2020531001 윤정인

[8주차 발표]

인공지능

CHAP 9 – 기계 학습

Contents

01

기계학습(Machine Learning) 이란?

기계 학습(Machine Learning)

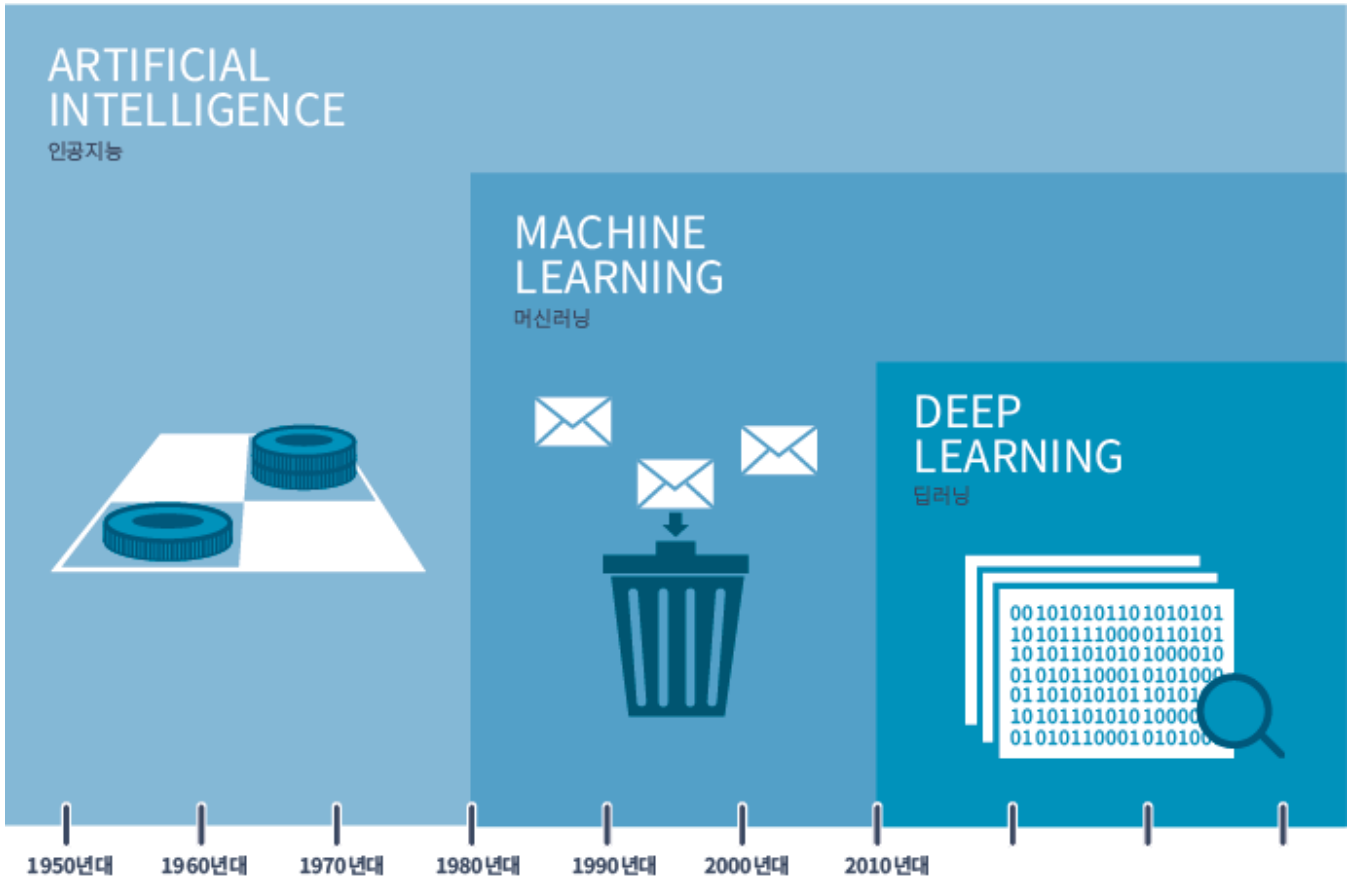
인공지능의 한 분야로, 컴퓨터에 학습 기능을 부여하기 위한 연구 분야

패턴 인식 및 계산 학습 이론에서 진화하여 컴퓨터가 학습하는 알고리즘을 연구

학습 데이터가 많아지면 알고리즘 성능 향상

인공지능, 기계학습, 딥러닝 간의 관계

인공지능 ⊃ 기계학습 ⊃ 딥러닝



http://www.tcpschool.com/deep2018/deep2018_ai_vs

01. 기계학습(Machine Learning) 이란?

기계 학습의 응용 분야

기계학습의 역사

- 1957 — 퍼셉트론 (Perceptron) 차후 딥러닝의 기반
- 1985 — 역전파 학습 알고리즘
- 1990~ — 기계학습이 독립적인 분야로 재편 통계와 확률 이론으로 초점
- 1995 — SVM의 인기 (Support Vector Machine)
- 1998 — 필기체 인식
- 2010~ — 딥러닝의 인기
- 2012 — ImageNet 에서의 우승 딥러닝의 기계학습 주도
- 2017 — Google의 TensorFlow 발표

02

기계학습의 종류

기계학습의 종류

지도 학습

"교사"에 의해 주어진 예제(샘플)와
정답(레이블)을 제공받음

목표는 입력을 출력에 매핑하는
일반적인 규칙(함수)을 학습하는 것

Ex.
강아지와 고양이를 구분하는 문제라면
강아지와 고양이에 대한 영상을 제공한 후,
교사가 어떤 영상이 강아지인지
어떤 영상이 고양이인지를 알려주는 것

비지도 학습

외부에서 정답(레이블)이 주어지지 않고
학습 알고리즘이 스스로 입력에서
어떤 구조를 발견하는 학습

비지도 학습을 사용하면 데이터에서
숨겨진 패턴 발견 가능

강화 학습

보상 및 처벌의 형태로 학습데이터가 주어져
보상과 처벌을 통하여 학습이 이루어짐

주로 차량 운전이나 상대방과의 경기 같은
동적인 환경에서 프로그램의 행동에 대한 피드백만 제공

Ex.
바둑에서 어떤 수를 두어서 승리하였다면 보상이 주어짐

03

기계학습의 용어들

특징 (Feature)

- 학습 모델에게 공급하는 입력
- 가장 간단한 경우는 입력 자체가 특징이 됨
- Ex. 이메일에 “검찰”이라는 문자 포함 여부 (YES or NO)
  이메일 발신자의 도메인 (문자열)
  이미지로만 이루어진 이메일 (YES or NO)

샘플

- 기계 학습에 주어지는 특정한 예제
- y = f(X)에서 x에 해당
- 레이블이 있는 샘플과 레이블이 없는 샘플도 존재
- 지도 학습을 시키려면 레이블이 있어야 함

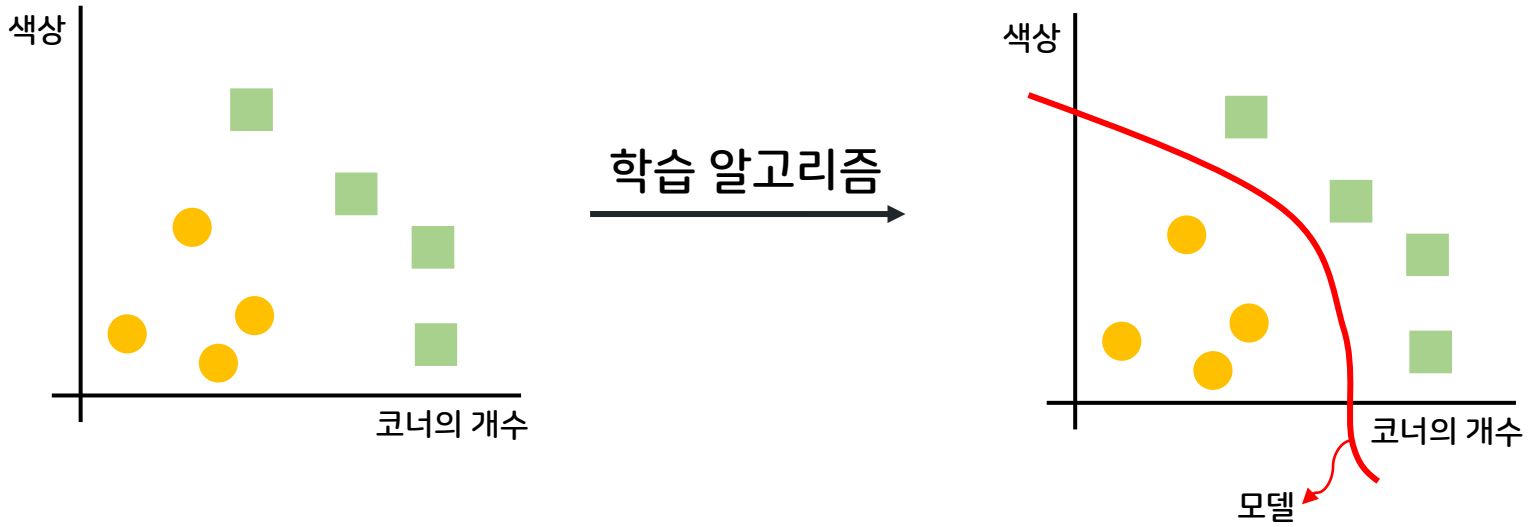
레이블(Label)

- 기계학습으로 예측하는 항목
- y = f(X)에서 y 변수에 해당
- Ex. 농작물의 향후 가격
  사진에 표시되는 동물의 종류
  동영상의 의미 등등

학습과 예측

- 학습 : 모델을 만들거나 배우는 것
  - 레이블이 있는 샘플을 모델에 보여주고
    모델이 특징과 레이블의 관계를 점차적으로 학습
- 예측 : 학습된 모델을 레이블에 없는 샘플에 적용하는 것
  - 학습된 모델을 사용하여 유용한 예측을 해내는 것
    예측을 통하여 레이블이 없는 새로운 샘플의 레이블의 추론이 가능

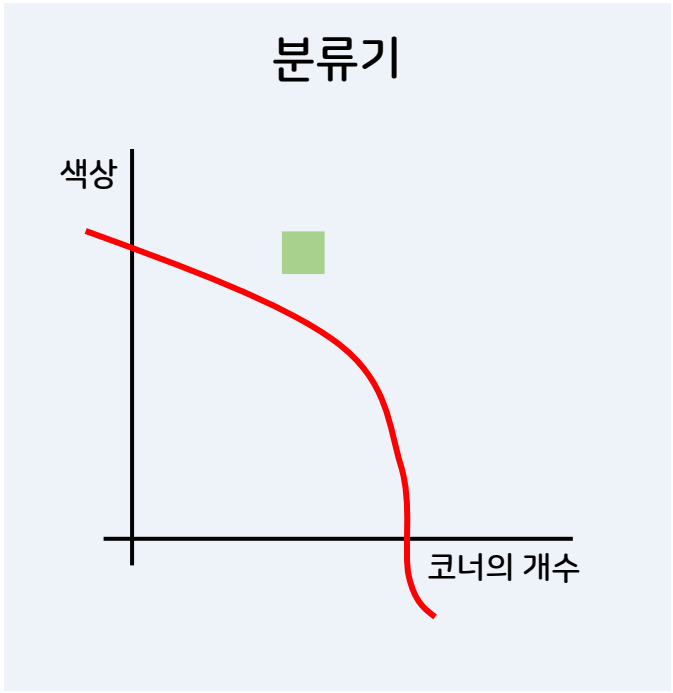
학습 데이터와 테스트 데이터



학습 알고리즘은 입력 데이터의 특징에 따라 입력을 “원”과 “사각형”으로 분류할 수 있는 모델을 내부적으로 생성

학습 데이터와 테스트 데이터

새로운 테스트 데이터



출력

“사각형”

학습이 끝나면 한 번도 본적이 없는 새로운 데이터로 시스템을 테스트

04

지도 학습

지도 학습

회귀

주어진 입력 – 출력 쌍을 학습한 후에 새로운 입력 값이 들어왔을 때
합리적인 출력 값을 예측하는 것

학습시키는 데이터가 이산적이 아니고 연속적
입력과 출력이 모두 실수

학습이 끝나면 매핑 함수가 만들어지고 새로운 입력데이터가 있을 때
매핑 함수를 통하여 해당 데이터에 대한 출력 변수 값을 예측함

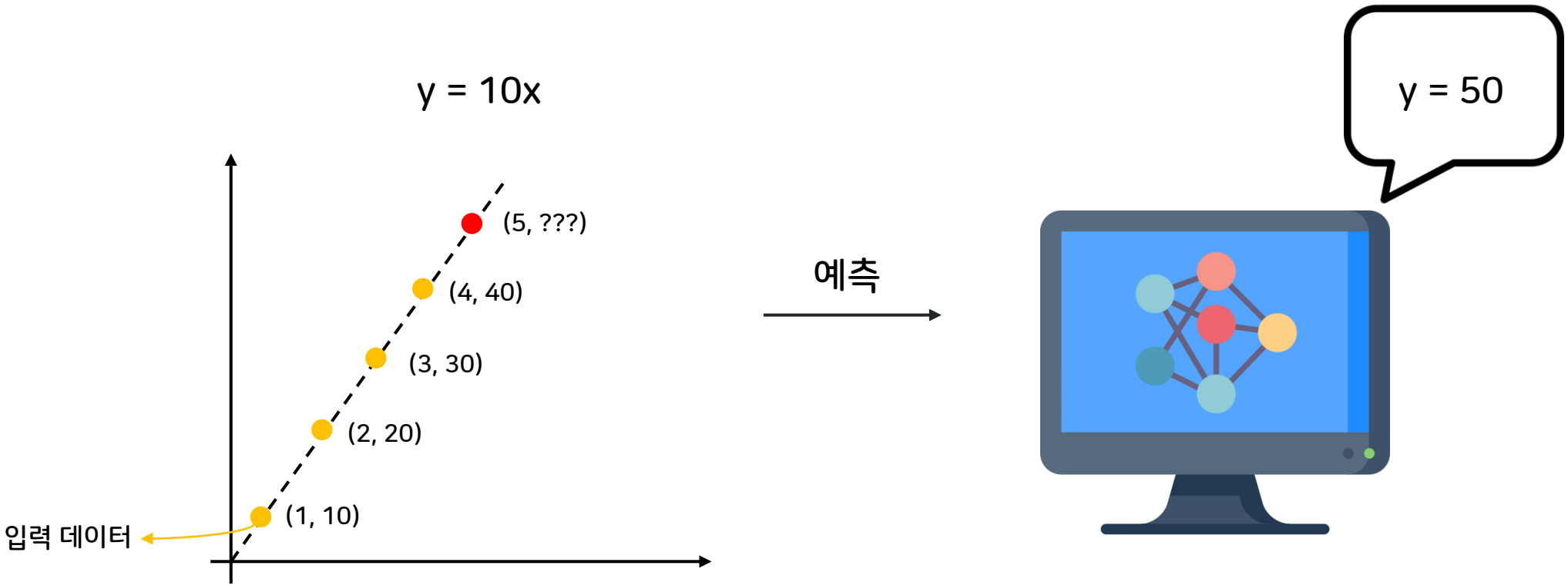
분류

입력을 두 개 이상의 레이블(유형)으로 분할하는 것
해당 모델을 학습시킬 때 레이블을 제공해야 함

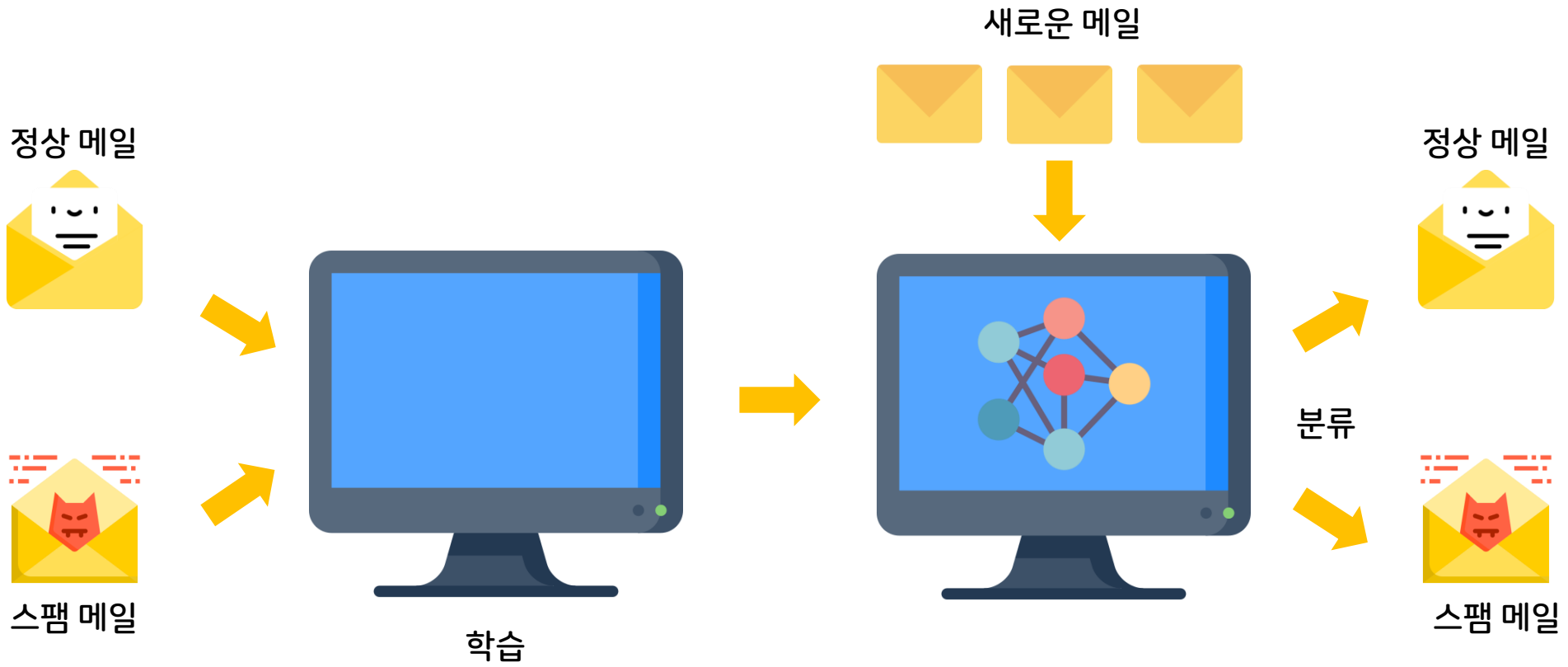
학습시에는 교사가 있어 입력의 올바른 레이블을 알려줌

학습이 끝나면 학습자가 한 번도 보지 못한 입력을
이들 레이블 중의 하나로 분류하는 시스템

회귀



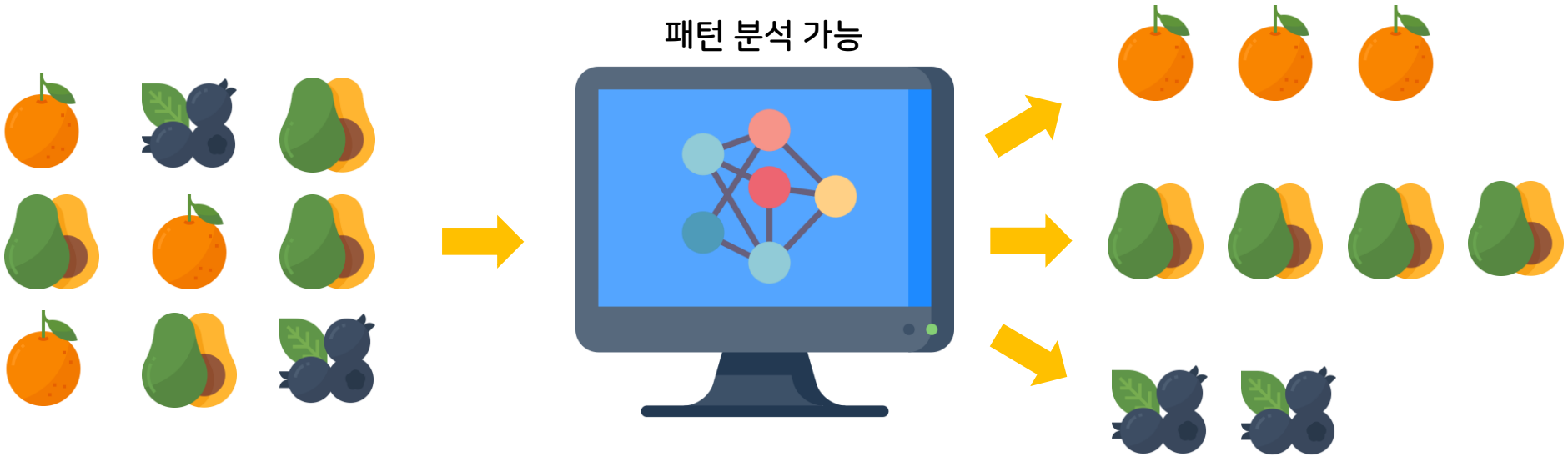
분류



05

비지도 학습

비지도 학습

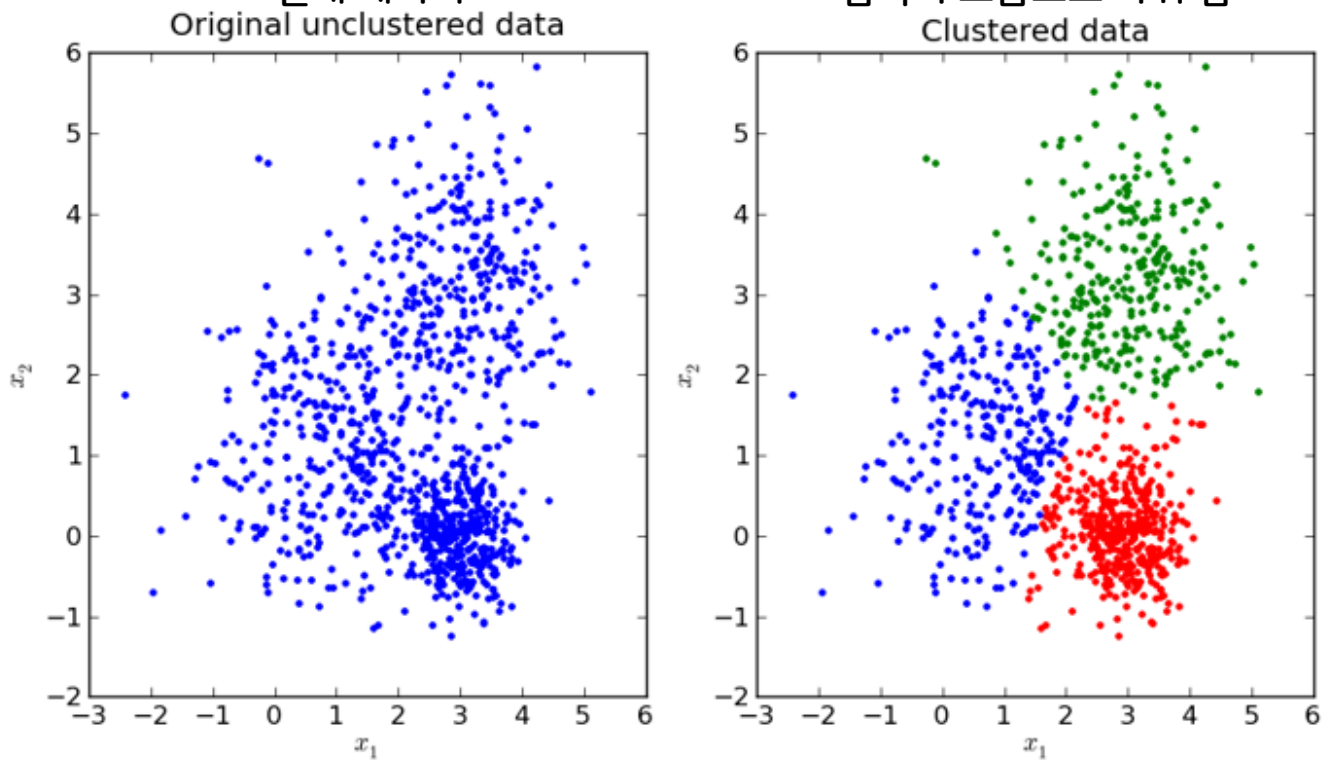


비지도 학습

거리 계산

원래 데이터

입력이 그룹으로 나눠 짐

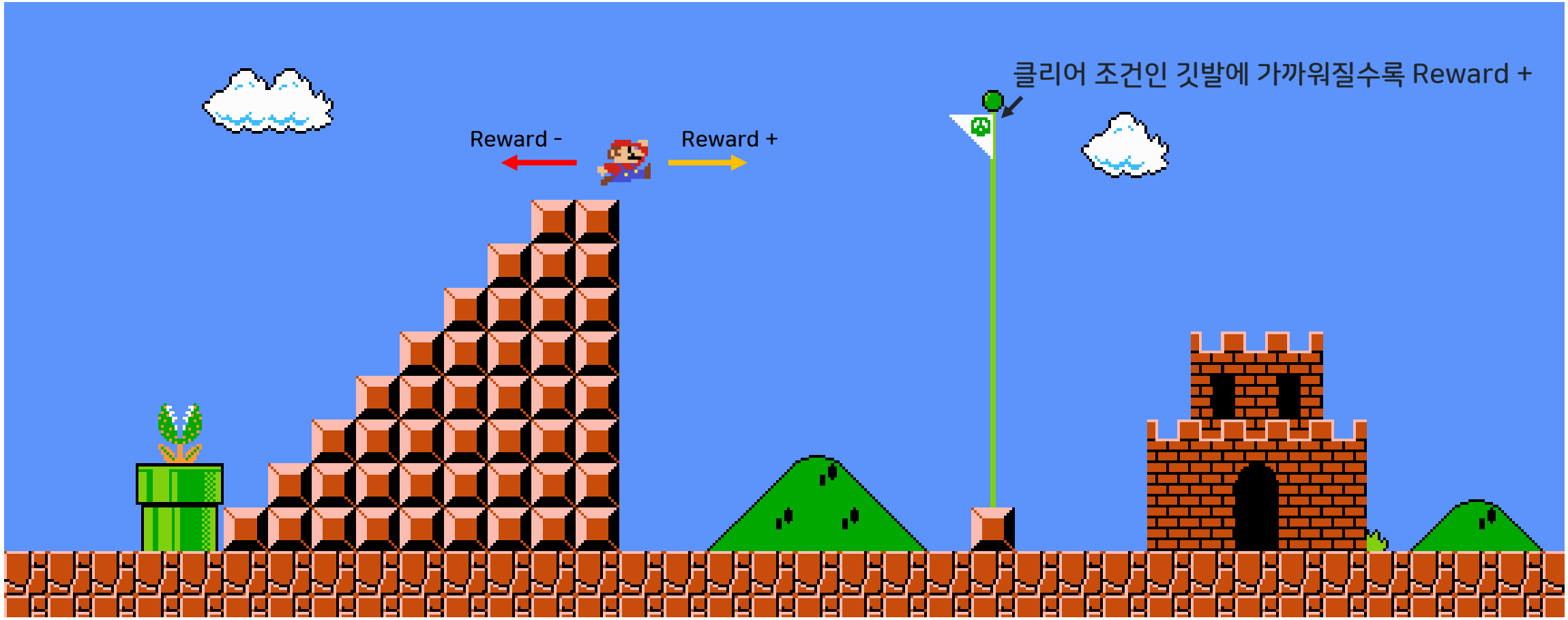


https://bioinformaticsandme.tistory.com/131

06

강화 학습

강화 학습



BUT

컴퓨터가 행동을 취했다고 해서 그 결과가 즉시 나타나지 않을 수 있음

이를 지연보상 문제라고 하며, 강화학습에서는 당장의 보상 값도 고려해야 하지만 나중에 얻을 이익까지도 계산해 넣어야 함

07

기계학습의 실용적인 가치

기계 학습의 실용적인 가치

프로그래밍 시간 단축

맞춤형 제품을 쉽게 개발할 수 있음

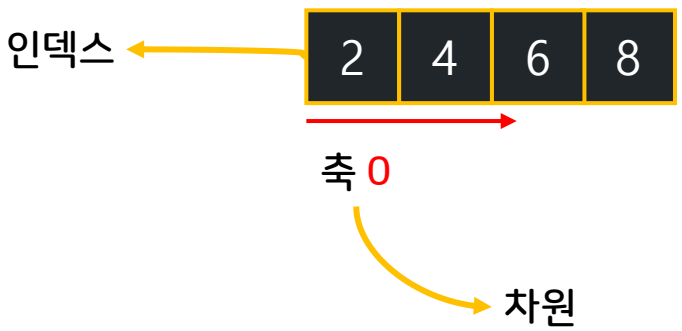
프로그래머로 시도할 알고리즘이 떠오르지 않는 문제 해결 가능

08

넘파이(Numpy)

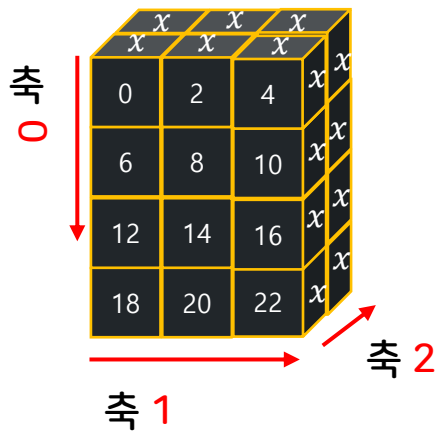
넘파이(Numpy) – 다차원 배열

1차원 배열



2차원 배열

3차원 배열



리스트 vs 넘파이 배열

리스트

```
#리스트를 이용한 시험성적 저장 프로그램

mid_scores = [10,20,30]
final_scores = [70,80,90]

total = (mid_scores + final_scores)

print(total);
```

```
[10, 20, 30, 70, 80, 90]
```

2개의 리스트를 합한 형태로 출력

넘파이 배열

```
#넘파이 배열을 이용한 시험성적 저장 프로그램

import numpy as np

mid_scores = np.array([10,20,30])
final_scores = np.array([70,80,90])

total = (mid_scores + final_scores)

print(total)
```

```
[ 80 100 120]
```

배열과 배열 간에 수학적인 연산 적용 가능

인덱싱과 슬라이싱

인덱싱

특정한 요소를 얻는 방법

0 1 2 3

grades = [88, 72, 93, 94]

```
[8]  grades = np.array([88, 72, 93, 94])

     grades[2] #2번 요소 얻기
```

```
93
```

```
[9]  grades[-1] #맨 마지막 요소 얻기
```

```
94
```

슬라이싱

요소 집합을 선택하는 방법

0 1 2 3 4

grades = [88, 72, 93, 94]

```
[11] grades = np.array([88, 72, 93, 94])

     grades[1:3] #인덱스 1~3까지의 요소 출력
```

```
array([72, 93])
```

```
[12] grades[:2] #시작, 종료 인덱스는 생략 가능
```

```
array([88, 72])
```

논리적인 인덱싱

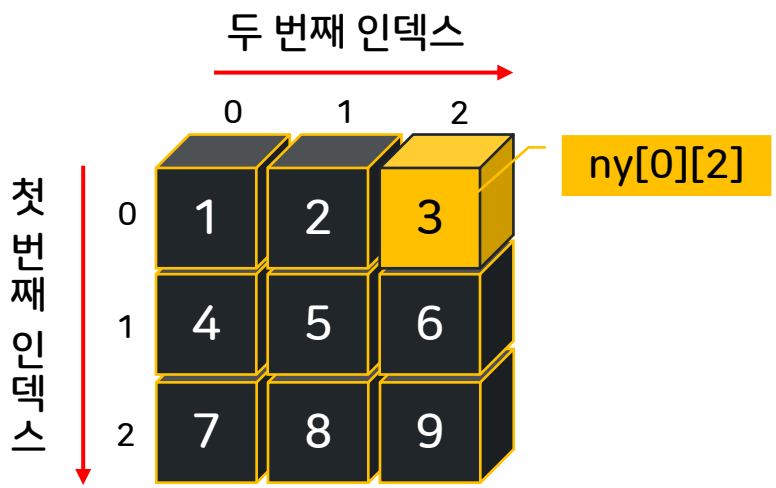
```
ages = np.array([18, 19, 25, 30, 28])

y = ages >20
y #결과는 부울형의 넘파이 배열
```

```
array([False, False,  True,  True,  True])
```

```
[18] ages[ages>20] #부울형 배열을 인덱스로 하여 배열 ages에 보내기
```

```
array([25, 30, 28])
```

2차원 배열




```
#넘파이 2차원 배열 생성하기

import numpy as np
y = [[1,2,3],[4,5,6],[7,8,9]] #파이썬의 2차원 리스트

ny = np.array(y) #넘파이 2차원 배열로 호출
print(ny)

print(ny[0][2]) #인덱스[0],[2]에 있는 요소 출력
```

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
3
```

arange() 함수

np.arange(start, stop, step)

시작값 종료값 간격

```
[28] #arange()함수

     import numpy as np

     np.arange(5) #0~4까지 넘파이 배열 생성
```

```
array([0, 1, 2, 3, 4])
```

```
[29] np.arange(1,6) #시작값과 종료값 지정 1~5
```

```
array([1, 2, 3, 4, 5])
```

```
[30] np.arange(1,10,2) #1~9까지 증가값 2로 배열 생성
```

```
array([1, 3, 5, 7, 9])
```

linspace() 함수

np.linspace(start, stop, step)

시작값 종료값 개수

```
#linspace() 함수

import numpy as np

np.linspace(0,10,50) #0~10까지 균일한 간격으로 50개 생성
```

```
array([ 0.        ,  0.20408163,  0.40816327,  0.6122449 ,  0.81632653,
        1.02040816,  1.2244898 ,  1.42857143,  1.63265306,  1.83673469,
        2.04081633,  2.24489796,  2.44897959,  2.65306122,  2.85714286,
        3.06122449,  3.26530612,  3.46938776,  3.67346939,  3.87755102,
        4.08163265,  4.28571429,  4.48979592,  4.69387755,  4.89795918,
        5.10204082,  5.30612245,  5.51020408,  5.71428571,  5.91836735,
        6.12244898,  6.32653061,  6.53061224,  6.73469388,  6.93877551,
        7.14285714,  7.34693878,  7.55102041,  7.75510204,  7.95918367,
        8.16326531,  8.36734694,  8.57142857,  8.7755102 ,  8.97959184,
        9.18367347,  9.3877551 ,  9.59183673,  9.79591837, 10.        ])
```

logspace() 함수

np.logspace(x, y, n)

```
#logspace() 함수

import numpy as np

np.logspace(0,5,10)
```

→ $10^0 \sim 10^5$ 까지 10개

```
array([1.00000000e+00, 3.59381366e+00, 1.29154967e+01, 4.64158883e+01,
       1.66810054e+02, 5.99484250e+02, 2.15443469e+03, 7.74263683e+03,
       2.78255940e+04, 1.00000000e+05])
```

reshape() 함수

```
new_array = old_array.reshape((3,4))
```

새로운 배열 — 원래의 배열 — 행의 개수 — 열의 개수

```
[ ]  y = np.arange(12) #1차원 배열 생성
     y

     array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11])

[ ]  y.reshape(3,4) #1차원 배열을 3행 4열짜리 2차원 배열로 재생성

     array([[ 0,  1,  2,  3],
            [ 4,  5,  6,  7],
            [ 8,  9, 10, 11]])

     new_array = y.reshape(6,-1) #데이터의 갯수에 맞춰서 자동으로 배열의 형태 결정
     new_array

     array([[ 0,  1],
            [ 2,  3],
            [ 4,  5],
            [ 6,  7],
            [ 8,  9],
            [10, 11]])
```

난수 생성 – rand()

```
#난수 생성하기

import numpy as np

np.random.seed(100) #난수 시드 결정

print("난수 5개 출력", np.random.rand(5),"₩n")

print("난수 2차원 배열(크기 = 5 X 3) 출력₩n", np.random.rand(5,3))
```

```
난수 5개 출력 [0.54340494 0.27836939 0.42451759 0.84477613 0.00471886]

난수 2차원 배열(크기 = 5 X 3) 출력
 [[0.12156912 0.67074908 0.82585276]
 [0.13670659 0.57509333 0.89132195]
 [0.20920212 0.18532822 0.10837689]
 [0.21969749 0.97862378 0.81168315]
 [0.17194101 0.81622475 0.27407375]]
```

```
a =10; b=20
print("10~20사이 난수 5개 :",(b-a)*np.random.rand(5)+a,"₩n")

print("10~20사이 정수 난수 5개 : ", np.random.randint(10,21,size = 5))

print("₩n10~20 사이에 크기가 4x7인 2차원 배열₩n",np.random.randint(10,21,size=(4,7)))
```

```
10~20사이 난수 5개 : [15.62379405 11.87743677 15.73114708 11.35465055 13.7562767 ]

10~20사이 정수 난수 5개 :  [17 17 20 14 11]

10~20 사이에 크기가 4x7인 2차원 배열
 [[11 18 17 20 18 18 18]
 [17 19 10 20 16 10 17]
 [14 15 11 15 15 13 14]
 [13 16 15 20 10 10 16]]
```

정규분포 난수 생성 – randn()

```
[78] #정규분포 난수 생성하기 평균값 0, 표준편차 1.0

     import numpy as np

     print("정규분포 난수 1차원배열 : ",np.random.randn(5))

     print("₩n정규분포 난수 2차원 배열 : ₩n", np.random.randn(5,4))
```

```
정규분포 난수 1차원배열 :  [ 0.95295943 -0.98336264 -1.09455719 -0.24134875 -0.59511384]

정규분포 난수 2차원 배열 :
 [[-0.98281189  0.08822993 -0.60275937  1.12555545]
 [ 0.9270533  -2.26915699 -0.34817064 -0.1640254 ]
 [ 1.21658059 -0.6268539  -0.27493981  1.14745744]
 [ 1.15996897  0.70218451 -1.28390481  1.52230777]
 [-0.17996332 -0.37614737  0.46034921  1.45542146]]
```

```
#평균값 10, 표준편차 2로 변경
m = 10; sigma = 2
m+sigma*np.random.randn(5)
```

```
array([10.48301378,  9.92466666, 11.66306606, 14.59873894,  8.12079511])
```

THANK YOU