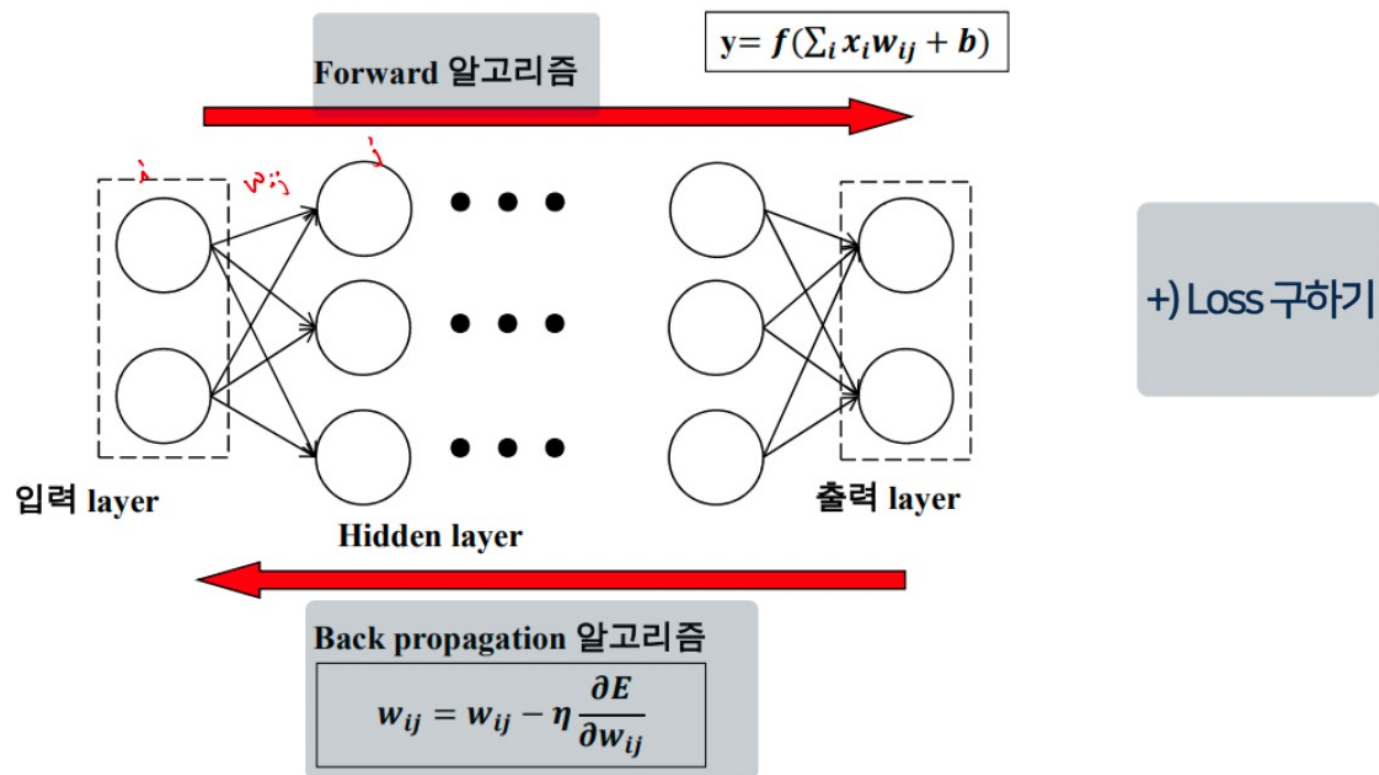
# 04 Deep Learning

- 딥러닝에서 학습이란?  
: 학습 자료(training data)를 이용하여 적절한  $W$ (가중치)를 추정하는 것



# 04 Deep Learning

- 순전파(Forward Propagation): training data에 대한 예측값을 구하는 과정
- 역전파(Backward Propagation): 예측값과 실제 정답값을 비교하여 적절한 가중치 값들을 학습해 가는 과정



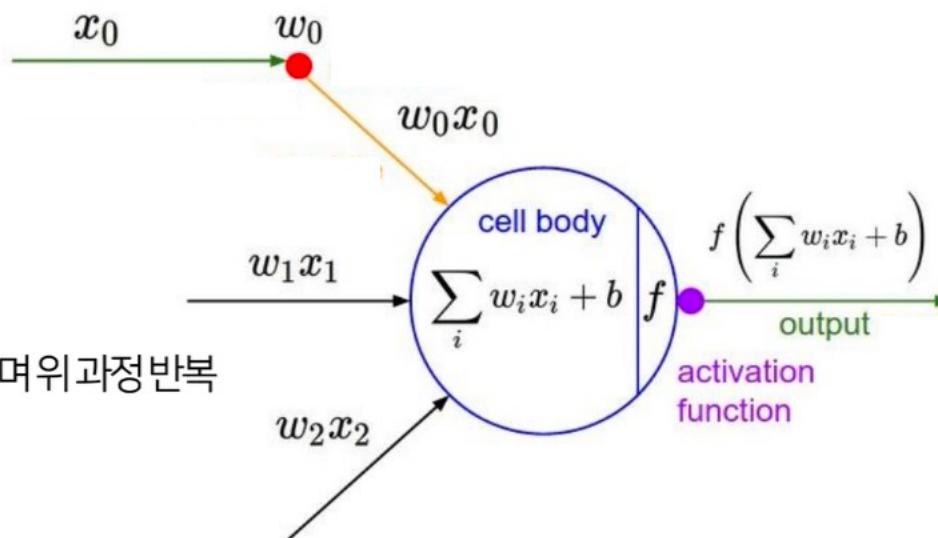
# 04 Deep Learning

## Forward Propagation

- 가중치를 이용한 연산을 통해 실제  $y$  값에 대한 예측값을 연산하는 과정
- $Y$ 에 대한 예측값을 통해 가중치에 대한 손실을 구할 수 있다

[순전파 과정]

1. Weight initialization
2. Weighted sum을 구한다
3. 활성화 함수(activation function) 적용하기
4. 해당 cell의 output 출력
5. 5.의 output 이 다음 층의 cell에 입력으로 주어지며 위 과정 반복
6. NN 전체의 output 도출





# 04 Deep Learning

## Weight initialization

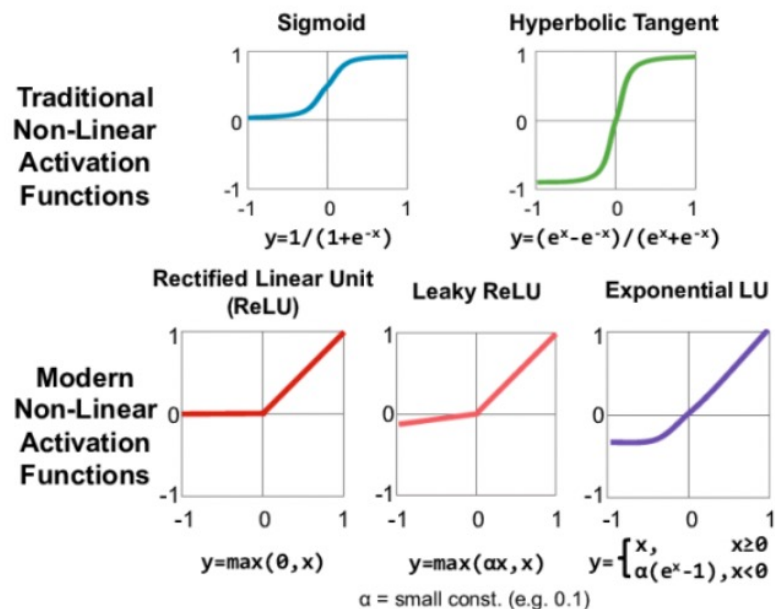
- random initialization: 가중치 값들을 처음에 random하게 initialization을 해주어 임의의 가중치 값에 대한 손실을 계산해서 최적의 가중치 값을 찾아나감
- 같은 모델을 훈련시키더라도 가중치가 초기에 어떤 값을 가졌느냐에 따라서 결과가 달라짐
- 가중치 초기화의 여러 방법: xavier initialization, he initialization etc.
- bias는 통상적으로 0으로 initialization하는 것이 효율적



# 04 Deep Learning

## Activation Function

- Activation Function을 사용하는 것은 선형성(linearity)을 깨뜨리기 위함
- Hidden layer-ReLU
- Output layer-이진분류시 sigmoid, 다중분류시 softmax function을 자주 사용



### \*Non-linearity(비선형성) 이해하기\*

선형함수인  $h(x)=cx$ 를 활성화함수로 사용한 3층 네트워크

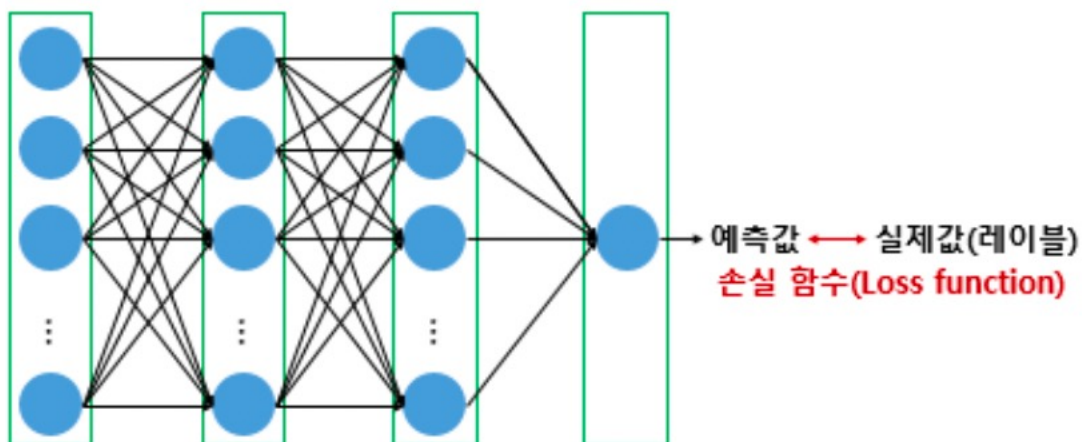
: 이를 식으로 나타내면  $y(x)=h(h(h(x)))$

But,  $y(x)=ax$ 와 똑같은 식. ( $a=c_3$ )

즉, 은닉층이 없는 네트워크로 표현할 수 있다.  
NN에서 층을 쌓는 혜택을 얻고 싶다면 Activation Function으로는 반드시 **비선형 함수**를 사용해야 한다.

# 04 Deep Learning

## Loss Function



나는 자전거를 \_\_\_\_\_  
빈칸에 들어갈 다음 단어는?

산다	0.22
부신다	0.02
고친다	0.20
탄다	0.51
보낸다	0.05

예측  
확률 분포

산다	0.22
부신다	0.02
고친다	0.20
탄다	0.51
보낸다	0.05

실제  
정답

1
0
0
0
0

이 둘의 차이를 수치화하는 함수

- 손실 함수: 실제값과 예측값의 차이를 수치화하는 함수
- 오차가 클수록 손실함수 값이 큼!
- Mean Squared Error, Cross-Entropy function 등

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2$$

# 04 Deep Learning

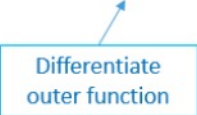
## Back Propagation

- Optimizer: NN이 손실함수 값을 줄여나가는 방법 (ex. Gradient Descent, SGD, Adam, RMSprop etc.)
- 각 가중치 값( $\theta$ )에 대한 loss function의 gradient(미분값)를 연산하는 과정
- 가중치에 대한 미분값을 구하기 위해 Chain Rule 사용

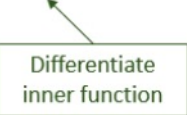
**Chain Rule**

If  $f$  and  $g$  are both differentiable and  $F(x)$  is the composite function defined by  $F(x) = f(g(x))$  then  $F$  is differentiable and  $F'$  is given by the product

$$F'(x) = f'(g(x)) g'(x)$$



Differentiate  
outer function



Differentiate  
inner function

Backpropagation: a simple example

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

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

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

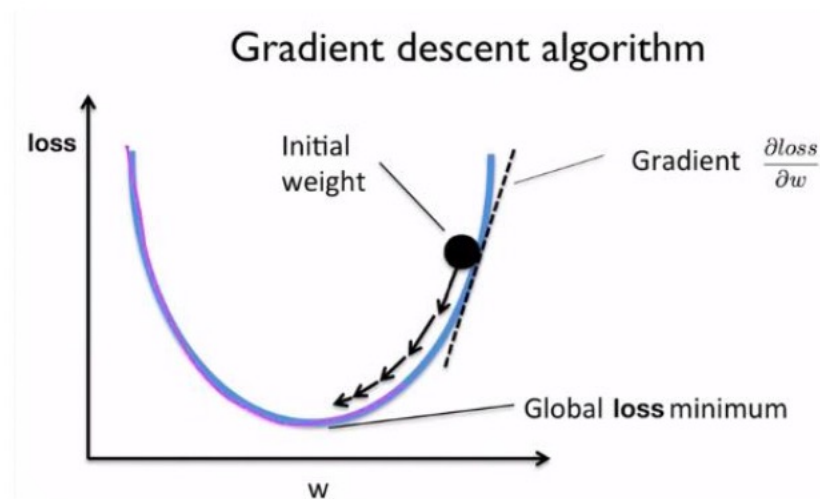
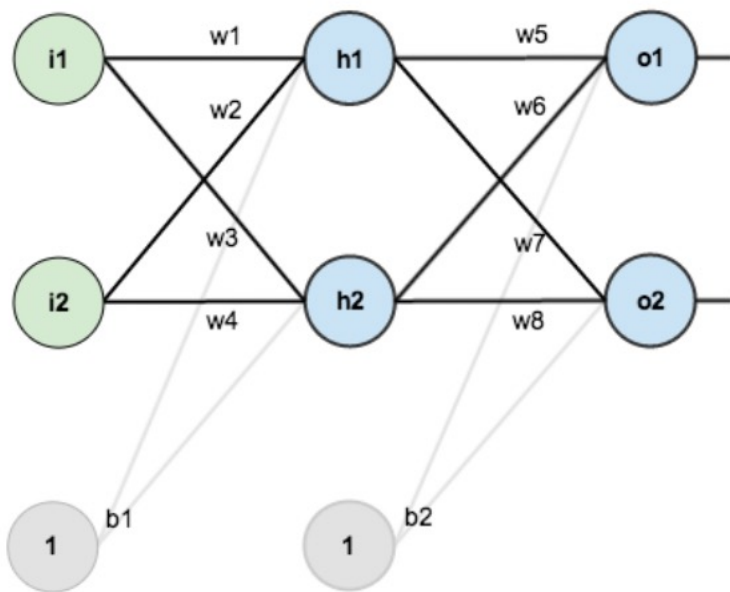
$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

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

# 04 Deep Learning

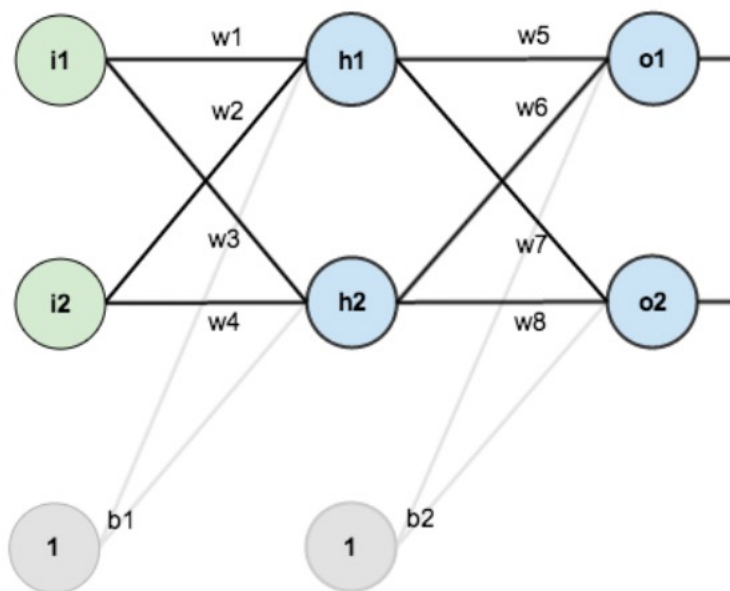
## Back Propagation

- 각 가중치 값( $\theta$ )에 대한 loss function의 gradient(미분값)를 연산하는 과정
- 가중치에 대한 미분값을 구하기 위해 Chain Rule 사용



# 04 Deep Learning

## Back Propagation



1. 순전파연산
2. Loss function 구하기
3. Loss function의 각 가중치에 대한 미분값 구하기
4. 가중치 업데이트하기
5. Repeat until convergence

# 04 Deep Learning

## 더 알아보기

### Hyperparameters

- 신경망 학습을 통해 튜닝되는 변수가 아니라 사용자가 직접 지정해야 하는 값들
- 하이퍼파라미터 종류: Learning rate, batch size, epoch, cost function, early stopping, hidden layer개수, weight initialization
- Hyperparameter의 최적화 방법: Manual Search, Grid Search, Random Search, Bayesian optimization 등 여러 방법이 제시됨

### Overfitting

- 통계나 ML에서, 제한된 데이터에 모델이 너무 특화되어 새로운 샘플에 대해서는 오히려 예측결과가 나빠지거나, 학습 효과가 나타나지 않는 경우
- 해결방법: 데이터 증식, drop-out, 가중치 규제, 모델 복잡도 줄이기

### Gradient Vanishing Problem

- DNN의 역전파 과정에서 입력층으로 갈 수록 gradient가 점차 작아지는 현상
- 입력층에 가까운 layer에서 가중치 업데이트가 제대로 되지 않아 최적의 모델을 찾기 어려워지는 문제
- Batch Normalization, 가중치 초기화 방법, gradient clipping, 활성화 함수 선택 등의 해결책이 있음

# 04 Deep Learning

## 더 공부하기

- 딥러닝을 이용한 자연어처리 입문: <https://wikidocs.net/61375>
- 활성화함수, 하이퍼 파라미터 튜닝, 옵티마이저, 가중치 초기화 등에 대한 자세한 설명:  
<https://ratsgo.github.io/deep%20learning/2017/04/22/NNtricks/>
- 역전파 연산과정 상세: <https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/>
- Stanford CS231n: computer vision, rnn, cnn 등 평강
- Coursera: Neural Networks and Deep Learning
- 3Blue 1Brown 유튜브 채널: <https://youtu.be/aircAruvnKk>
- Keras: <https://keras.io/ko/callbacks/>

# 05 실습



끝!

감사합니다