K-nearest neighbor 알고리즘

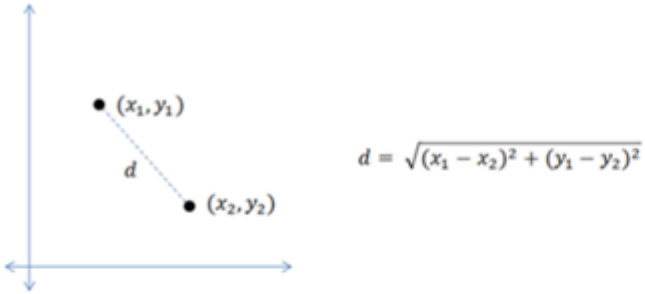
비슷하면서도 다른 K-NN과 K-Means

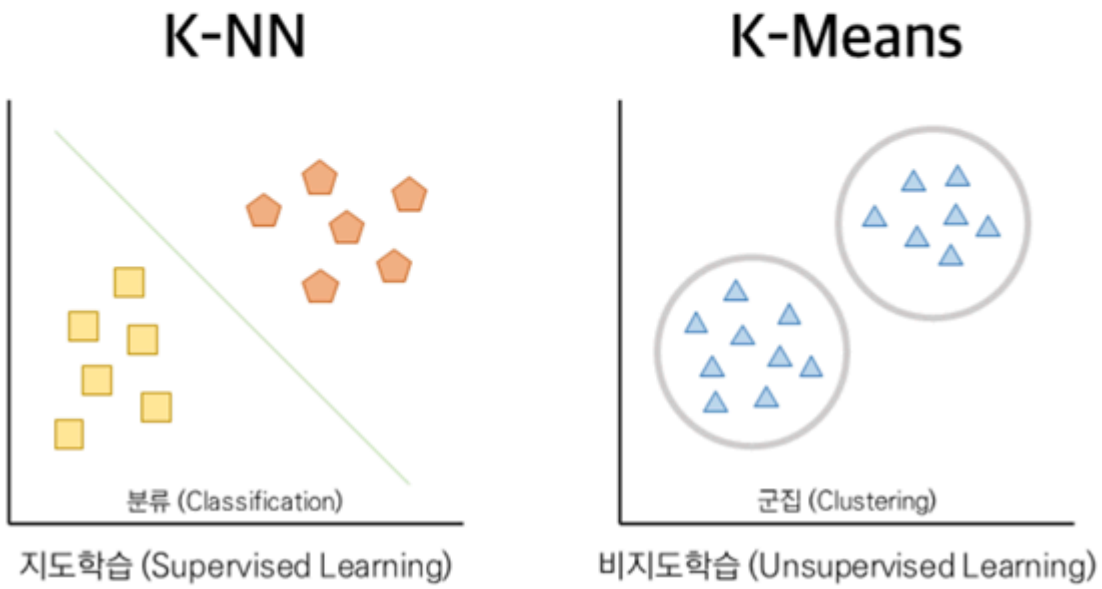
- 두 알고리즘은 특정 데이터를 '거리'를 기반으로 해, 그룹으로 나타낸다는 점에서 유사

  - '거리'는 유클리디안(Euclidean) 거리 - 데이터를 좌표상에 나타냈을 때 직선거리를 의미
  - 거리가 가까울수록 특성(Feature)의 유사도(Similarity)가 높다는 것을 뜻함

- 두 알고리즘의 큰 차이점은 '레이블(Lable)'의 유무

  - K-NN - 미리 레이블이 붙어 있는 데이터들을 학습하여 이를 바탕으로 새로운 데이터에 대해 분류를 수행
  - K-Means - 레이블을 모르더라도 비슷한 특징을 가진 데이터끼리 묶어주는 군집을 수행





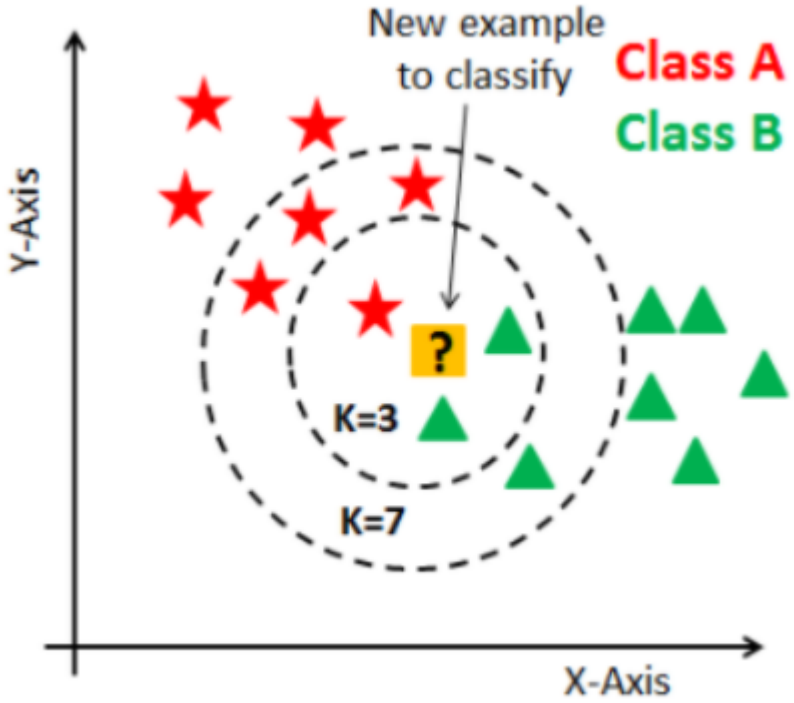
K-NN(K-Nearest Neighbor)

- K-최근접 이웃 - K개의 가까운 이웃의 속성에 따라 분류

  - K = 분류하고자 하는 하나의 데이터와 거리상으로 근접한 이웃의 개수
  - K개의 데이터가 속한 클래스 레이블에 따라 다수결 원칙에 의해 분류가 이루어짐

- 그림에서 물음표로 표시된 새로운 포인트는 근접한 이웃의 숫자에 따라 다른 클래스로 분류됨

  - 새로운 포인트는 K=3일 경우 근접한 이웃의 클래스가 ★는 1개, ▲는 2개로 Class B에 속하게 됨
  - K=7일 경우 근접한 이웃의 클래스가 ★는 4개, ▲는 3개로 Class A에 속하게 됨



```
#데이터 셋을 만듬(=10개)
#[x, y, type]
dataset = [[2.7810836,2.550537003,0],
    [1.465489372,2.362125076,0],
    [3.396561688,4.400293529,0],
    [1.38807019,1.850220317,0],
```

```
    [3.06407232,3.005305973,0],
    [7.627531214,2.759262235,1],
    [5.332441248,2.088626775,1],
    [6.922596716,1.77106367,1],
    [8.675418651,-0.242068655,1],
    [7.673756466,3.508563011,1]]
```

```
#거리 계산하기
#KNN의 거리를 구하는 공식은 유클리드 거리 공식을 사용

from math import sqrt
#calculate the Euclidean distance between two vectors
#row = [x, y, type]

def euclidean_distance(row1, row2): #row에는 type까지 포함되어 있기 때문에 마지막 range는 포함하지 않고 계산
    distance = 0.0
    for i in range(len(row1)-1):
        distance += (row1[i] - row2[i])**2
    return sqrt(distance)
```

```
#거리가 정상적으로 측정되는지 테스트
#row0은 새로 들어온 데이터이며 이 좌표와 DataSet에 있는 좌표간의 거리를 계산

row0 = [3, 3]
for row in dataset:
    distance = euclidean_distance(row0, row)
    print(distance) #10개 출력
```

```
0.2189163999999999
1.534510628
0.3965616879999998
1.61192981
0.06407232000000018
4.627531214
2.3324412480000003
3.922596716
5.675418650999999
```

```
4.673756466
```

```
#가장 근처에 있는 요소 뽑기
#train 변수는 데이터 셋, test_row는 측정하고자 하는 좌표, num_neighbors 변수가 K를 의미
#Locate the most similar neighbors

def get_neighbors(train, test_row, num_neighbors):
    distances = list()
    for train_row in train:
        dist = euclidean_distance(test_row, train_row)
        distances.append((train_row, dist))
    distances.sort(key=lambda tup: tup[1])
    neighbors = list()
    for i in range(num_neighbors):
        neighbors.append(distances[i][0])
    return neighbors
```

```
#함수를 선언하고 테스트하면 가장 가까운 3개의 좌표가 나오게 됨
#해당 테스트는 K=3일때 테스트
#이 리스트를 보면 아래와 같이 출력
#0 type이 3개 이므로 우리가 측정하고자 하는 값은 0 type이라는 것
neighbors = get_neighbors(dataset, row0, 3)
for neighbor in neighbors:
    print(neighbor)
```

```
[3.06407232, 3.005305973, 0]
[2.7810836, 2.550537003, 0]
[3.396561688, 4.400293529, 0]
```

```
#예측하기 - 예측된 type을 출력해주는 함수
# Make a classification prediction with neighbors
def predict_classification(train, test_row, num_neighbors):
    neighbors = get_neighbors(train, test_row, num_neighbors)
    for neighbor in neighbors:
        print(neighbor)
    output_values = [row[-1] for row in neighbors]
```

```
    prediction = max(set(output_values), key=output_values.count)
    return prediction
```

```
row0 = [3,3,0]

prediction = predict_classification(dataset, row0, 3)
print('Expected %d, Got %d.' % (row0[-1], prediction))
```

```
[3.06407232, 3.005305973, 0]
[2.7810836, 2.550537003, 0]
[3.396561688, 4.400293529, 0]
Expected 0, Got 0.
```

```
row0 = [6,5,0]
prediction = predict_classification(dataset, row0, 3)
print('Expected %d, Got %d.' % (row0[-1], prediction))
```

```
[7.673756466, 3.508563011, 1]
[3.396561688, 4.400293529, 0]
[7.627531214, 2.759262235, 1]
Expected 0, Got 1.
```

1-3-KNN-마켓과 머신러닝(생선분류)

<<<참조자료 사이트>>>

1. 머신러닝 알고리즘, K-NN과 K-Means란?
2. K-최근접 이웃(K-NN) 알고리즘
3. KNN(K-Nearest Neighbor) 알고리즘 - Python 예제
4. K-NN 알고리즘 (K-최근접 이웃) 개념
