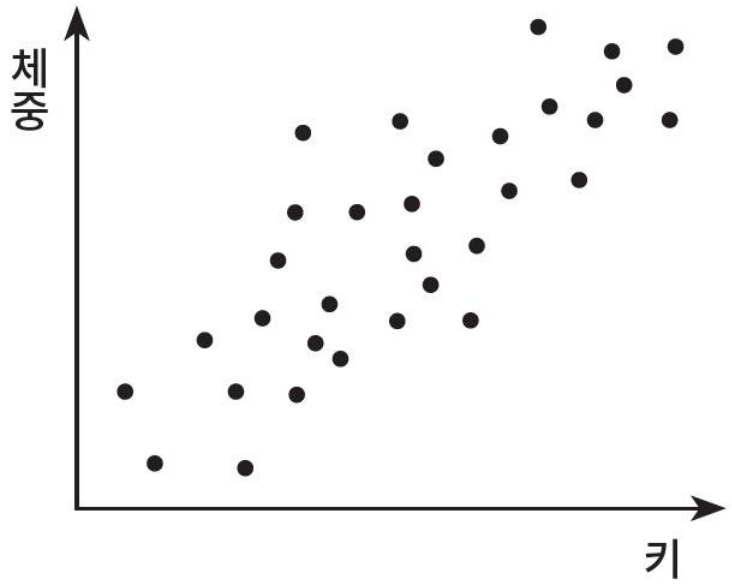
# K-means 클러스터링

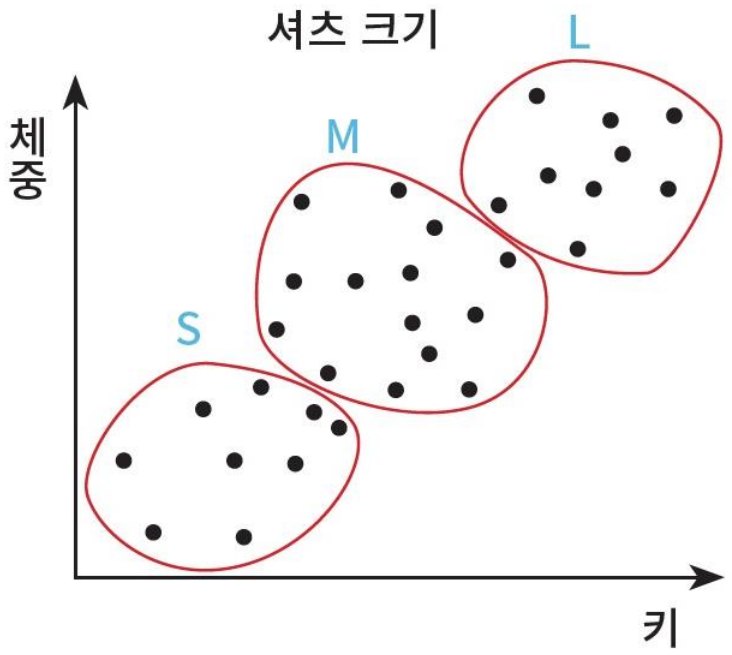
- 비지도 학습 중에서 가장 대표적인 것이 K-means 알고리즘이다.

- K-means 알고리즘(K-means algorithm)은 주어진 n개의 관측값을 k개의 클러스터로 분할하는 알고리즘으로, 관측값들은 거리가 최소인 클러스터로 분류된다.

# K-means 클러스터링

- 셔츠를 만들어서 판매하는 회사를 생각해보자. 회사는 시장에 새로운 셔츠 모델을 공개하여야 한다. 회사는 사람들의 키와 체중을 조사하여 그래프로 그려보았다고 하자.



# K-means 클러스터링 알고리즘

입력값

1. k: 클러스터 수
2. D: n개의 데이터

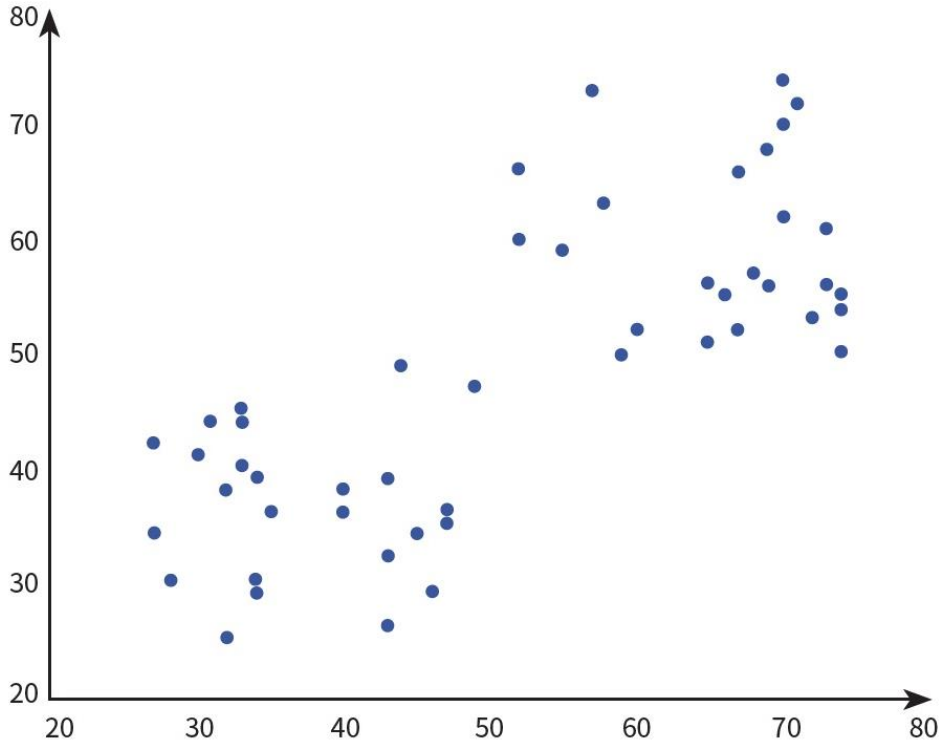
출력값: k개의 클러스터

알고리즘

1. 집합 D에서 k개의 데이터를 임의로 추출하고, 이 데이터들을 각 클러스터의 중심 (centroid) 으로 설정한다. (초기값 설정)
2. 집합 D의 각 데이터에 대해 k개의 클러스터 중심과의 거리를 계산하고, 각 데이터가 어느 중심점 (centroid)와 가장 유사도가 높은지 알아낸다. 그리고 그렇게 찾아낸 중심점으로 각 데이터들을 할당한다.
3. 클러스터의 중심점을 다시 계산한다. 즉, 2에서 재할당된 클러스터들을 기준으로 중심점을 다시 계산한다.
4. 각 데이터의 소속 클러스터가 바뀌지 않을 때까지 과정 2, 3을 반복한다.

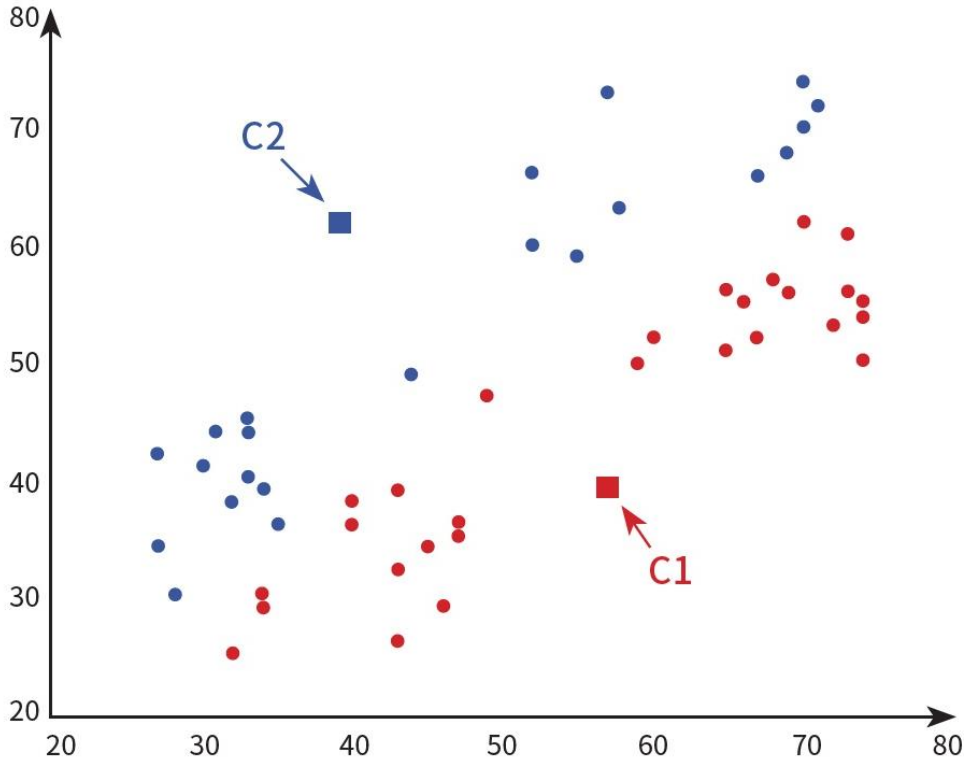
# 알고리즘 #1

- 알고리즘을 설명하기 위하여 다음과 같이 데이터들이 주어졌다고 하자. 우리는 데이터를 2개의 그룹으로 나누어야 한다. 즉 k=2이다.



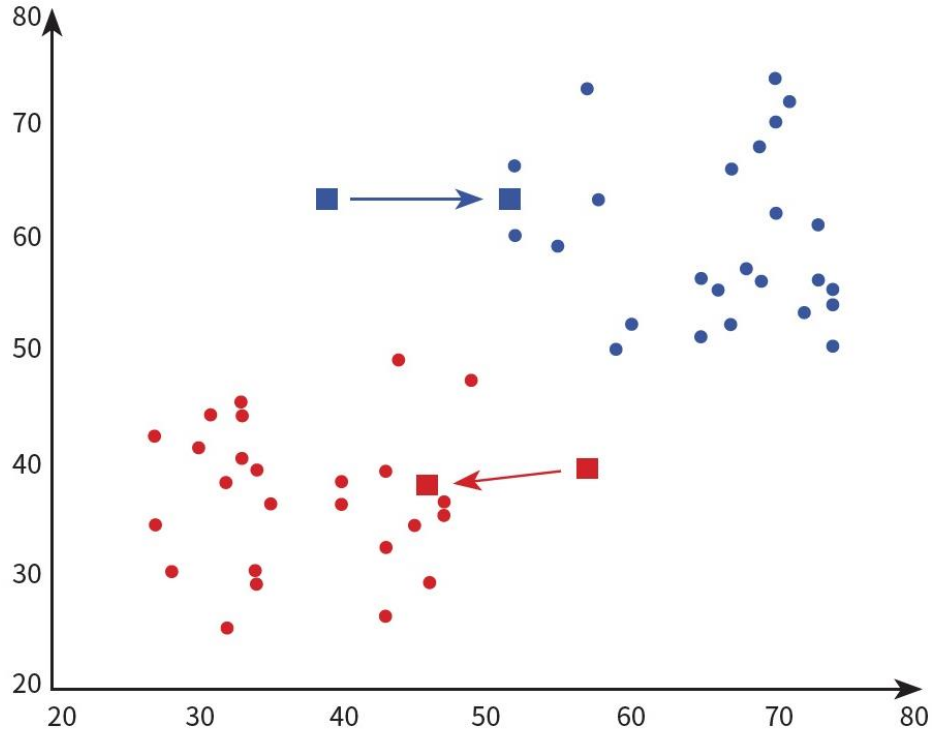
# 알고리즘 #2

- 알고리즘은 무작위로 2개의 중심점을 선택한다. 이것을 C1과 C2라고 하자.



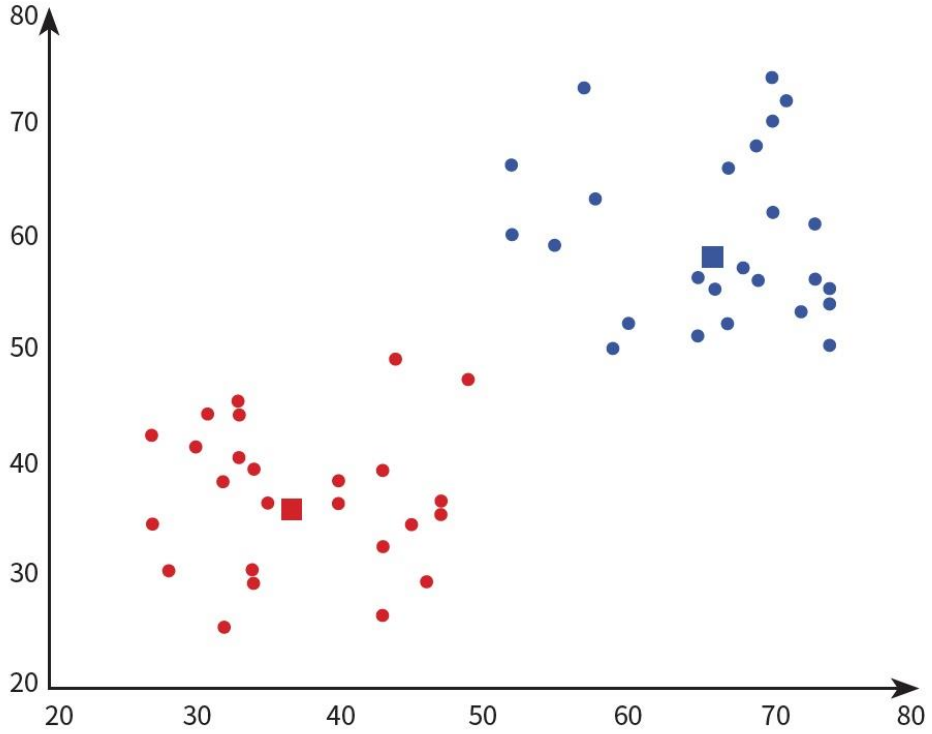
# 알고리즘 #3

- 모든 파란색 점과 빨간색 점의 평균을 따로 계산한다. 이 점이 클러스터의 새로운 중심이 된다.



# 알고리즘 #4

- 2개의 중심점의 위치가 변하지 않을 때까지 2단계와 3단계를 반복한다.



# sklearn을 이용한 K-means 클러스터링

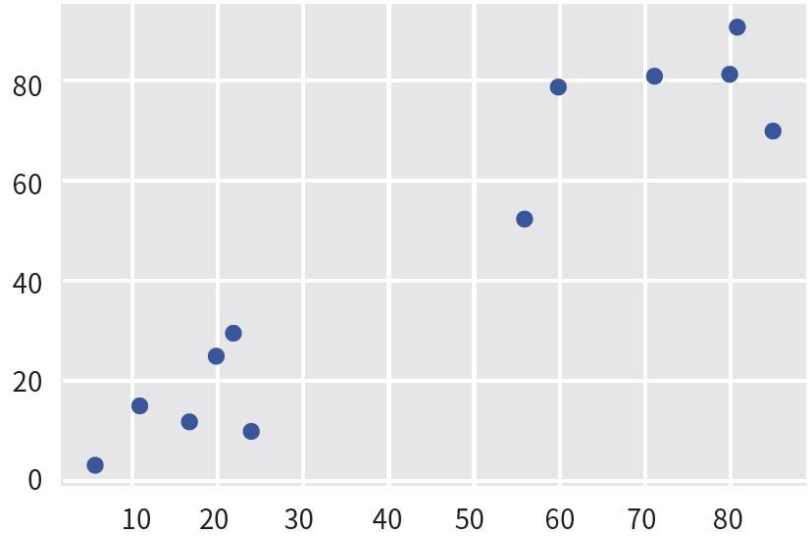
- 여기서는 sklearn 라이브러리를 이용하여 K-means 클러스터링 알고리즘을 실습해보자.

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.cluster import KMeans

X = np.array([
     [6,3], [11,15], [17,12], [24,10], [20,25], [22,30],
     [85,70], [71,81], [60,79], [56,52], [81,91], [80,81]])

plt.scatter(X[:,0],X[:,1])
```

# sklearn을 이용한 K-means 클러스터링



# sklearn을 이용한 K-means 클러스터링

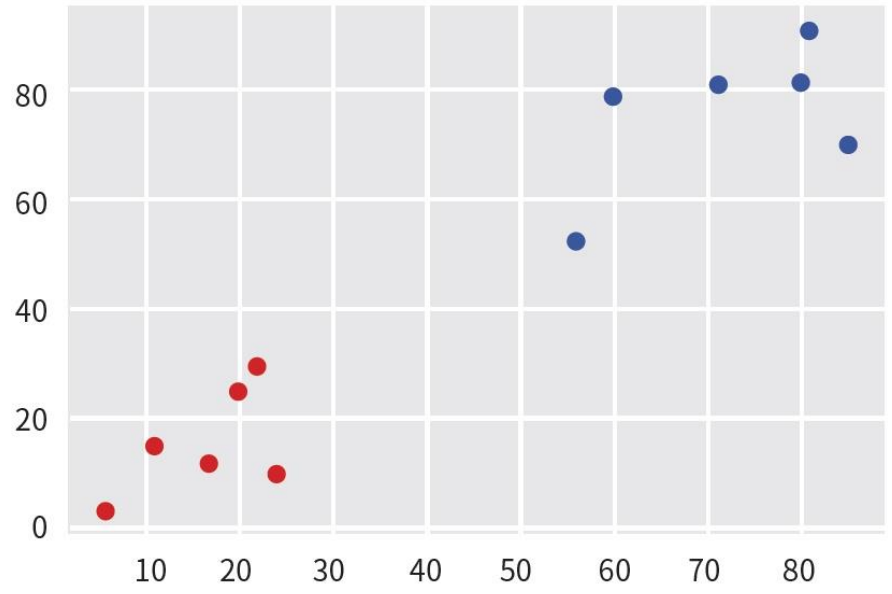
```
kmeans = KMeans(n_clusters=2)
kmeans.fit(X)

print(kmeans.cluster_centers_)
```

```
[[72.16666667 75.66666667]
 [16.66666667 15.83333333]]
```

# sklearn을 이용한 K-means 클러스터링

```
print(kmeans.labels_)
plt.scatter(X[:,0],X[:,1], c=kmeans.labels_, cmap='rainbow')
```



# Lab: K-means 알고리즘 실습

- sklearn.datasets.samples_generator의 make_blobs()하는 함수 이용

```
import matplotlib.pyplot as plt
import seaborn as sns; sns.set()
import numpy as np
from sklearn.cluster import KMeans
from sklearn.datasets.samples_generator import make_blobs

X, y_true = make_blobs(n_samples=300, centers=4,
                       cluster_std=0.60, random_state=0)
plt.scatter(X[:, 0], X[:, 1], s=50);

kmeans = KMeans(n_clusters=4)
kmeans.fit(X)
y_kmeans = kmeans.predict(X)

plt.scatter(X[:, 0], X[:, 1], c=y_kmeans, s=50, cmap='viridis')

centers = kmeans.cluster_centers_
plt.scatter(centers[:, 0], centers[:, 1], c='black', s=200, alpha=0.5);
```

# Lab: K-means 알고리즘 실습