서울시립대학교 토목공학과

파이썬의 기본 문법부터 딥러닝 맛보기까지

박상수 박사과정

2020년 8월 28일

목차

- 신경망 기초, 퍼셉트론, 다층 퍼셉트론
- 다층 퍼셉트론 예제 (붓꽃/와인 데이터)

신경망 기초

신경망은 무엇이고, **무엇이** 좋을 것일까 **?**

신경망이 적용된 다양한 어플리케이션

- **비전, 자연어 처리 등 다양한 분야에서 뛰어난 성능을 보여주는 신경망**
  - **Style Transfer**: Generative Adversarial Network (GAN) 신경망 사용, 이미지 생성
  - **Super Resolution**: 품질이 좋지 못한 이미지의 품질을 개선하기 위해 신경망 사용

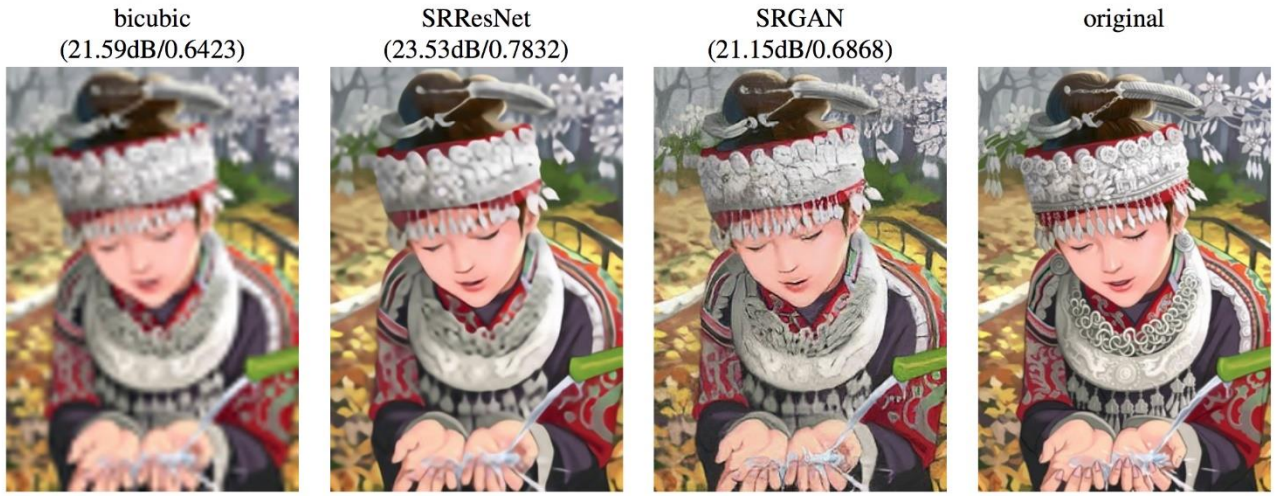
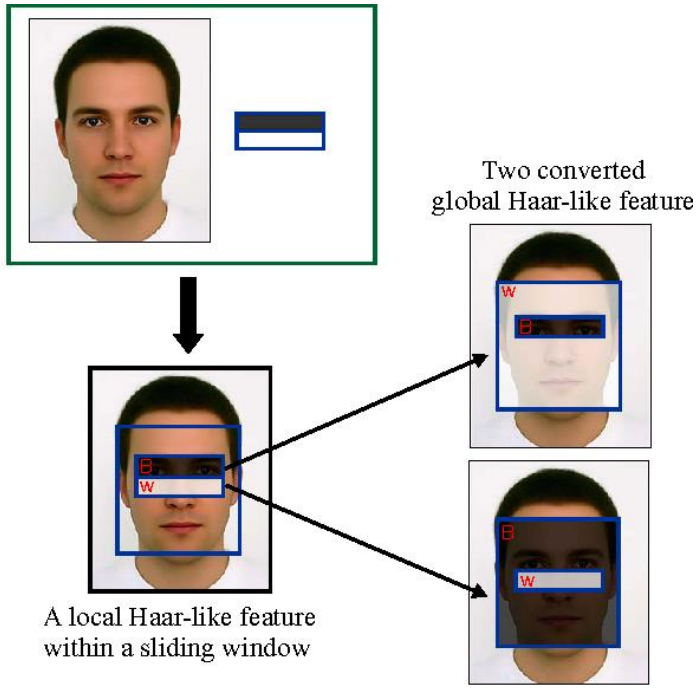


Figure 2: From left to right: bicubic interpolation, deep residual network optimized for MSE, deep residual generative adversarial network optimized for a loss more sensitive to human perception, original HR image. Corresponding PSNR and SSIM are shown in brackets. [4× upscaling]

신경망이 개발되기 이전의 연구

- **어플리케이션에 발생되는 패턴을 조건문을 사용하여 처리**
  - **Harr like feature:** 눈과 코를 검은색, 흰색으로 구분하여 처리
  - 발생하는 모든 경우에 대응하려고 하지만, 예외 상황은 존재

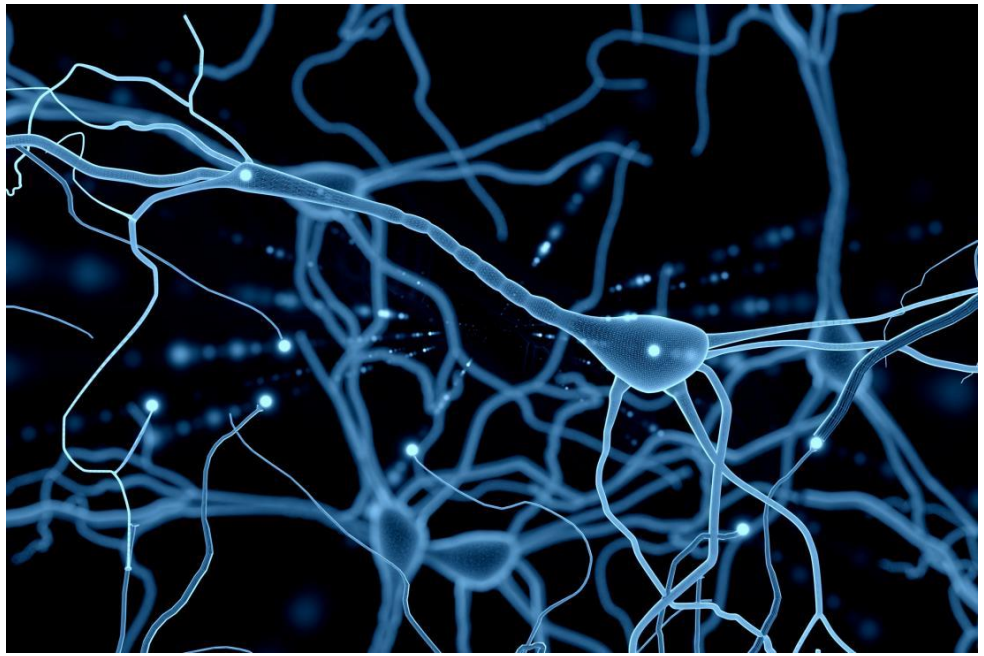


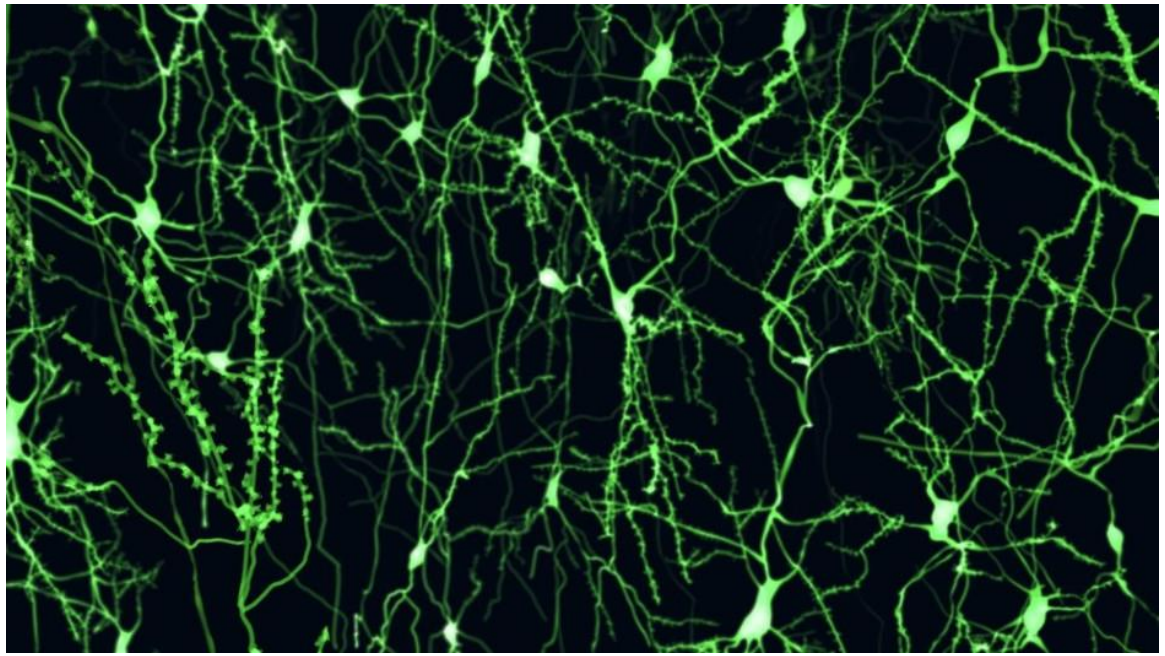

```
if () then {
else if () then {
} else if () then {
} else if () then {} else if () then {}
……
```

신경망 (Neuron system)

사람의 신경 시스템

- **신경세포: 전기적인 방법으로 신호를 전달하는 세포**
- 인접한 신경세모와 시냅스라는 구조를 통해 신호를 주고 받음

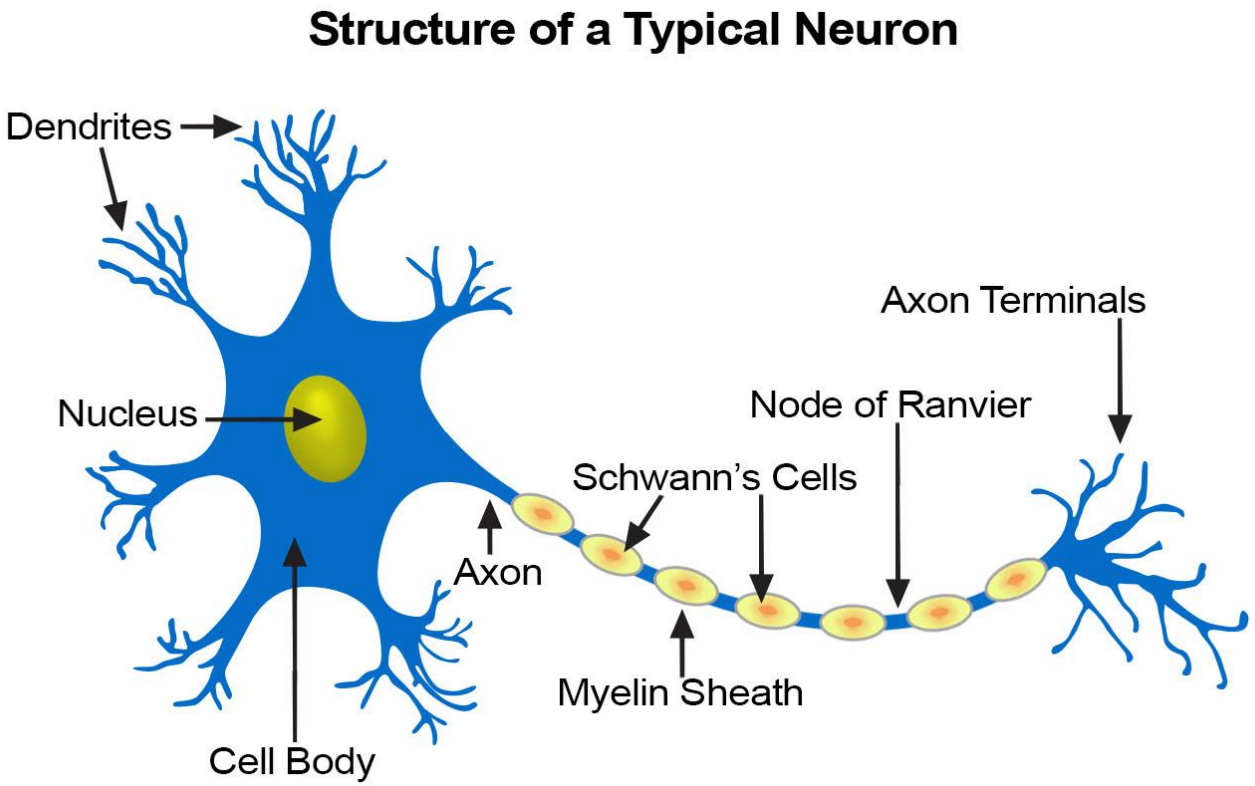




신경망 (Neuron system)

신경세포 (뉴런)

- **두뇌의 가장 작은 정보 처리 단위**
  - **세포체 (Cell body): 간단한 연산, 수상돌기 (Dendrite): 신호수신, 축삭 (Axon): 처리 결과를 전송**
  - 사람은 **10^{11}개의 뉴런**을 가지며, 각 뉴런은 10^{3}개 가량 다른 뉴런과 연결



최초의 인공 신경망 (Artificial Neuron)

맥클럭-피치 모델

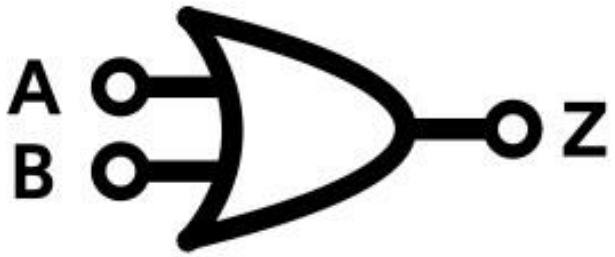
뉴런 세포를 논리적인 값으로 모델링 (1943년)

- 뉴런은 **활성화**되거나 **활성화되지 않는** 2가지 상태 존재
- 뉴런이 흥분된 상태일때, **2개 이상의 고정된 수의 시냅스 활성화**

AND gate

A	B	Z
0	0	0
0	1	0
1	0	0
1	1	1

OR gate



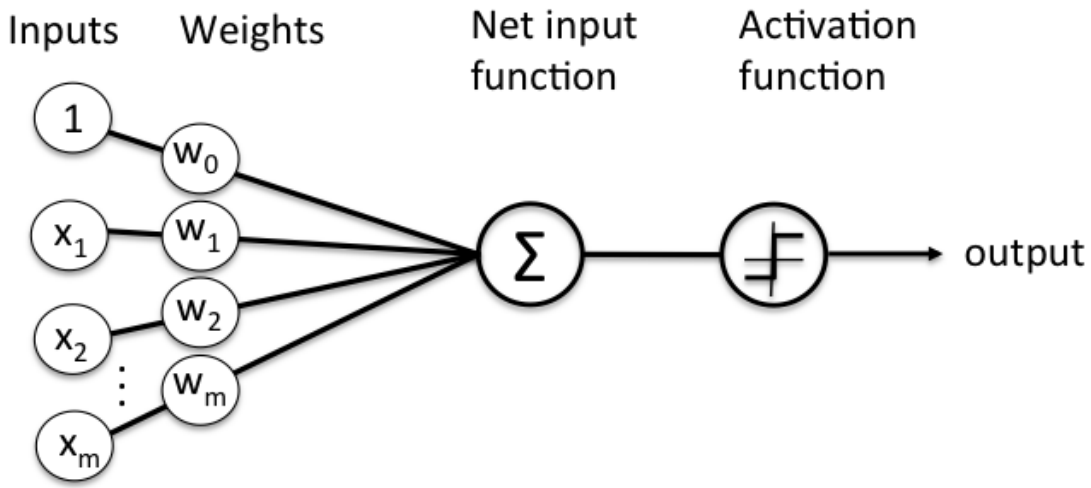
A	B	Z
0	0	0
0	1	1
1	0	1
1	1	1

퍼셉트론/다층 퍼셉트론

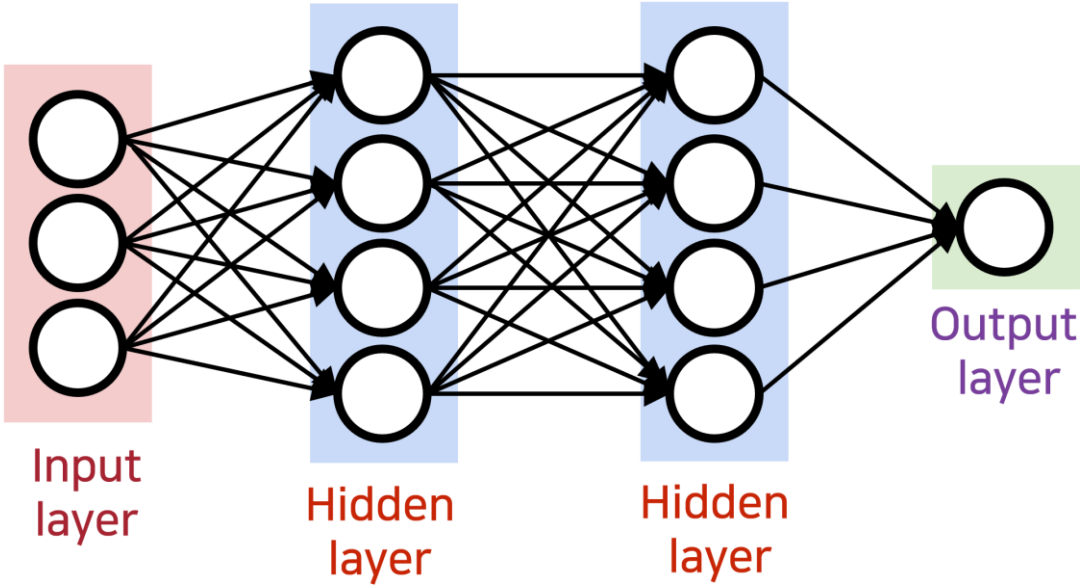
퍼셉트론 (Perceptron)

선형분류기와 동일한 가능을 하는 퍼셉트론 (1958년)

- Input과 Weight의 곱을 모두 더한 뒤, **활성 함수**를 통해 최종 결과를 생성
- **다층 퍼셉트론**은 **퍼셉트론**을 여러 개 연결하고 쌓은 구조



〈퍼셉트론〉

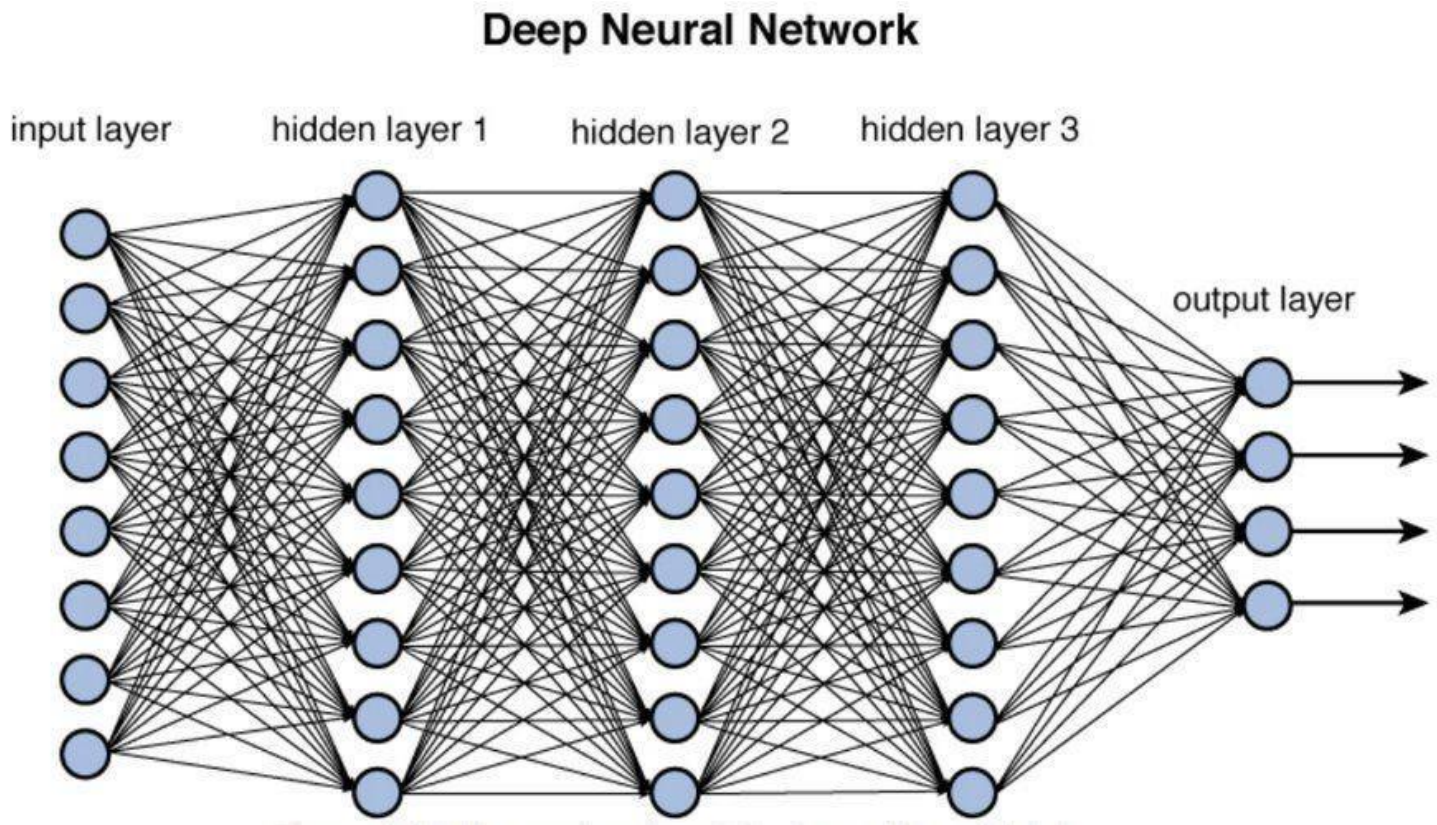


〈다층 퍼셉트론〉

딥러닝

딥러닝 (Deep Neural Network)

- **입력과 출력 사이에 잇는 인공 뉴런들을 여러 개 층층이 쌓고 연결한 인공 신경망**
- 학습 알고리즘과 컴퓨터의 성능이 향상되어 구현 가능

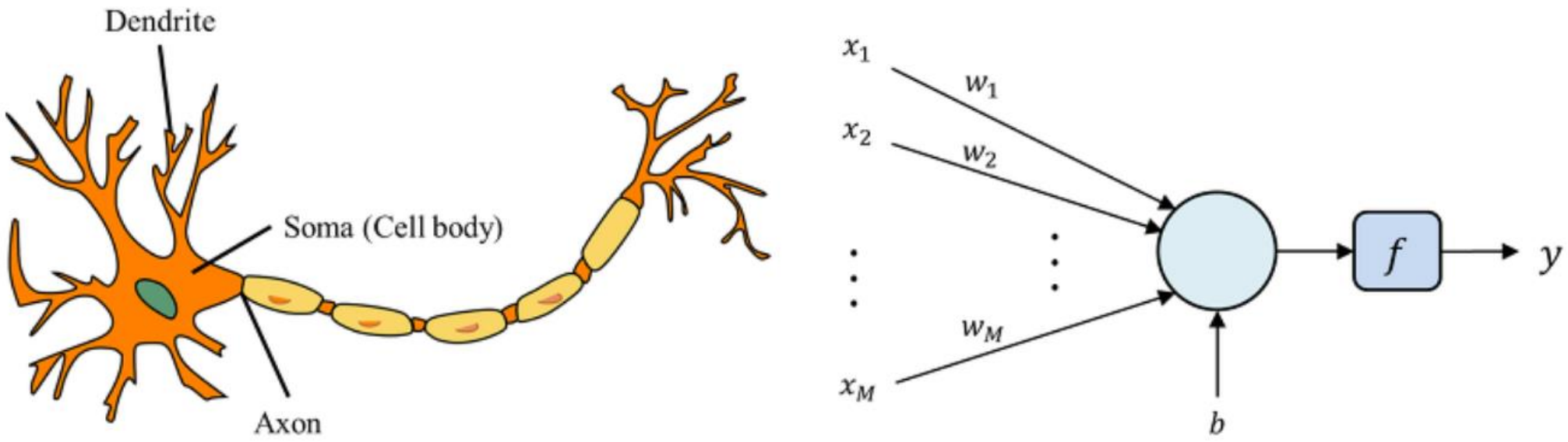


퍼셉트론

Percept: 지각된 것; 지각[인지]의 대상;

퍼셉트론

- 인공신경망 시스템은 사람의 신경계 시스템을 모사하여 설계

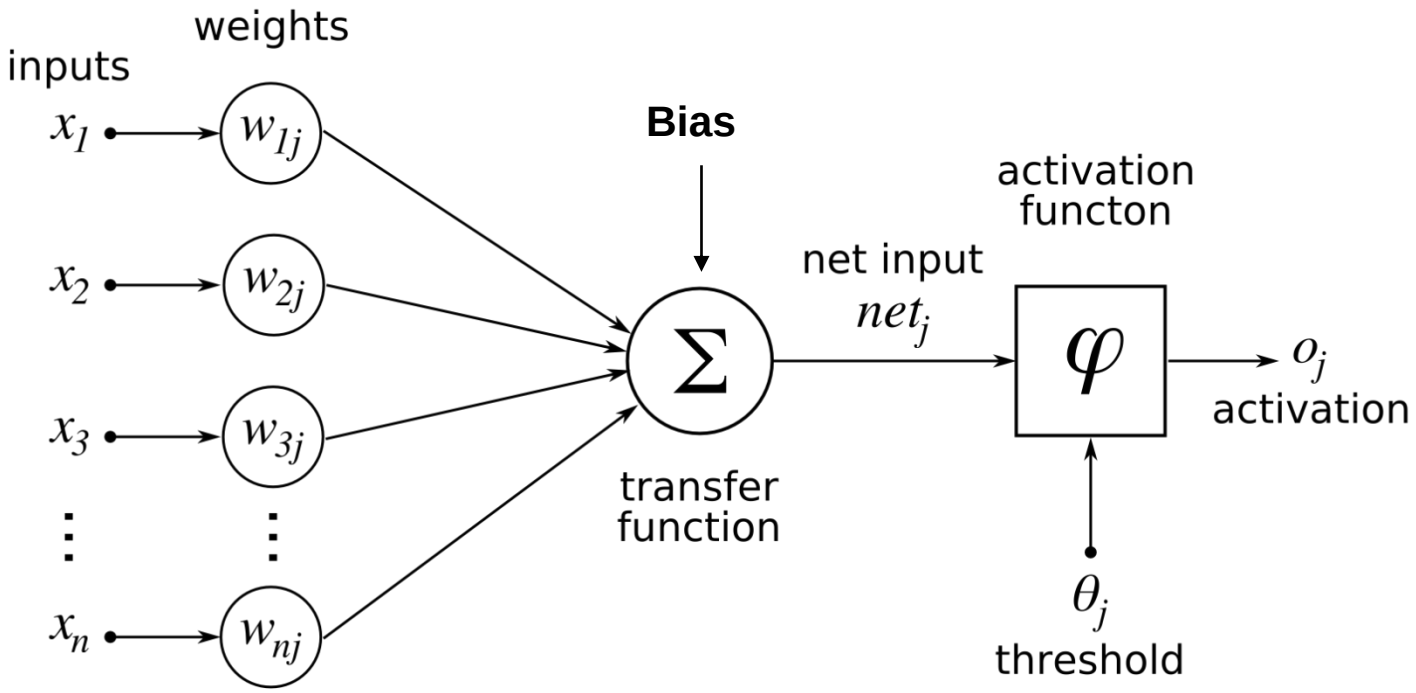


생물체의 neuron (좌)과 artificial neuron (우)

퍼셉트론 구조

다수의 신호를 입력 받아 하나의 신호를 출력하는 구조

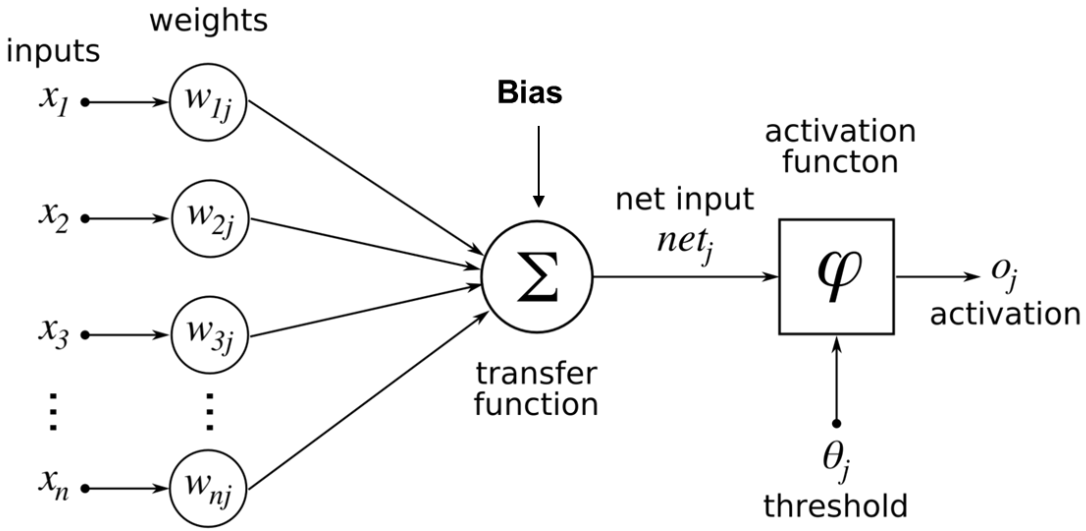
- 입력 신호 (x_1~ x_n), 출력 신호 (O_j)
- 신호를 전달하는 역할 (W_{1j}~ W_{nj}): 가중치 (weight)
- 퍼셉트론의 동작 경향성 (Bias): 바이어스
- 신호의 전달 여부 결정 (activation function): 활성함수



퍼셉트론 예시

활성 함수 문제

- net_j의 값이 얼마 일 때, 여친을 보기 위해 외출을 할까 ?
- **활성함수의 조건**
  - 값이 0보다 크다면, 값을 그대로 출력
  - 값이 0보다 작다면, 값을 0으로 출력

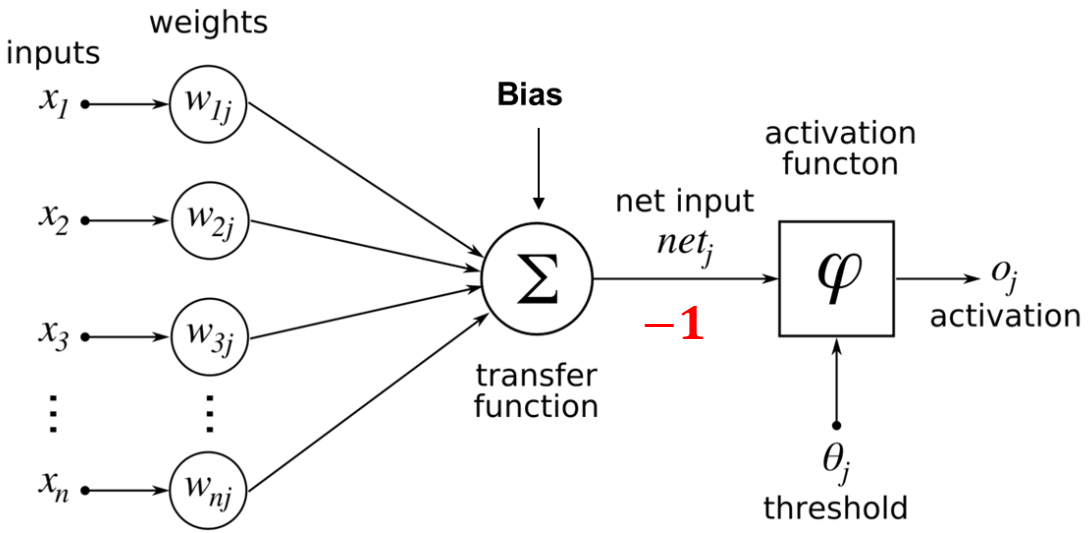


x, w	의미
x_1	비가 많이 온다
x_2	여친이 만나자고 한다
w_1	비를 좋아하는 정도
w_2	여친을 좋아하는 정도
net_j	외출을 최종적으로 결정하기 전
o_j	외출의 유무

퍼셉트론 예시

활성 함수 문제

- $\boldsymbol{net_j = x_1 * w_1 + x_2 * w_2 + Bias}$ 의 값이 얼마 일 때, 여친을 보기 위해 외출을 할까 ?
- $\boldsymbol{-1 = x_1(1) * w_1(-5) + x_2(1) * w_2(6) + Bias}$ $\boldsymbol{(-2, I\ don't\ like\ rainy\ days)}$
- **활성함수의 조건 (0보다 작다면)** 때문에, 외출 안 한다



$\boldsymbol{x, w}$	의미
$\boldsymbol{x_1}$	비가 많이 온다
$\boldsymbol{x_2}$	여친이 만나자고 한다
$\boldsymbol{w_1}$	비를 좋아하는 정도
$\boldsymbol{w_2}$	여친을 좋아하는 정도
$\boldsymbol{net_j}$	외출을 최종적으로 결정하기 전
$\boldsymbol{o_j}$	외출의 유무

퍼셉트론 예시

선형 분류로 표현이 가능한 퍼셉트론

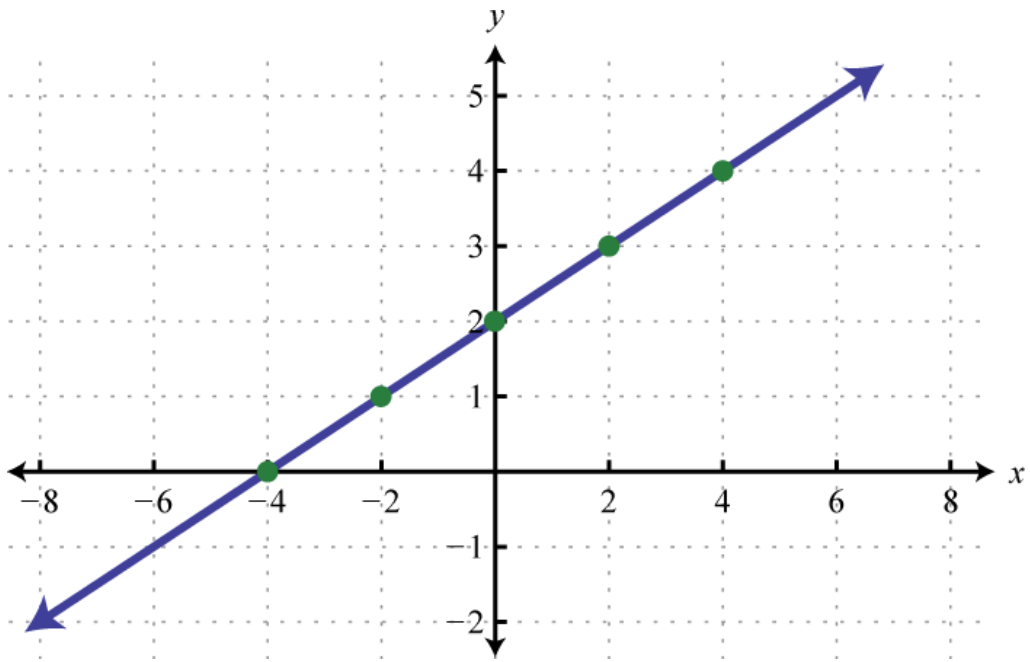
- 퍼셉트론은 **선형 방정식**으로 표현 가능, 즉 **선형 분류기**로 동작
- 퍼셉트론을 사용하여 논리적인 연산의 최소 단위인 **논리연산 (AND, OR, NAND)** 구현 가능

$$y = 1 * (-5) + (1) * (6) = 1$$

난 태생적으로 비가 싫어 (bias): -2

$$y = 1 - 2 = -1$$

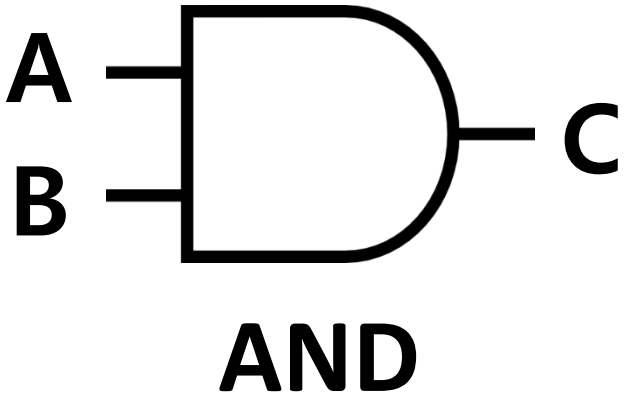
$$y = x_1 * w_1 + x_2 * w_2 + bias$$



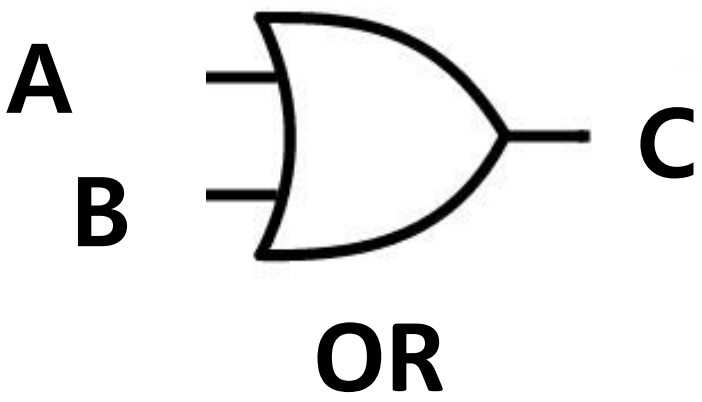
퍼셉트론: 논리연산

논리 연산 (AND, OR, NAND)

- 불 대수 (Bool algebra)는 **두개의 상태 (참, 거짓)**으로 동작하는 연산



A/B	C
$0/0$	0
$1/0$	0
$0/1$	0
$1/1$	1

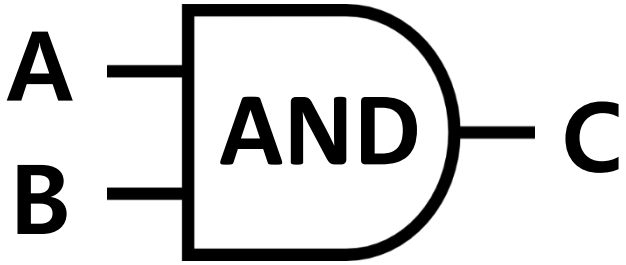


A/B	C
$0/0$	0
$1/0$	1
$0/1$	1
$1/1$	1

퍼셉트론: 논리연산 예제 #1

논리 연산 (AND)

- 두종류의 입력을 받고 (x, w) 가중치 합을 계산 (numpy.sum)
  - 입력의 값이 모두 1인 경우에만 결과값 1
  - numpy.sum(입력1, 입력2): 입력1과 입력2를 서로 각각 곱하여 하나의 합으로 계산
  - 예제에서는 **x와 w의 가중치 합 (x*w)**
- 가중치 합의 값이 임계값 (theta)보다 크다면 1, 그렇지 않다면 0



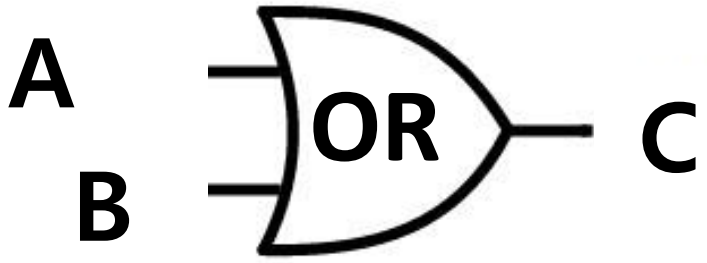
$$C > threshold, 1, else\ 0$$
$$(w1, w2, threshold) = (0.5, 0.5, 0.2)$$

```python
import numpy as np
from __future__ import print_function

def AND(x1, x2):
    x = np.array([x1,x2])
    w = np.array([0.5,0.5])
    theta = 0.7
    if np.sum(w*x) <= theta:
        return ?
    else:
        return ?
inputData = np.array([[0,0],[1,0],[0,1],[1,1]])

for x in inputData:
 print(x[0],", ",x[1]," ==> ",AND(x[0],x[1]), sep='')
```

퍼셉트론: 논리연산 예제 #2

논리 연산 (OR)

- 두종류의 입력을 받고 (x, w) 가중치 합을 계산 (numpy.sum)
  - 입력의 값이 둘 중에 하나라도 1인 경우 결과값 1
  - numpy.sum(입력1, 입력2): 입력1과 입력2를 서로 각각 곱하여 하나의 합으로 계산
  - 예제에서는 **x와 w의 가중치 합 (x*w)**
- 가중치 합의 값이 임계값 (theta)보다 크다면 1, 그렇지 않다면 0



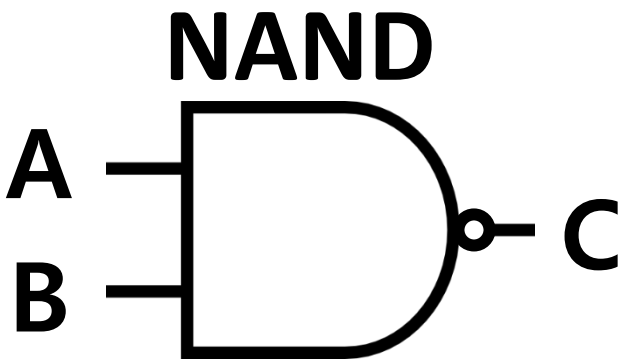
$$C > threshold, 1, else\ 0$$
$$(w1, w2, threshold) = (0.5, 0.5, 0.2)$$

```python
import numpy as np
from __future__ import print_function

def OR(x1, x2):
    x = np.array([x1,x2])
    w = np.array([0.5,0.5])
    theta = 0.2
    if np.sum(w*x) <= theta:
        return ?
    else:
        return ?
inputData = np.array([[0,0],[1,0],[0,1],[1,1]])

for x in inputData:
 print(x[0],", ",x[1]," ==> ",OR(x[0],x[1]), sep='')
```

퍼셉트론: 논리연산 예제 #3

논리 연산 (NAND)

- 두종류의 입력을 받고 (x, w) 가중치 합을 계산 (numpy.sum)
  - 입력의 값이 둘 다 1이면 결과값 0
  - numpy.sum(입력1, 입력2): 입력1과 입력2를 서로 각각 곱하여 하나의 합으로 계산
  - 예제에서는 **x와 w의 가중치 합 (x*w)**
- 가중치 합의 값이 임계값 (theta)보다 크다면 1, 그렇지 않다면 0

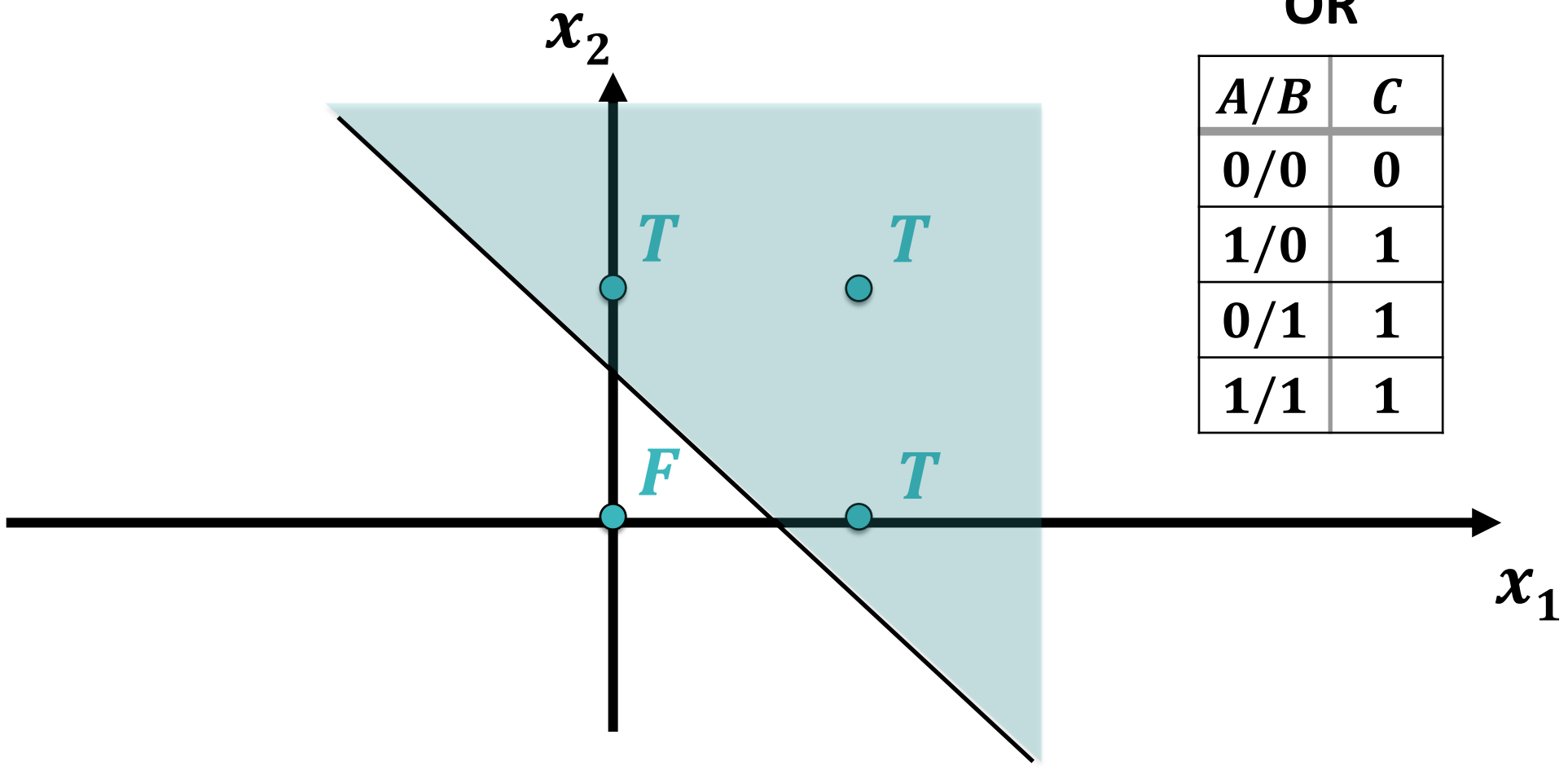


$$C > threshold, 1, else\ 0$$
$$(w1, w2, threshold) = (-0.5, -0.5, -0.7)$$

```
1   import numpy as np
2   from __future__ import print_function
3
4   def NAND(x1, x2):
5       x = np.array([x1,x2])
6       w = np.array([-0.5,-0.5])
7       theta = -0.7
8       if np.sum(w*x) <= theta:
9           return ?
10      else:
11          return ?
12  inputData = np.array([[0,0],[1,0],[0,1],[1,1]])
13
14  for x in inputData:
15   print(x[0],", ",x[1]," ==> ",NAND(x[0],x[1]), sep='')
```

퍼셉트론: 논리연산

- 퍼셉트론의 논리 연산은 선형적인 그래프로 표현 가능
  - 선형적인 문제 해결 가능 !



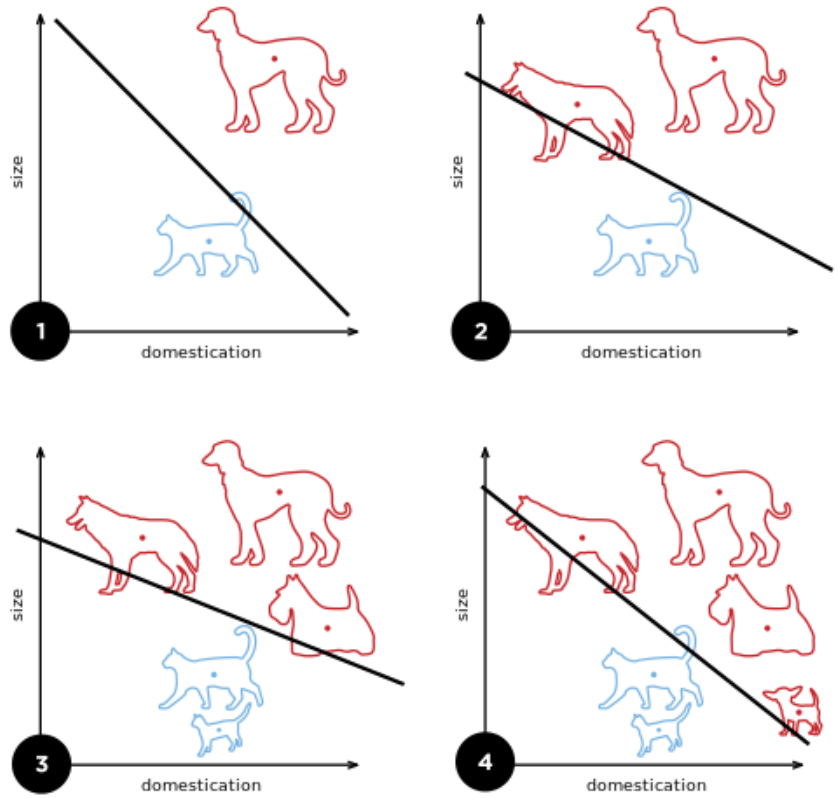
OR

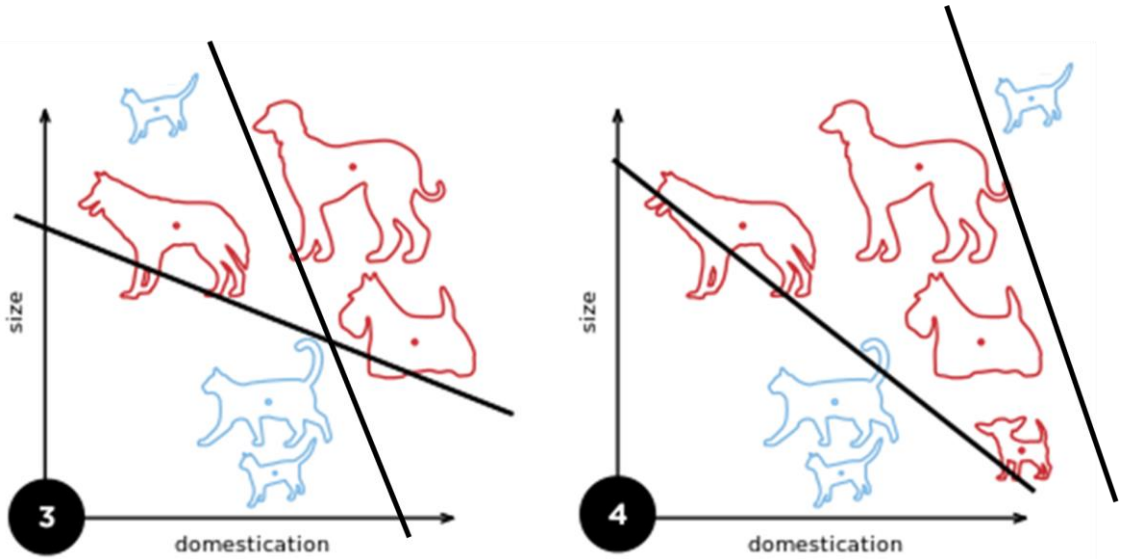
A/B	C
0/0	0
1/0	1
0/1	1
1/1	1

퍼셉트론의 한계

비 선형적인 문제는 해결 불가능

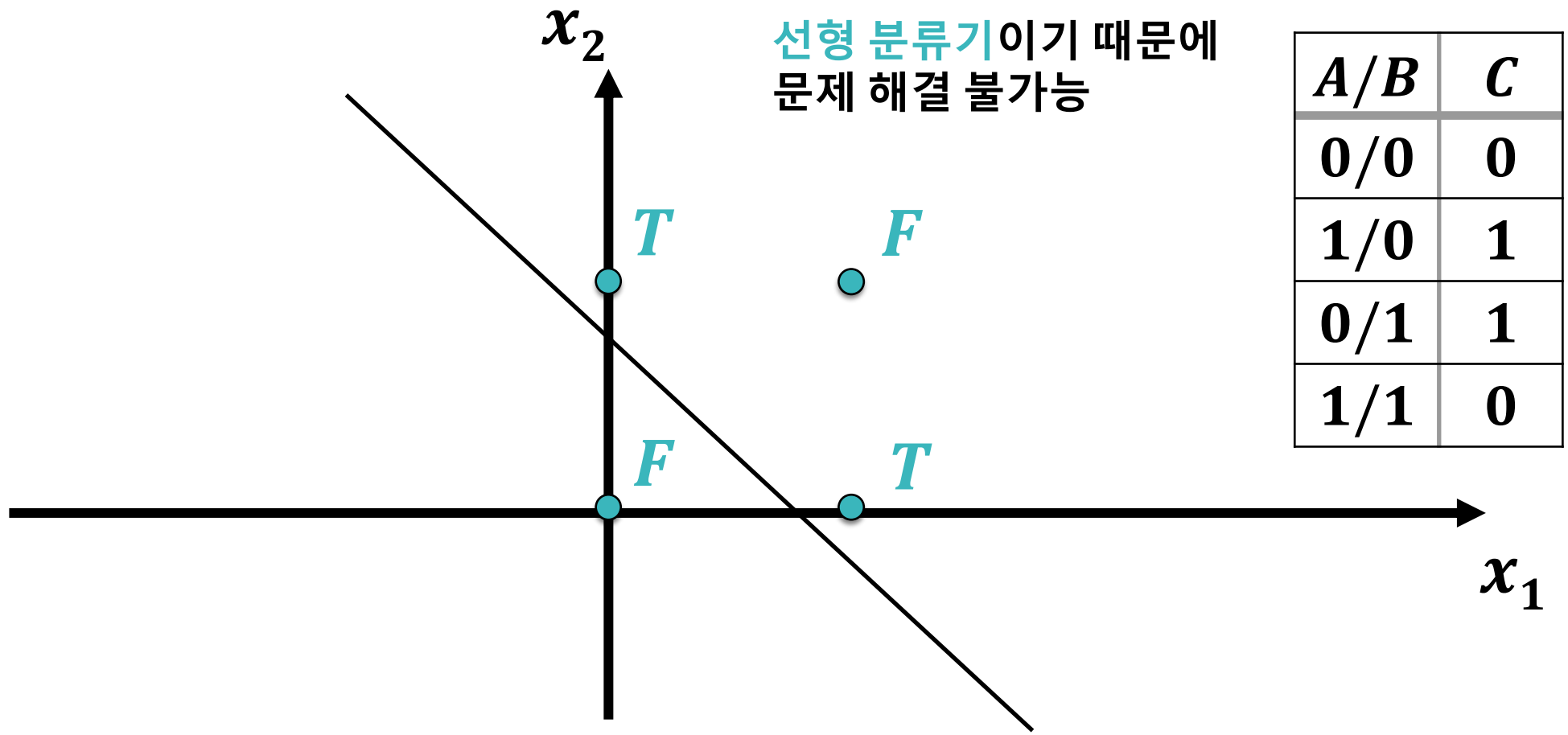
- **선형적인 문제:** 선형 방정식 하나로 해결 가능
- **비 선형적인 문제:** 선형 방정식 하나로 해결 불가능





퍼셉트론의 한계

- **비 선형적인 문제는 해결 불가능 (XOR 문제)**
  - **단층 퍼셉트론은 다양한 문제의 해결 불가능**
  - 따라서 선형적인 특선을 **벗어난 비 선형적인 접근 방법 필요**



선형 분류기이기 때문에
문제 해결 불가능

A/B	C
0/0	0
1/0	1
0/1	1
1/1	0

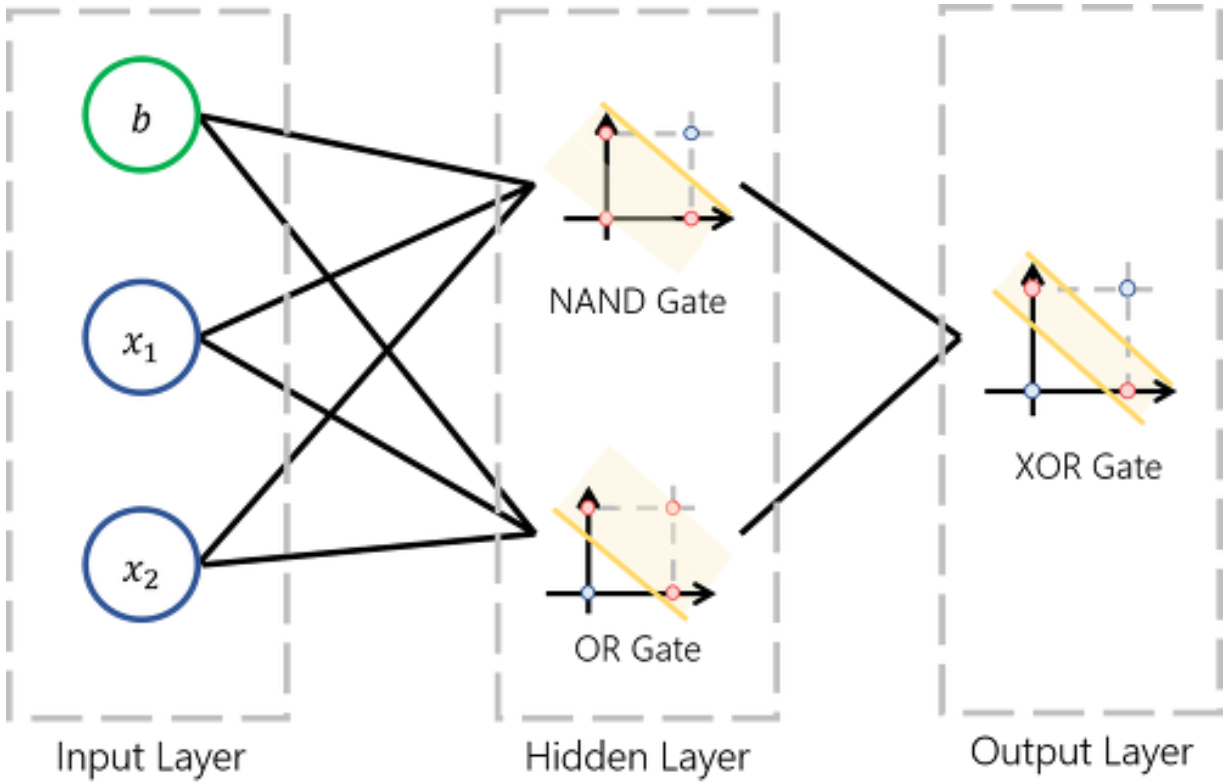
다층 퍼셉트론

더 복잡하고 어려운 문제를 해결하는 방법

퍼셉트론을 여러 개 쌓는 방법

비 선형적인 문제는 단층 퍼셉트론을 사용하여 해결 불가능

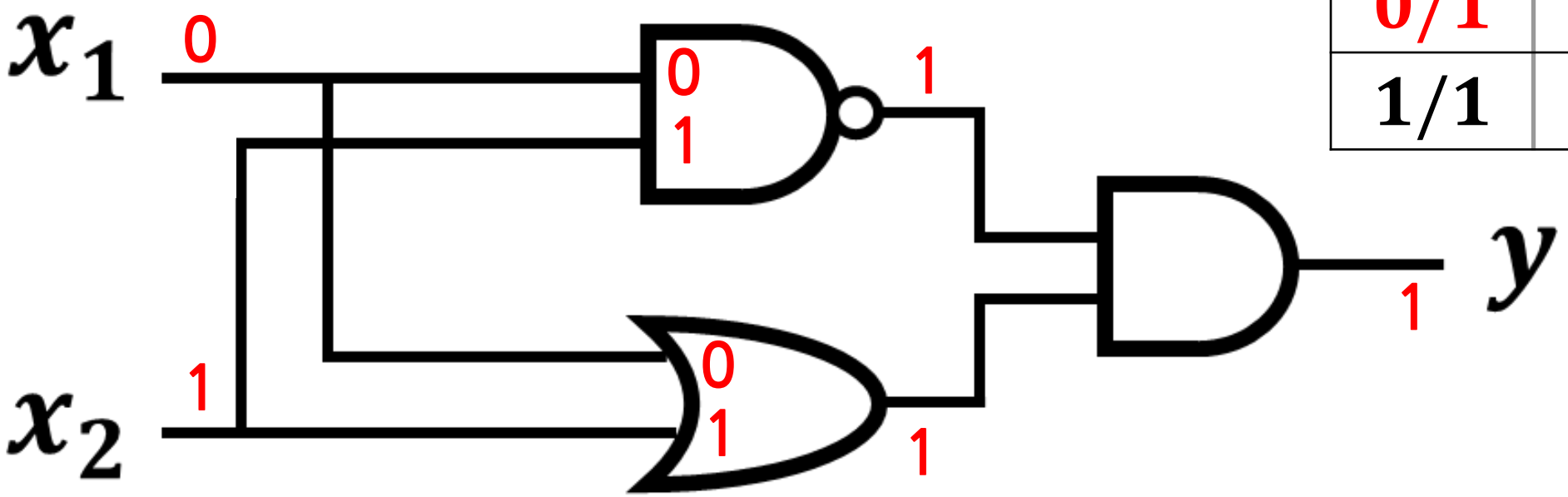
- XOR 논리연산은 하나의 레이어 (단층)을 사용하여 해결하는 것은 불가능
- 하지만, NAND, OR, AND 연산을 사용하여 해결 가능 (다층 퍼셉트론)



XOR=NAND+OR+AND

- **비 선형적인 문제는 단층 퍼셉트론을 사용하여 해결 불가능**
  - XOR 논리연산은 하나의 레이어 (단층)을 사용하여 해결하는 것은 불가능
  - 하지만, NAND, OR, AND 연산을 사용하여 해결 가능

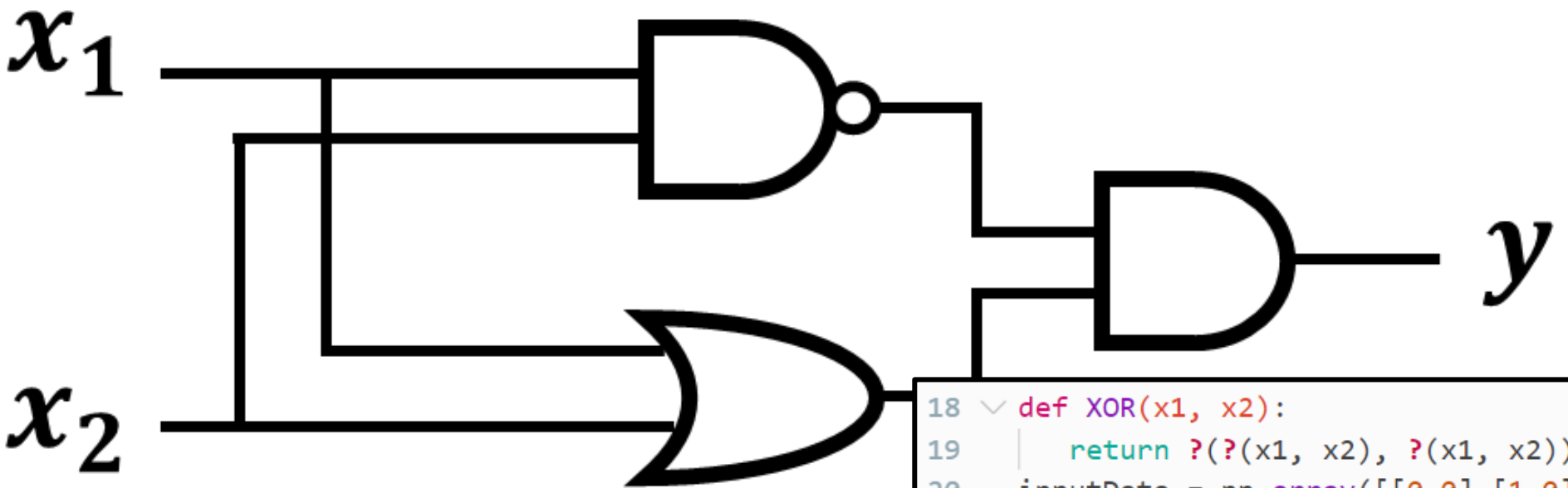
x_1/x_2	y
0/0	0
1/0	1
0/1	1
1/1	0



다층 퍼셉트론 예제 #1

여러 종류의 논리연산을 조합하여 XOR 연산 구현

- XOR 연산은 NAND와 OR연산의 결과를 AND로 연산하여 구현 가능

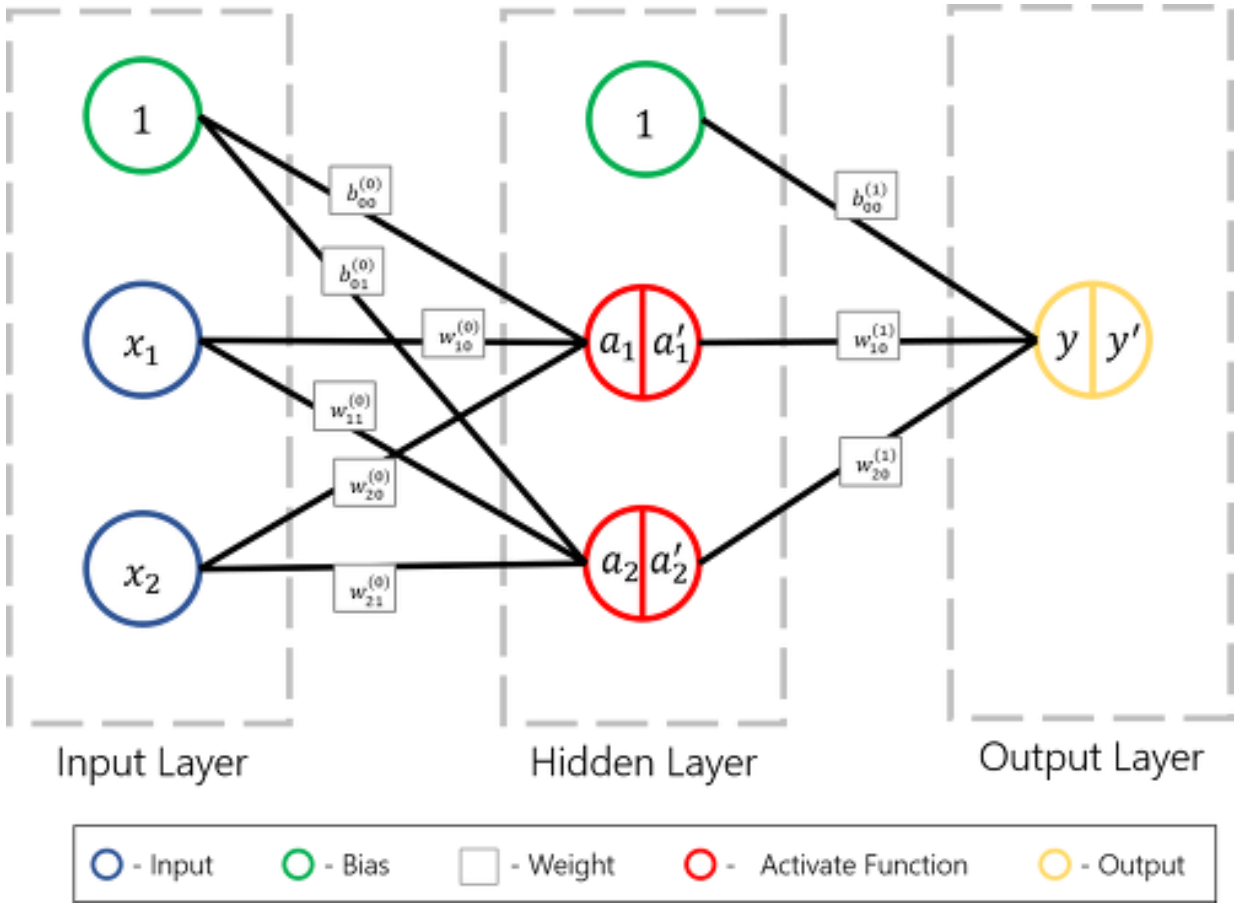



```python
def XOR(x1, x2):
    return ?(?(x1, x2), ?(x1, x2))
inputData = np.array([[0,0],[1,0],[0,1],[1,1]])

for x in inputData:
  print(x[0],", ",x[1]," ==> ",XOR(x[0],x[1]), sep = '')
```

다층 퍼셉트론 (Multi Layer Perceptron, MLP)

여러 층으로 구성된 퍼셉트론

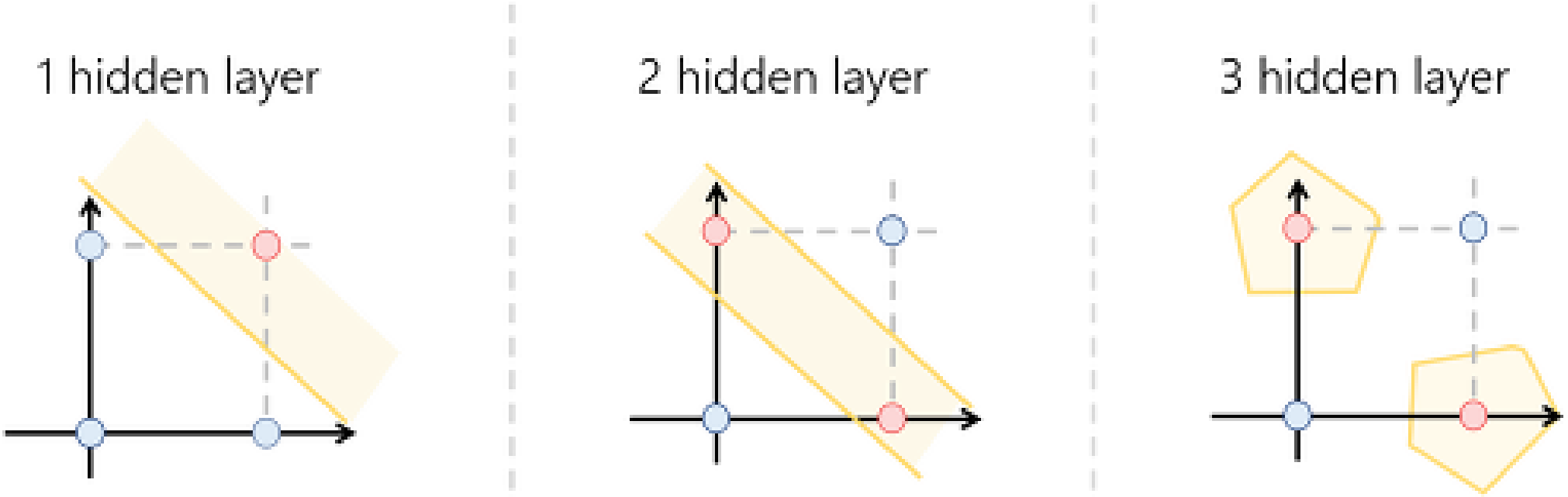
- 처음 (Input), 결과 출력 (Output), 중간 부분 (Hidden)
- Hidden Layer가 3층 이상이 되면 Deep Learning



다층 퍼셉트론 (Multi Layer Perceptron, MLP)

MLP의 개수에 따라 결정할 수 있는 영역

- 1층인 경우 **선형분리**, 2층은 **구역분리**
- 3층은 더 **세분화된 분리** 가능



다층 퍼셉트론 예제

더 복잡하고 어려운 문제를 해결하는 방법

붓꽃 데이터를 사용한 다층 퍼셉트론 예제

학습 성능 평가

- 본 예제에서는 **Precision**, **Recall**, **F1-Score**를 사용하여 성능을 평가
  - Precision과 Recall을 측정하기 위해서는 **Confusion matrix** 필요
  - **정확도 (Accuracy)**: 예측이 정답과 얼마나 정확한가 ?
  - 정밀도 (Precision): 예측한 것 중에 정답의 비율은 ?
  - 재현율 (Recall): 찾아야 할 것 중에서 실제로 찾은 비율은 ?
  - F1-Score: 정밀도와 재현율의 평균
- **정확도, 정밀도, 재현율이 높다고 성능이 좋은 것은 아님, F1-Score의 값이 높으면 성능이 좋다고 할 수 있음 !**

$$F1 = 2. \frac{Precision \times Recall}{Precision + Recall}$$

혼동 행렬 (Confusion matrix)

4가지 정보를 바탕으로 3가지 척도를 계산

- **정확도 (Accuracy):** 정확도는 1을 1로, 0을 0로 **정확하게 분류**한 것을 의미
- **정밀도 (Precision):** 모델을 1이라고 분류한 그룹 A가 있을 때, **믿을 만한 정도**로 A를 만들어 냈는지 평가
  - 예) 어부가 그물을 던져 물고기를 잡을 때, 그물안에 1이라는 물고기가 얼마나 있을지에 대한 척도
- **재현도 (Recall):** 정밀도와 비교되는 척도, **전체 예측 중에 TP가 얼마나 많은 것인가에 대한 정보**
  - 관심있는 영역만을 추출했는지를 의미하는 것으로 모형의 실용성과 관련된 척도

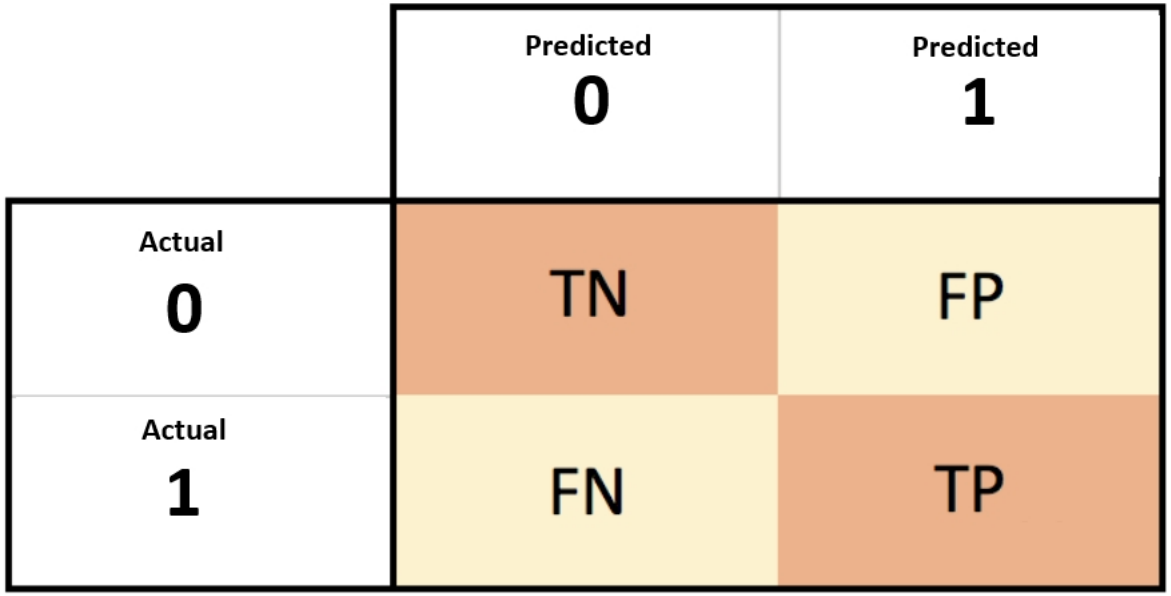
- $\text{Accuracy(정확도)} = \dfrac{TP + TN}{TP + TN + FP + FN}$
- $\text{Precision(정밀도)} = \dfrac{TP}{TP + FP}$
- $\text{Recall(재현도)} = \dfrac{TP}{TP + FN}$

	Predicted 0	Predicted 1
Actual 0	TN	FP
Actual 1	FN	TP

혼동 행렬 (Confusion matrix)

훈련된 모델의 성능을 측정하기 위한 Matrix

- 모델을 **평가하는 지표** (정밀함, 실용적인 분류, 정확한 분류)
  - **레이블 0, 1을 가진 데이터를 분류**한다고 할 때, 관심 범주를 1이라고 가정
  - True Positives (TP): 1인 레이블을 1이라고 함. (관심 범주를 정확하게 분류)
  - False Negatives (FN): 1인 레이블을 0이라고 함. (관심 범주가 아닌 것으로 잘못 분류)
  - False Positives (FP): 0인 레이블을 1이라고 함. (관심 범주라고 잘못 분류)
  - True Negatives (TN): 0인 레이블을 0이라고 함. (관심 범주가 아닌 것을 정확하게 분류)



정확도를 계산하는 방법 예제

너의 목소리가 보여를 사용한 예제

- 6명의 출연자가 있으며, **실력자**와 **음치**를 **다양한 단서**를 통해 가려내는 프로그램

정확도를 계산하는 방법 예제

- **노래를 부르는 모습을 보고 감으로 예측**
  - 예측 번호: [1, 2, 3, 4, 5, 6]
  - 정답: [음치, 음치, 음치, 음치, 정상, 음치]
  - 예측: [음치, 음치, 정상, 정상, 정상, 정상
  - **음치를 찾는 것이 목적이라면**
    - **정확도**: 1, 2, 5, 6는 맞추고, 3, 4번 틀림
      - 6명 중에서 4명을 맞췄음으로, 2/3=0.66
    - **정밀도**: 음치라고 예측한 사람 중에, 진짜 음치는 ?
      - 음치라 예측한 1, 2번이 둘다 음치, 2/2=1.00
    - **재현율**: 전체 음치 중에 맞춘 음치의 비율은 ?
      - 음치가 4명 있는데 그중에서 2명을 맞춤, 2/4=0.5
    - **F1-Score**: 정밀도와 재현율의 평균
      - 2*정밀도*재현율/(정밀도+재현율)=2*1.00*0.5/(1.00+0.5)=0.66

정확도를 계산하는 방법 예제

어렵게 하는 법과 쉽게 하는 방법

- numpy.equal (data1, data2)
  - data1과 data2의 동일 경우 카운트
- numpy.sum (조건)
  - 조건에 해당되는 것들만 합에 계산
- **Sklearn의 함수를 사용하여 쉽게 구현 가능**
  - accuracy_score(test, target)
    - test와 target를 사용하여 정확도 계산
  - precision_score()
  - recall_score()
  - f1_score()

```
1   # 성능 측정을 위한 계산
2   # 직업 계산을 하도록 프로그래밍 하여 구현
3
4   accuracy = np.mean(        (y,p))
5   right = np.sum(            )
6   precision = right / np.sum(p)
7   recall = right / np.sum(y)
8   f1 = 2 * precision*recall/(precision+recall)
9
10  print('accuracy',accuracy)
11  print('precision', precision)
12  print('recall', recall)
13  print('f1', f1)
14
15  # sklearn 을 이용하면 전부 계산해줍니다.
16
17  print('accuracy', metrics.accuracy_score(y,p) )
18  print('precision', metrics.                (y,p) )
19  print('recall', metrics.recall_score(y,p) )
20  print('f1', metrics.        (y,p) )
21
22  print(metrics.classification_report(y,p))
23  print(metrics.confusion_matrix(y,p))
```

붓꽃 데이터를 사용한 다층 퍼셉트론 예제

붓꽃 데이터 셋

- setosa, versicolor, virginica라는 품종의 데이터
- 학습을 위해서는 **Training Data Set**과 **Test Data Set**으로 구분되어야 함
- **먼저 iris 데이터의 키 값 확인** !
  - Data: iris 데이터가 있는 부분
  - Feature_names: iris data set의 특징을 기술하는 부분
  - **Target_names**과 **Target**은 다층 퍼셉트론의 이용해 분석하려는 **목적 변수**
  - Target에는 setosa, verginia, virginica 등이 존재
  - DESCR: iris data에 대한 설명을 포함

```
iris=load_iris()
iris.keys()
```

```
dict_keys(['data', 'target', 'target_names', 'DESCR', 'feature_names', 'filename'])
```

붓꽃 데이터를 사용한 다층 퍼셉트론 예제

데이터 확인 및 셋 분리

- Iris data의 0번째부터 9번째까지 슬라이싱
  - 슬라이싱: 배열에서 특정 부분만을 확인하는 방법
  - 출력된 값들은 iris 데이터를 분류하는 특징의 값에 해당
- Iris 데이터 셋의 data를 X변수 할당, target 변수는 y변수에 할당

```
In [56]: iris['data'][0:10]

Out[56]: array([[5.1, 3.5, 1.4, 0.2],
                [4.9, 3. , 1.4, 0.2],
                [4.7, 3.2, 1.3, 0.2],
                [4.6, 3.1, 1.5, 0.2],
                [5. , 3.6, 1.4, 0.2],
                [5.4, 3.9, 1.7, 0.4],
                [4.6, 3.4, 1.4, 0.3],
                [5. , 3.4, 1.5, 0.2],
                [4.4, 2.9, 1.4, 0.2],
                [4.9, 3.1, 1.5, 0.1]])
```

```
1    X = iris['data']
2    y = iris['target']
```

붓꽃 데이터를 사용한 다층 퍼셉트론 예제

Train, Test 데이터 셋 구분

- 다층 퍼셉트론을 학습하기 위해서는 **학습과 테스트 셋** 필요
  - sklearn.model_selection 라이브러리 안에 있는 train_test_split 모듈 사용
- 분할된 훈련 데이터 셋과 테스트 셋은 각각 x_train, X_test, y_train, y_test에 할당
  - 할당 이후에 **정규화** 과정을 진행 (정규화 방법 명시, fit, transform 순으로 진행)

```
 5   from sklearn.model_selection import train_test_split
 6   X_train, X_test, y_train, y_test =  train_test_split(X,y)
 7
 8   from sklearn.preprocessing import StandardScaler
 9   scaler =                    ()
10
11   scaler.    (X_train)
12
13   X_train = scaler.           (X_train)
14   X_test = scaler.           (X_test)
```

붓꽃 데이터를 사용한 다층 퍼셉트론 예제

MLP 알고리즘 로드 및 Hidden Layer 할당

- 다층 퍼셉트론을 학습하기 위해서 **Hidden Layer의 설정** 필요
  - sklearn.neural_network 라이브러리의 MLPClassifier 사용
- MLPClassifier는 다중신경망 분류 알고리즘을 저장하고 있는 모듈
  - Hidden Layer의 크기를 설정
  - 예) hidden_layer_sizes=(10,10,10)은 3개의 Hidden Layer를 만들고 각 계층별로 노드 10개씩 할당
  - **이후 fit ()함수를 사용하여 학습하고, predict ()를 사용하여 성능을 평가**

```
17    from sklearn.neural_network import MLPClassifier
18    mlp = MLPClassifier(                    =(10,10,10))
19
20    mlp.    (X_train, y_train)
21    predictions = mlp.        (X_test)
```

붓꽃 데이터를 사용한 다층 퍼셉트론 예제

성능 평가

- 성능 평가의 지표는 직접 계산을 하거나 API를 사용하여 쉽게 계산 가능
  - sklearn.metrics 라이브러리의 confusion_matrix, classification_report 함수 사용
- confusion_matrix (y_ture, y_pred)
  - y_true (정답), y_pred (predict ()에서 얻은 예측값)을 사용하여 계산
- classification_report (y_ture, y_pred)
  - y_true (정답), y_pred (predict ()에서 얻은 예측값)을 사용하여 계산

```
2    from sklearn.metrics import classification_re
3
4    print(confusion_matrix(y_test, predictions))
5    print(                        (y_test, predictions))
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	17
1	0.73	0.62	0.67	13
2	0.50	0.62	0.56	8
accuracy			0.79	38
macro avg	0.74	0.75	0.74	38
weighted avg	0.80	0.79	0.79	38

와인 데이터를 사용한 다층 퍼셉트론 예제

와인의 화학성분을 입력으로 사용하여 어떤 품종인지 예측하는 신경망

- 알고 싶은 결과는 첫 번째 행에 해당되는 Cultivator
  - Alcohol ~ Proline 까지의 데이터가 입력으로 들어 왔을 때, **Cultivator (품종)**을 예측하는 모델을 설계 !
- **wine.shape**
  - 데이터 셋의 크기를 확인 (178개의 데이터와 14개의 속성 값)

	Cultivator	Alchol	Malic_Acid	Ash	Alcalinity_of_Ash	Magnesium	Total_phenols	Falvanoids	Nonflavanoid_phenols	Proanthocyanins	Color_intensity
0	1	14.23	1.71	2.43	15.6	127	2.80	3.06	0.28	2.29	5.64
1	1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26	1.28	4.38
2	1	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30	2.81	5.68
3	1	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24	2.18	7.80
4	1	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39	1.82	4.32

와인 데이터를 사용한 다층 퍼셉트론 예제

Train, Test 데이터 셋 구분

- 다층 퍼셉트론을 학습하기 위해서는 **학습과 테스트 셋** 필요
  - sklearn.model_selection 라이브러리 안에 있는 train_test_split 모듈 사용
- 분할된 훈련 데이터 셋과 테스트 셋은 각각 x_train, X_test, y_train, y_test에 할당
  - 할당 이후에 **정규화** 과정을 진행 (정규화 방법 명시, fit, transform 순으로 진행)

```
# Data and Label
X = wine.drop('Cultivator',axis=1)
y = wine['Cultivator']

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y)

# Regularization
from sklearn.preprocessing import StandardScaler
scaler =                ()

# Fit only to the training data
scaler.    (X_train)

StandardScaler(copy=True, with_mean=True, with_std=True)

# Now apply the transformations to the data:
X_train = scaler.          (X_train)
X_test = scaler.          (X_test)
```

와인 데이터를 사용한 다층 퍼셉트론 예제

MLP 알고리즘 로드 및 Hidden Layer 할당

- 다층 퍼셉트론을 학습하기 위해서 **Hidden Layer의 설정** 필요
  - sklearn.neural_network 라이브러리의 MLPClassifier 사용
- MLPClassifier는 다중신경망 분류 알고리즘을 저장하고 있는 모듈
  - Hidden Layer의 크기를 설정
  - 예) hidden_layer_sizes=(13,13,13)은 3개의 Hidden Layer를 만들고 각 계층별로 노드 13개씩 할당
  - Max_iter 옵션은 학습의 반복 횟수와 연관 있는 옵션 (1장씩 학습한다면, 500장의 이미지를 학습)
  - **이후 fit ()함수를 사용하여 학습하고, predict ()를 사용하여 성능을 평가**

```
21    from sklearn.neural_network import MLPClassifier
22
23    mlp = MLPClassifier(                                ,max_iter=500)
24    mlp.fit(X_train,y_train)
25    predictions = mlp.        (X_test)
```

와인 데이터를 사용한 다층 퍼셉트론 예제

성능 평가

- 성능 평가의 지표는 직접 계산을 하거나 API를 사용하여 쉽게 계산 가능
  - sklearn.metrics 라이브러리의 confusion_matrix, classification_report 함수 사용
- confusion_matrix (y_ture, y_pred)
  - y_true (정답), y_pred (predict ()에서 얻은 예측값)을 사용하여 계산
- classification_report (y_ture, y_pred)
  - y_true (정답), y_pred (predict ()에서 얻은 예측값)을 사용하여 계산

```
2   from sklearn.metrics import classification_report
3
4   print(                (y_test, predictions))
5   print(                      (y_test, predictions))
```

	precision	recall	f1-score	support
1	0.93	1.00	0.96	13
2	0.89	0.94	0.92	18
3	1.00	0.86	0.92	14
accuracy			0.93	45
macro avg	0.94	0.93	0.93	45
weighted avg	0.94	0.93	0.93	45

감사합니다.

오늘하루 수업 듣느라 고생하셨습니다 !