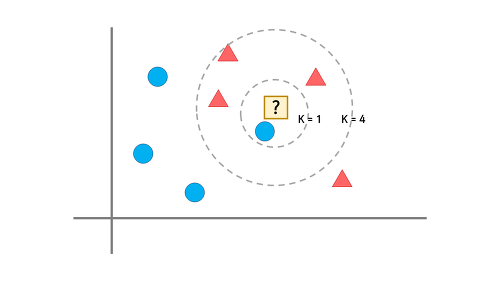
**K-Nearest Neighbors Algorithm(KNN, k-최근접 이웃법)**

* 단순히 주변에 무엇이 가장 가까이 있는가를 보는 것이 아니라 주변에 있는 몇 개의 것들을 같이 봐서 가장 많은 것의 그룹에 주어진 데이터를 분류하는 방식을 사용하게 된다
* 이것이 K-Nearest Neighbors(KNN)방법이다. 이 때, k는 주변의 개수를 의미한다



* K가 1일 때는 "?"를 파란 동그라미라고 판단하겠지만 K를 4로 변경한다면 빨간 세모라고 판단할 것이다.
* 그렇다면 최선의 K값은 무엇일까? K값이 커질수록 분류에서 이상치(위 그림에서는 파란동그라미)의 영향이 줄어들지만 K가 계속 커진다면 분류자체를 못하게 되는 상황이 발생한다.

**python 코드를 이용한 예제**

*# 관련 라이브러리 import*

**import** **numpy** **as** **np**

**import** **matplotlib.pyplot** **as** **plt**

* 아래는 **학습데이터 셋**이다
* 카테고리는 **'과일', '단백질', '채소'** 이렇게 3가지로 구분되며, 데이터의 **단맛**과 **아삭거림**의 정도에 따라 카테고리가 정해진다
* 단맛과 아삭거림의 정도를 사용자로부터 입력을 받아, target이라는 새롭게 분류할 데이터를 생성

grape = [8, 5]

fish = [2, 3]

carrot = [7, 10]

orange = [7, 3]

celery = [3, 8]

cheese = [1, 1]

category = ['과일', '단백질', '채소', '과일', '채소', '단백질']

*# 분류대상*

dan = int(input('단맛 입력(1~10):'))

asac = int(input('아삭거림 입력(1~10):'))

target = [dan, asac]

단맛 입력(1~10):2

아삭거림 입력(1~10):7

* 아래는 데이터를 분석하기 쉬운 dataset으로 만드는 함수이다
* **분류 집단**을 np.array()를 이용하여 행렬형태로 만들고, **분류대상**과 **분류범주**를 각각 생성한다

**def** data\_set():

dataset = np.array([grape,fish,carrot,orange,celery,cheese]) *# 분류집단*

size = len(dataset)

class\_target = np.tile(target, (size, 1)) *# 분류대상*

class\_category = np.array(category) *# 분류범주*

**return** dataset, class\_target, class\_category

*# dataset 생성*

dataset, class\_target, class\_categoty = data\_set() *# data\_set()함수 호출*

* 위에서 생성한 dataset을 바탕으로 유클리드 거리 계산식을 이용하여 **분류할 대상과 분류 범주와의 거리**를 구한다.
* 그런 다음 가까운 거리에 있는 순으로 오름차순 정렬한 후 그 값에 따라 어떤 카테고리에 가까운지 분류한다

**def** classify(dataset, class\_target, class\_categoty, k):

*# 유클리드 거리 계산*

diffMat = class\_target - dataset *# 두 점의 차*

sqDiffMat = diffMat\*\*2 *# 차에 대한 제곱*

row\_sum = sqDiffMat.sum(axis=1) *# 차에 대한 제곱에 대한 합*

distance = np.sqrt(row\_sum) *# 차에 대한 제곱에 대한 합의 제곱근(최종거리)*

*# 가까운 거리 오름차순 정렬*

sortDist = distance.argsort()

*# 이웃한 k개 선정*

class\_result = {}

**for** i **in** range(k):

c = class\_categoty[sortDist[i]]

class\_result[c] = class\_result.get(c, 0) + 1

**return** class\_result

* 이제 위의 함수를 호출하여 분류하려는 대상이 어떤 카테고리에 더 많이 가까운지 확인한다 (k값을 얼마로 할지도 사용자로부터 입력받는다)

*# 함수 호출*

k = int(input('k값 입력(1~3):'))

class\_result = classify(dataset, class\_target, class\_categoty, k) *# classify()함수호출*

print(class\_result)

k값 입력(1~3):3

{'채소': 2, '단백질': 1}

* 일단 출력결과 위와 같은 분류결과가 나왔다. 즉, 내가 분류할 대상은 '채소' 카테고리에 더 많이 가까운 것을 알 수있다.
* 이렇게 만으로도 분류결과를 파악할 수 있지만, 아래처럼 출력결과도 함수로 만들어보기 쉽게 구현하였다.

*# 분류결과 출력 함수 정의*

**def** classify\_result(class\_result):

protein = fruit = vegetable = 0

**for** c **in** class\_result.keys():

**if** c == '단백질':

protein = class\_result[c]

**elif** c =='과일':

fruit = class\_result[c]

**else** :

vegetable = class\_result[c]

**if** protein > fruit **and** protein > vegetable:

result = "분류대상은 단백질 입니다."

**elif** fruit > protein **and** fruit > vegetable:

result = "분류대상은 과일 입니다"

**else**:

result = "분류대상은 채소 입니다."

**return** result

a = classify\_result(class\_result)

print(a)

분류대상은 채소 입니다.

* 또한 위의 거리를 쉽게 파악할 수 있도록 시각화도 해볼 수 있다.
* plt.scatter() 함수 이용

*# 시각화(o= 과일, +=단밸질, \*=채소*

plt.scatter(8, 5, marker='o', s = 200)

plt.scatter(2, 3, marker='+', s = 200)

plt.scatter(7, 10, marker='\*', s = 200)

plt.scatter(7, 3, marker='o', s = 200)

plt.scatter(3, 8, marker='+', s = 200)

plt.scatter(1, 1, marker='\*', s = 200)

plt.scatter(dan, asac, color='red', s = 200) *# 분류대상 -> B집단*

plt.show()

