Python

K-Means

K-Means

❖ 교사 학습

- 관측된 데이터가 특징 벡터 x와 관측값이 속해있는 클래스 ω로 이루어진 변수 쌍 {x,ω}으로 구성될 경우의 학습

❖ 비교사 학습

- 클래스 라벨 ωi가 주어지지 않고 특징 벡터 x={x1, x2,…,xN} 만으로 이루어진 데이터 집합이 주어질 경우의 학습

❖ 비교사 학습의 이용

- 첫째, 표본의 수가 너무 많아서 각 표본마다 일일이 라벨링하는 것이 비용이 많이 드는 지루한 과정일 경우
- 둘째, 데이터 마이닝(data mining)과 같이 원천적으로 데이터에 대한 클래스 라벨들이 주어지지 않을 경우
- 셋째, 많은 데이터 집합들이 작은 프로토타입(원형) 집합으로 압축되어 질 수 있는 경우. (kNNR)

K-Means

❖ k-means의 계산 절차

1. 시작: 데이터 집합 $[\mathbf{x}_1, ..., \mathbf{x}_N]$에서 K개의 벡터를 임의로 선택하여 K개의 초기 중심 집합 $[y_1, ..., y_K]$를 만든다.
2. E단계: 만약 데이터 $\mathbf{x}_n$이 $\mathbf{y}_i$에 가장 가깝다면 클러스터 X_i에 속하도록 라벨링한다. 결국 데이터 집합은 K개의 클러스터들 $\{X_1, ..., X_K\}$로 나누어진다.

$$X_i = \{\mathbf{x}_n \mid d(\mathbf{x}_n, \mathbf{y}_i) \le d(\mathbf{x}_n, \mathbf{y}_j),\ j = 1, ..., K\} \tag{7.5}$$

3. M단계: E단계에서 구한 새로운 클러스터들에서 각각의 중심을 갱신한다.

$$\mathbf{y_i} = c(X_i),\ i = 1, ..., K \tag{7.6}$$

4. 데이터와 가장 가까운 클러스터 중심들과 거리의 합으로 총 왜곡(distortion)을 구한다.

$$D = \sum_{n=1}^{N} d(\mathbf{x_n}, \mathbf{y_{i(n)}}), \tag{7.7}$$

※ $\mathbf{x_n} \in X_k$ 라면 $i(n) = k$

5. 총 왜곡이 적절하게 변하지 않거나 설정된 반복 횟수에 도달할 때까지 2~4단계를 반복한다.

$$\Delta\mathbf{D} = \frac{\mathbf{D_{prev}} - \mathbf{D_{curr}}}{\mathbf{D_{prev}}} < 10^{-4} \quad : \text{왜곡 검증} \tag{7.8}$$

K-Means

- K-Means

```
dataNum = 1000
X = np.zeros((dataNum*3, 2))

X[0:dataNum, :] = np.random.randn(dataNum, 2)
X[dataNum:dataNum*2, :] = np.random.randn(dataNum, 2) + 2
X[dataNum*2:dataNum*3, :] = np.random.randn(dataNum, 2) + 4

plt.figure(1)
plt.plot(X[:, 0], X[:, 1], '*g')
N = X.shape[0]
K = 2
m = np.zeros((K, 2))
Xlabel = np.zeros((N, 1))
i = 0
cmode = ['*g', '*r']
```

K-Means

- K-Means

```
#대표 벡터를 선택
while i < K:
    t = int(np.floor(np.random.rand(1) * N))
    if (X[t, :] != m[0, :]).all and (X[t, :] != m[1, :]).all:
        m[i, :] = X[t, :]
        plt.plot(m[i, 0], m[i, 1], 'Dk')
        i = i + 1

plt.show()
```

K-Means

- K-Means

```
for iteration in range(0, 10, 1):
    # 각 데이터를 가까운 클러스터에 할당
    plt.figure(iteration+(K*10))
    Xlabel = np.zeros((1, N))

    for i in range(0, N, 1):
        dist = np.zeros((1, K))
        for j in range(0, K, 1):
            dist[0, j] = np.dot(X[i, :] - m[j, :], X[i, :] - m[j, :].T)
        Xlabel[0, i] = np.argmin(dist)

    # 대표벡터를 다시 계산
    for i in range(0, K, 1):
        I = np.where(Xlabel == i)
        m[i, :] = np.mean(X[I[1], :])
```

K-Means

- K-Means

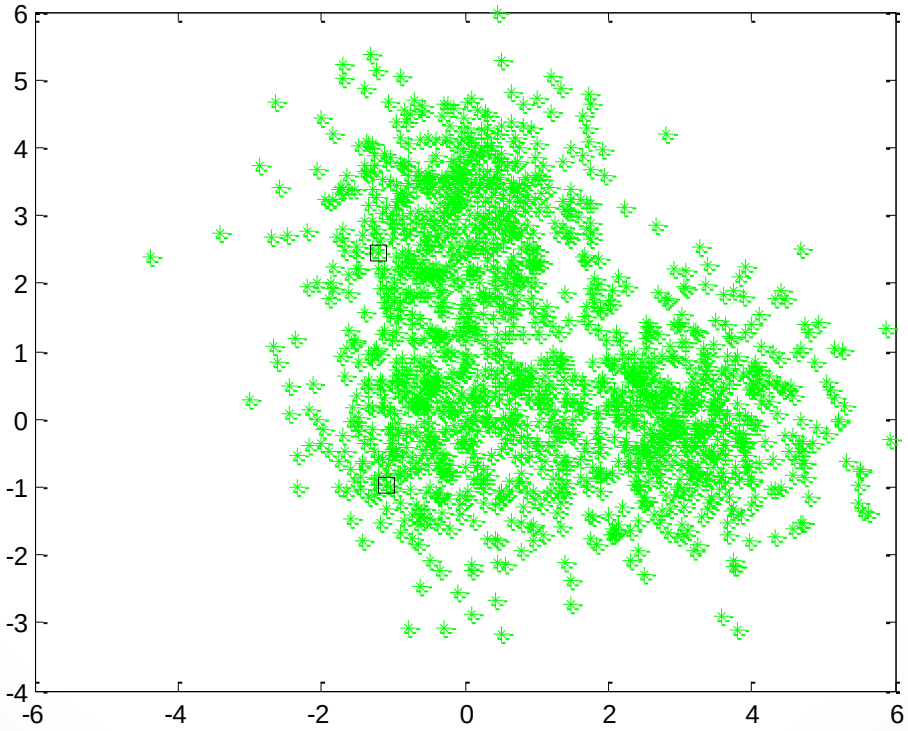
```
for i in range(0, K, 1):
    c1 = np.where(Xlabel == i)
    plt.plot(X[c1[1], 0], X[c1[1], 1], cmode[i])

for i in range(0, K, 1):
    plt.plot(m[i, 0], m[i, 1], 'Dk')

plt.show()
```

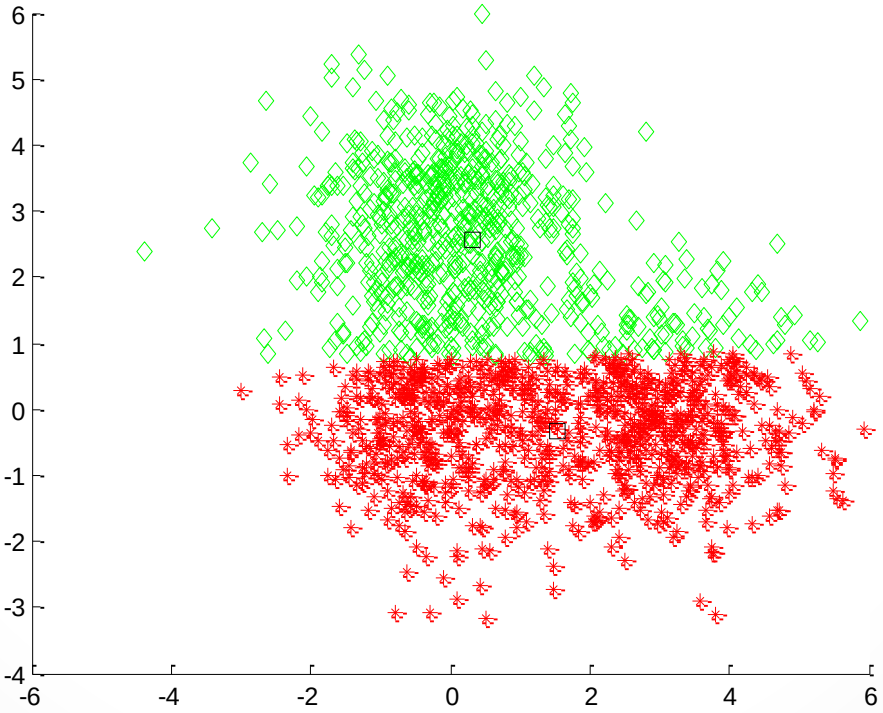
K-Means

- K-Means(초기 중심)



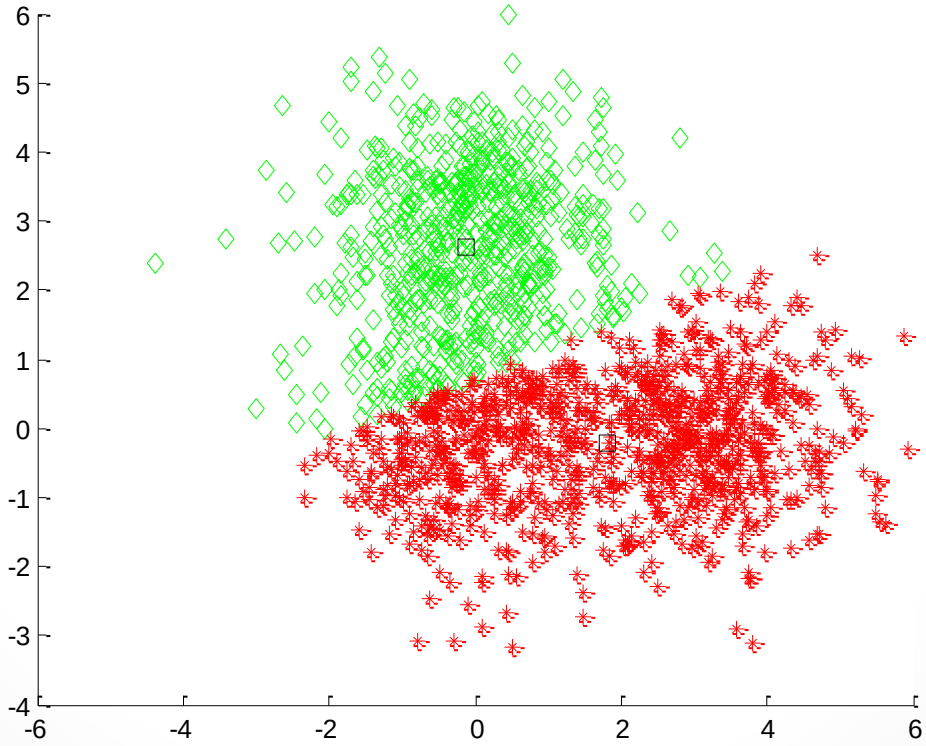
K-Means

- K-Means(Step 1)



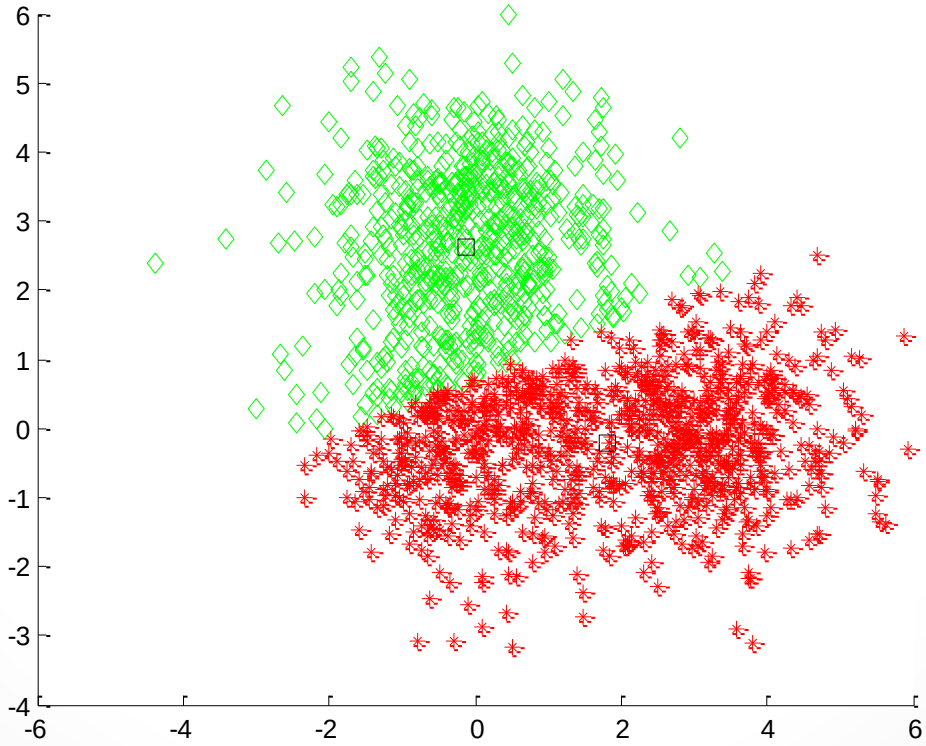
K-Means

- K-Means(Step 2)



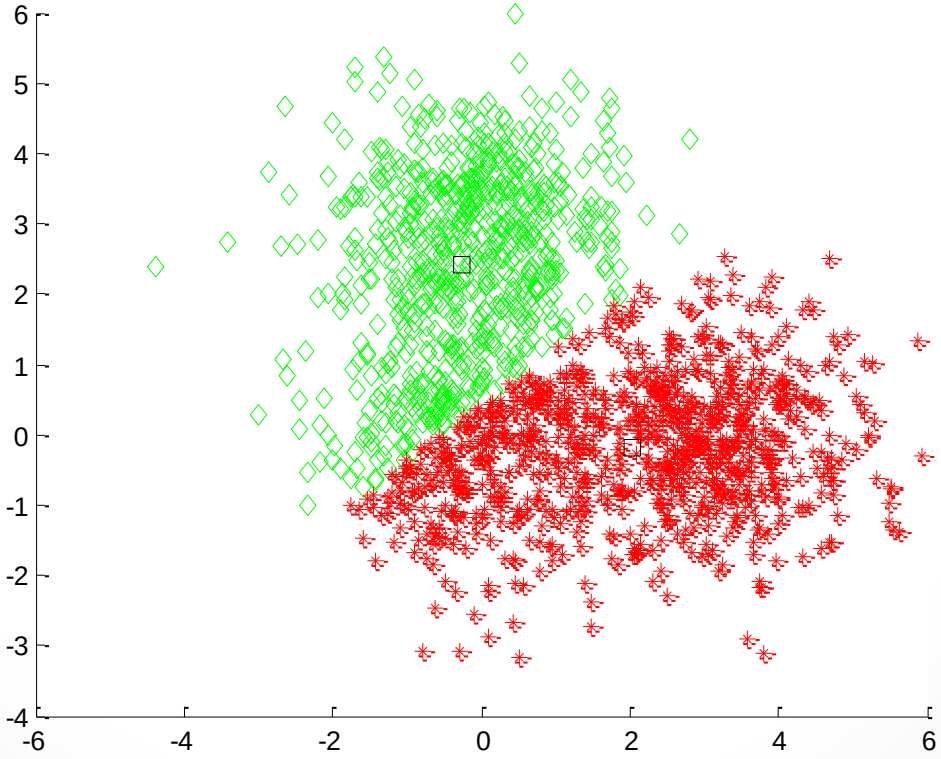
K-Means

- K-Means(Step 3)



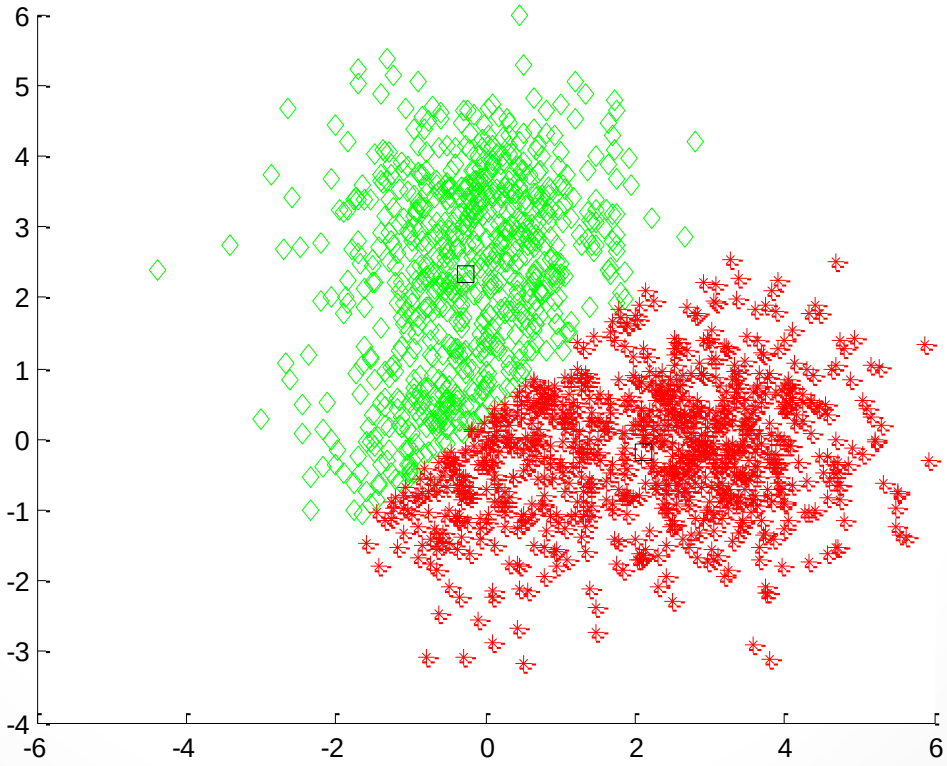
K-Means

- K-Means(Step 4)



K-Means

- K-Means(Step 5)



K-Means

- Exercise
  - 주어진 코드를 이용하여 5개의 클래스를 가지는 클러스터링을 수행하고, 왜곡검증을 추가하시오.