클러스터링
(Clustering)

한국공학대학교
전자공학부
채승호 교수

기계학습의 종류

Machine Learning

Supervised Learning (지도학습)

- 레이블된 데이터로 학습
- 미래 데이터 예측
- y = f(x)

✓ Regression (회귀)
✓ Classification (분류)

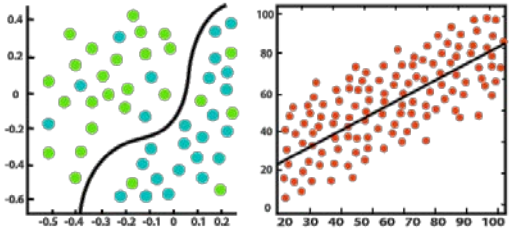
Unsupervised Learning (비지도학습)

- 레이블 없이 학습
- 데이터의 숨겨진 구조/ 특징 발견
- x~p(x)

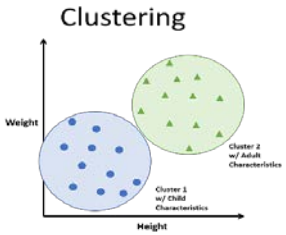
✓ Clustering
✓ Density estimation
✓ Dimentionality Reduction

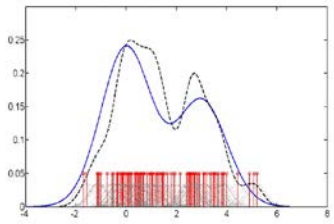
Reinforcement Learning (강화학습)

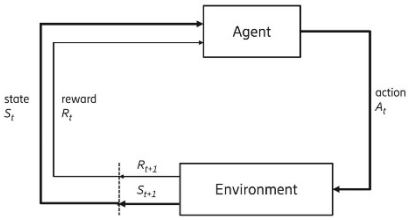
- 보상 시스템으로 학습
- 의사 결정을 위한 최적의 액션 선택



참고) 임의 추출된 데이터를 사용



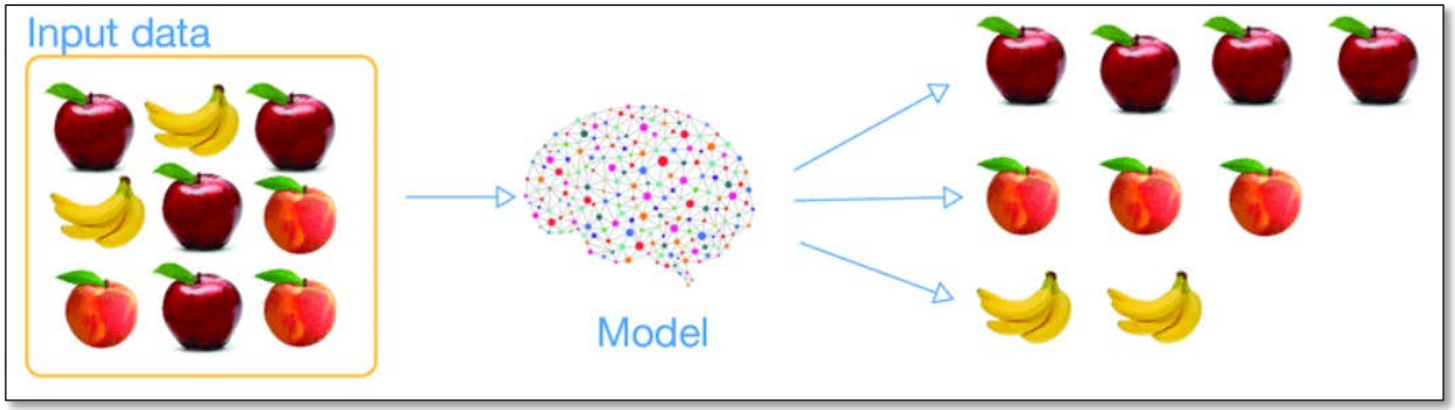




참고) 임의 추출이 아닌 전략적으로 선택된 데이터를 이용해 학습

비지도학습

- 비지도학습(Unsupervised Learning)
  - 데이터가 어떻게 구성되어 있는지(ex 데이터의 패턴)를 유추하는 기계학습 알고리즘
  - 훈련데이터는 입력값만으로 구성 (데이터 셋은 레이블링 되어 있지 않음)
  - ex) 통계적 밀도 분석, 클러스터링 등에 사용



데이터 유사성 척도

- Sample 간의 거리
  - 데이터 샘플을 하나의 벡터로 해석
  - 샘플 사이의 거리를 측정하는 객관적 방법 필요
  - $\mathrm{dist}(\mathbf{x}_i, \mathbf{x}_j)$: 두 벡터 사이의 거리
    - $\mathbf{x}_i = \left[x_{i0}\ x_{i1}\ \cdots\ x_{i(N-1)}\right]^T$
    - $\mathbf{x}_j = \left[x_{j0}\ x_{j1}\ \cdots\ x_{j(N-1)}\right]^T$
  - 서로 다른 벡터 사이의 거리가 가까울수록 두 벡터가 유사함을 전제

거리

- 민코프스키 거리 (Minkowski distance)

$$\mathrm{dist}_{mk}(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\|_p = \left(\sum_{u=0}^{N-1} |x_{iu} - x_{ju}|^p\right)^{\frac{1}{p}} \quad \text{(p-norm)}$$

  - $p = 2$일 때, 일반적인 유클리디안(Euclidean distance) 거리와 같아짐

$$\mathrm{dist}_{ed}(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\|_2 = \sqrt{\sum_{u=0}^{N-1} |x_{iu} - x_{ju}|^2}$$

  - $p = 1$일 때, 맨해튼 거리(Manhattan dist.) 또는 격자 거리(grid dist.)라고 함

$$\mathrm{dist}_{max}(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\|_1 = \sum_{u=0}^{N-1} |x_{iu} - x_{ju}|$$

  - 수치적 속성의 거리 측정에 사용 가능

K-평균 클러스터링

- K-평균 클러스터링 알고리즘 (K-means clustering algorithm)
  - 훈련 샘플을 K개의 클러스터로 분류
  - 분류한 뒤 각 클러스터에 대한 평균 오차를 최소화
    - 각 클러스터에 속한 샘플과 각 클러스터에 속한 모든 샘플들의 평균과의 오차의 합

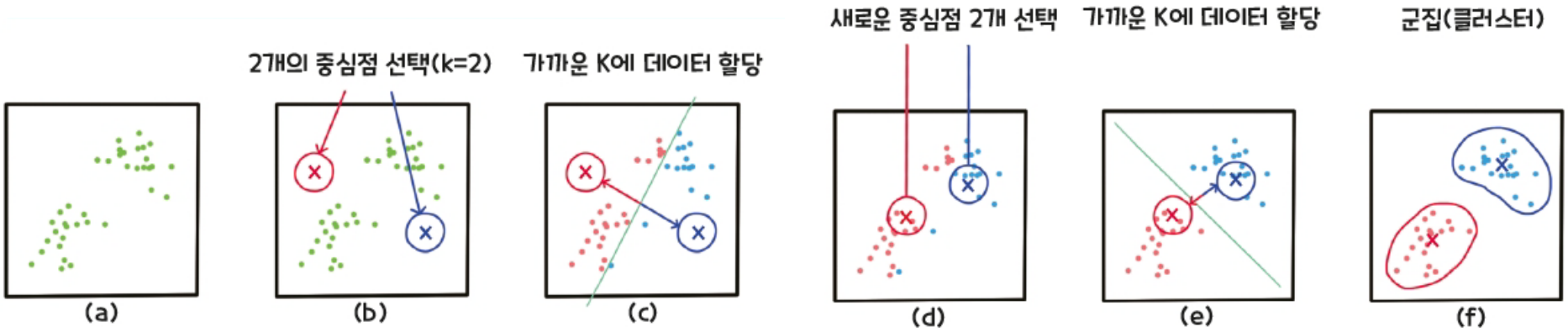
$$E = \sum_{i=0}^{K-1} \sum_{x \in C_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|_2 \ , \qquad \boldsymbol{\mu} = \frac{1}{|C_i|} \sum_{x \in C_i} \mathbf{x}_i$$

    - 클러스터 내의 샘플들이 클러스터 평균 벡터를 중심으로 밀집된 정도를 나타냄
    - E값이 작을수록 클러스터 내 샘플들의 유사도가 높아짐
    - 모든 클러스터 구조에 대해 E를 구하고 최적화 → 불가능

      → 반복적 최적화 알고리즘 사용

K-평균 군집화(K-means Clustering)

- 간략한 프로세스 설명



- (a) : 일반적인 데이터 분포입니다.
- (b) : 데이터셋에서 K개의 중심점을 임의로 지정하는데, 여기에서는 K=2의 값으로 중심점 2개를 설정했습니다.
- (c) : 데이터들을 가장 가까운 중심점에 할당합니다.
- (d) : (c)에서 할당된 결과를 바탕으로 중심점을 새롭게 지정합니다.
- (e) : 중심점이 더 이상 변하지 않을 때까지 (c)~(d) 과정을 반복합니다.
- (f) : 최종적인 군집이 형성됩니다.

몇 개의 군집 (k)으로 나눌지 설정하는 것이 핵심

K-평균 군집화(K-means Clustering)

- 알고리즘

입력: 샘플 세트 $D = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m\}$
클러스터 k

과정:

1: D에서 랜덤으로 k개의 샘플을 선택해 초기 평균 벡터mean vector $\{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_k\}$로 정한다
2: **repeat**
3: $C_i = \varnothing \ (1 \leqslant i \leqslant k)$로 설정한다
4: **for** $j = 1, 2, \ldots, m$ **do**
5: 샘플 $\boldsymbol{x}_j$와 각 평균 벡터 $\boldsymbol{\mu}_i \ (1 \leqslant i \leqslant k)$ 간의 거리를 계산한다: $d_{ji} = \|\boldsymbol{x}_j - \boldsymbol{\mu}_i\|_2$
6: 거리가 가장 가까운 평균 벡터를 기반으로 $\boldsymbol{x}_j$의 클러스터 레이블을 정한다: $\lambda_j = \operatorname{argmin}_{i \in \{1,2,\ldots,k\}} d_{ji}$
7: 샘플 $\boldsymbol{x}_j$를 상응하는 클러스터에 포함한다 $C_{\lambda_j} = C_{\lambda_j} \bigcup \{\boldsymbol{x}_j\}$
8: **end for**
9: **for** $i = 1, 2, \ldots, k$ **do**
10: 새로운 평균 벡터 $\boldsymbol{\mu}'_i = \frac{1}{|C_i|} \sum_{\boldsymbol{x} \in C_i} \boldsymbol{x}$를 계산한다
11: **if** $\boldsymbol{\mu}'_i \neq \boldsymbol{\mu}_i$ **then**
12: 평균 벡터 $\boldsymbol{\mu}_i$를 $\boldsymbol{\mu}'_i$로 갱신한다
13: **else**
14: 현재 평균 벡터를 변하지 않도록 보존한다
15: **end if**
16: **end for**
17: **until** 평균 벡터가 갱신되지 않을 때까지

출력: 클러스터 분할 $C = \{C_1, C_2, \ldots, C_k\}$

K-평균 군집화(K-means Clustering)

- $K = 3$ 예시

입력: 샘플 세트 $D = \{x_1, x_2, \ldots, x_m\}$
클러스터 k
과정:
1: D에서 랜덤으로 k개의 샘플을 선택해 초기 평균 벡터mean vector $\{\mu_1, \mu_2, \ldots, \mu_k\}$로 정한다
2: repeat
3: $C_i = \varnothing$ $(1 \leqslant i \leqslant k)$로 설정한다
4: for $j = 1, 2, \ldots, m$ do
5: 샘플 x_j와 각 평균 벡터 μ_i $(1 \leqslant i \leqslant k)$ 간의 거리를 계산한다: $d_{ji} = \|x_j - \mu_i\|_2$
6: 거리가 가장 가까운 평균 벡터를 기반으로 x_j의 클러스터 레이블을 정한다: $\lambda_j = \text{argmin}_{i \in \{1,2,\ldots,k\}} d_{ji}$
7: 샘플 x_j를 상응하는 클러스터에 포함한다 $C_{\lambda_j} = C_{\lambda_j} \bigcup \{x_j\}$
8: end for
9: for $i = 1, 2, \ldots, k$ do
10: 새로운 평균 벡터 $\mu'_i = \frac{1}{|C_i|} \sum_{x \in C_i} x$를 계산한다
11: if $\mu'_i \neq \mu_i$ then
12: 평균 벡터 μ_i를 μ'_i로 갱신한다
13: else
14: 현재 평균 벡터를 변하지 않도록 보존한다
15: end if
16: end for
17: until 평균 벡터가 갱신되지 않을 때까지
출력: 클러스터 분할 $C = \{C_1, C_2, \ldots, C_k\}$

번호	밀도	당도	번호	밀도	당도	번호	밀도	당도
1	0.697	0.460	11	0.245	0.057	21	0.748	0.232
2	0.774	0.376	12	0.343	0.099	22	0.714	0.346
3	0.634	0.264	13	0.639	0.161	23	0.483	0.312
4	0.608	0.318	14	0.657	0.198	24	0.478	0.437
5	0.556	0.215	15	0.360	0.370	25	0.525	0.369
6	0.403	0.237	16	0.593	0.042	26	0.751	0.489
7	0.481	0.149	17	0.719	0.103	27	0.532	0.472
8	0.437	0.211	18	0.359	0.188	28	0.473	0.376
9	0.666	0.091	19	0.339	0.241	29	0.725	0.445
10	0.243	0.267	20	0.282	0.257	30	0.446	0.459

1단계: 주어진 클러스터 개수에 대해 랜덤하게 초기값 설정 (초기 평균 벡터; centroid)

3개 클러스터의 초기 평균 벡터를 6, 12, 24번 샘플로 랜덤 선택

$$\boldsymbol{\mu}_1 = (0.403, 0.237)$$
$$\boldsymbol{\mu}_2 = (0.343, 0.099)$$
$$\boldsymbol{\mu}_3 = (0.478, 0.437)$$

K-평균 군집화(K-means Clustering)

입력: 샘플 세트 $D = \{x_1, x_2, \ldots, x_m\}$
클러스터 k

과정:

1: D에서 랜덤으로 k개의 샘플을 선택해 초기 평균 벡터mean vector $\{\mu_1, \mu_2, \ldots, \mu_k\}$로 정한다
2: **repeat**
3: $C_i = \varnothing \ (1 \leqslant i \leqslant k)$로 설정한다
4: **for** $j = 1, 2, \ldots, m$ **do**
5: 샘플 x_j와 각 평균 벡터 $\mu_i \ (1 \leqslant i \leqslant k)$ 간의 거리를 계산한다: $d_{ji} = \|x_j - \mu_i\|_2$
6: 거리가 가장 가까운 평균 벡터를 기반으로 x_j의 클러스터 레이블을 정한다: $\lambda_j = \text{argmin}_{i \in \{1,2,\ldots,k\}} d_{ji}$
7: 샘플 x_j를 상응하는 클러스터에 포함한다 $C_{\lambda_j} = C_{\lambda_j} \bigcup \{x_j\}$
8: **end for**
9: **for** $i = 1, 2, \ldots, k$ **do**
10: 새로운 평균 벡터 $\mu'_i = \frac{1}{|C_i|} \sum_{x \in C_i} x$를 계산한다
11: **if** $\mu'_i \neq \mu_i$ **then**
12: 평균 벡터 μ_i를 μ'_i로 갱신한다
13: **else**
14: 현재 평균 벡터를 변하지 않도록 보존한다
15: **end if**
16: **end for**
17: **until** 평균 벡터가 갱신되지 않을 때까지

출력: 클러스터 분할 $C = \{C_1, C_2, \ldots, C_k\}$

번호	밀도	당도	번호	밀도	당도	번호	밀도	당도
1	0.697	0.460	11	0.245	0.057	21	0.748	0.232
2	0.774	0.376	12	0.343	0.099	22	0.714	0.346
3	0.634	0.264	13	0.639	0.161	23	0.483	0.312
4	0.608	0.318	14	0.657	0.198	24	0.478	0.437
5	0.556	0.215	15	0.360	0.370	25	0.525	0.369
6	0.403	0.237	16	0.593	0.042	26	0.751	0.489
7	0.481	0.149	17	0.719	0.103	27	0.532	0.472
8	0.437	0.211	18	0.359	0.188	28	0.473	0.376
9	0.666	0.091	19	0.339	0.241	29	0.725	0.445
10	0.243	0.267	20	0.282	0.257	30	0.446	0.459

2단계: 모든 샘플에 대해 각 클러스터 평균 벡터와의 거리를 계산하여 가장 가까운 평균 벡터의 클러스터 소속으로 분류

1번 샘플 (0.697, 0.460)과 각 평균 벡터와의 거리

$$\|\mathbf{x}_1 - \boldsymbol{\mu}_1\|_2 = 0.369$$

$$\|\mathbf{x}_1 - \boldsymbol{\mu}_2\|_2 = 0.506$$

$$\|\mathbf{x}_1 - \boldsymbol{\mu}_3\|_2 = 0.220$$

1번 샘플은 3번 클러스터 소속

K-평균 군집화(K-means Clustering)

입력: 샘플 세트 $D = \{x_1, x_2, \ldots, x_m\}$
클러스터 k

과정:

1: D에서 랜덤으로 k개의 샘플을 선택해 초기 평균 벡터mean vector $\{\mu_1, \mu_2, \ldots, \mu_k\}$로 정한다
2: **repeat**
3: $C_i = \varnothing \ (1 \leqslant i \leqslant k)$로 설정한다
4: **for** $j = 1, 2, \ldots, m$ **do**
5: 샘플 x_j와 각 평균 벡터 $\mu_i \ (1 \leqslant i \leqslant k)$ 간의 거리를 계산한다: $d_{ji} = \|x_j - \mu_i\|_2$
6: 거리가 가장 가까운 평균 벡터를 기반으로 x_j의 클러스터 레이블을 정한다: $\lambda_j = \text{argmin}_{i \in \{1,2,\ldots,k\}} d_{ji}$
7: 샘플 x_j를 상응하는 클러스터에 포함한다 $C_{\lambda_j} = C_{\lambda_j} \bigcup \{x_j\}$
8: **end for**
9: **for** $i = 1, 2, \ldots, k$ **do**
10: 새로운 평균 벡터 $\mu'_i = \frac{1}{|C_i|} \sum_{x \in C_i} x$를 계산한다
11: **if** $\mu'_i \neq \mu_i$ **then**
12: 평균 벡터 μ_i를 μ'_i로 갱신한다
13: **else**
14: 현재 평균 벡터를 변하지 않도록 보존한다
15: **end if**
16: **end for**
17: **until** 평균 벡터가 갱신되지 않을 때까지

출력: 클러스터 분할 $C = \{C_1, C_2, \ldots, C_k\}$

번호	밀도	당도	번호	밀도	당도	번호	밀도	당도
1	0.697	0.460	11	0.245	0.057	21	0.748	0.232
2	0.774	0.376	12	0.343	0.099	22	0.714	0.346
3	0.634	0.264	13	0.639	0.161	23	0.483	0.312
4	0.608	0.318	14	0.657	0.198	24	0.478	0.437
5	0.556	0.215	15	0.360	0.370	25	0.525	0.369
6	0.403	0.237	16	0.593	0.042	26	0.751	0.489
7	0.481	0.149	17	0.719	0.103	27	0.532	0.472
8	0.437	0.211	18	0.359	0.188	28	0.473	0.376
9	0.666	0.091	19	0.339	0.241	29	0.725	0.445
10	0.243	0.267	20	0.282	0.257	30	0.446	0.459

2단계: 모든 샘플에 대해 각 클러스터 평균 벡터와의 거리를 계산하여 가장 가까운 평균 벡터의 클러스터 소속으로 분류

$$C_1 = \{\mathbf{x}_3, \mathbf{x}_5, \mathbf{x}_6, \mathbf{x}_7, \mathbf{x}_8, \mathbf{x}_9, \mathbf{x}_{10}, \mathbf{x}_{13}, \mathbf{x}_{14}, \mathbf{x}_{17}, \mathbf{x}_{18}, \mathbf{x}_{19}, \mathbf{x}_{20}, \mathbf{x}_{23}\}$$

$$C_2 = \{\mathbf{x}_{11}, \mathbf{x}_{12}, \mathbf{x}_{16}\}$$

$$C_3 = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_4, \mathbf{x}_{15}, \mathbf{x}_{21}, \mathbf{x}_{22}, \mathbf{x}_{24}, \mathbf{x}_{25}, \mathbf{x}_{26}, \mathbf{x}_{27}, \mathbf{x}_{28}, \mathbf{x}_{29}, \mathbf{x}_{30}\}$$

첫번째 클러스터 완성!

K-평균 군집화(K-means Clustering)

입력: 샘플 세트 $D = \{x_1, x_2, \ldots, x_m\}$
클러스터 k

과정:

1: D에서 랜덤으로 k개의 샘플을 선택해 초기 평균 벡터mean vector $\{\mu_1, \mu_2, \ldots, \mu_k\}$로 정한다
2: **repeat**
3: $C_i = \varnothing \ (1 \leqslant i \leqslant k)$로 설정한다
4: **for** $j = 1, 2, \ldots, m$ **do**
5: 샘플 x_j와 각 평균 벡터 $\mu_i \ (1 \leqslant i \leqslant k)$ 간의 거리를 계산한다: $d_{ji} = \|x_j - \mu_i\|_2$
6: 거리가 가장 가까운 평균 벡터를 기반으로 x_j의 클러스터 레이블을 정한다: $\lambda_j = \text{argmin}_{i \in \{1,2,\ldots,k\}} d_{ji}$
7: 샘플 x_j를 상응하는 클러스터에 포함한다 $C_{\lambda_j} = C_{\lambda_j} \bigcup \{x_j\}$
8: **end for**
9: **for** $i = 1, 2, \ldots, k$ **do**
10: 새로운 평균 벡터 $\mu'_i = \frac{1}{|C_i|} \sum_{x \in C_i} x$를 계산한다
11: **if** $\mu'_i \neq \mu_i$ **then**
12: 평균 벡터 μ_i를 μ'_i로 갱신한다
13: **else**
14: 현재 평균 벡터를 변하지 않도록 보존한다
15: **end if**
16: **end for**
17: **until** 평균 벡터가 갱신되지 않을 때까지

출력: 클러스터 분할 $C = \{C_1, C_2, \ldots, C_k\}$

번호	밀도	당도	번호	밀도	당도	번호	밀도	당도
1	0.697	0.460	11	0.245	0.057	21	0.748	0.232
2	0.774	0.376	12	0.343	0.099	22	0.714	0.346
3	0.634	0.264	13	0.639	0.161	23	0.483	0.312
4	0.608	0.318	14	0.657	0.198	24	0.478	0.437
5	0.556	0.215	15	0.360	0.370	25	0.525	0.369
6	0.403	0.237	16	0.593	0.042	26	0.751	0.489
7	0.481	0.149	17	0.719	0.103	27	0.532	0.472
8	0.437	0.211	18	0.359	0.188	28	0.473	0.376
9	0.666	0.091	19	0.339	0.241	29	0.725	0.445
10	0.243	0.267	20	0.282	0.257	30	0.446	0.459

3단계: 완성된 각 클러스터의 새로운 평균 벡터 계산 후 2단계 반복

$$C_1 = \{\mathbf{x}_3, \mathbf{x}_5, \mathbf{x}_6, \mathbf{x}_7, \mathbf{x}_8, \mathbf{x}_9, \mathbf{x}_{10}, \mathbf{x}_{13}, \mathbf{x}_{14}, \mathbf{x}_{17}, \mathbf{x}_{18}, \mathbf{x}_{19}, \mathbf{x}_{20}, \mathbf{x}_{23}\}$$

$$\boldsymbol{\mu}'_1 = \frac{1}{|C_1|} \sum_{\mathbf{x} \in C_1} \mathbf{x} = (0.493, 0.207)$$

$$\boldsymbol{\mu}'_2 = \frac{1}{|C_2|} \sum_{\mathbf{x} \in C_2} \mathbf{x} = (0.394, 0.066)$$

$$\boldsymbol{\mu}'_3 = \frac{1}{|C_3|} \sum_{\mathbf{x} \in C_3} \mathbf{x} = (0.602, 0.396)$$

K-평균 군집화(K-means Clustering)

입력: 샘플 세트 $D = \{x_1, x_2, \ldots, x_m\}$
클러스터 k

과정:

1: D에서 랜덤으로 k개의 샘플을 선택해 초기 평균 벡터mean vector $\{\mu_1, \mu_2, \ldots, \mu_k\}$로 정한다
2: **repeat**
3: $C_i = \varnothing\ (1 \leqslant i \leqslant k)$로 설정한다
4: **for** $j = 1, 2, \ldots, m$ **do**
5: 샘플 x_j와 각 평균 벡터 $\mu_i\ (1 \leqslant i \leqslant k)$ 간의 거리를 계산한다: $d_{ji} = \|x_j - \mu_i\|_2$
6: 거리가 가장 가까운 평균 벡터를 기반으로 x_j의 클러스터 레이블을 정한다: $\lambda_j = \mathrm{argmin}_{i \in \{1,2,\ldots,k\}} d_{ji}$
7: 샘플 x_j를 상응하는 클러스터에 포함한다 $C_{\lambda_j} = C_{\lambda_j} \bigcup \{x_j\}$
8: **end for**
9: **for** $i = 1, 2, \ldots, k$ **do**
10: 새로운 평균 벡터 $\mu'_i = \frac{1}{|C_i|} \sum_{x \in C_i} x$를 계산한다
11: **if** $\mu'_i \neq \mu_i$ **then**
12: 평균 벡터 μ_i를 μ'_i로 갱신한다
13: **else**
14: 현재 평균 벡터를 변하지 않도록 보존한다
15: **end if**
16: **end for**
17: **until** 평균 벡터가 갱신되지 않을 때까지

출력: 클러스터 분할 $C = \{C_1, C_2, \ldots, C_k\}$

번호	밀도	당도	번호	밀도	당도	번호	밀도	당도
1	0.697	0.460	11	0.245	0.057	21	0.748	0.232
2	0.774	0.376	12	0.343	0.099	22	0.714	0.346
3	0.634	0.264	13	0.639	0.161	23	0.483	0.312
4	0.608	0.318	14	0.657	0.198	24	0.478	0.437
5	0.556	0.215	15	0.360	0.370	25	0.525	0.369
6	0.403	0.237	16	0.593	0.042	26	0.751	0.489
7	0.481	0.149	17	0.719	0.103	27	0.532	0.472
8	0.437	0.211	18	0.359	0.188	28	0.473	0.376
9	0.666	0.091	19	0.339	0.241	29	0.725	0.445
10	0.243	0.267	20	0.282	0.257	30	0.446	0.459

3단계: 완성된 각 클러스터의 새로운 평균 벡터 계산 후 2단계 반복

$$\boldsymbol{\mu}'_1 = (0.493, 0.207)$$

$$\boldsymbol{\mu}'_2 = (0.394, 0.066)$$

$$\boldsymbol{\mu}'_3 = (0.602, 0.396)$$

$$C_1 = \{\mathbf{x}_5, \mathbf{x}_6, \mathbf{x}_7, \mathbf{x}_8, \mathbf{x}_9, \mathbf{x}_{10}, \mathbf{x}_{13}, \mathbf{x}_{14}, \mathbf{x}_{15}, \mathbf{x}_{16}, \mathbf{x}_{17}, \mathbf{x}_{19}, \mathbf{x}_{20}, \mathbf{x}_{23}\}$$

$$C_2 = \{\mathbf{x}_{10}, \mathbf{x}_{11}, \mathbf{x}_{12}, \mathbf{x}_{18}\}$$

$$C_3 = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_{21}, \mathbf{x}_{22}, \mathbf{x}_{24}, \mathbf{x}_{25}, \mathbf{x}_{26}, \mathbf{x}_{27}, \mathbf{x}_{28}, \mathbf{x}_{29}, \mathbf{x}_{30}\}$$

K-평균 군집화(K-means Clustering)

입력: 샘플 세트 $D = \{x_1, x_2, \ldots, x_m\}$
클러스터 k

과정:

1: D에서 랜덤으로 k개의 샘플을 선택해 초기 평균 벡터mean vector $\{\mu_1, \mu_2, \ldots, \mu_k\}$로 정한다
2: **repeat**
3: $C_i = \varnothing \ (1 \leqslant i \leqslant k)$로 설정한다
4: **for** $j = 1, 2, \ldots, m$ **do**
5: 샘플 x_j와 각 평균 벡터 $\mu_i \ (1 \leqslant i \leqslant k)$ 간의 거리를 계산한다: $d_{ji} = \|x_j - \mu_i\|_2$
6: 거리가 가장 가까운 평균 벡터를 기반으로 x_j의 클러스터 레이블을 정한다: $\lambda_j = \mathrm{argmin}_{i \in \{1,2,\ldots,k\}} d_{ji}$
7: 샘플 x_j를 상응하는 클러스터에 포함한다 $C_{\lambda_j} = C_{\lambda_j} \bigcup \{x_j\}$
8: **end for**
9: **for** $i = 1, 2, \ldots, k$ **do**
10: 새로운 평균 벡터 $\mu'_i = \frac{1}{|C_i|} \sum_{x \in C_i} x$를 계산한다
11: **if** $\mu'_i \neq \mu_i$ **then**
12: 평균 벡터 μ_i를 μ'_i로 갱신한다
13: **else**
14: 현재 평균 벡터를 변하지 않도록 보존한다
15: **end if**
16: **end for**
17: **until** 평균 벡터가 갱신되지 않을 때까지

출력: 클러스터 분할 $C = \{C_1, C_2, \ldots, C_k\}$

번호	밀도	당도	번호	밀도	당도	번호	밀도	당도
1	0.697	0.460	11	0.245	0.057	21	0.748	0.232
2	0.774	0.376	12	0.343	0.099	22	0.714	0.346
3	0.634	0.264	13	0.639	0.161	23	0.483	0.312
4	0.608	0.318	14	0.657	0.198	24	0.478	0.437
5	0.556	0.215	15	0.360	0.370	25	0.525	0.369
6	0.403	0.237	16	0.593	0.042	26	0.751	0.489
7	0.481	0.149	17	0.719	0.103	27	0.532	0.472
8	0.437	0.211	18	0.359	0.188	28	0.473	0.376
9	0.666	0.091	19	0.339	0.241	29	0.725	0.445
10	0.243	0.267	20	0.282	0.257	30	0.446	0.459

4단계: 클러스터의 변화가 더 이상 없을 경우 종료

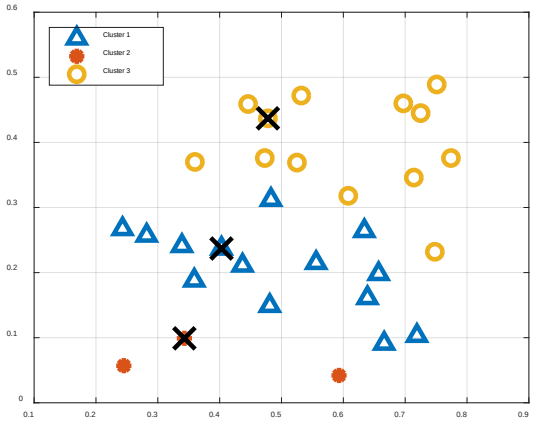
4회 반복후 최종 클러스터링 결과

$$C_1 = \{\mathbf{x}_5, \mathbf{x}_6, \mathbf{x}_7, \mathbf{x}_9, \mathbf{x}_{13}, \mathbf{x}_{14}, \mathbf{x}_{16}, \mathbf{x}_{17}, \mathbf{x}_{21}\}$$

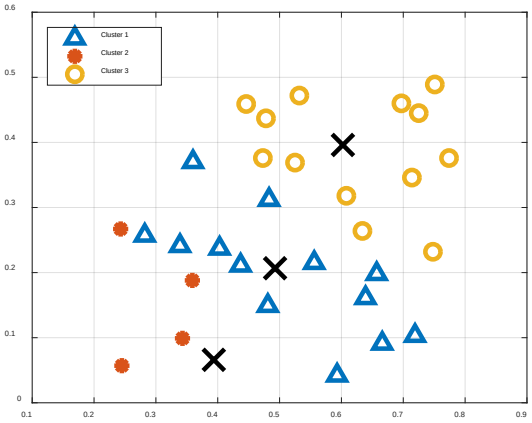
$$C_2 = \{\mathbf{x}_6, \mathbf{x}_8, \mathbf{x}_{10}, \mathbf{x}_{11}, \mathbf{x}_{12}, \mathbf{x}_{15}, \mathbf{x}_{18}, \mathbf{x}_{19}, \mathbf{x}_{20}\}$$

$$C_3 = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_4, \mathbf{x}_{22}, \mathbf{x}_{23}, \mathbf{x}_{24}, \mathbf{x}_{25}, \mathbf{x}_{26}, \mathbf{x}_{27}, \mathbf{x}_{28}, \mathbf{x}_{29}, \mathbf{x}_{30}\}$$

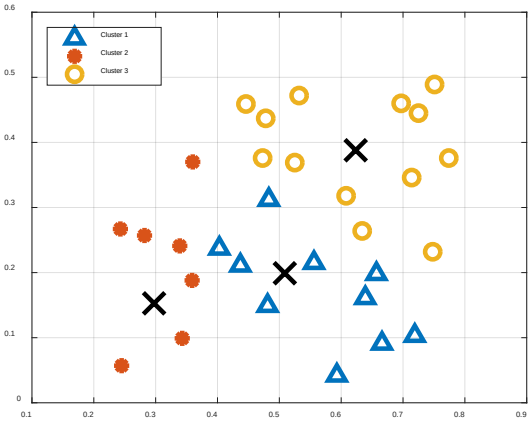
K-평균 군집화(K-means Clustering)



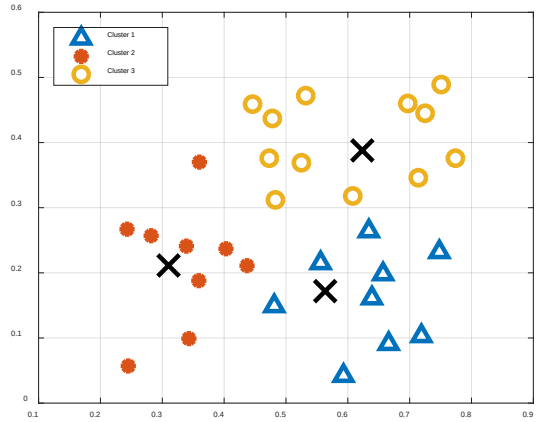
1회



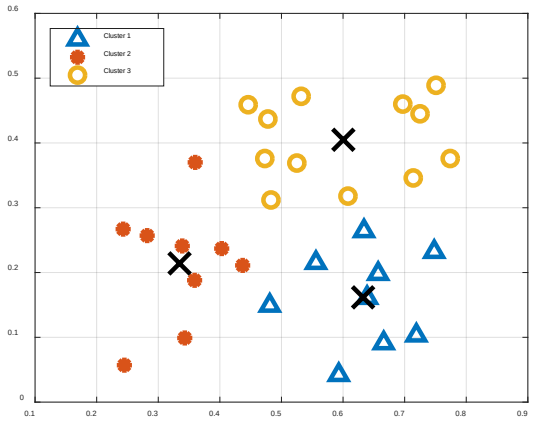
2회



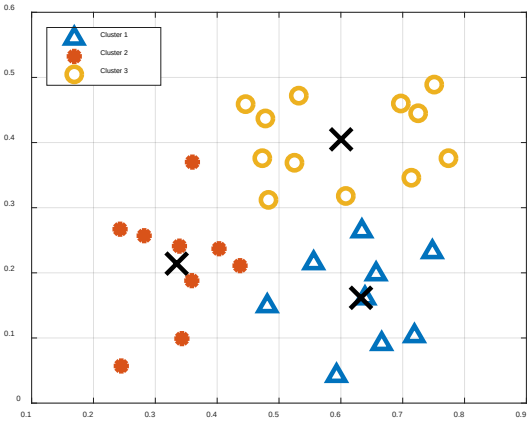
3회



4회



5회



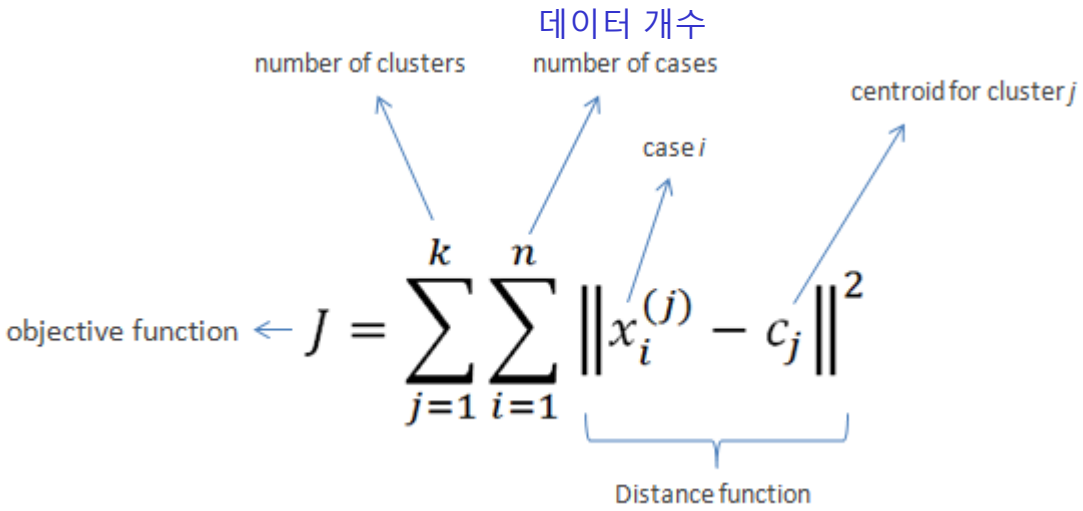
6회

K-평균 군집화(K-means Clustering)

- K-평균 군집화 알고리즘의 장단점
  - 장점
    - 쉽고 빠르게 연산이 가능 (간단한 알고리즘)
    - 대용량 데이터에 적합
  - 단점
    - k값 & 시작 중심점(centroid) 임의 지정 → 시작 중심점을 어떻게 정하느냐에 따라 cluster 결과가 민감해짐
      - 한 번에 k개의 중심점을 랜덤하게 생성하므로 각 중심점 사이의 거리가 짧으면 분류가 제대로 이루어지지 않음
    - local minimum으로 수렴
    - Outlier에 민감함 → 굉장히 멀리 떨어진 몇 개의 점도 반영하여 centroid 결정
    - 원(혹은 구)형의 cluster만 찾을 수 있음 → 원형이 아닌 cluster의 경우 정확한 결과를 도출할 수 없음

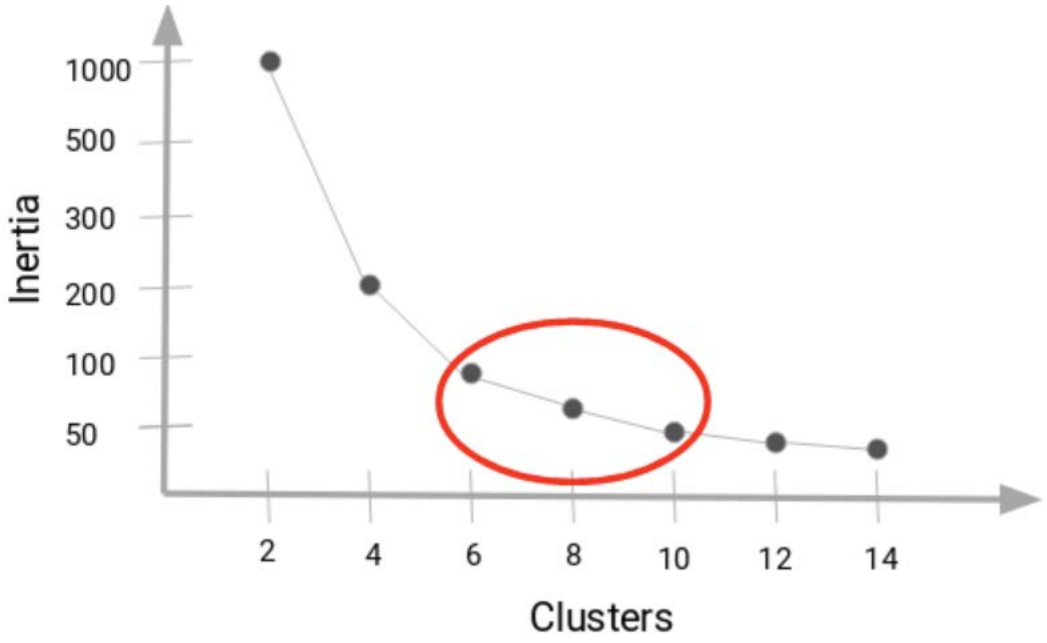
Elbow Method

- KMeans의 K 값을 정하는 기준
  - Elbow Method, Silhouette Score, ...
- Elbow Method
  - 이너셔(Inertia): 각 샘플과 가장 가까운 센트로이드 사이의 거리 제곱들의 합
    - 클러스터에 속한 샘플이 얼마나 가깝게 모여있는지를 나타내는 값
    - K-Means 클러스터링 성능 지표



Elbow Method

- 클러스터 수와 이너셔는 반비례 관계
- 이너셔가 급격하게 감소하다가 어느 지점에서 완만하게 감소
  - 해당 지점을 엘보우(Elbow)라고 함
- 이를 바탕으로 모델의 적정 클러스터 개수 지정 가능



실습

- 실습데이터 (파일명: MallCustomer.csv)
  - 총 200개의 데이터
  - id : 고윳값
  - gender : 성별
  - income : 소득
  - spendig score : 쇼핑몰에서 부여한 고객의 점수 (소비금액 및 행동 패턴 기반)

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
195	196	Female	35	120	79
196	197	Female	45	126	28
197	198	Male	32	126	74
198	199	Male	32	137	18
199	200	Male	30	137	83

실습

실습 #1 - 1

- Scikit-learn 라이브러리를 사용하여 K-Means 클러스터링을 구현 하시오.
  - 사이킷런은 파이썬에서 머신러닝 분석시 유용하게 사용할 수 있는 라이브러리

```python
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.preprocessing import MinMaxScaler # data scaling 사용
from sklearn.cluster import KMeans   # KMeans Clustering 함수

import os
current_path = os.path.dirname(os.path.abspath(__file__))

df = pd.read_csv('Mall_Customers.csv')
print(df)

data = df[['Annual Income (k$)', 'Spending Score (1-100)']]

k = 3 # clustering 개수

# 그룹 수, k-means++ 방식, random_state(학습결과 동일성을 위한 난수고정)
model = KMeans(n_clusters = k, init = 'k-means++', random_state = 10)

# 클러스터 중심 계산 및 각 샘플에 대한 클러스터 인덱스 예측
df['cluster'] = model.fit_predict(data)

# 클러스터 최종 중심값 도출 속성
final_centroid = model.cluster_centers_
print(final_centroid)

plt.figure(figsize = (8,8))
for i in range(k):
    plt.scatter(df.loc[df['cluster'] == i, 'Annual Income (k$)'], df.loc[df['cluster'] == i, 'Spending Score (1-100)'], label = 'cluster' + str(i))

plt.scatter(final_centroid[:,0], final_centroid[:,1], s = 50, c = 'violet', marker = 'x', label = 'Centroids')
plt.legend()
plt.title(f'K = {k} results', size = 15)
plt.xlabel('Annual Income', size = 12)
plt.ylabel('Spending Score', size = 12)
plt.show()
```

중심점 랜덤 지정 : init = ‘random’ → Classical K-Means 방법

.loc: 행 데이터 가져오기

참고)

- Classical K-Means
  - 1) K개의 임의의 중심점(centroid)을 배치
  - 2) 각 데이터들을 가장 가까운 중심점으로 할당
  - 3) 군집으로 지정된 데이터들을 기반으로 해당 군집의 중심점 업데이트
  - 4) 수렴 될 때까지 (즉, 더이상 중심점 업데이트 X) 2번, 3번 단계 반복
- K-Means ++ 클러스터링의 원리
  - K-Means 첫번째 단계에서 중심점 배치를 랜덤이 아닌 특정 방식으로 결정
    - 1) 데이터 포인트중에서 무작위로 1개를 선택하여 중심점 지정
    - 2) 나머지 데이터 포인트들에 대해 그 첫번째 중심점까지의 거리 계산
    - 3) 지정된 중심점으로부터 거리가 가정 먼 데이터 포인트를 그 다음 중심점 지정
    - 4) 중심점이 k개가 될 때까지 2, 3번을 반복
  - 초기 중심점을 전략적 배치 → classical K-Means보다 더 최적의 군집화
  - 알고리즘 수렴하는 속도가 빠름

실습

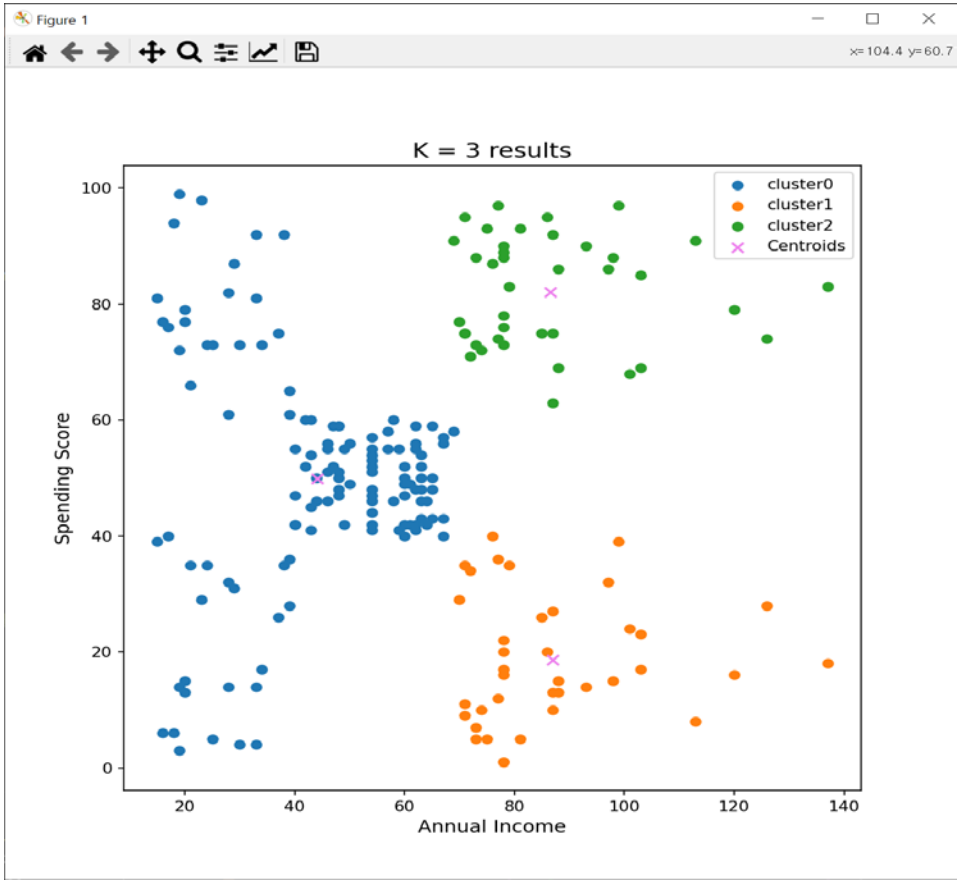
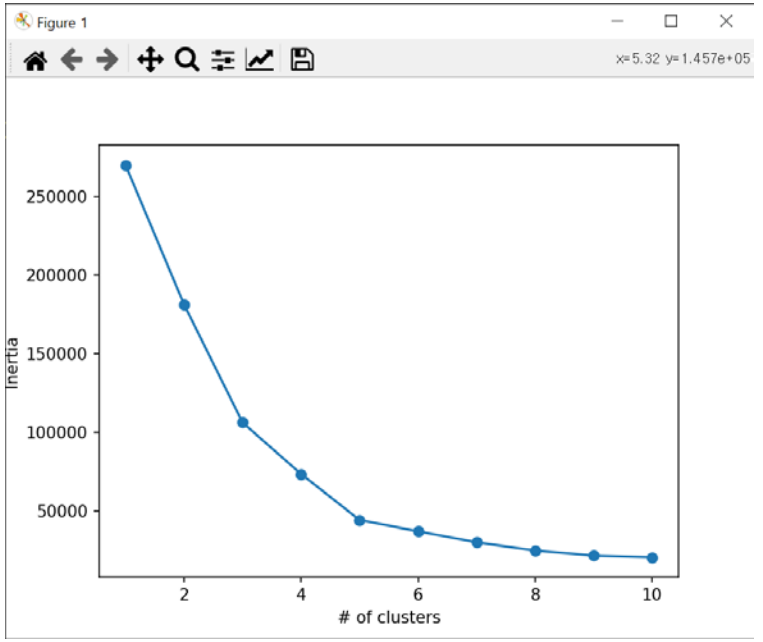


Figure 1
x=104.4 y=60.7
K = 3 results
cluster0
cluster1
cluster2
Centroids
Spending Score
Annual Income
0
20
40
60
80
100
20
40
60
80
100
120
140

실습

- 실습 #1 - 2
  - elbow method를 구현하고 K값 증가에 따른 inertia 값을 그래프로 그리시오.

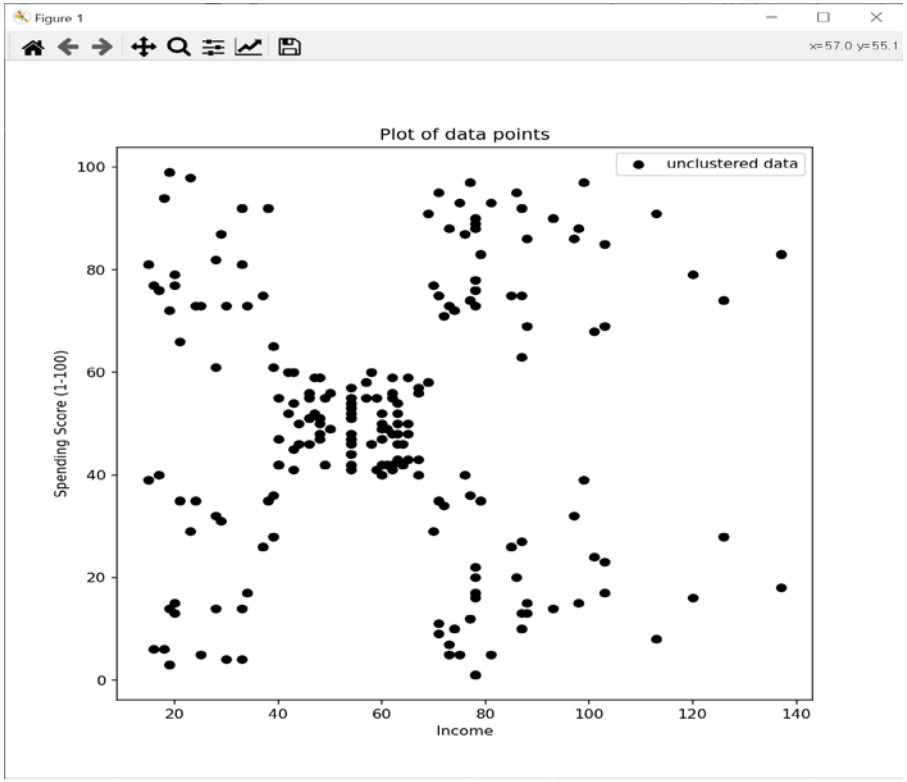
```python
def elbow(X):
    sse = []
    for i in range(1, 11):
        km = KMeans(n_clusters = i, init = 'k-means++', random_state = 0)
        km.fit(X)
        sse.append(km.inertia_)

    plt.plot(range(1,11), sse, marker = 'o')
    plt.xlabel('# of clusters')
    plt.ylabel('Inertia')
    plt.show()

elbow(data)
```



- KMeans 객체는 내부적으로 .inertia_ 속성을 가지고 있음 → intertia_ 속성으로 확인 가능

실습

- 실습 #2
  - ‘MallCustomer.csv’에서 Income과 Spending Score(1-100) 데이터를 그래프에 표시하시오.



실습

- 실습 #3
  - K-Means Clustering 알고리즘을 Raw level에서 구현하고, 클러스터 중심과 클러스터링 된 결과를 그래프로 그리시오.
  - K=3을 가정, iteration = 100회
  - 초기 중심점은 data point 중에서 random하게 K개 설정