

다층 신경망

MLP : **M**ulti-**L**ayer **P**erceptron

MLP

- MLP_1
 - NN 구조 - 입출력
 - NN 구조 - node, weight, bias
- MLP_2
 - NN 구조 - Layer
 - NN 학습 프로세스 - Backpropagation
 - NN 학습 프로세스 - 경사하강법, chain-rule

MLP

MLP - Multi-Layer Perceptron

Q. 퍼셉트론? “뇌의 시각 자극 반응을 모방하는 인공 시스템”

Perceptron

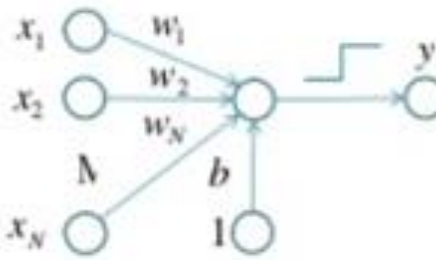
- 1957년 프랭크 로젠블랫(Frank Rosenblatt)에 의해 처음 소개
- 입력(features)을 받아 각각의 가중치(weights)를 적용하고, 모든 가중치가 적용된 합을 출력하는 간단한 형태의 신경망 모델
- 단일 뉴런 = 퍼셉트론
- 인지(perception) -> perceptron

MLP

MLP - Multi-Layer Perceptron

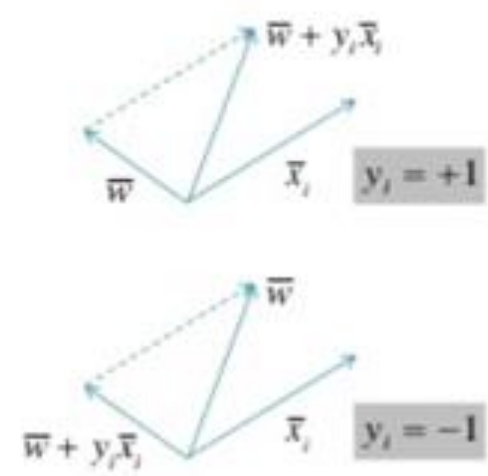
Q. 퍼셉트론? “뇌의 시각 자극 반응을 모방하는 인공 시스템”


Perceptron



decision function
 $\hat{f}(\bar{x}) = \text{sign}(\bar{w} \cdot \bar{x} + b)$
 $\bar{x} = (x_1, x_2, \dots, x_N)$ features of the customer \bar{x}
approve credit if $\sum_{i=1}^N w_i x_i > \text{threshold}$
deny credit if $\sum_{i=1}^N w_i x_i < \text{threshold}$

input: $T = \{(\bar{x}_i, y_i) | i = \overline{1, p}\} \subset \mathbb{R}^N \times \{+1, -1\}$
 $\bar{w} \leftarrow \vec{0}, b \leftarrow 0$
repeat
 for $i=1$ to p
 if $\text{sign}(\bar{w} \cdot \bar{x}_i + b) \neq y_i$ then
 $\bar{w} \leftarrow \bar{w} + y_i \bar{x}_i$
 $b \leftarrow b + y_i$
 end if
 end for
until $\text{sign}(\bar{w} \cdot \bar{x}_j + b) = y_j, \forall j = \overline{1, p}$
return \bar{w}, b





Frank Rosenblatt

COMMERCIAL AIRCRAFTS CORPORATION, INC.
BOSTON, U.S.A.

MEMBER: ILLINOIS
THE UNIVERSITY
A PROGRAM IN RESEARCH AND DEVELOPMENT
DECEMBER 1962
JANUARY, 1967

Presented by *Frank Rosenblatt*
Frank Rosenblatt
Project Engineer

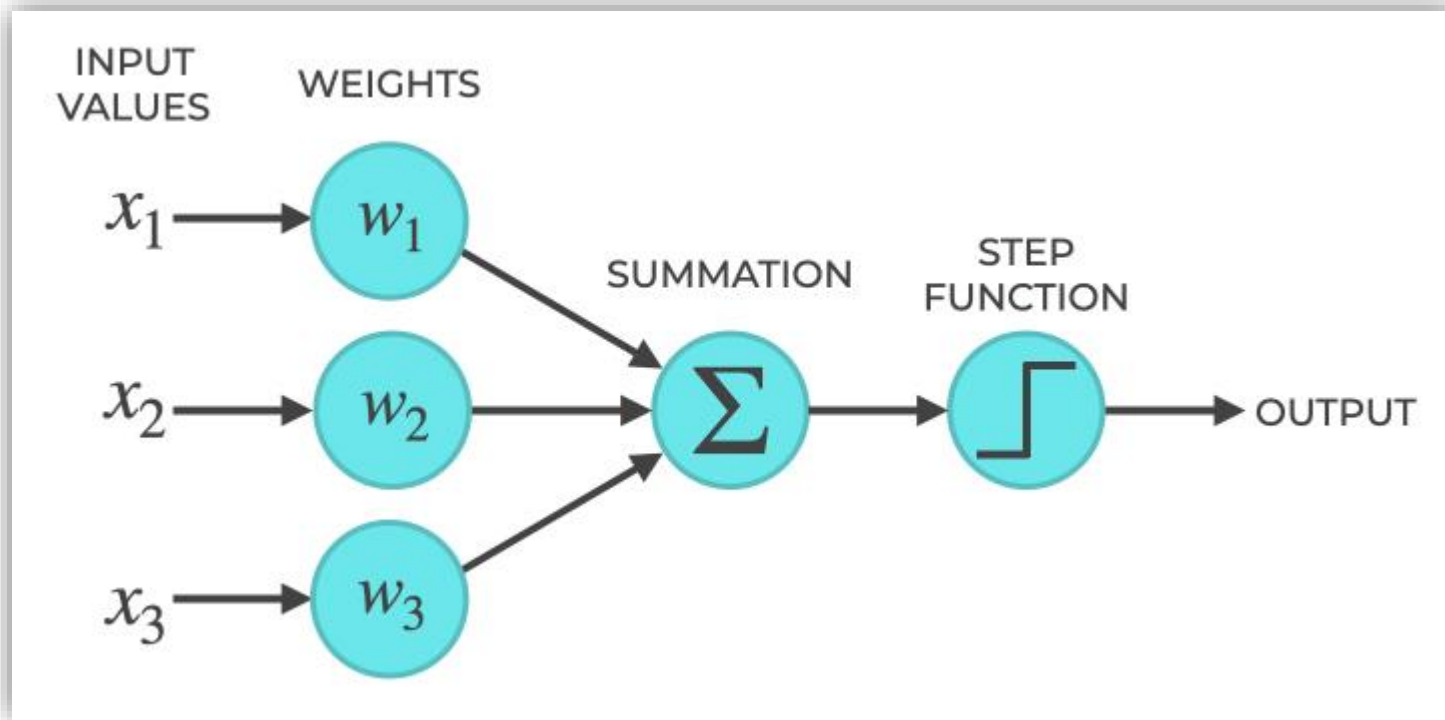
MLP

MLP - Multi-Layer Perceptron

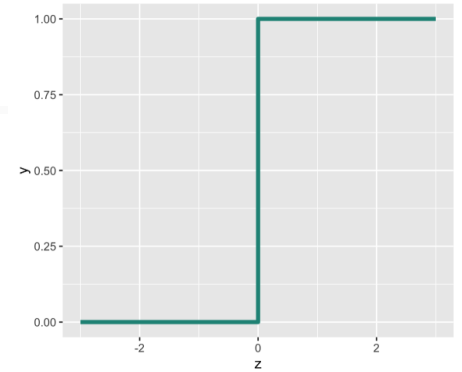
퍼셉트론

Perceptron

입력, 가중치
가중치가 적용된 입력의 합이 특정 임계값에 도달하면 출력 생성



<https://www.sharpsightlabs.com/blog/python-perceptron-from-scratch/>



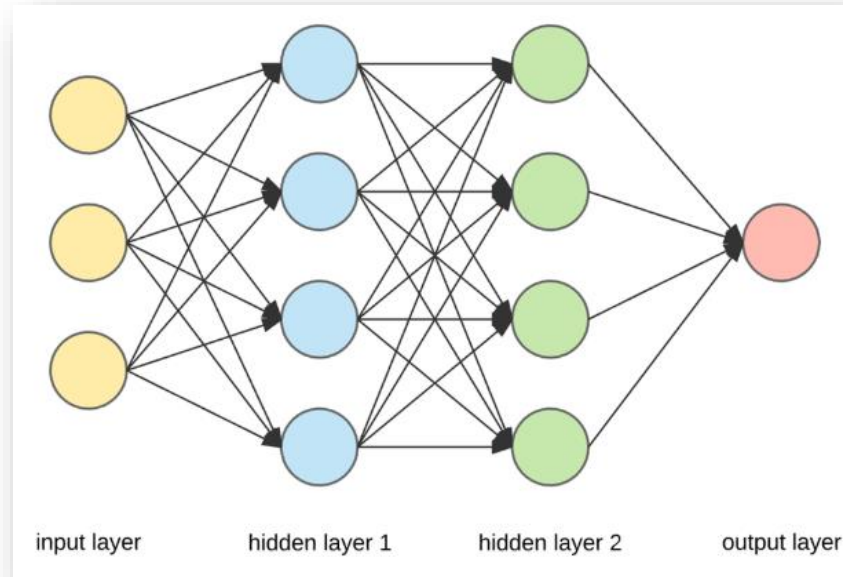
특정 임계값에
도달하면
출력 생성

MLP

MLP - Multi-Layer Perceptron

Perceptron *vs* Neural Network

- **Neural Network (신경망)** : 1943년, McCulloch & Pitts "신경망 모델" 을 수학적으로 정의
 - 퍼셉트론들이 모여 구성된 전체 구조
 - cf, Perceptron : 신경망의 한 개 노드



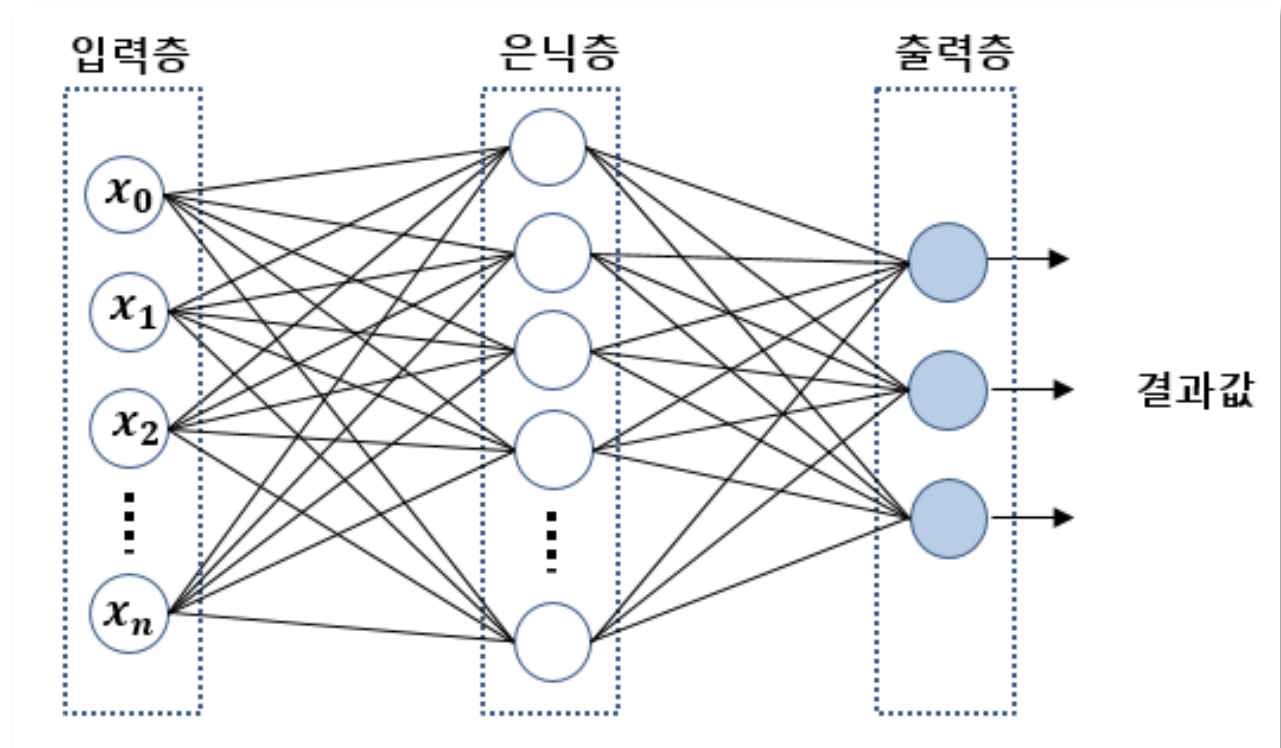
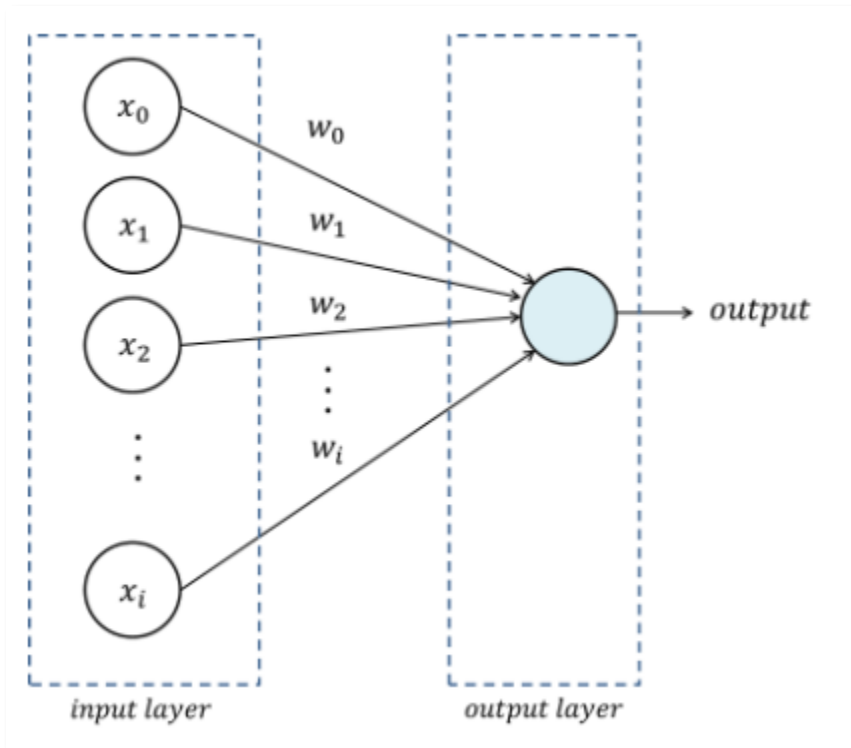
MLP

MLP - Multi-Layer Perceptron

단층

VS

다층

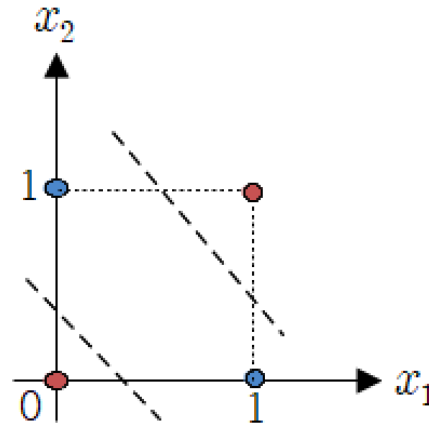
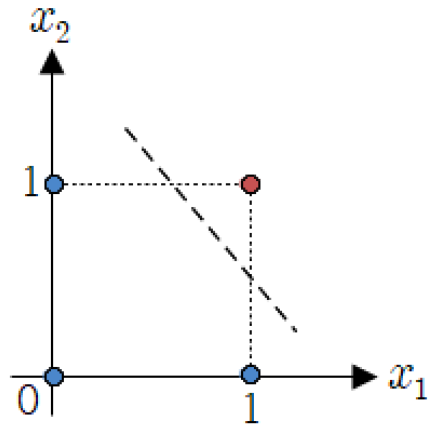


MLP

MLP - Multi-Layer Perceptron

- 단층 신경망의 한계

- Minsky가 단층 신경망은 선형 분리가 불가능한 문제를 풀 수 없다고 지적(1969년)



XOR 문제

선형 분리 불가능(Linearly inseparable) 문제

MLP

MLP - Multi-Layer Perceptron

- 단층 신경망의 한계

- 선형으로 분류되지 않는 데이터들에 대해 적절한 모델링 어려움
- 많은 데이터는 비정형 데이터(이미지, 음성 데이터 등)이며 대부분이 선형적으로 모델링 되지 않음

1. 선형 분리성의 제한:

XOR 문제와 같은 선형으로 분리 불가능한 문제들에 적용 안됨

2. 복잡한 문제 해결 능력 부족:

입력과 출력 사이의 복잡한 관계를 모델링하는 데 한계. 실세계의 많은 문제들은 복잡하고, 비선형적인 관계를 포함하고 있음 -> 단층 신경망보다 더 복잡한 구조 필요

3. 특성 추출 능력 부족:

고차원 데이터에서 유의미한 특성을 자동으로 추출하고 학습하는 능력이 매우 제한적

MLP

MLP - Multi-Layer Perceptron

- 단층 신경망의 한계

선형 *VS* 비선형

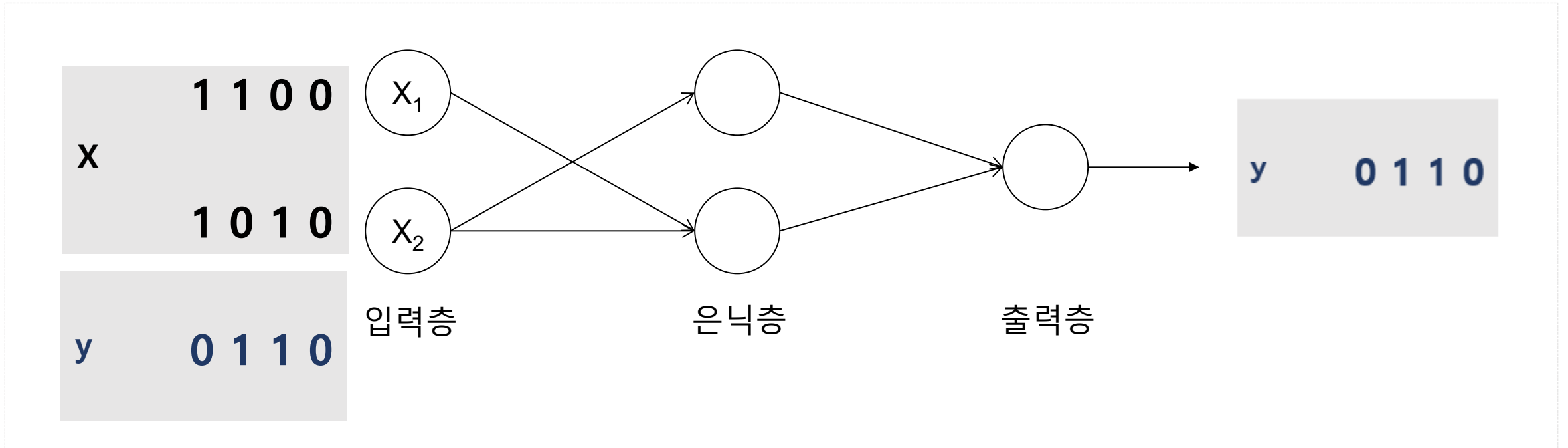
deeplearning.linear.nonlinear

MLP

MLP - Multi-Layer Perceptron

- 단층 신경망의 한계

- XOR 문제 -> 은닉층 추가 -> Multi-Layer Perceptron



MLP

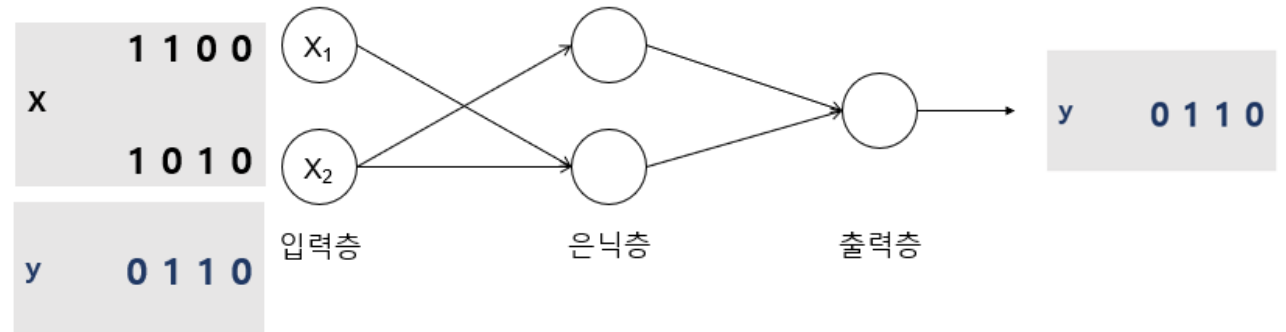
MLP - Multi-Layer Perceptron

- 단층 신경망의 한계

- XOR 문제 -> 은닉층 추가 -> Multi-Layer Perceptron

- 실습 - XOR 문제 해결
5.03.MLP_XOR.ipynb

cf, 4.03.NN. Backpropagation.ipynb



MLP

MLP - Multi-Layer Perceptron

| 축약어 | 용어 | 설명 |
|-----|----------------------------|------------------------------|
| NN | Neural Network | 인공 신경망(네트워크), 가장 일반적으로 사용 |
| ANN | Artificial Neural Network | "인공"이라는 표현 강조 (보통 NN과 동일 의미) |
| FNN | Feedforward Neural Network | 순방향 신경망 (순차적인 데이터 흐름 강조) |
| MLP | Multi-Layer Perceptron | 다층 퍼셉트론 (은닉층이 1개 이상인 FNN) |

MLP

MLP - Multi-Layer Perceptron

- **Single Layer Perceptron (SLP, 단층 신경망)**
- **Multi-Layer Perceptron (MLP, 다층 신경망)**
- **Shallow Neural Network (얕은 신경망)**
- **Deep Neural Network (DNN, 심층 신경망)**

MLP

MLP - Multi-Layer Perceptron - 정의

- 하나 이상의 은닉층(hidden layers)을 포함하는 feedforward 신경망
- 복잡한 데이터 패턴을 인식하고 분류, 회귀, 패턴 인식 등의 다양한 작업에 사용
- 입력층(input layer)
 - + 하나 또는 여러 개의 은닉층(hidden layer)
 - + 출력층(output layer) 으로 구성

MLP

MLP - Multi-Layer Perceptron - 특징

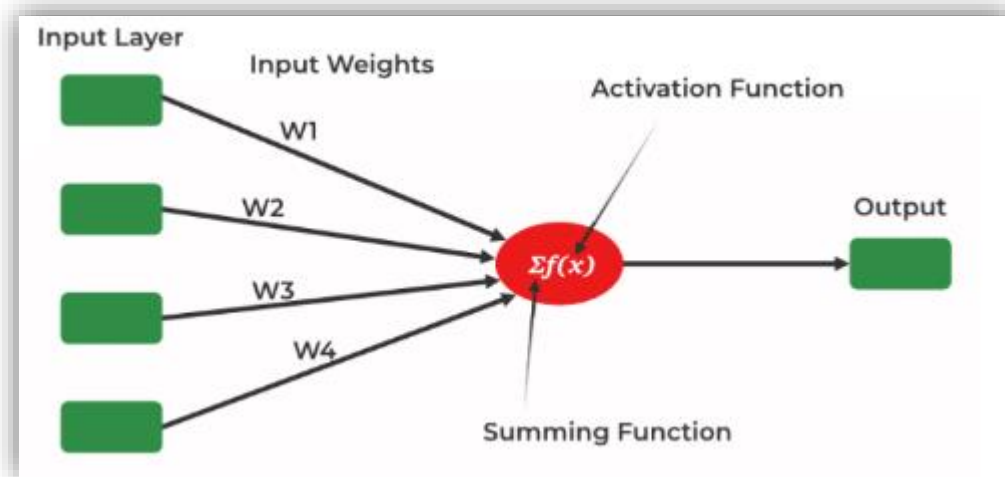
- **비선형성:** 비선형 활성화 함수를 사용하여 선형적으로 구분되지 않는 데이터 처리
(활성화 함수: 시그모이드(sigmoid), 하이퍼볼릭 탄젠트(tanh), 렐루(ReLU) 등)
- **Feed forward 구조:** 입력층에서 출력층으로 data가 순차적으로 전달되는 전향적 구조
데이터가 여러 은닉층을 거치며 변환
- **역전파 알고리즘:** backpropagation. 출력층에서 입력층 방향으로 오차 전파
가중치(weights)와 편향(biases)을 조정 -> 주어진 입력에 대한 예측 오류를 최소화하도록 학습
- **범용 근사자:** 매우 복잡한 함수나 패턴 학습 가능

MLP

- 실습 – 단층 신경망(Single Layer Perceptron)
5.03.MLP_SLP.toy.ipynb

5.03.MLP_SLP.ipynb (iris)

cf) Multi-Layer Perceptron



MLP

NN

기본 딥러닝 알고리즘

- Feedforward neural network (FNN)
- Convolutional neural network (CNN)
- Recurrent neural network (RNN)

MLP

NN

기본 딥러닝 알고리즘

- **FNN (Feedforward Neural Network)**

입력부터 출력까지 정보가 앞으로 전달

순환(loop) 없음

- **DNN (Deep Neural Network, 깊은 신경망):**

여러 개의 은닉층을 포함하는 신경망

복잡한 문제 모델링 가능

- **CNN (Convolutional Neural Network, 합성곱 신경망)**

이미지나 비디오 데이터 처리에 적합

합성곱 계층을 사용하여 이미지에서 패턴을 추출

- **RNN (Recurrent Neural Network, 순환 신경망):**

시계열 데이터나 자연어 처리와 같이 데이터가 순서에 의미가 있는 경우에 유용

이전의 정보를 다음 상태의 입력으로 사용하는 순환 구조

MLP

NN

딥러닝 알고리즘 – advanced

- **CNN base**
 - CNN 모델의 발전
 - Pre-trained Model
 - Transfer Learning
 - Object detection (YOLO, SSD, R-CNN family)
- **RNN base**
 - Seq2seq
 - LSTM, GRU
- **LM**
 - Transformer (Attention)
 - BERT & BERT variants
 - GPT models

인공 신경망 학습 프로세스

1. 초기화(Initialization)

> 학습 과정을 시작하기 위한 준비 단계 . (보통)무작위 값으로 초기화.

2. 순전파(Forward Propagation)

> 입력 데이터가 신경망의 입력층으로 주어짐, 각 층의 뉴런을 통과하면서 **활성화 함수**에 의해 처리

> 신경망의 출력층까지 과정 수행

* **활성화 함수** : 신경망에 비선형성을 도입하여 복잡한 문제 해결 수행

3. 오차 계산(Error Calculation)

> 신경망의 출력과 실제 타겟 값 사이의 차이 계산(**손실 함수** - 평균 제곱 오차(MSE)나 교차 엔트로피(Cross-Entropy) 사용)

> 신경망의 예측이 얼마나 정확한지를 수치적으로 표현

4. 오류 역전파(Error **Backpropagation**)

> 계산된 오차는 출력층에서부터 입력층 방향으로 역전파 되어 각 가중치가 오차에 얼마나 기여하는지 계산

각 가중치가 오차에 미치는 영향 계산 -> 가중치 조정

인공 신경망 학습 프로세스

5. 가중치 업데이트(Weight Update)

- > **경사 하강법**을 사용하여 각 가중치를 업데이트.
- > 오차를 최소화하는 방향으로 가중치를 조정하여 신경망의 성능 개선

6. 과적합(Overfitting) 및 정규화(Regularization)

- > 학습 과정 중 **과적합** 발생 가능(학습 데이터에 대해 너무 잘 맞추어져 새로운 데이터에 대한 일반화 성능이 떨어지는 현상)
- > **정규화** 기법(L1, L2 규제, 드롭아웃 등)을 통해 모델의 복잡도 제어, 과적합 줄임

7. 반복 학습(Iterative Learning)

- > 위의 과정(**순전파** → **오차 계산** → **오류 역전파** → **가중치 업데이트**) 계속 반복, 최적화 진행
- 위 학습 단계를 통하여 입력 데이터에 내재된 복잡한 패턴과 구조를 **학습**,
다양한 문제에 대한 **예측**(Predict) 수행

MLP

인공 신경망 학습 프로세스

순전파_(활성화함수) → 오차 계산_(손실함수) → 오류 역전파 → 가중치 업데이트
(과적합 방지 -> 정규화, 일반화)

MLP

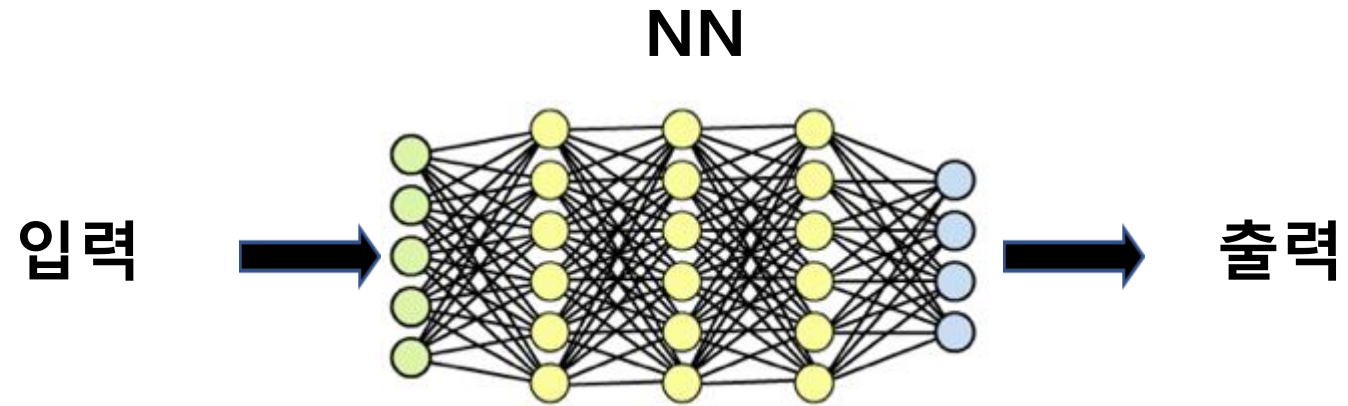
인공 신경망 학습 프로세스

NN Keyword

- 순전파
- 활성화 함수
- 비용함수
- 손실함수
- 역전파
- 경사하강법
- 경사소실
- 과적합
- Optimizer
- 정규화(Regularization) 일반화 (Generalization)

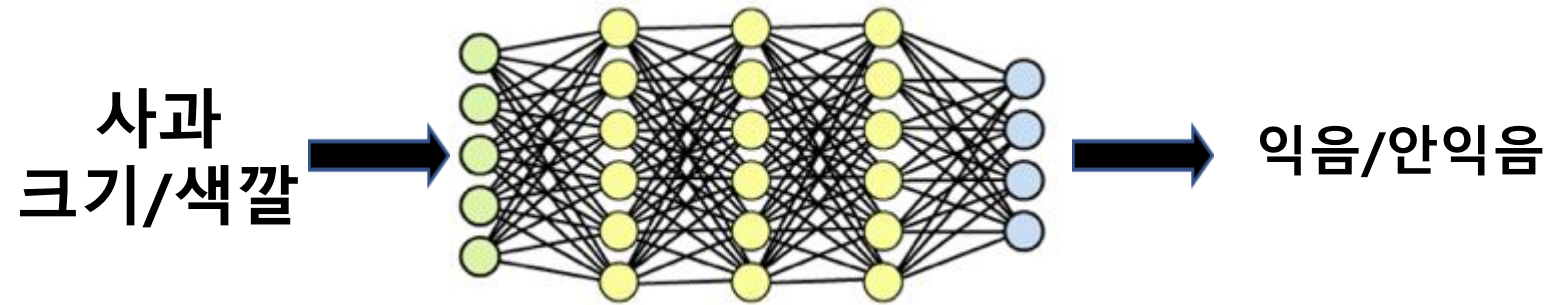
MLP

NN 구조 - 입출력



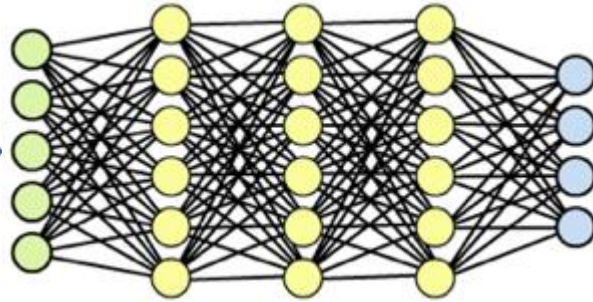
MLP

NN 구조 - 입출력

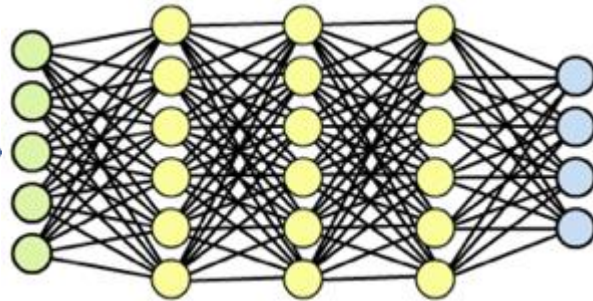


MLP

NN 구조 - 입출력



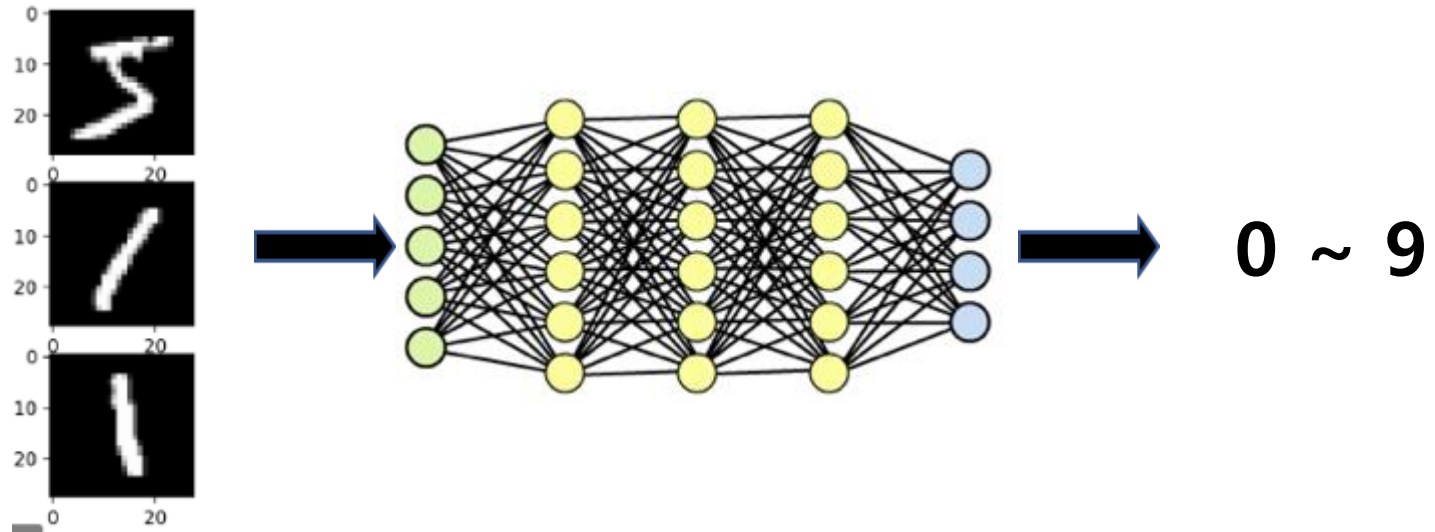
0(고양이)



1(강아지)

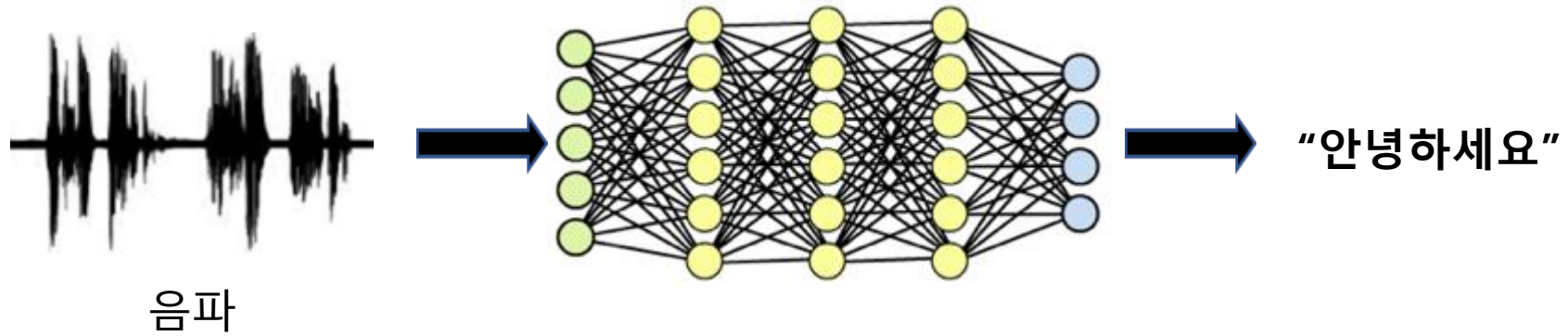
MLP

NN 구조 - 입출력



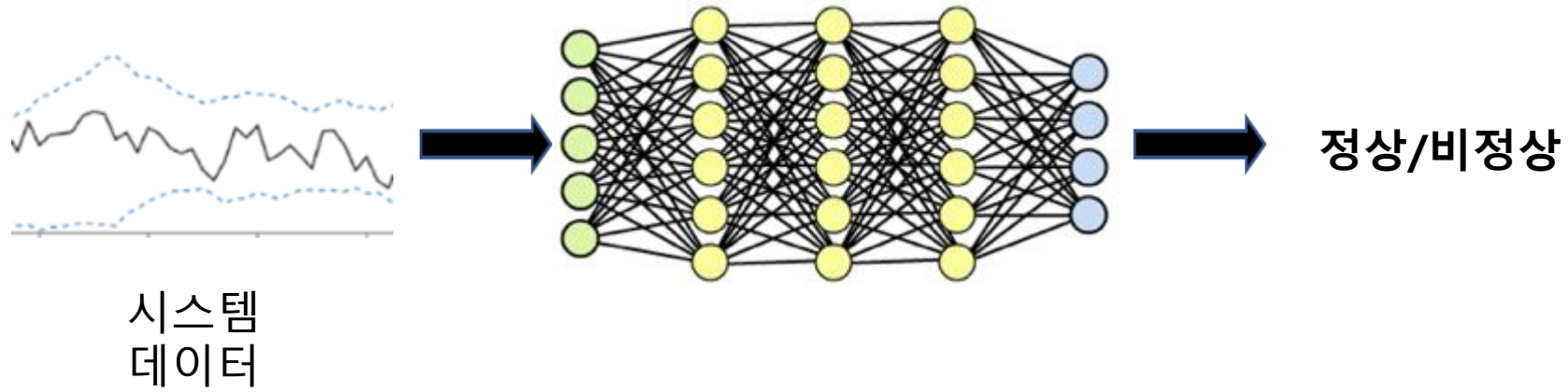
MLP

NN 구조 - 입출력



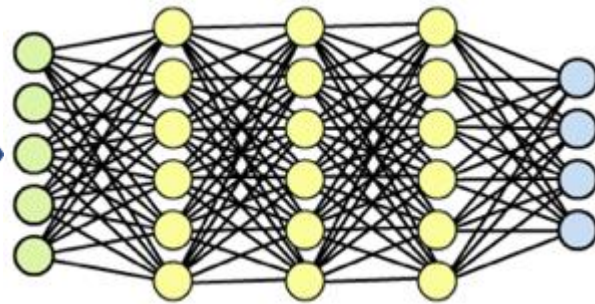
MLP

NN 구조 - 입출력



MLP

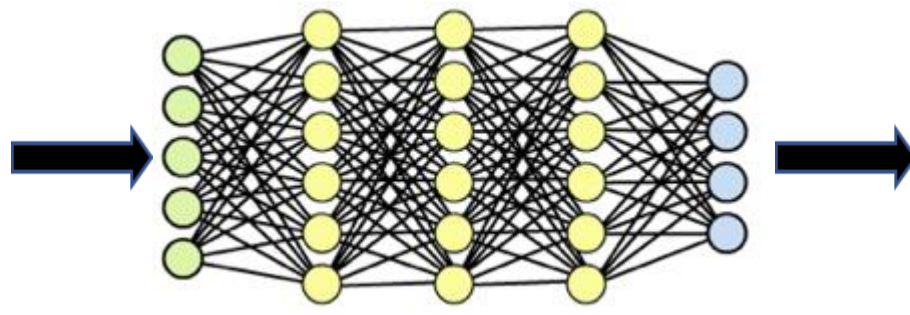
NN 구조 - 입출력



다음 수

MLP

NN 구조 - 입출력



다음 수

- 실습 - 바둑 다음 수 예측

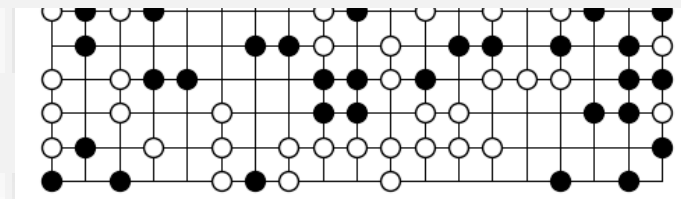
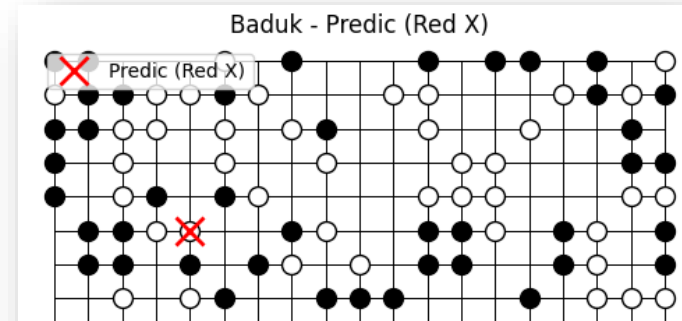
5.03.MLP_Input.Output_Baduk.ipynb

1. 바둑 데이터 (임의 데이터)

```
X = np.random.choice([-1, 0, 1], size=(1000, 19, 19, 1), p=[0.3, 0.4, 0.3]) # 백, 빈, 흑  
y = np.random.randint(0, 361, size=(1000,))  
y = to_categorical(y, num_classes=361)
```

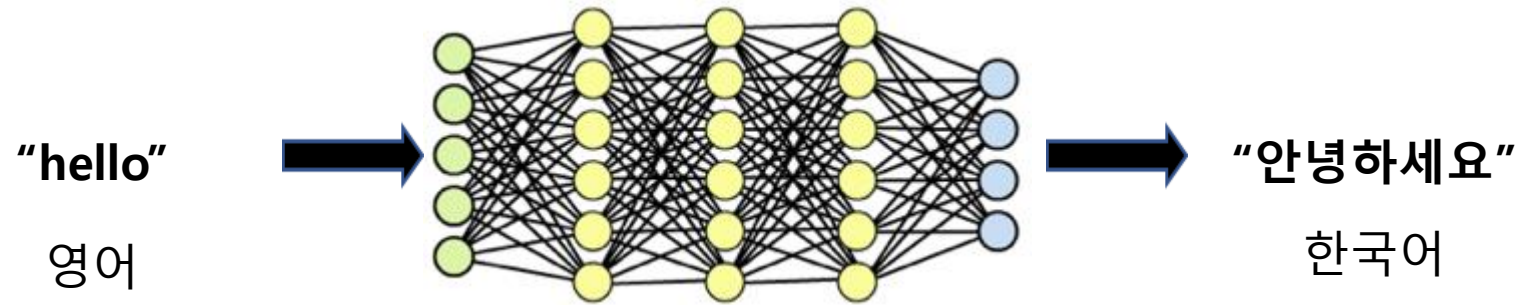
3. 학습

```
model.fit(X, y, epochs=10, batch_size=32, verbose=1)
```



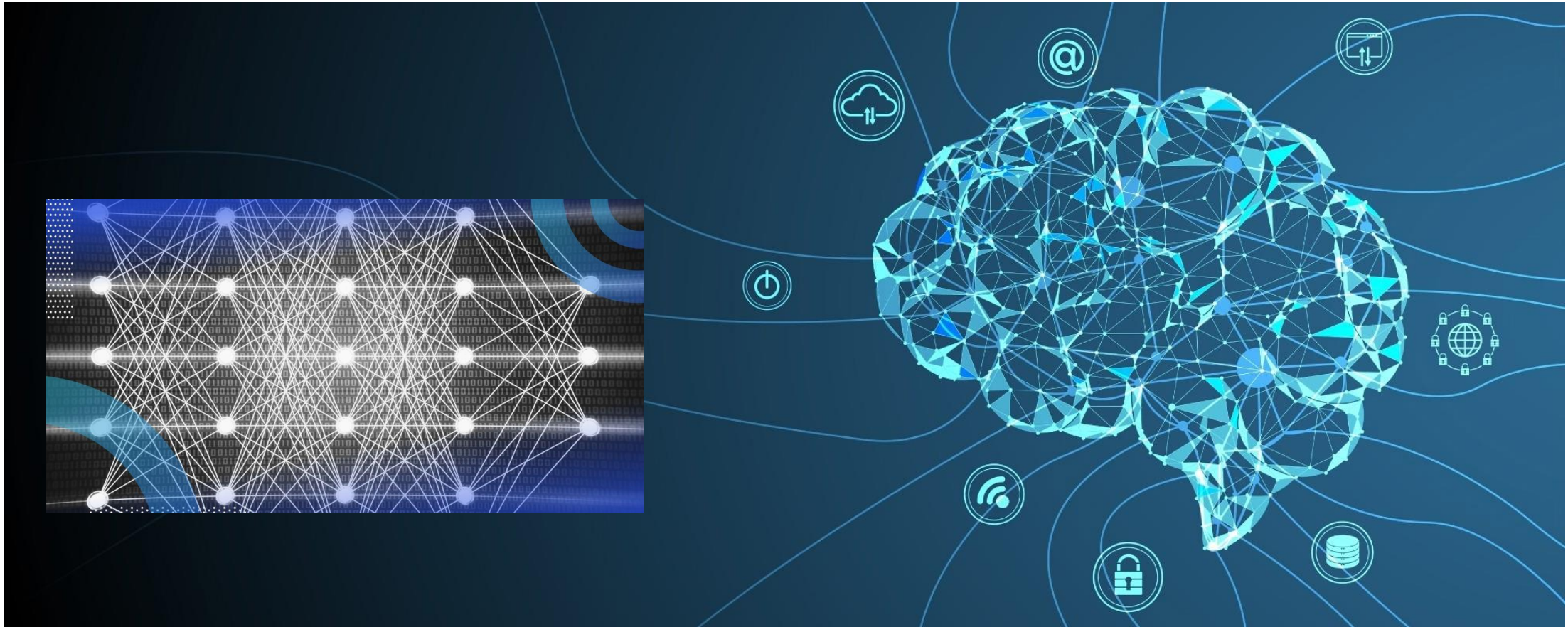
MLP

NN 구조 - 입출력



MLP

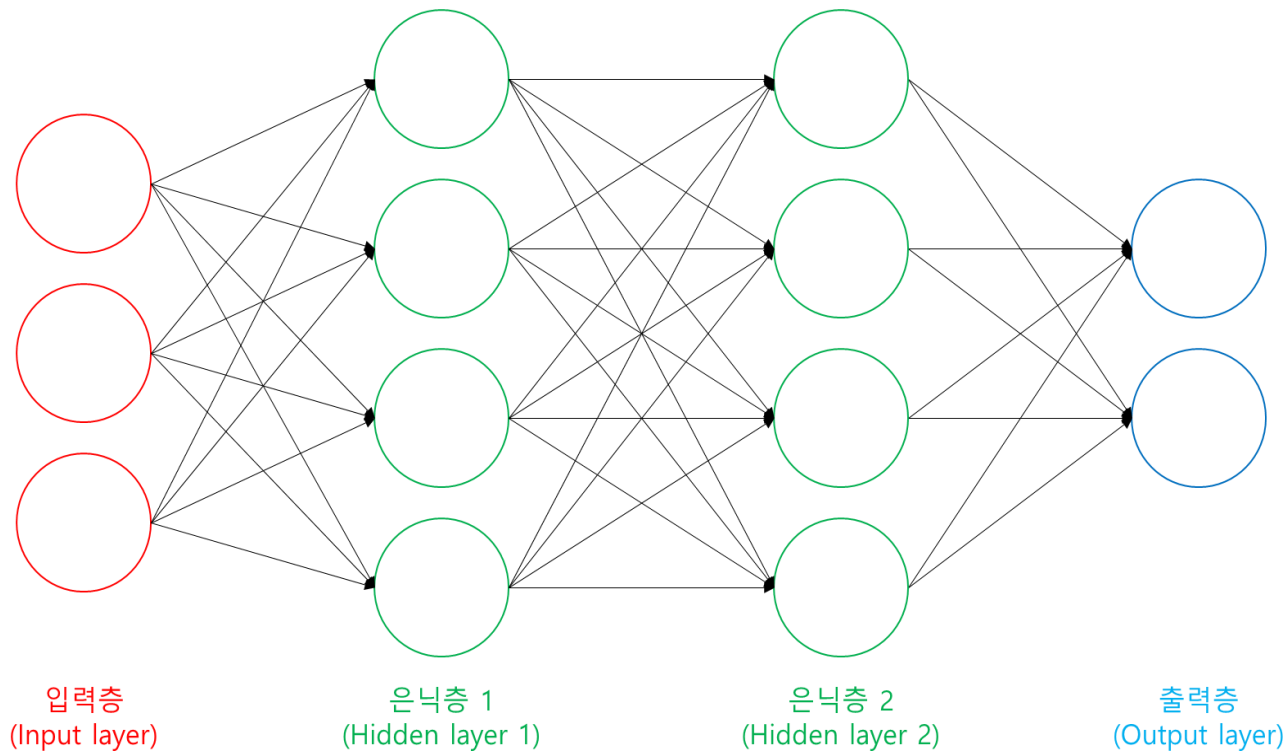
NN 구조



MLP

NN 구조

- Node
- Weight
- Bias
- Layer



MLP

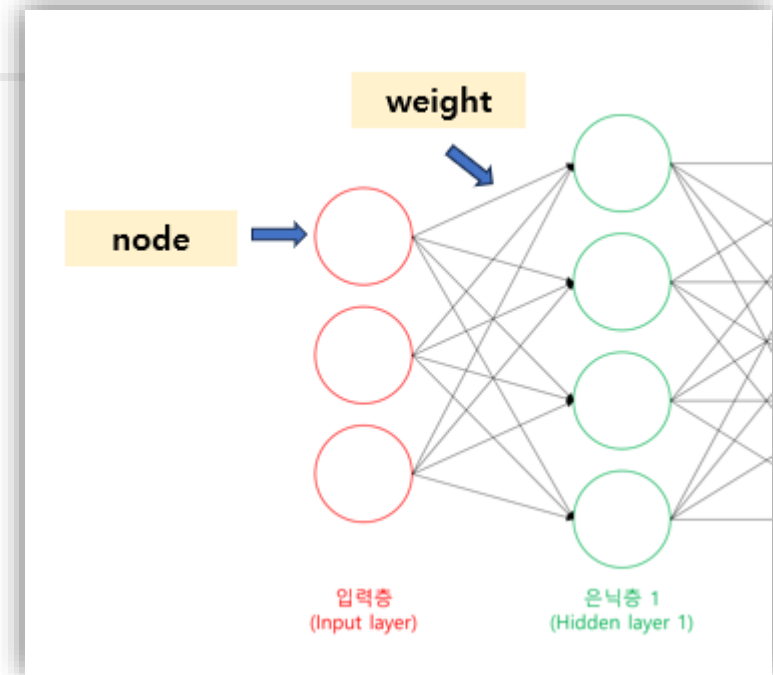
NN 구조 - node

node(뉴런)

- 정보를 입력 받아 처리 후 출력하는 신경망의 기본 계산 단위
- 각 노드는 여러 입력을 받아 하나의 출력 생성

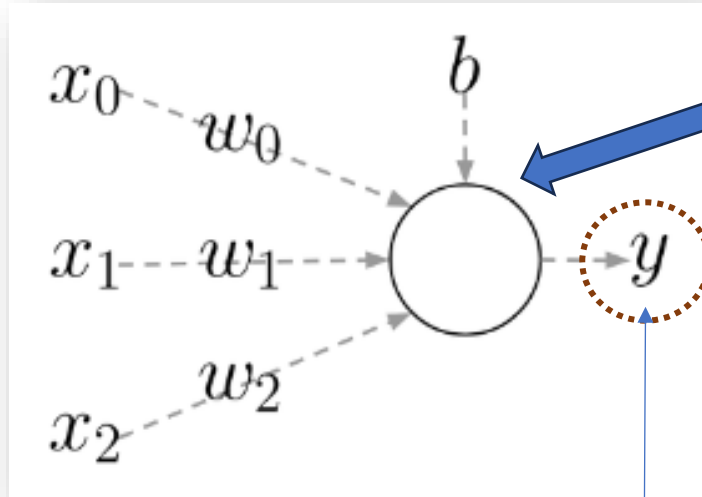
역할

- **신호 처리:** 각 노드는 입력된 신호(데이터)에 가중치를 곱하고, 그 결과를 합산하여 활성화 함수에 전달. 활성화 함수는 이 합산된 입력 신호를 기반으로 노드의 출력을 결정
- **비선형성 추가:** 활성화 함수를 통해 비선형성을 도입 -> 복잡한 패턴과 함수 학습 가능
- **정보 통합:** 신경망의 각 층에 있는 노드들은 이전 층의 출력을 입력으로 받아 처리
-> 이를 통해 정보 통합



MLP

NN 구조 - node



- 원 : 노드
- 화살표 : 신호의 흐름
- x_n : 외부에서 들어오는 정보(신호) - Input
- w_n : 신호를 연결하는 가중치(Weight)
- b : Bias
- y : 외부로 나가는 정보(신호) - Output

Q. v? y?

$$u = w_1 \times x_1 + w_2 \times x_2 + w_3 \times x_3 + b = \sum_{i=0}^3 (w_i \times x_i) + b$$

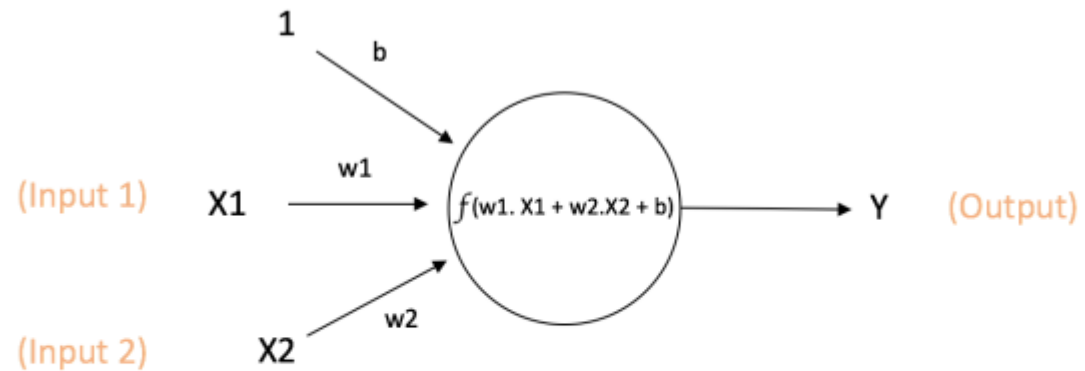
MLP

NN 구조 - node

Q. v ? y ?

-> $y = f(v)$

f : 활성화 함수



Output of neuron = $Y = f(w1.X1 + w2.X2 + b)$

MLP

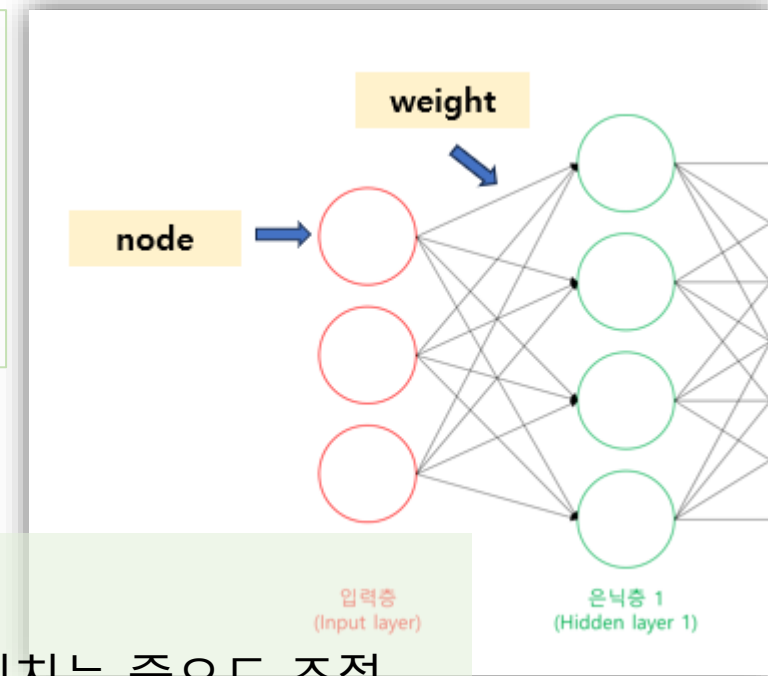
NN 구조 - weight

weight(가중치)

- 신경망의 각 연결선에 할당된 값
- 노드의 출력이 다음 layer의 노드에 얼마나 많은 영향을 미칠지 결정
-> 각 입력이 출력에 미치는 영향력

역할

- **입력 중요도 조절:** 가중치는 각 입력 신호의 중요도를 조절
중요한 특성에는 높은, 덜 중요한 특성에는 낮은 가중치 부여 -> 결과에 미치는 중요도 조절
- **학습을 통한 최적화:** 학습 과정에서 가중치 조정
비용 함수를 최소화하는 방향으로 학습(=가중치 조정) -> 데이터에 내재된 규칙, 관계 모델링
- **신호 전달**



MLP

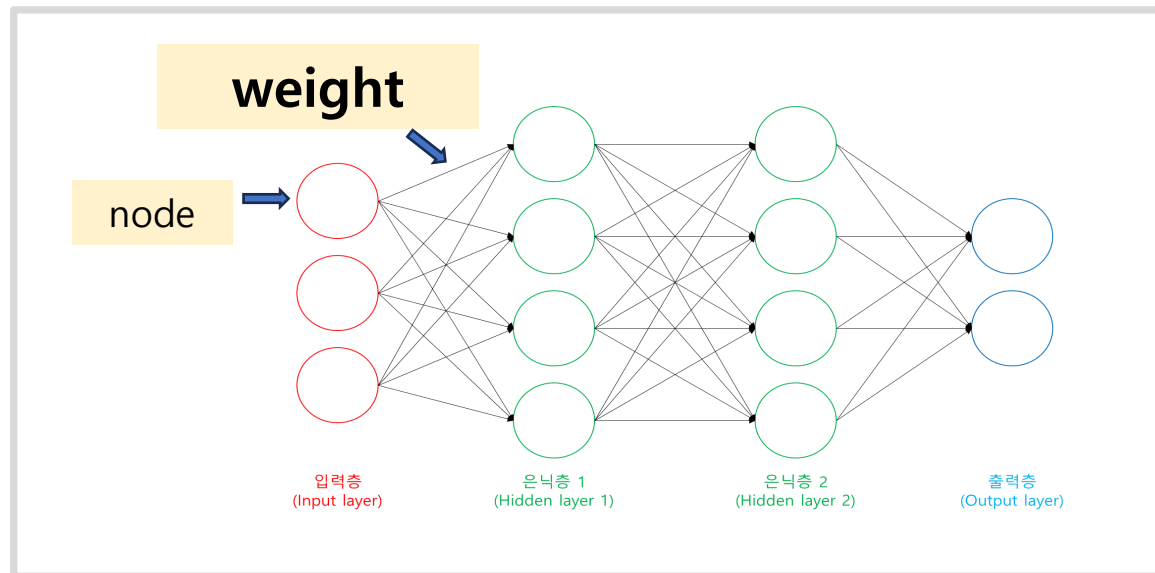
NN 구조 - weight

<https://towardsdatascience.com/whats-the-role-of-weights-and-bias-in-a-neural-network-4cf7e988a0f>

- Weights play an important role in changing the orientation or slope of the line that separates two or more classes of data points.
- Weights tell the importance of a feature in predicting the target value.
- Weights tell the relationship between a feature and a target value

분류

회귀



MLP

NN 구조 - weight

<https://towardsdatascience.com/whats-the-role-of-weights-and-bias-in-a-neural-network-4cf7e9888a0f>

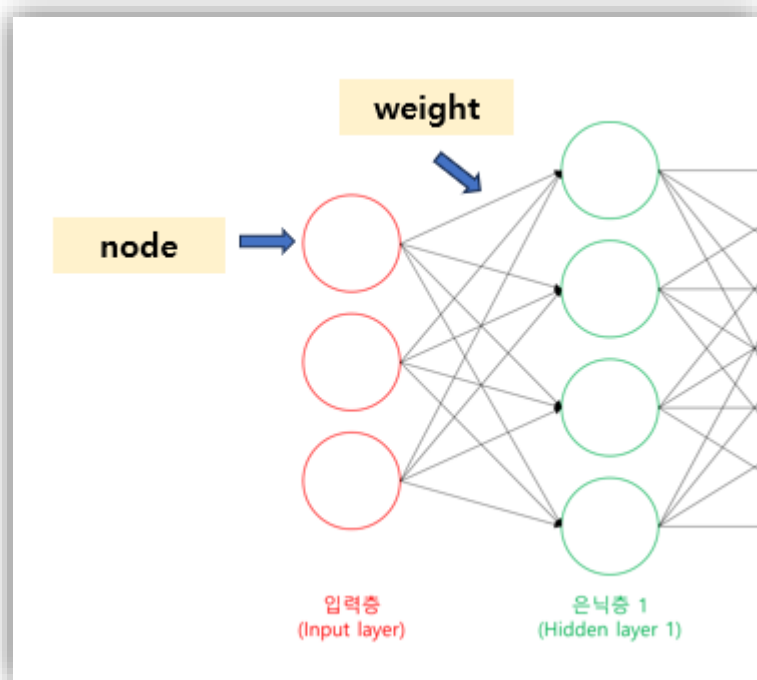
What do the weights in a Neuron convey to us?

1. Importance of the feature

Weights associated with each feature, convey the importance of that feature in predicting the output value. Features with weights that are close to zero said to have lesser importance in the prediction process compared to the features with weights having a larger value.

In the above example the w_2 is 0 that means we don't need w_2 for predicting the value of y and the value of w_1 is 1, that means we can predict the value of y using x_1 alone.

2. Tells the relationship between a particular feature in the dataset and the target value.



MLP

Q. $X?$ $y?$

NN 구조 - weight

2. Tells the relationship between a particular feature in the dataset and the target value.

let us consider an example of finding the likelihood of buying a car, with the dataset containing two input features like

1. Car price
2. Car Popularity

and let us suppose that people often tend to buy a car within their budget and the most popular one among many.

$$\text{Buying Car} \propto \frac{1}{\text{price of Car}}$$

자동차 구매 가능성은 가격의 역수에 비례

$$\text{buying car} \propto \text{Popularity of Car}$$

자동차 구매 가능성은 인기에 비례

the equation of line looks like...

$$w_1 * (\text{price of car}) + w_2 * (\text{popularity of car}) + b = 0$$

< - 특징과 (자동차 가격과 인기)
목표 변수(자동차 구매 가능성)
사이 관계 설명

MLP

NN 구조 - weight

so if the weight associated with a feature is positive it implies that there is a direct relationship between that feature and the target value, and if the weight associated with the feature is negative it implies that there is an inverse relationship between the feature and the target value.

MLP

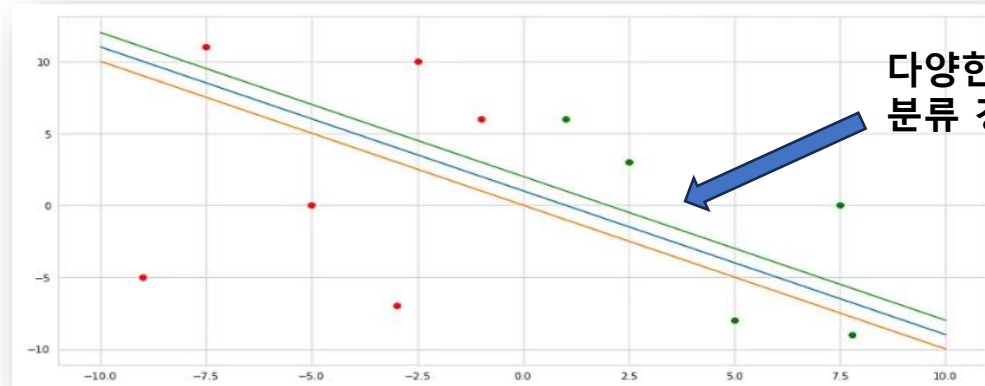
NN 구조 - weight

분류 문제

| | x1 | x2 | y |
|----|----|----|---|
| 0 | -9 | -5 | 0 |
| 1 | 1 | 6 | 1 |
| 2 | -5 | 0 | 0 |
| 3 | -3 | -7 | 0 |
| 4 | 5 | -8 | 1 |
| 5 | -7 | 11 | 0 |
| 6 | 2 | 3 | 1 |
| 7 | -2 | 10 | 0 |
| 8 | 7 | 0 | 1 |
| 9 | -7 | 9 | 0 |
| 10 | -1 | 6 | 0 |

weight 역할 : 결정 경계 학습

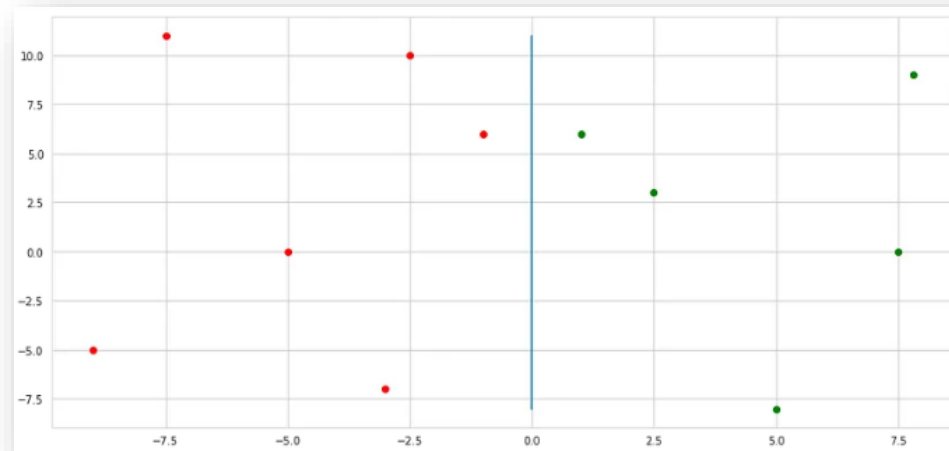
x1, x2의 값과 목표 변수(y)를 기반으로 한 선형 모델 시각화



$$y = w_1 \cdot x_1 + w_2 \cdot x_2 + b$$

Nonlinear Transformation

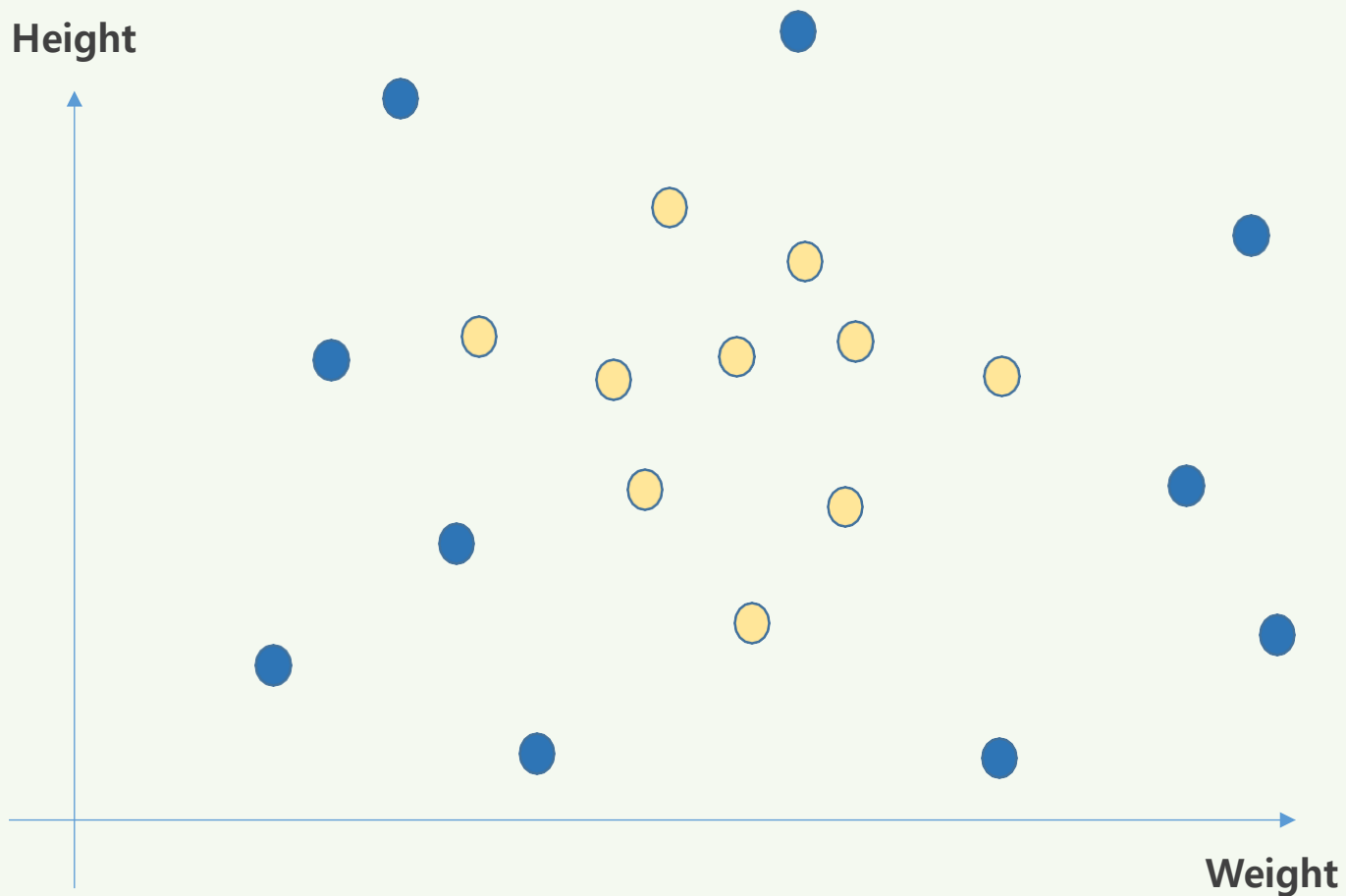
NN 모델



MLP

비선형 변환(Nonlinear Transformation)

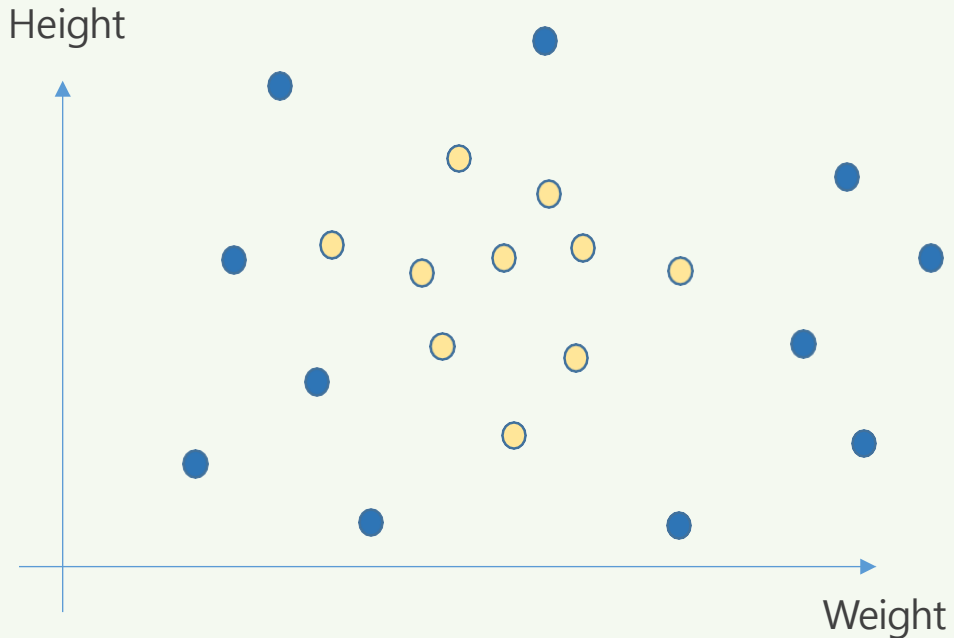
아래 분포의 데이터를 하나의 직선으로 선형 분류?



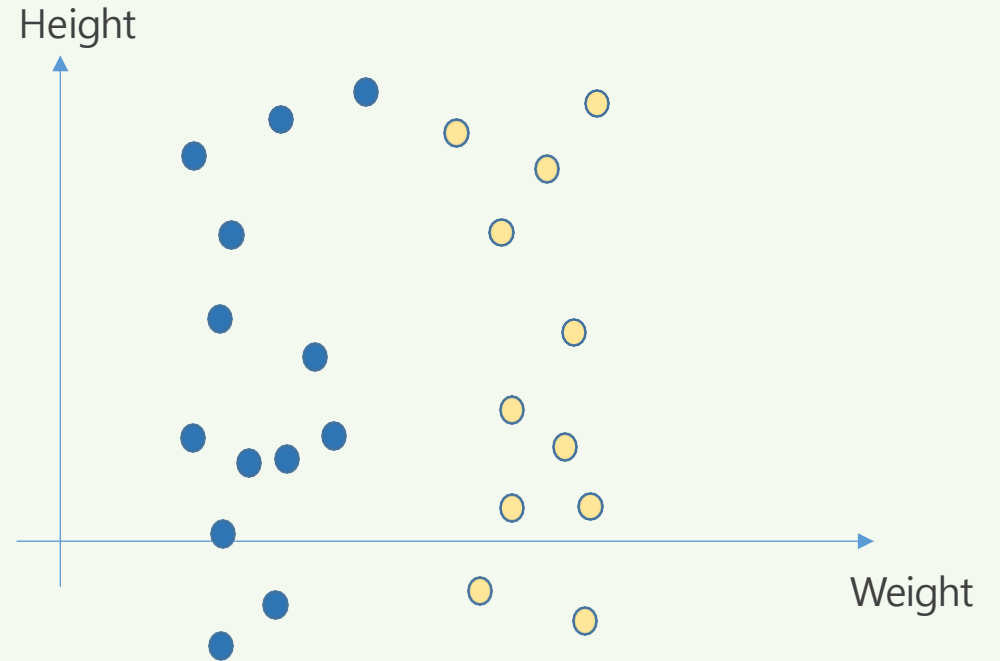
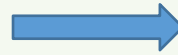
MLP

비선형 변환(Nonlinear Transformation)

: 입력 데이터에 대해 선형이 아닌 관계(예: 곱셈, 지수, 로그, 사인, 코사인 등)를 적용하는 변환
데이터의 복잡한 구조와 패턴을 보다 잘 표현



비선형 변환
데이터 포인트들 사이의 관계를 변화시켜,
특정 분석이나 분류 작업에 있어
데이터를 더욱 명확하게 구분



비선형 변환이 적용된 후의 데이터 분포
원 공간에서 선형으로 분리할 수 없는
데이터 포인트를 분리, 데이터 표현을 변경,
패턴 인식, 분류 등과 같은 복잡한 작업을
더 쉽게 수행 가능

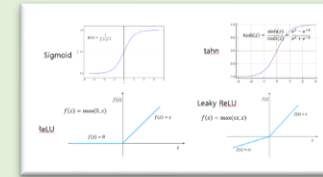
MLP

비선형 변환(Nonlinear Transformation)

비선형 변환..... 단순한 선형 관계보다 복잡한 데이터의 패턴과 관계를 모델링 가능 -> 복잡한 문제 해결

> 신경망에서의 비선형 변환은 주로 활성화 함수를 통해 이루어짐

> 신경망의 각 노드에 적용되어 입력 데이터를 비선형으로 변환



> 선형 변환만을 사용하는 모델은 입력 데이터의 선형 조합을 통해 예측 수행

but 많은 실 세계 문제들은 선형 모델로는 충분히 표현할 수 없는 복잡한 비선형 구조를 지님

ex) 이미지 인식, 자연어 처리, 음성 인식 등 (데이터 간 복잡한 관계와 패턴이 존재)

> 비선형 변환 이점

- 복잡한 데이터 구조 학습
- 특징 추출: 데이터에서 고차원적인 특징 추출 -> 문제 해결의 key
- 범용 함수 근사자: 신경망 구조의 복잡성 + 충분한 데이터 + 비선형 활성화 함수

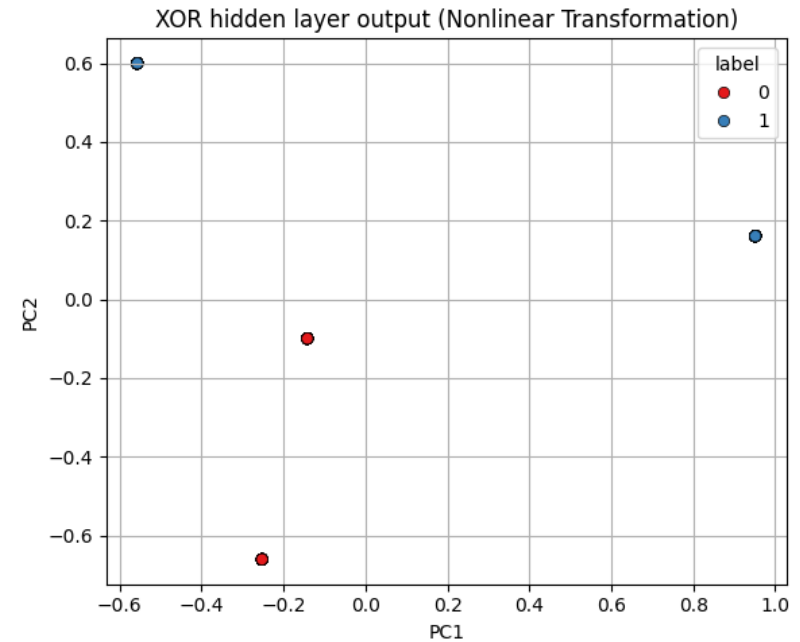
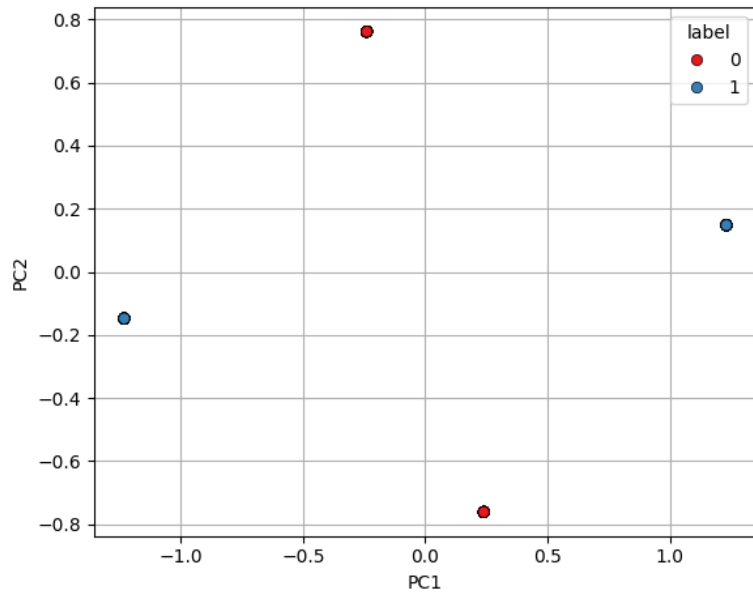
-> (이론적으로) 어떤 함수든 근사 -> 매우 다양한 유형의 문제 해결 가능

MLP

NN 구조 - weight

- 실습 - 가중치(weight) 학습에 따른 분류 경계선 최적화

5.03.MLP.NonlinearTransformation.ipynb

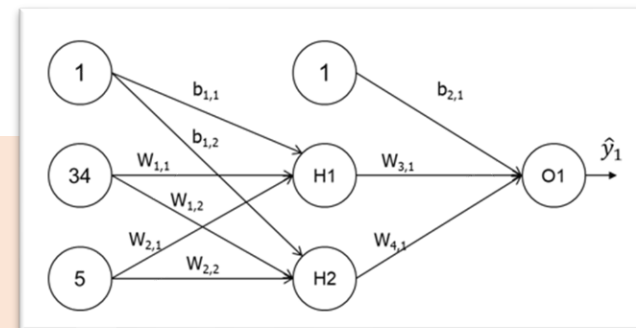


MLP

NN 구조 - weight 초기화

1. 초기화(Initialization) - Weight

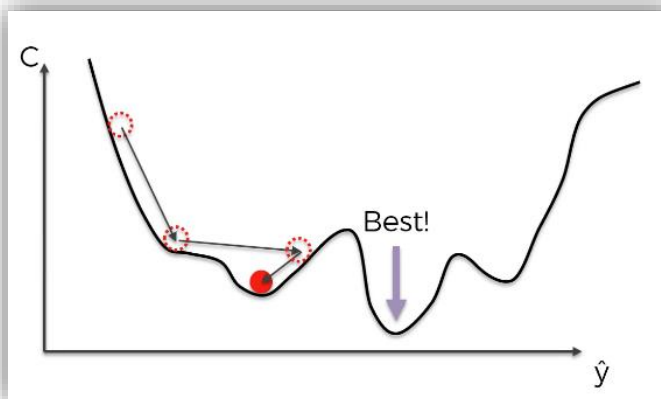
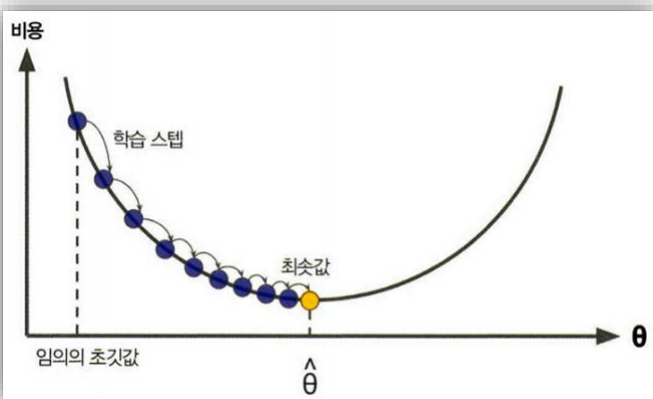
> 학습 과정을 시작하기 위한 준비 단계. (보통) 무작위 값으로 초기화.



적절한 초기화를 통해 신경망이 빠르게 수렴, **경사 하강법**이 효율적으로 작동하도록 마련

If, 부적절한 초기화일 경우

- 학습 속도가 늦어짐
- 신경망 학습 방해 (gradient vanishing/exploding),
- **지역 최솟값**(local minima) 문제 발생



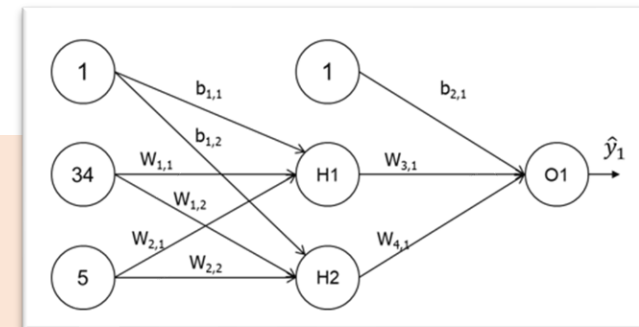
```
print("Weight of first hidden layer:", model.fcl.weight,
      [-0.0058, -0.2836, -0.0217, 0.2652],
      [ 0.3448, 0.0891, 0.1539, -0.1527],
      [ 0.2777, 0.4313, -0.0363, 0.3461],
      [-0.2031, -0.2037, 0.1142, -0.3359],
      [ 0.4055, -0.4603, 0.3420, -0.4732],
      [-0.0758, -0.3533, 0.1142, 0.0566],
      [-0.3811, 0.0252, -0.0242, -0.4098],
      [ 0.2326, 0.1702, -0.3825, 0.3353],
      [-0.3802, -0.4732, 0.0997, -0.2166],
      [-0.2036, 0.2465, -0.1772, -0.0342],
      [ 0.0769, -0.3079, -0.3603, 0.3320],
      [-0.1640, 0.0087, 0.1098, 0.0172],
      [-0.0333, 0.1915, 0.4446, 0.2531],
      [ 0.3915, 0.0048, -0.3163, -0.1294],
      [ 0.0320, 0.4789, 0.4915, 0.0685],
      [-0.1768, -0.1448, 0.3675, 0.0896],
      [ 0.4665, 0.4394, -0.4358, -0.4374],
      [ 0.0072, -0.2807, -0.4197, -0.1590],
      [ 0.4765, 0.4422, -0.4353, 0.1949])
```

MLP

NN 구조 – weight 초기화

1. 초기화(Initialization) - Weight

> 학습 과정을 시작하기 위한 준비 단계 . (보통)무작위 값으로 초기화.



• 효율적 가중치 초기화의 목적

1. Breaking Symmetry(대칭성):

if, 모든 가중치가 동일 값으로 초기화 -> 모든 뉴런이 동일하게 학습되어 대칭적인 가중치로 세팅

2. 기울기 소실/폭발 문제(Vanishing/Exploding Gradient):

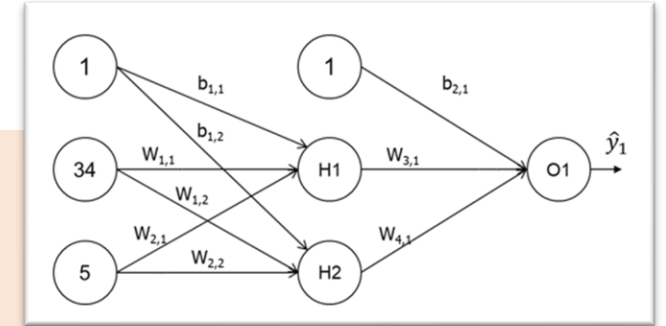
가중치를 너무 크게 또는 너무 작게 초기화하면, 기울기가 소실되거나 폭발하는 문제가 발생

MLP

NN 구조 – weight 초기화

1. 초기화(Initialization) - Weight

> 학습 과정을 시작하기 위한 준비 단계 . (보통)무작위 값으로 초기화.



• 초기화 방법

1. 무작위 초기화(Random Initialization):

- 가중치를 작은 난수로 설정하여 각 뉴런이 다른 값을 가지도록 함.
- ex, 작은 값에서 균등 분포(uniform distribution)나 정규 분포(normal distribution)로 무작위 초기화

2. Xavier 초기화(Glorot Initialization):

- 선형 및 비선형 활성화 함수(ReLU, Sigmoid 등)에 사용
- 입력값과 출력값의 분산을 비슷하게 유지하여 기울기 소실 문제를 완화

3. He 초기화(He Initialization):

- ReLU 계열의 활성화 함수에 적합한 초기화 방법.
- Xavier 초기화와 유사, 활성화 함수가 비선형인 경우(예: ReLU) 더 큰 분산을 사용.

4. 제로 초기화(Zero Initialization):

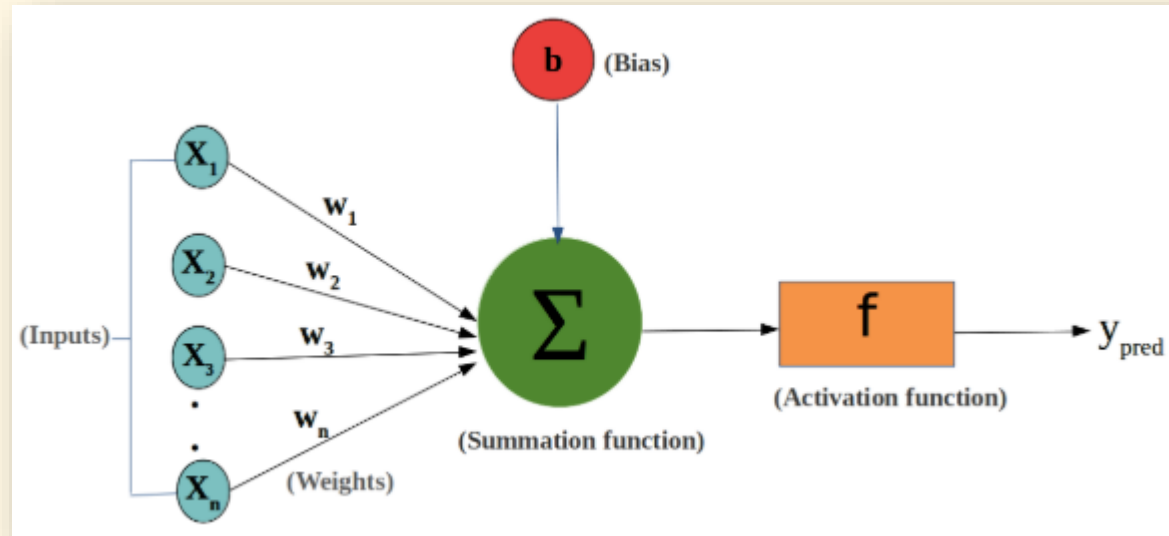
- 가중치를 모두 0으로 초기화

MLP

NN 구조 - bias

bias node (편향 노드) = 선형 회귀 모델에서의 절편(intercept)

- 모델 학습에 중요한 파라미터
- 각 node의 출력에 추가되는 값
- 입력 데이터에 독립적으로 node의 활성화 조절
- 신경망이 다양한 문제에 더 잘 일반화할 수 있도록 도움

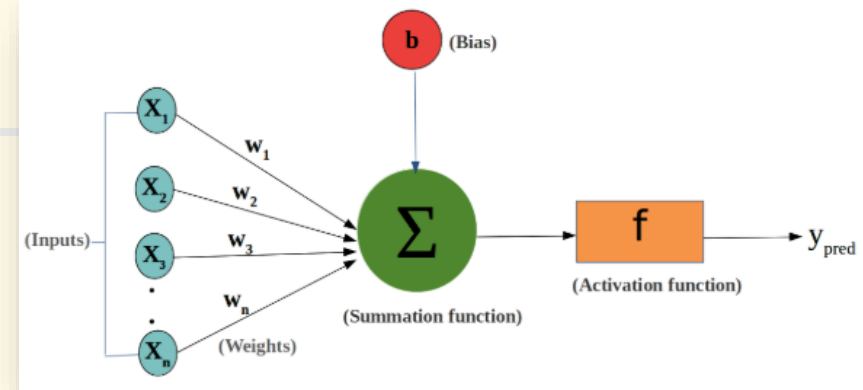


$$Z = w_1 X_1 + w_2 X_2 + \dots + w_n X_n + b$$

MLP

NN 구조 - bias

bias node (편향 노드) - 역할



$$Z = W_1 X_1 + W_2 X_2 + \dots + W_n X_n + \mathbf{b}$$

- 활성화 임계값 조정:

[입력 신호의 총합 + **bias 값**] -> 활성화 함수에 전달
활성화 함수가 결정하는 임계값 이상일 때 노드 활성화
=> bias 값이 노드의 활성화 조절

- 모델의 유연성 증가:

네트워크는 입력이 없을 때(즉, 모든 입력이 0일 때)에도 **bias 값**으로 인해 출력 가능
=> 모델이 더 유연하게 데이터의 특성을 학습하고, 더 복잡한 함수를 모델링할 수 있도록 함

MLP

NN 구조 - bias

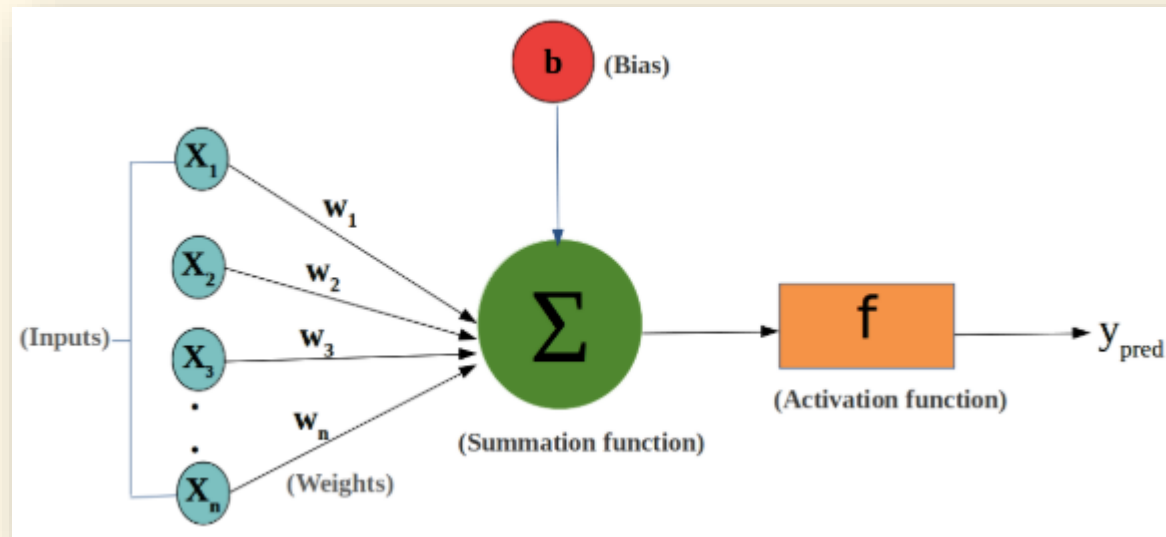
bias node (편향 노드) = 선형 회귀 모델에서의 절편(intercept)

- 은닉층에 존재
- 선형 회귀 모형의 intercept 와 비슷한 역할
- bias node에서 출력되는 값 $\Rightarrow 1$

$$Z = W_1 X_1 + W_2 X_2 + \dots + W_n X_n + b$$

출력 = $f(\sum(\text{입력} \times \text{가중치}) + \text{편향})$

편향을 통해 모델은 입력 데이터가 없거나 0에 가까울 때도 활성화
-> 모델의 학습 능력과 일반화 능력 향상

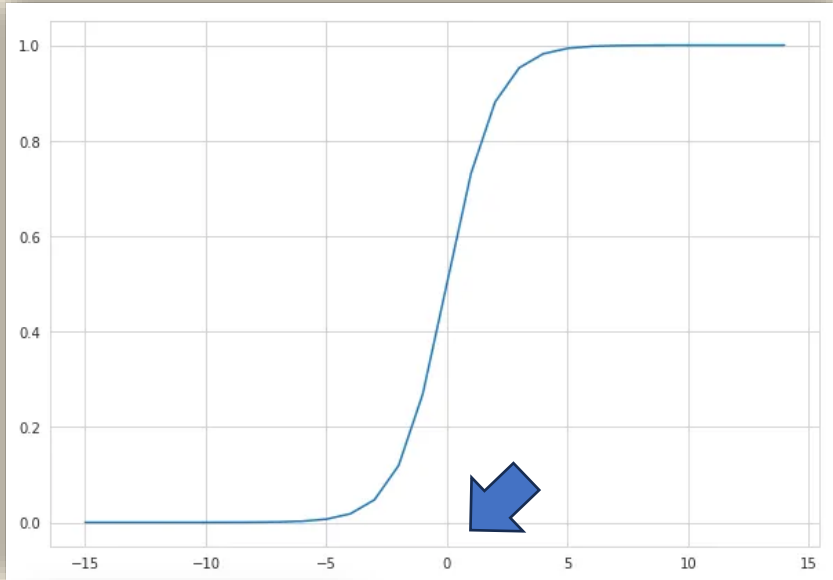


MLP

NN 구조 - bias

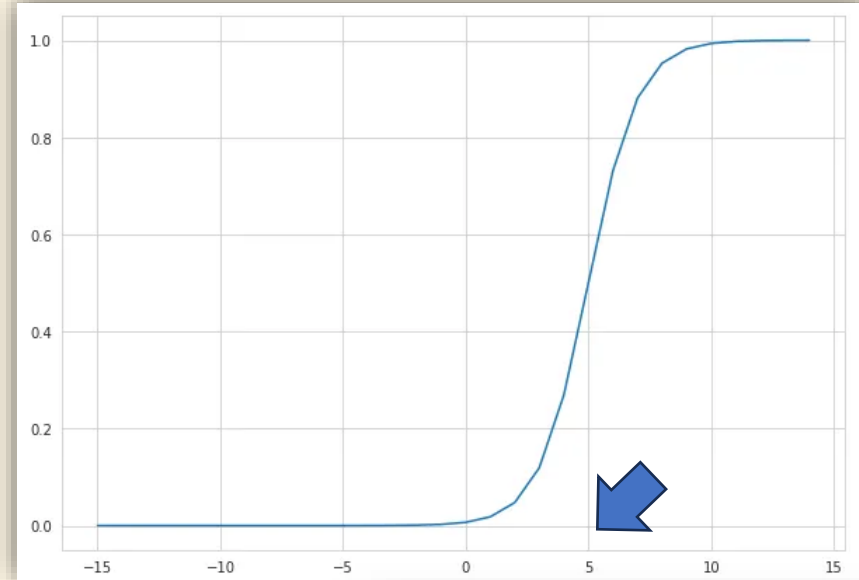
bias node (편향 노드) - 역할

- 활성화 임계값 조정:



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

what if we want y value to be 0 when $x < 5$?



$$y = \frac{1}{1 + e^{-(x-5)}} = \sigma(x - 5)$$

MLP

NN 구조 – bias

Why do we add bias?

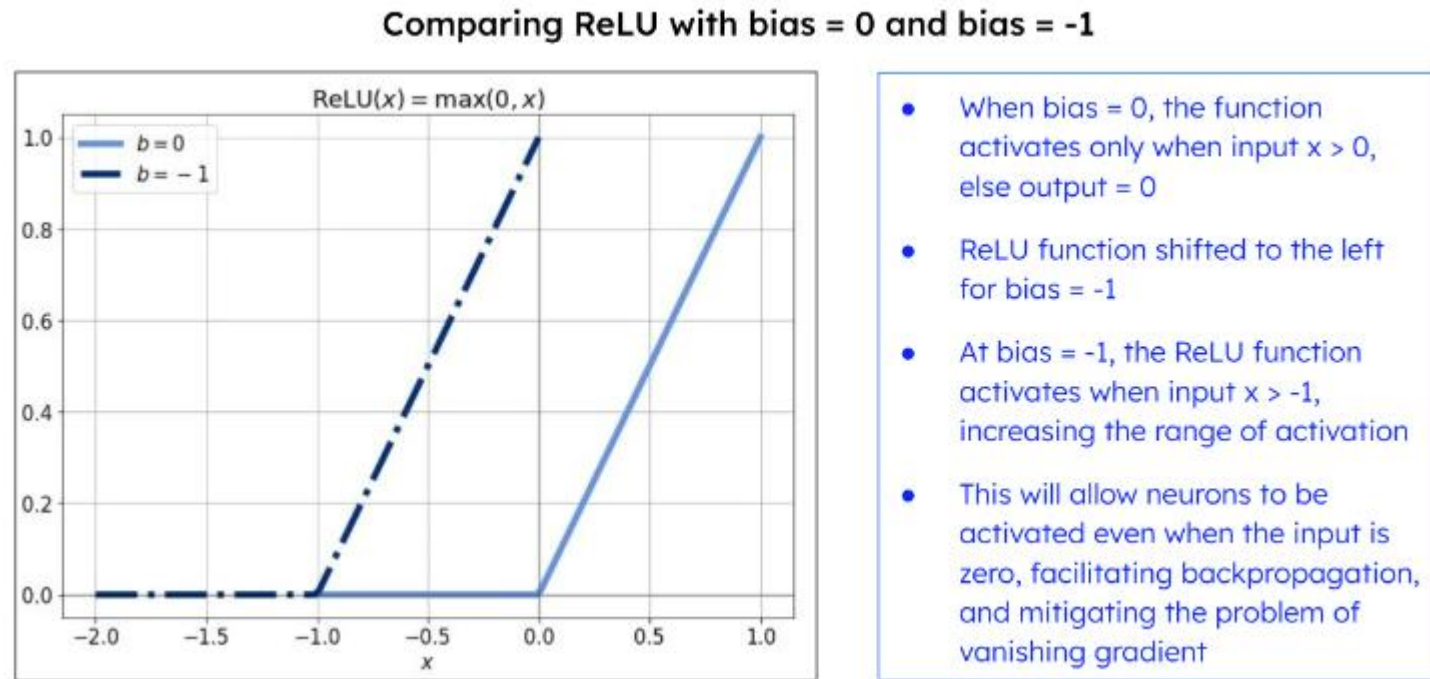
- Assists in achieving a better data fit and learning complex patterns
- Handling zero inputs and mitigating the **problem of vanishing gradient**

MLP

NN 구조 – bias

Why do we add bias?

- Handling zero inputs and mitigating the **problem of vanishing gradient**



$$\text{ReLU}(x+1) = \max(0, x+1)$$

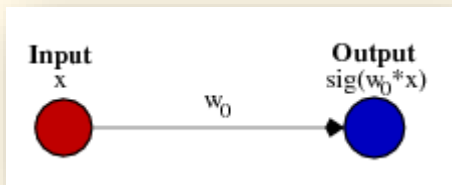
MLP

NN 구조 – bias

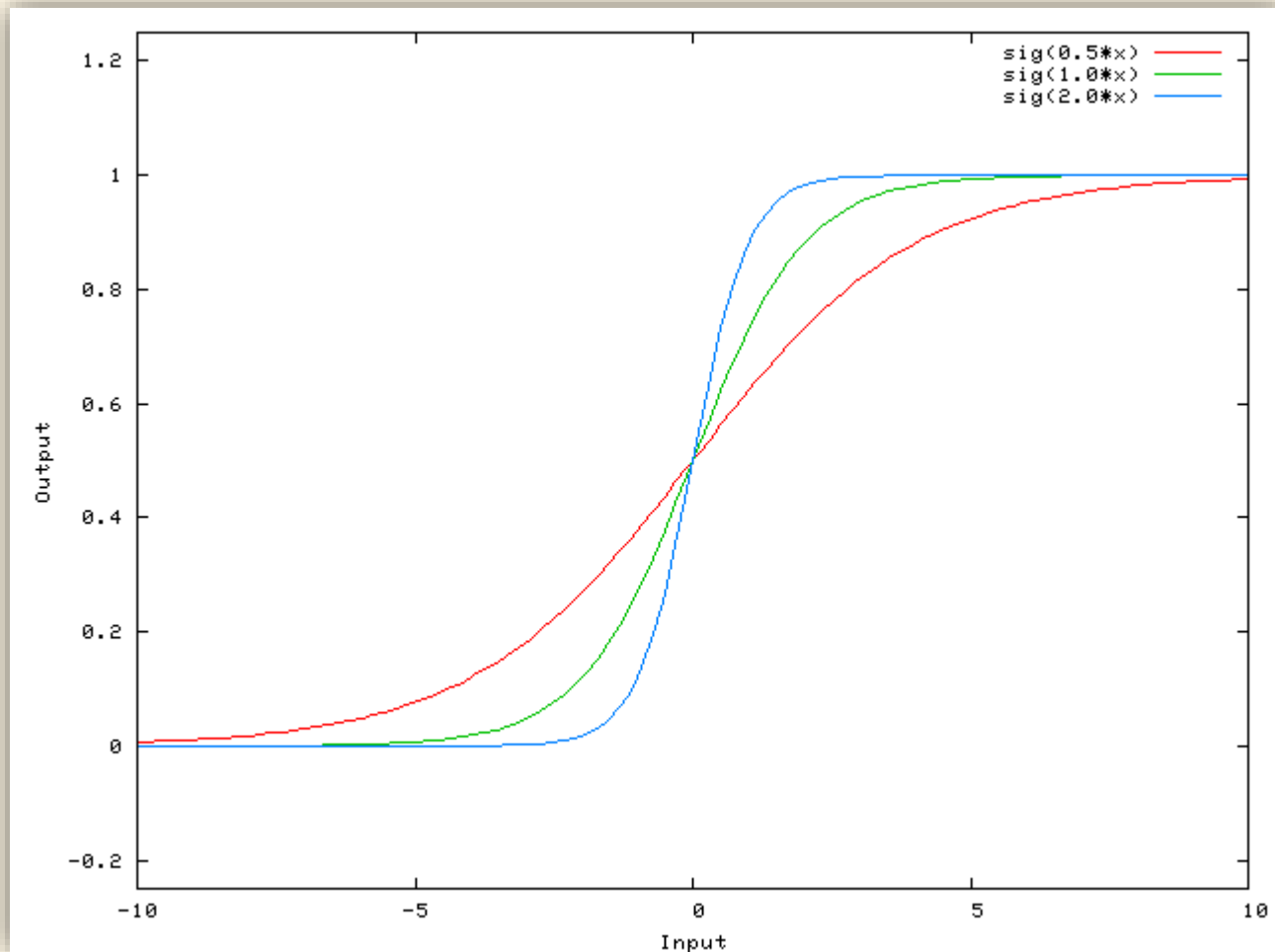
<https://stackoverflow.com/questions/2480650/what-is-the-role-of-the-bias-in-neural-networks>

- 1-input, 1-output network that has no bias

서로 다른 weight 값을 가진
sigmoid 함수 (3개)



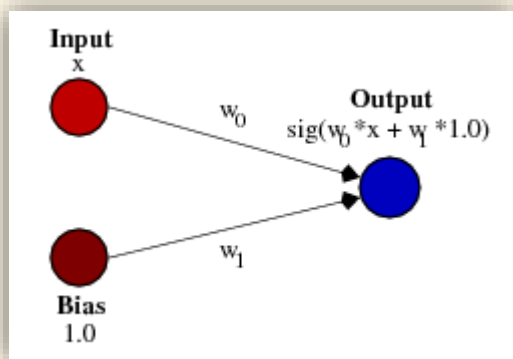
what if you wanted
the network to
output 0 when x is 2?



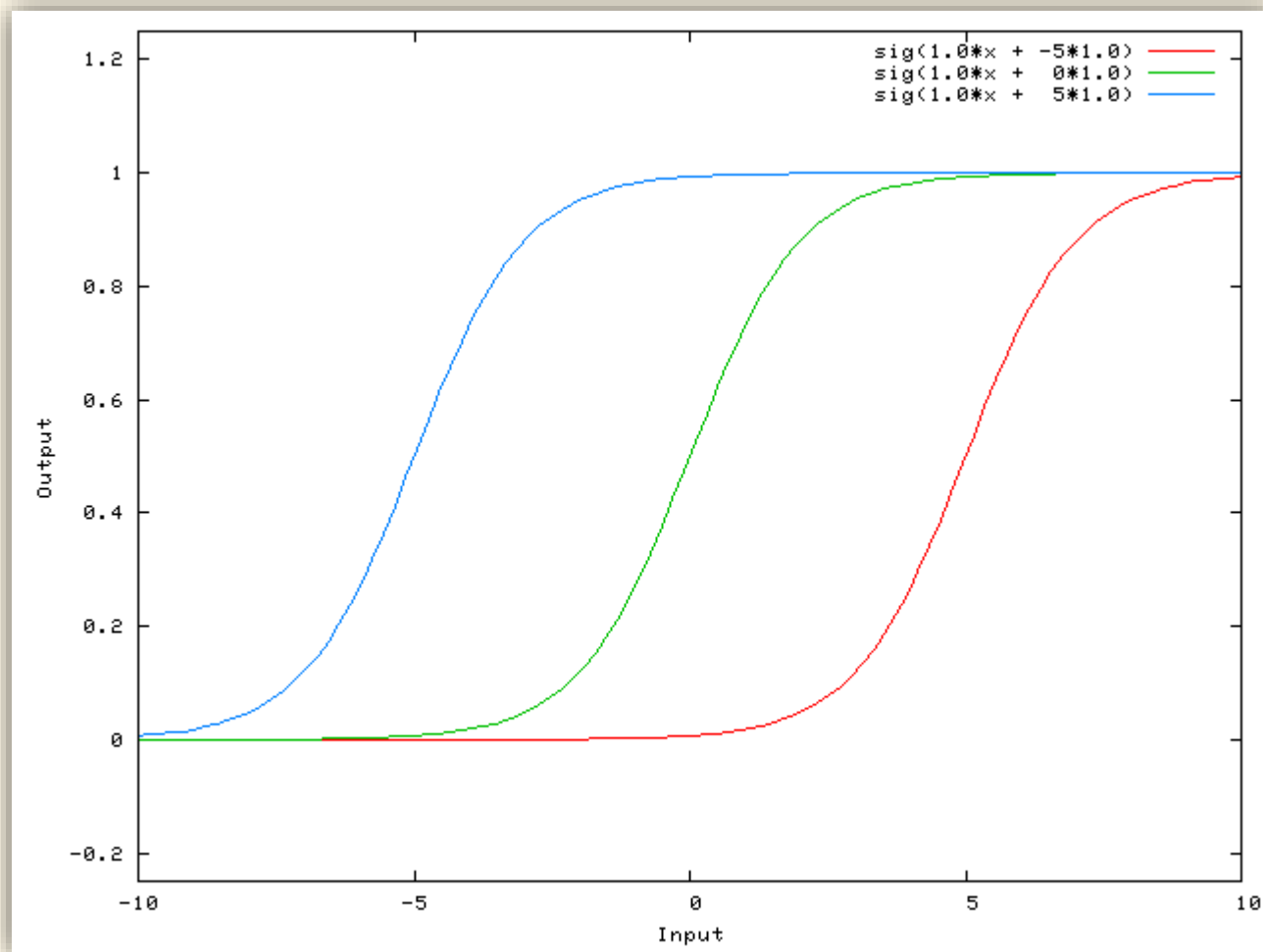
MLP

NN 구조 - bias

- 1-input, 1-output network that has bias



네트워크에 바이어스를 추가

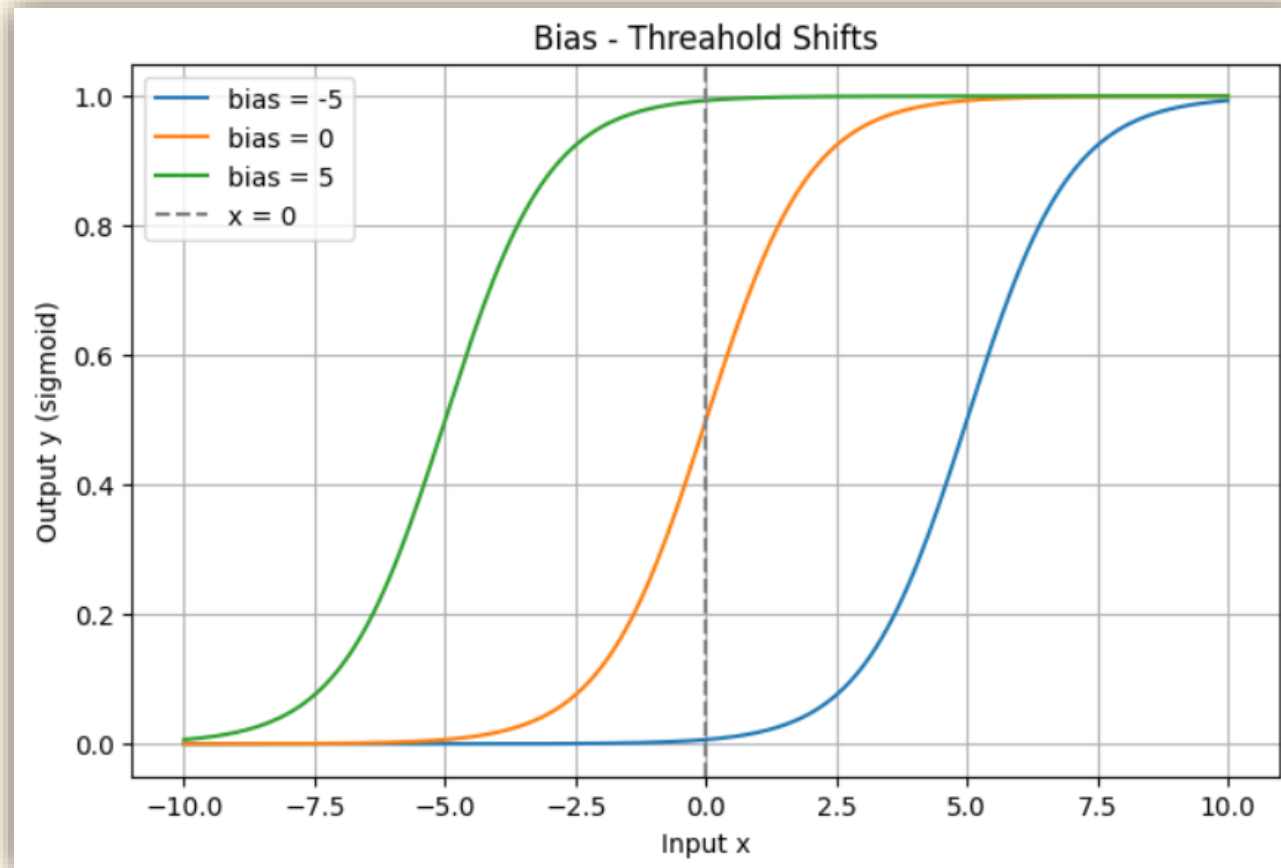


MLP

NN 구조 - bias

- 실습

5.03.Bias.Threahold_Shifts.ipynb



THANK YOU