

A background image of a kitchen wall. At the top, a wooden shelf holds several teapots and cups in various colors like blue, red, and white. Below the shelf, a small framed picture hangs on the wall. To the left, a wooden rack holds several long-handled kitchen tools. In the foreground, a large, abstract painting with green and brown tones is visible.

---

**Machine Learning**

# Clustering

---

김선녕(sykim.lecture@gmail.com)

---

## 군집화(Clustering)란?

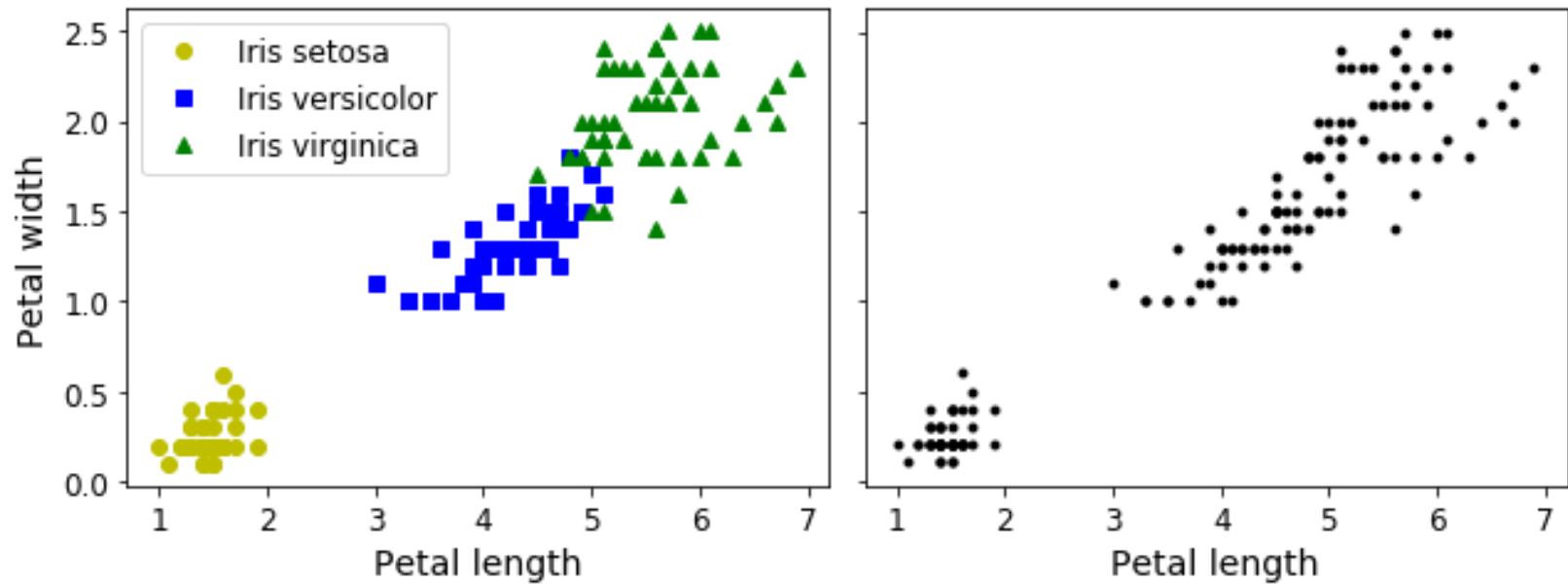
K-평균 알고리즘(K-Means Algorithm)

계층형 군집화(Hierarchical Clustering)

DBSCAN

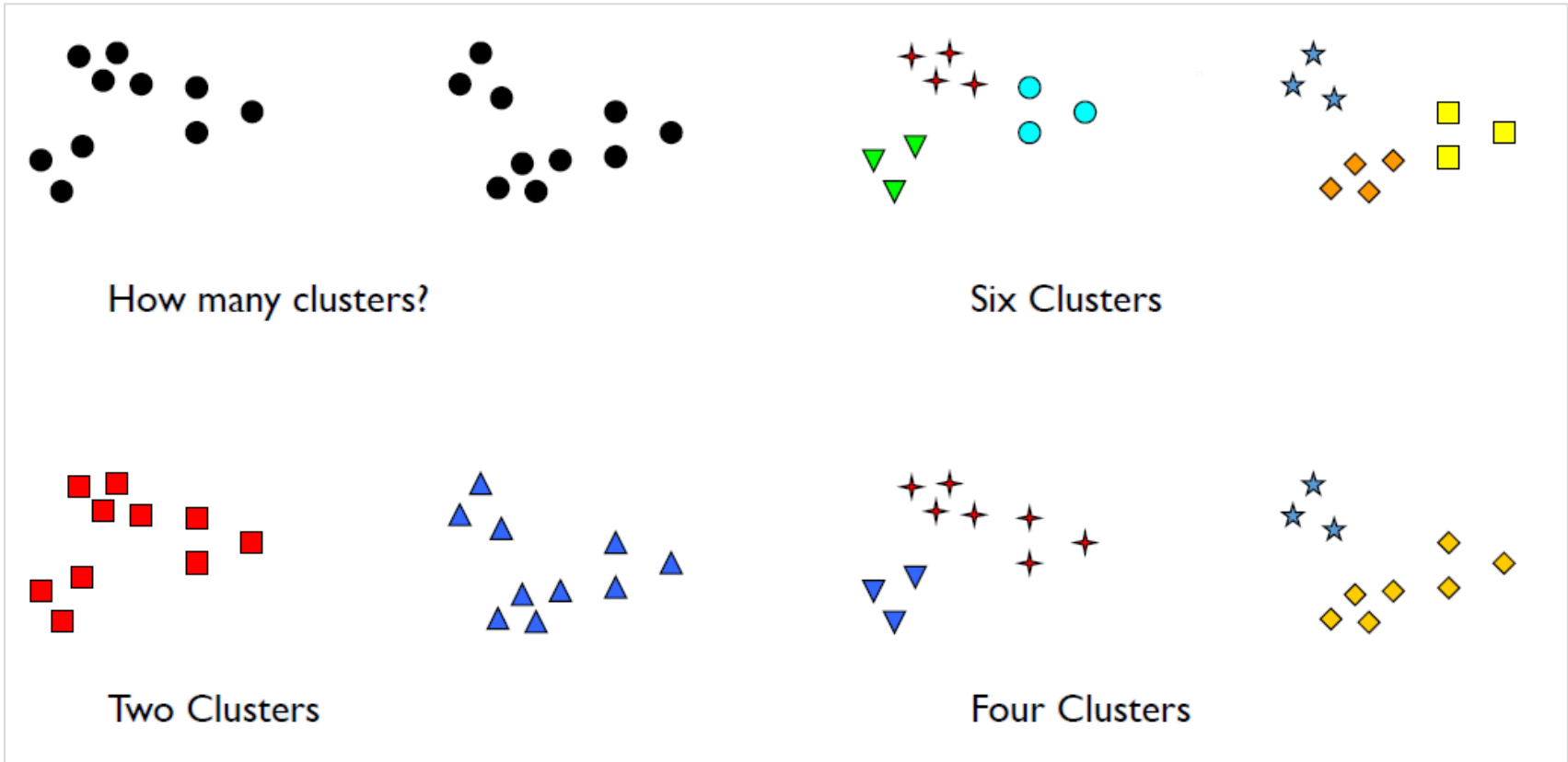
- 지식(정보)가 없는 상태에서 유사(similar)한 데이터를 모아 같은 그룹으로 묶는 일
  - 군집간 분산 최대화, 군집 내 분산 최소화
  - 유사도(similarity) 계산
- 레이블이 없는 대표적인 비지도학습(unsupervised learning)
  - 분류(classification) : 지도학습(supervised learning)
- Hierarchical Clustering 과 Partitional Clustering 가 있다.

- 레이블 유무



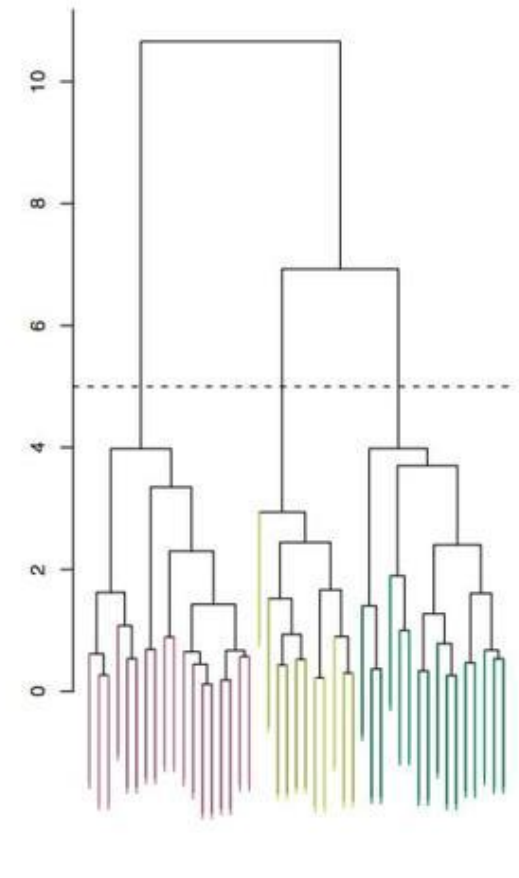
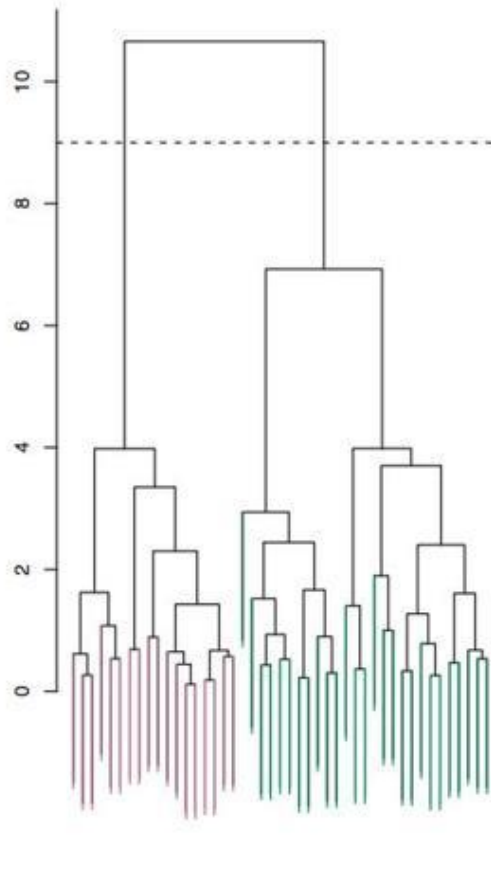
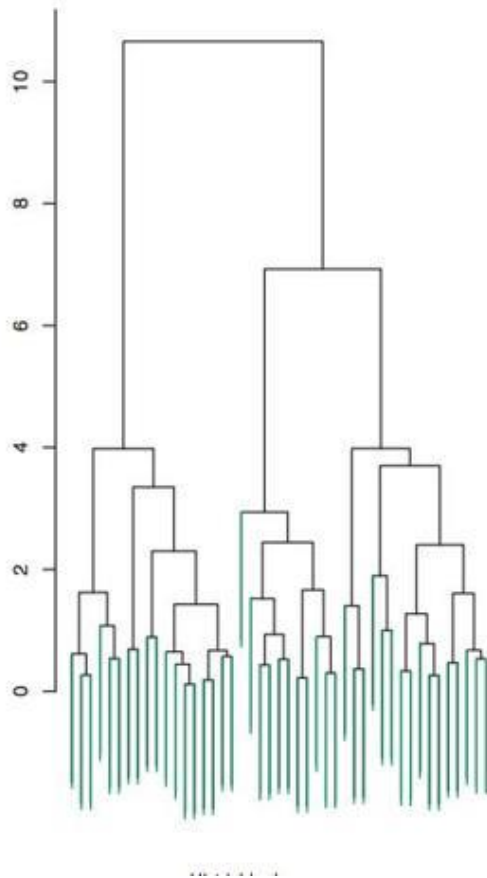
## 분할 군집화(Partitional Clustering)

- 계층을 고려하지 않고 평면적으로 클러스터하는 방법.
- 몇 개의 클러스터로 나눌 것이라 예상하고 클러스터 개수를 정한다.



## 계층형 군집화(Hierarchical Clustering)

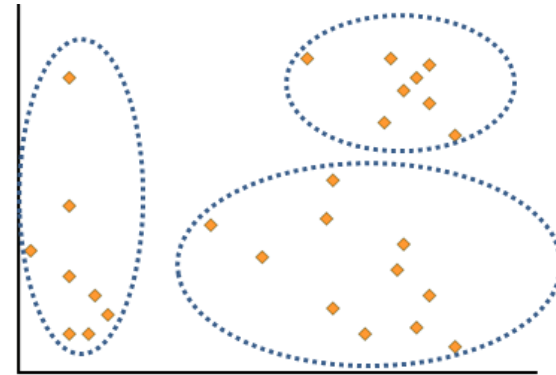
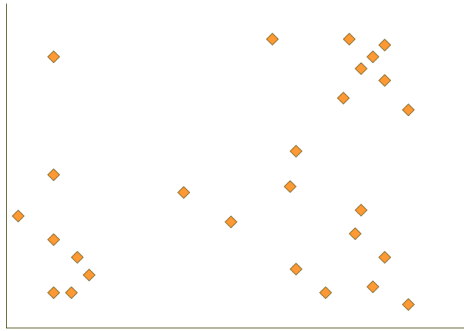
- 계층형 군집화는 군집의 개수를 미리 지정하지 않아도 된다.
- 데이터를 트리 모양으로 나타내는 덴드로그램(dendrogram)을 만들어 낸다.



- 문서분류
  - 문서에 나타난 키워드의 빈도수에 따라 문서의 주제별 분류
- 영상분류
  - 영상의 색조에 따라 분류
  - 위성사진으로 곡물 수확량, 화재 상황
- 고객분류
  - 구매이력이나 웹사이트 내 행동 등을 기반으로 그룹화
  - 고객 추천시스템
- 데이터 분석
  - 새로운 데이터셋을 분석할 때 군집화를 실행하고 각각의 군집을 따로 분석
- 이상치 탐지
  - 모든 군집에서 친화성이 낮은 샘플 데이터는 이상치일 가능성이 높다







정지영상을 비슷한 것끼리 군집화



영상 분할(segmentation)



군집화(Clustering)란?

**K-평균 알고리즘(K-Means Algorithm)**

계층형 군집화(Hierarchical Clustering)

DBSCAN

- 주어진 데이터셋에서  $k$ 개의 군집을 찾고자 하는 알고리즘

입력 : 훈련집합  $X = \{x_1, x_2, \dots, x_n\}$ ,      군집의 개수  $k$

출력 : 군집집합  $C = \{c_1, c_2, \dots, c_k\}$

[pseudo code]

1:  $k$ 개의 점들을 초기 중심점(*centroid*)으로 선택한다.

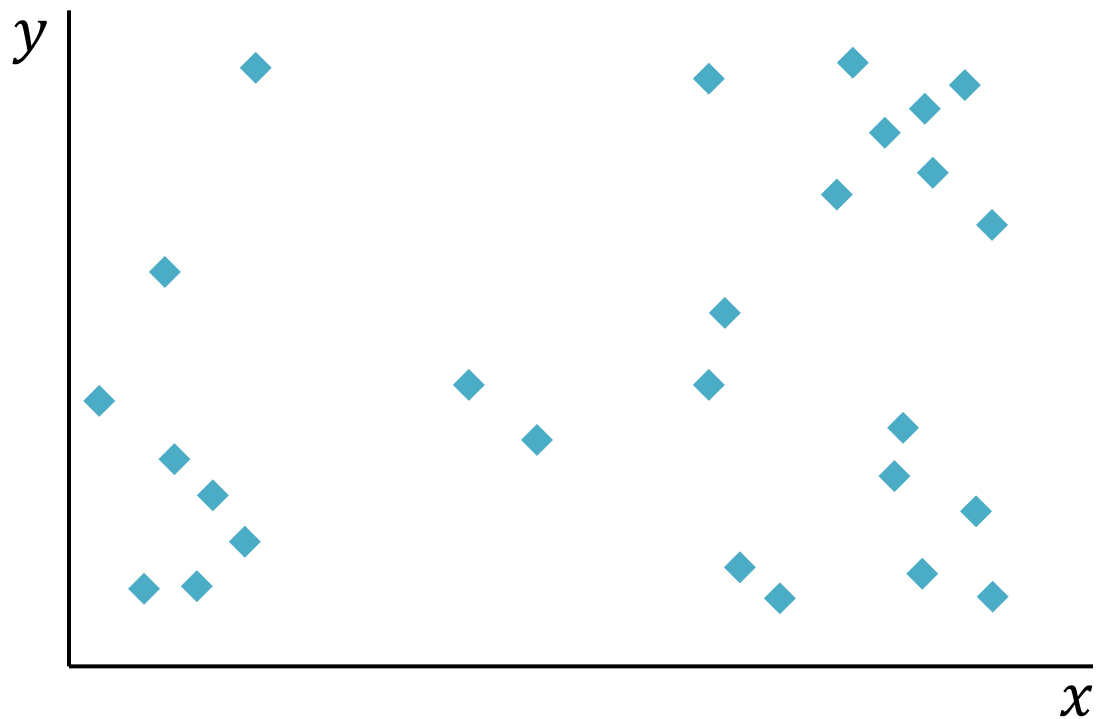
2: **repeat**

3:    각 점을 가장 가까운 중심점(*centroid*)에 할당하여  $k$ 개의 군집을 형성한다.

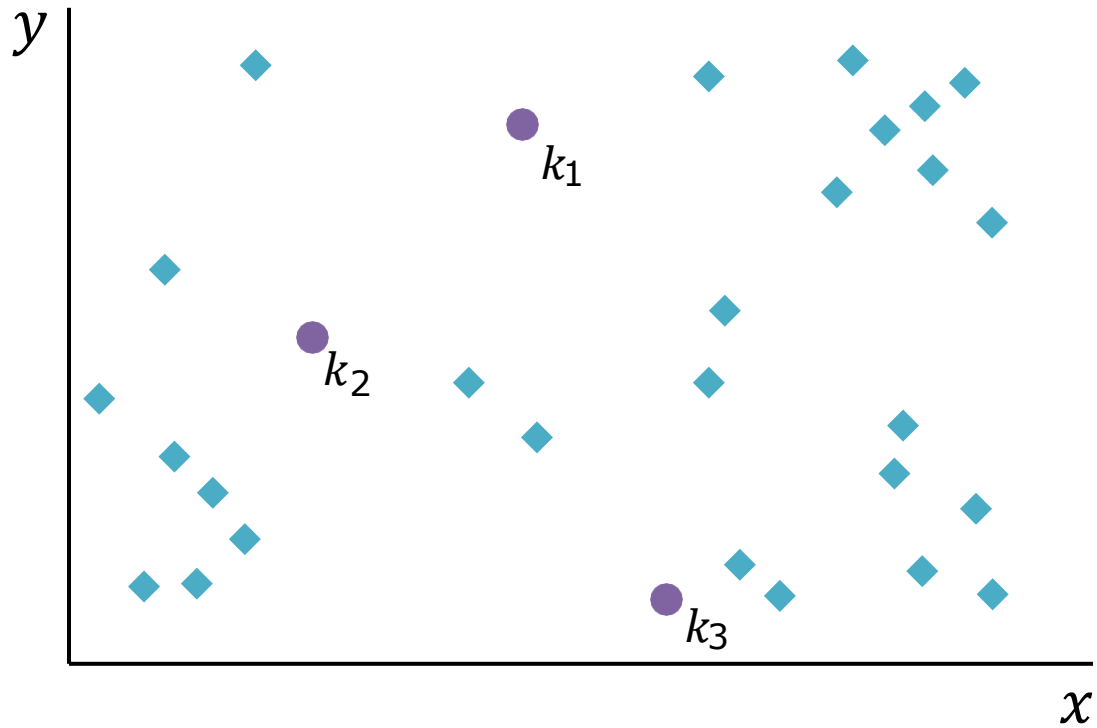
4:    각 군집의 중심점을 다시 계산한다.

5: **until** 중심점이 바뀌지 않을 때까지

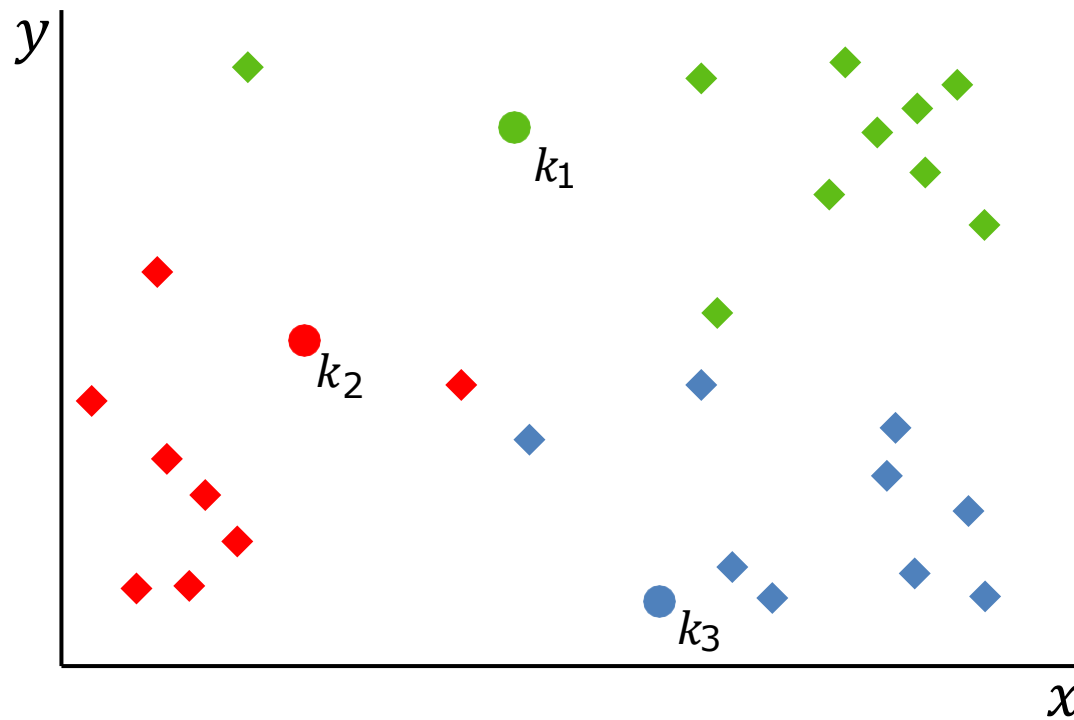
Suppose we wish to cluster these items



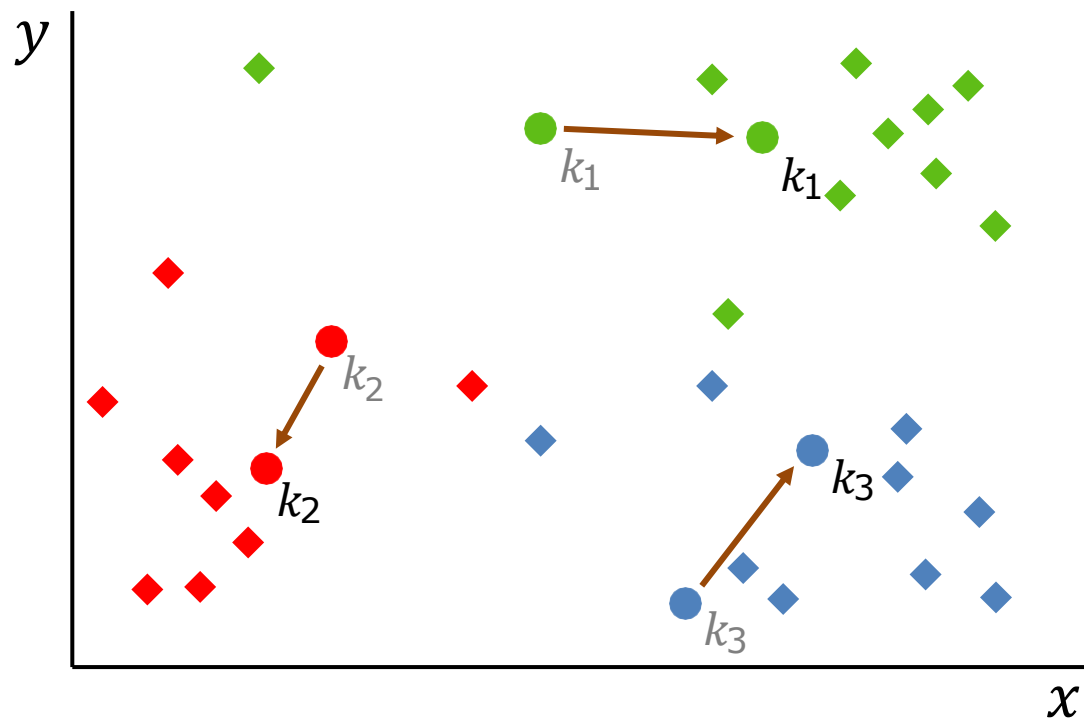
Pick 3 initial cluster centers at random



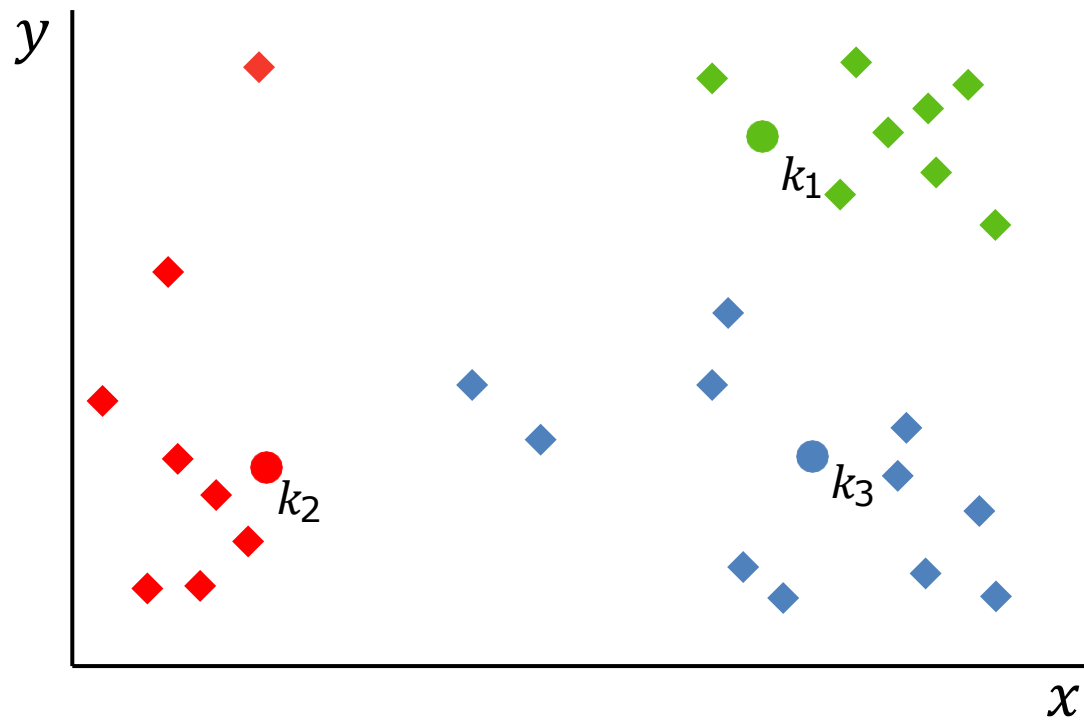
Assign each instance to the closest cluster center



Move each cluster center to the mean of each cluster.  
Reassign points closest to a different new cluster center.



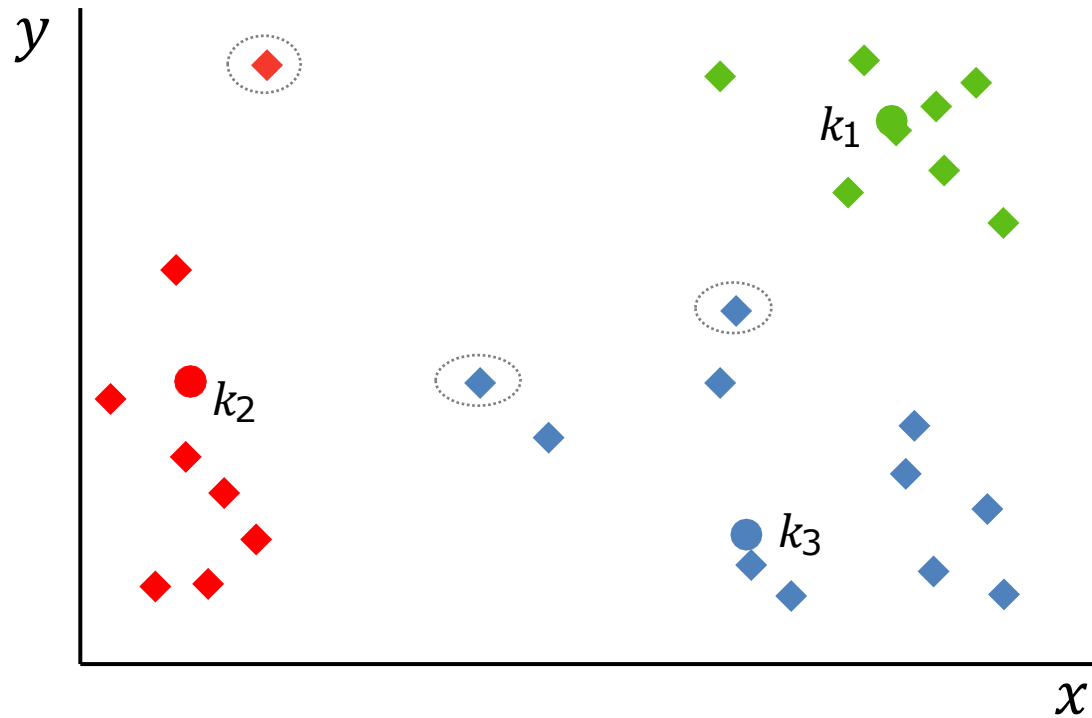
Recompute cluster means



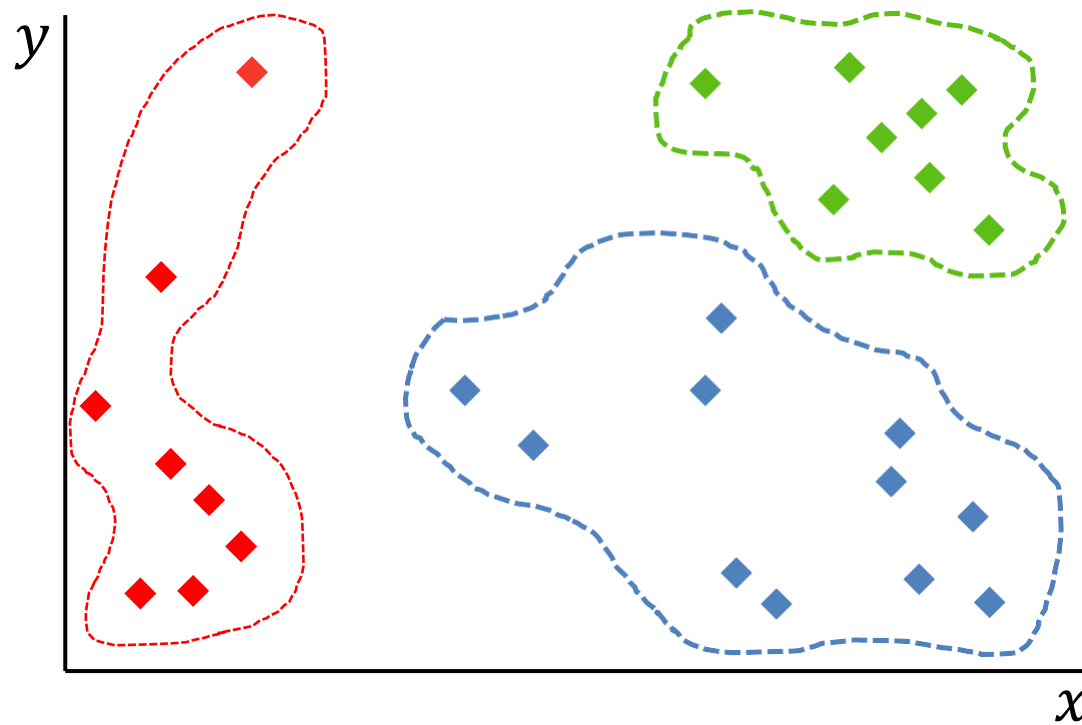


Reassign labels

**No change. Done!**



## Final clusters



- 군집화의 목표는 목적함수(*objective function*)에 의해 표현
  - 목적함수 : 점들 사이 또는 점과 군집 중심점 간의 인접성에 의존
  - 유클리디안 거리(*Euclidean Distance*)를 이용하면 클러스터의 품질은 오차 제곱의 합(*ESS : Error Sum of Squares*)으로 평가
  - 각각의 데이터점의 오차(*error*) - 가장 가까운 중심점과의 유클리디안 거리를 산출

$$ESS = \sum_{i=1}^k \sum_{x \in c_i} dist(c_i, x)^2$$

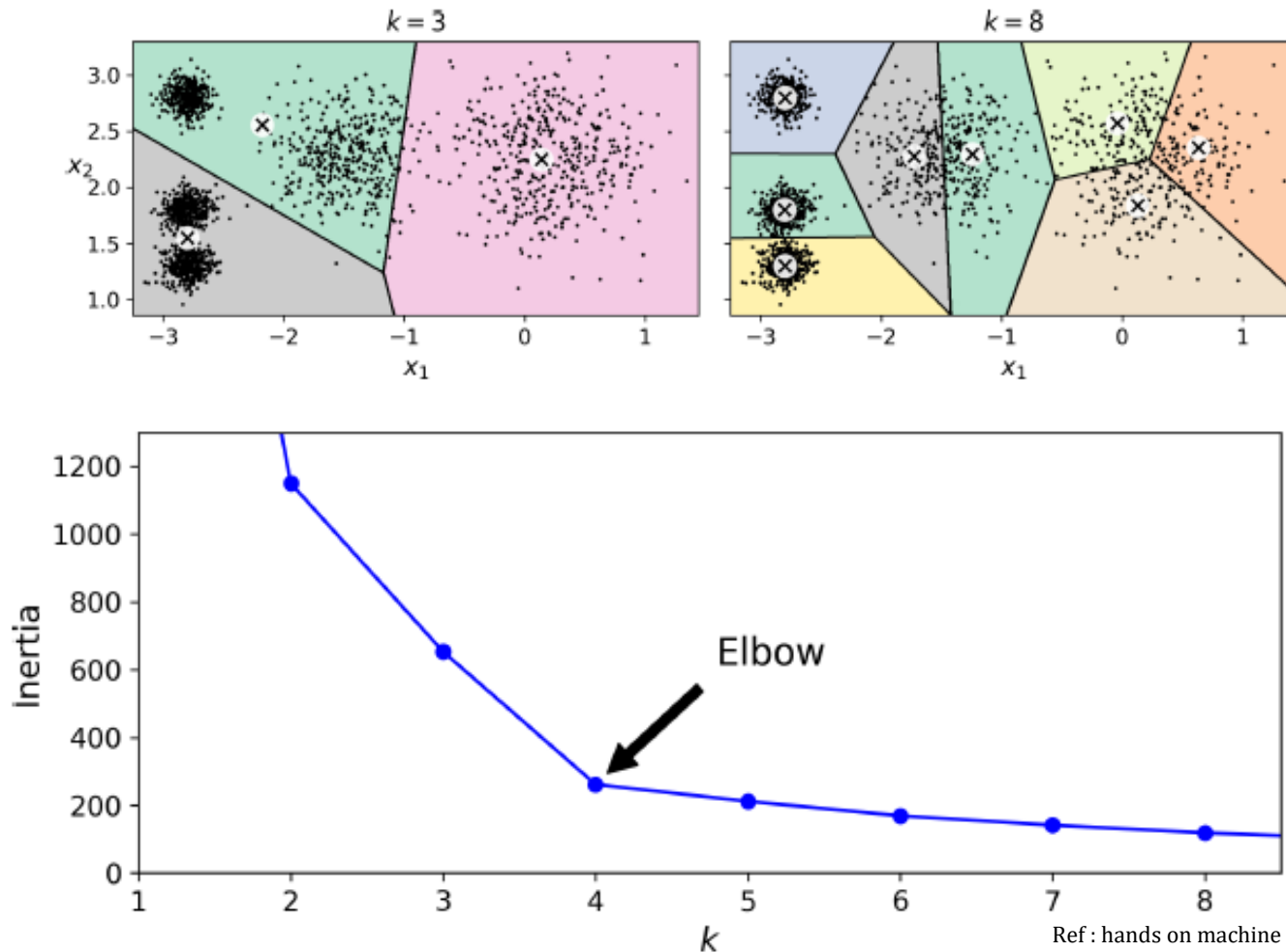
$i$ 번째 군집의 중심점(평균)은

$$c_i = \frac{1}{m_i} \sum_{x \in c_i} x$$

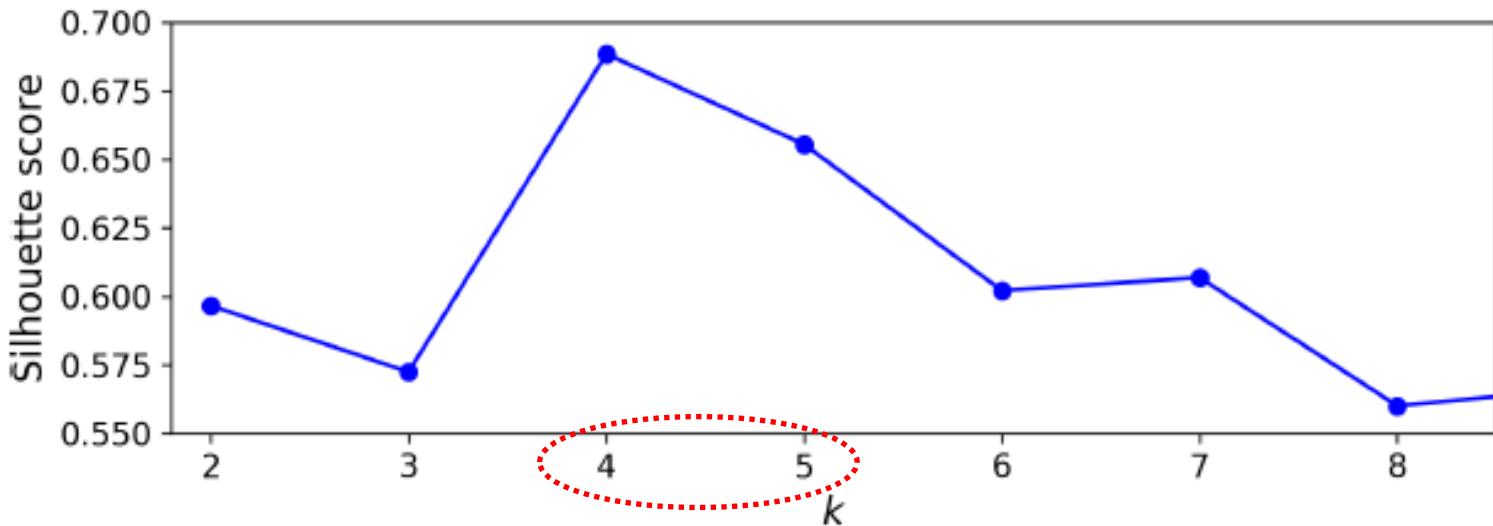
- $SSE$ 의 최소화(단계 3,4)하는  $k$ 값을 선택하는 것이 중요.

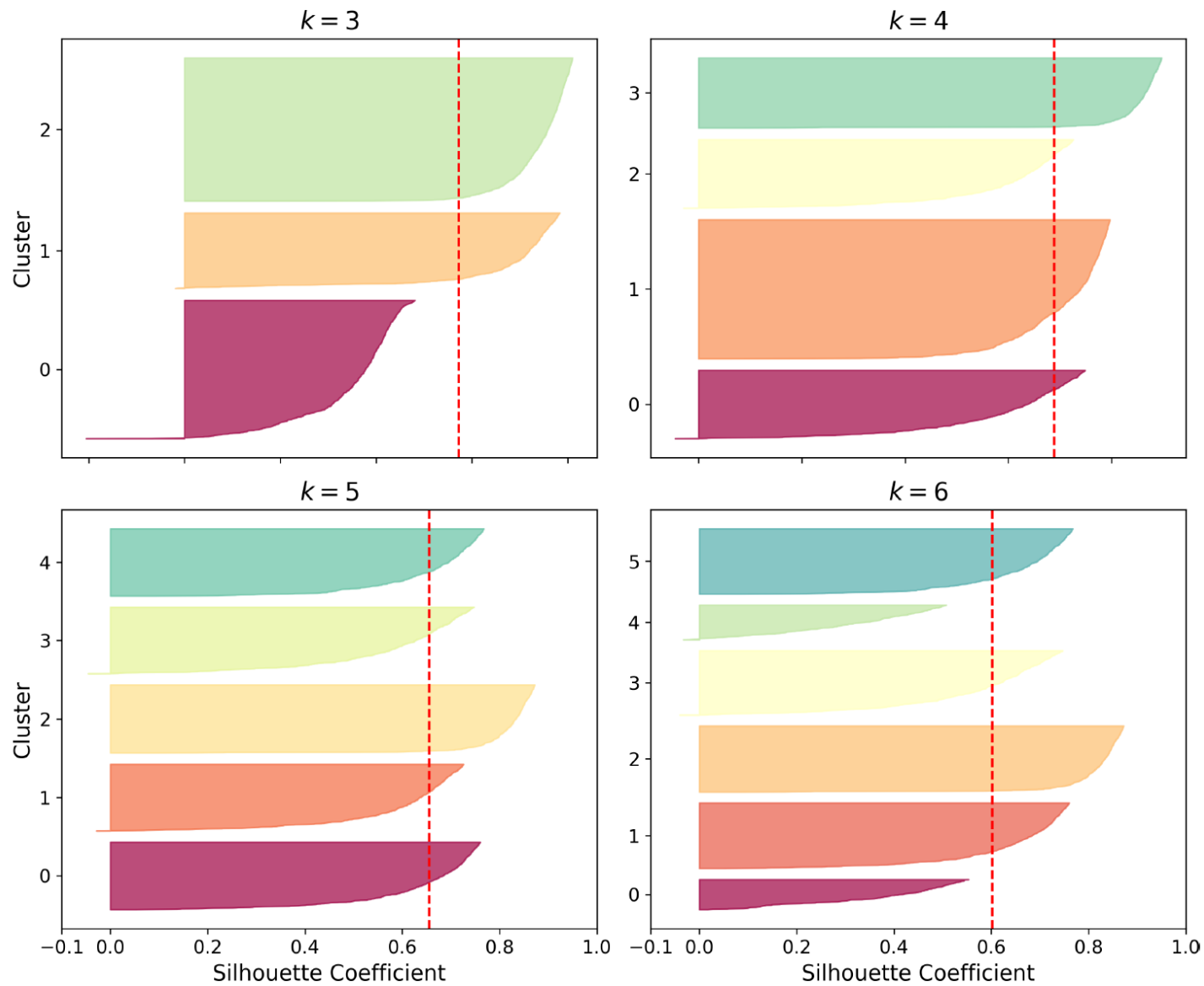
## 잘못된 클러스터 개수 선택

- $k$ 가 너무 작으면 별개의 클러스터를 합치고  $k$ 가 너무 크면 하나의 클러스터가 여러 개로 나뉜다
- 이너셔(inertia) : 각 샘플과 가장 가까운 센트로이드사이의 평균제곱거리



- 실루엣 계수 =  $(b - a) / \max(a, b)$ 
  - $a$  : 동일한 클러스터에 있는 다른 샘플까지 평균거리( 클러스터 내부의 평균 거리)
  - $b$  : 가장 가까운 클러스터까지 평균거리(가장 가까운 클러스터의 샘플까지 평균 거리)
  - +1에 가까우면 자신의 클러스터안에 잘 속해 있고 다른 클러스터와는 멀리 떨어져 있다는 뜻.
  - 0에 가까우면 클러스터 경계에 위치한다는 의미
  - -1에 가까우면 잘못된 클러스터라는 의미
- 실루엣 점수 : 실루엣 계수의 평균

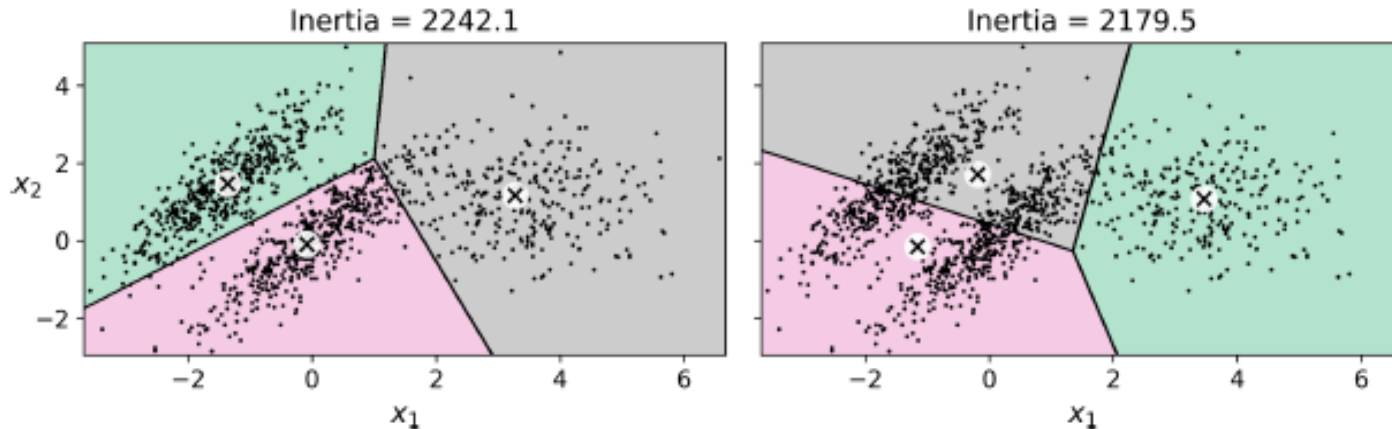




- 장점
  - 속도가 빠르고 확장이 용이
  - 비록 멀티로 수행되더라도 간단하며 다양한 데이터에 매우 효율적
- 단점
  - 모든 객체들 간 거리 계산으로 인한 부하 발생
  - 초기 중심점(centroid) 설정이 최종 분류에 큰 영향을 미침
  - 밀집도가 서로 다르거나 원형이 아닐 경우 잘 작동되지 않음
  - 평균값 계산시 이상치(outlier) 데이터가 미치는 영향이 크다

(예) 크기와 밀집도가 다른 세 개의 타원형 클러스터

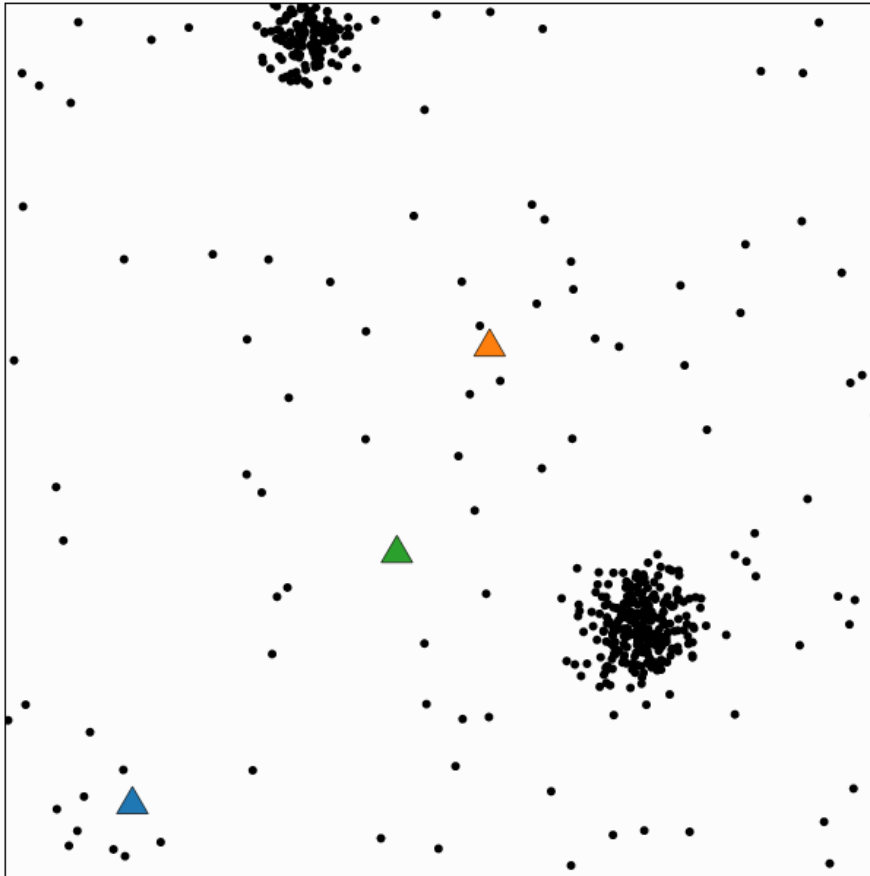
k-평균이 세 개의 타원형 클러스터를 적절히 구분하지 못한다



# $k$ -평균 알고리즘( $k$ -means algorithm) 데모

24

<https://stanford.edu/class/engr108/visualizations/kmeans/kmeans.html>



The  $k$ -means algorithm is an iterative method for clustering a set of  $N$  points (vectors) into  $k$  groups or clusters of points.

## Algorithm

Repeat until convergence:

### Find closest centroid

Find the closest centroid to each point, and group points that share the same closest centroid.

### Update centroid

Update each centroid to be the mean of the points in its group.

Find closest centroid

## Data

Clustered points  Random

Number of clusters :

Number of centroids :

New points

New centroids

Mean square point-centroid distance: not yet calculated



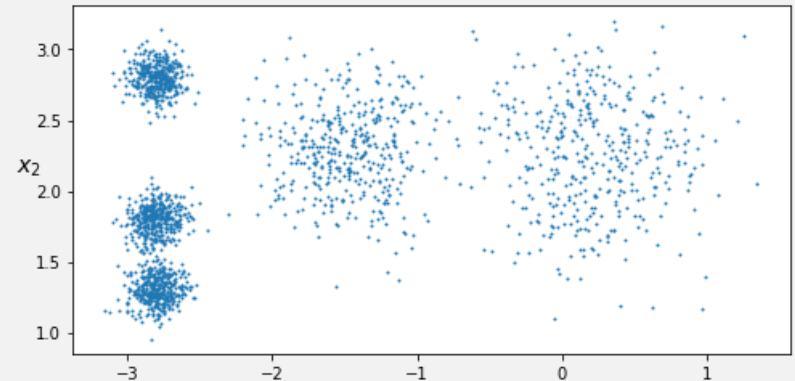
```
import tensorflow as tf
import numpy as np
import pandas.util.testing as pd
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.datasets import make_blobs
blob_centers = np.array (
    [[0.2, 2.3], [-1.5, 2.3], [-2.8, 1.8], [-2.8, 2.8], [-2.8, 1.3]]
)
blob_std = np.array([0.4, 0.3, 0.1, 0.1, 0.1])

# 가상데이터 생성
X, y = make_blobs(n_samples=2000, centers=blob_centers, cluster_std=blob_std, random_state=7)

def plot_clusters(X, y=None) :
    plt.scatter(X[:, 0], X[:, 1], c=y, s=1)
    plt.xlabel("$x_1$", fontsize=14)
    plt.ylabel("$x_2$", fontsize=14, rotation=0)

plt.figure(figsize=(8, 4))
plot_clusters(X)
plt.show()
```



```
# K-평균 군집 알고리즘을 훈련
from sklearn.cluster import KMeans

k = 5
kmeans = KMeans(n_clusters=k, random_state=42)

# Compute cluster centers and predict cluster index for each sample
y_pred = kmeans.fit_predict(X)
y_pred # Index of the cluster each sample belongs to

array([4, 1, 0, ..., 3, 0, 1], dtype=int32)
```

```
y_pred is kmeans.labels_
```

True

```
# 5개의 센트로이드 (즉 클러스터 중심)
```

```
kmeans.cluster_centers_

array([[ 0.20876306,  2.25551336],
       [-2.80389616,  1.80117999],
       [-1.46679593,  2.28585348],
       [-2.79290307,  2.79641063],
       [-2.80037642,  1.30082566]])
```

```
# KMeans 객체는 훈련한 샘플의 레이블을 가지고 있다.
# 여기에서 샘플의 레이블은 샘플에 할당한 클러스터의 인덱스
kmeans.labels_
```

```
array([4, 1, 0, ..., 3, 0, 1], dtype=int32)
```

```
# 새로운 샘플의 레이블
X_new = np.array([[0, 2], [3, 2], [-3, 3], [-3, 2.5]])
kmeans.predict(X_new)
```

```
array([0, 0, 3, 3], dtype=int32)
```

```
# 이 모델의 결정 경계 (보로노이 다이어그램)
```

```
def plot_data(X):
    plt.plot(X[:, 0], X[:, 1], 'k.', markersize=2)
```

```
def plot_centroids(centroids, weights=None, circle_color='w', cross_color='k'):
    if weights is not None:
        centroids = centroids[weights > weights.max() / 10]
    plt.scatter(centroids[:, 0], centroids[:, 1],
                marker='o', s=30, linewidths=8,
                color=circle_color, zorder=10, alpha=0.9)
    plt.scatter(centroids[:, 0], centroids[:, 1],
                marker='x', s=50, linewidths=50,
                color=cross_color, zorder=11, alpha=1)
```

```

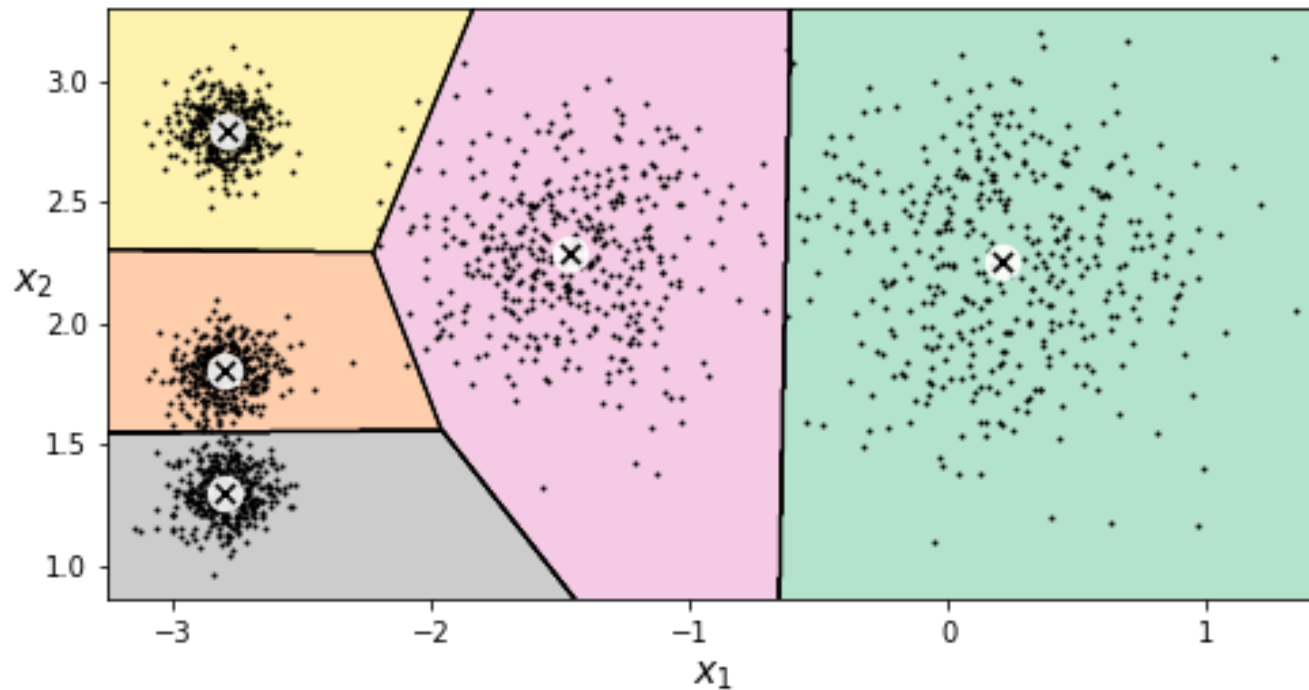
def plot_decision_boundaries(clusterer, X, resolution=1000,
                             show_centroids=True, show_xlabels=True, show_ylabels=True):
    mins = X.min(axis=0) - 0.1
    maxs = X.max(axis=0) + 0.1
    # np.meshgrid() : x 값의 배열과 y 값의 배열로 직사각형 격자를 만드는 것
    xx, yy = np.meshgrid(np.linspace(mins[0], maxs[0], resolution),
                          np.linspace(mins[1], maxs[1], resolution))
    Z = clusterer.predict(np.c_[xx.ravel(), yy.ravel()])
    # xx.shape => (1000, 1000)
    Z = Z.reshape(xx.shape)

    plt.contourf(Z, extent=(mins[0], maxs[0], mins[1], maxs[1]),
                 cmap="Pastel2")
    plt.contour(Z, extent=(mins[0], maxs[0], mins[1], maxs[1]),
                linewidths=1, colors='k')
    plot_data(X)
    if show_centroids:
        plot_centroids(clusterer.cluster_centers_)

    if show_xlabels:
        plt.xlabel("$x_1$", fontsize=14)
    else:
        plt.tick_params(labelbottom=False)
    if show_ylabels:
        plt.ylabel("$x_2$", fontsize=14, rotation=0)
    else:
        plt.tick_params(labelleft=False)

```

```
plt.figure(figsize=(8, 4))  
plot_decision_boundaries(kmeans, X) # 클러스터의 결정 경계를 그린다  
plt.show()
```



# K-평균 알고리즘을 1, 2, 3회 반복하고 센트로이드가 어떻게 움직이는지 확인해 보자

```
kmeans_iter1 = KMeans(n_clusters=5, init="random", n_init=1,  
                      algorithm="full", max_iter=1, random_state=1)  
kmeans_iter2 = KMeans(n_clusters=5, init="random", n_init=1,  
                      algorithm="full", max_iter=2, random_state=1)  
kmeans_iter3 = KMeans(n_clusters=5, init="random", n_init=1,  
                      algorithm="full", max_iter=3, random_state=1)  
  
kmeans_iter1.fit(X)  
kmeans_iter2.fit(X)  
kmeans_iter3.fit(X)
```

```
plt.figure(figsize=(10, 8))

plt.subplot(321)
plot_data(X)
plot_centroids(kmeans_iter1.cluster_centers_, circle_color='r', cross_color='w')
plt.ylabel("$x_2$", fontsize=14, rotation=0)
plt.tick_params(labelbottom=False)
plt.title("Update the centroids (initially randomly)", fontsize=14)

plt.subplot(322)
plot_decision_boundaries(kmeans_iter1, X, show_xlabels=False, show_ylabels=False)
plt.title("Label the instances", fontsize=14)

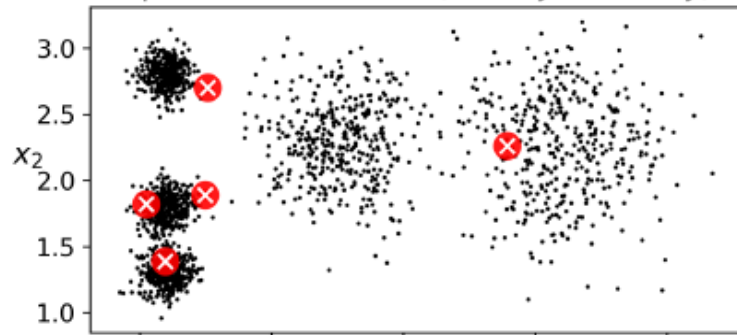
plt.subplot(323)
plot_decision_boundaries(kmeans_iter1, X, show_centroids=False, show_xlabels=False)
plot_centroids(kmeans_iter2.cluster_centers_)

plt.subplot(324)
plot_decision_boundaries(kmeans_iter2, X, show_xlabels=False, show_ylabels=False)

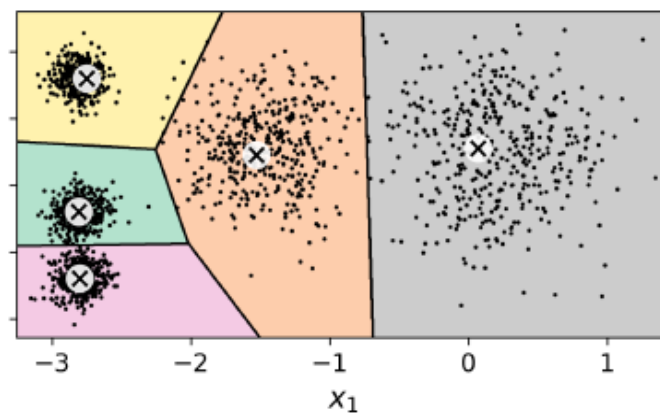
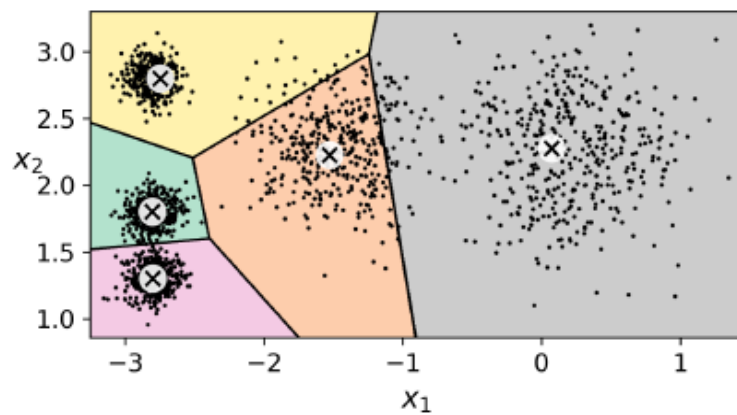
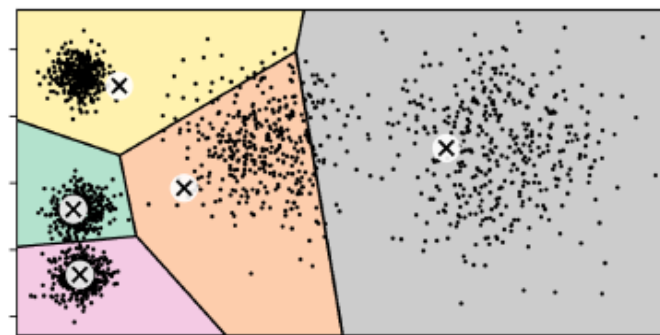
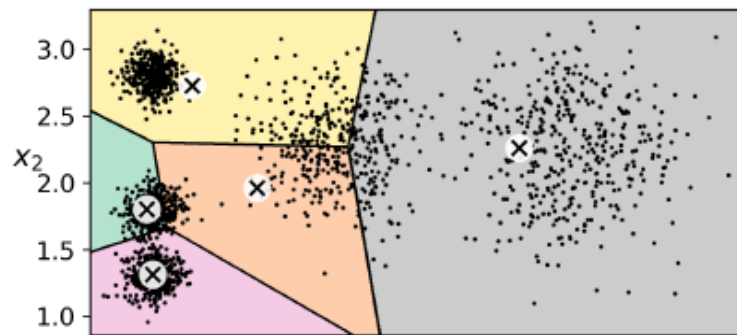
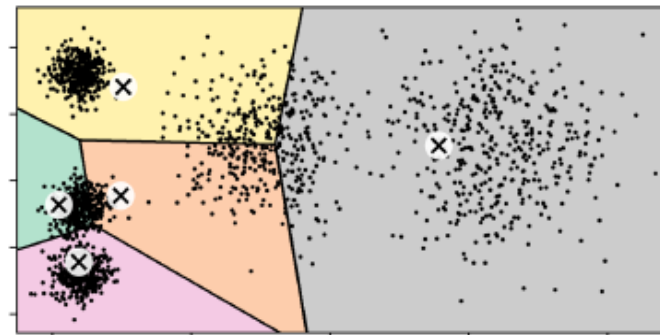
plt.subplot(325)
plot_decision_boundaries(kmeans_iter2, X, show_centroids=False)
plot_centroids(kmeans_iter3.cluster_centers_)

plt.subplot(326)
plot_decision_boundaries(kmeans_iter3, X, show_ylabels=False)
plt.show()
```

Update the centroids (initially randomly)



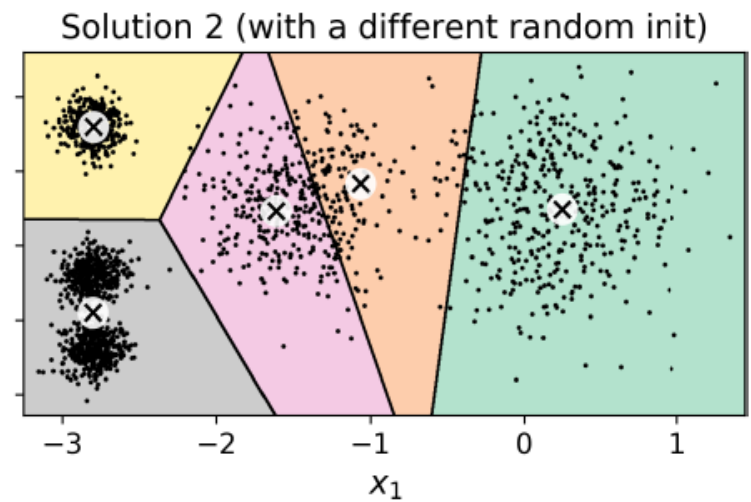
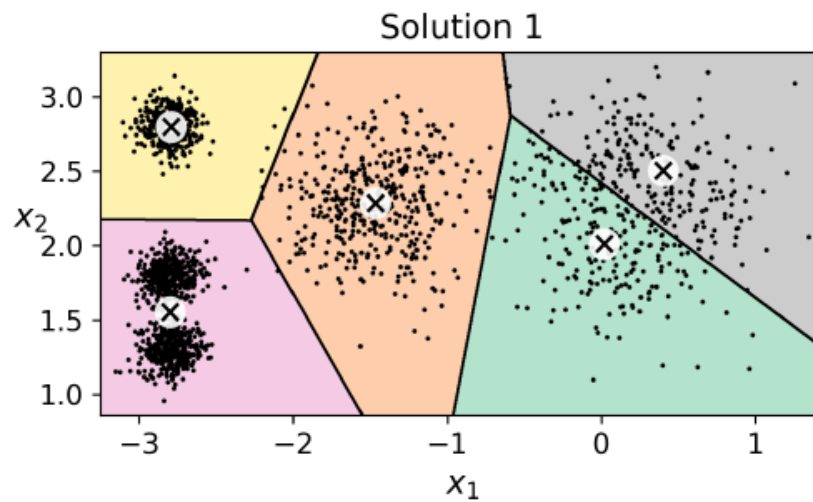
Label the instances






## 운 나쁜 센트로이드 초기화 : 최적이 아닌 솔루션

33





군집화(Clustering)란?  
K-평균 알고리즘(K-Means Algorithm)  
**계층형 군집화(Hierarchical Clustering)**  
DBSCAN

- 군집들 사이의 거리를 이용하여 가까운 거리에 있는 군집들을 차례로 묶거나, 먼 거리에 있는 군집들을 차례로 분리해 나가는 방법
- 합병에 의한 방법(*Agglomerative, Bottom – up*)
  - 포인트를 개별 클러스터로 시작하고 각 단계에서 가장 가까운 클러스터 쌍을 병합.
  - 가까운 개체끼리 묶어 새로운 군집을 이루어 나가며 결국은 한 개의 군집을 만듦
- 분할에 의한 방법(*Divisive, Top – down*)
  - 모든 것을 포함하는 클러스터에서 시작하여 각 단계에서 개별 점의 싱글 클러스터만 남을 때까지 클러스터를 분할
  - 전체를 두 개의 군집으로 분할하는 것으로 시작하여 상이한 개체들 부터 나누어 감. 결국은 개체들 각각이 군집이 됨

- 단일연결법(Single Linkage Method)

- MIN (Single Link) Proximity
- Defines cluster proximity as **the shortest distance between two points,  $x$  and  $y$ ,** that are in different clusters,  $A$  and  $B$ :

$$d(A, B) = \min_{x \in A, y \in B} d(x - y)$$

- 완전연결법(Complete Linkage Method)

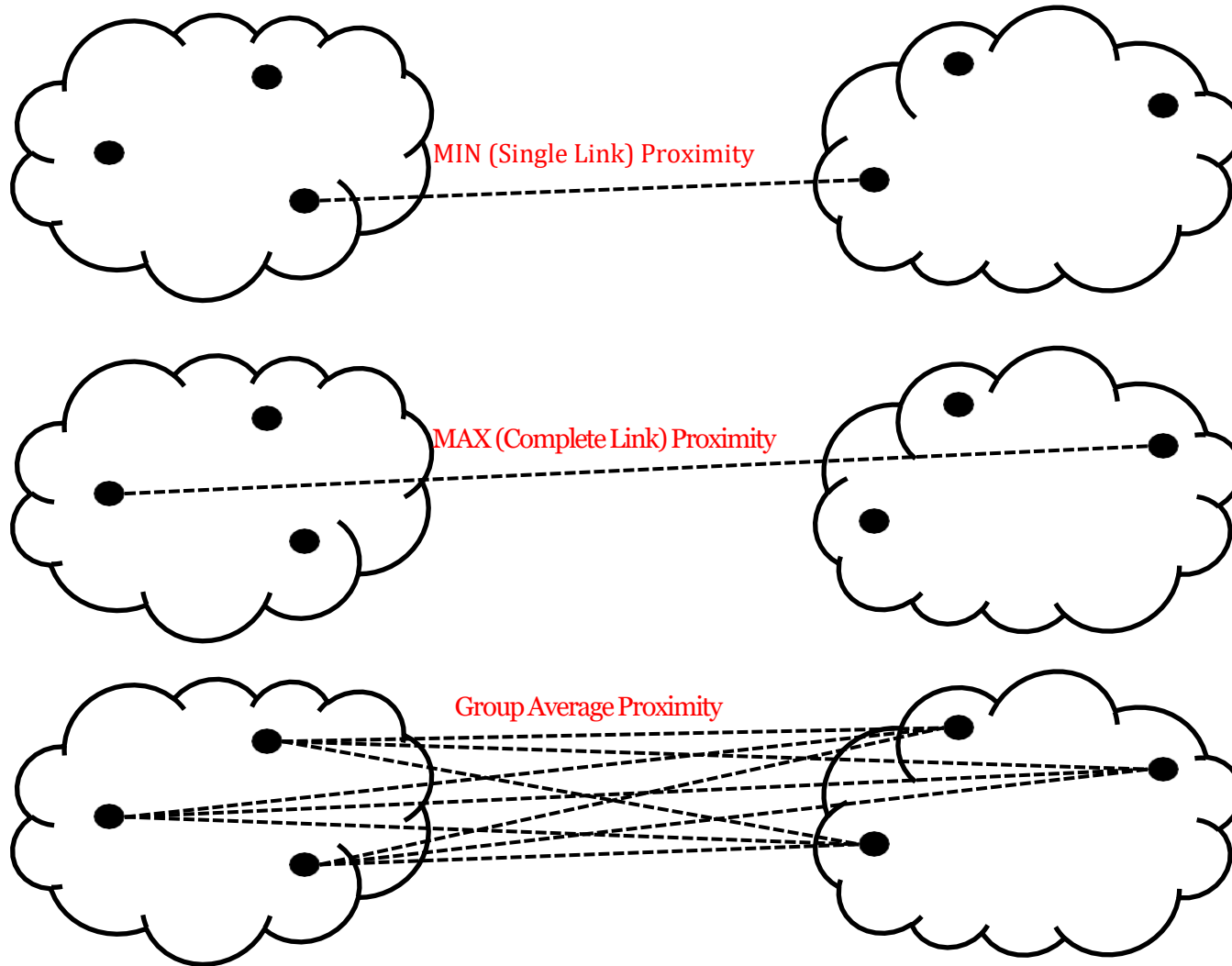
- MAX (Complete Link) Proximity
- Defines cluster proximity as **the furthest distance between two points,  $x$  and  $y$ ,** that are in different clusters,  $A$  and  $B$ :

$$d(A, B) = \max_{x \in A, y \in B} d(x - y)$$

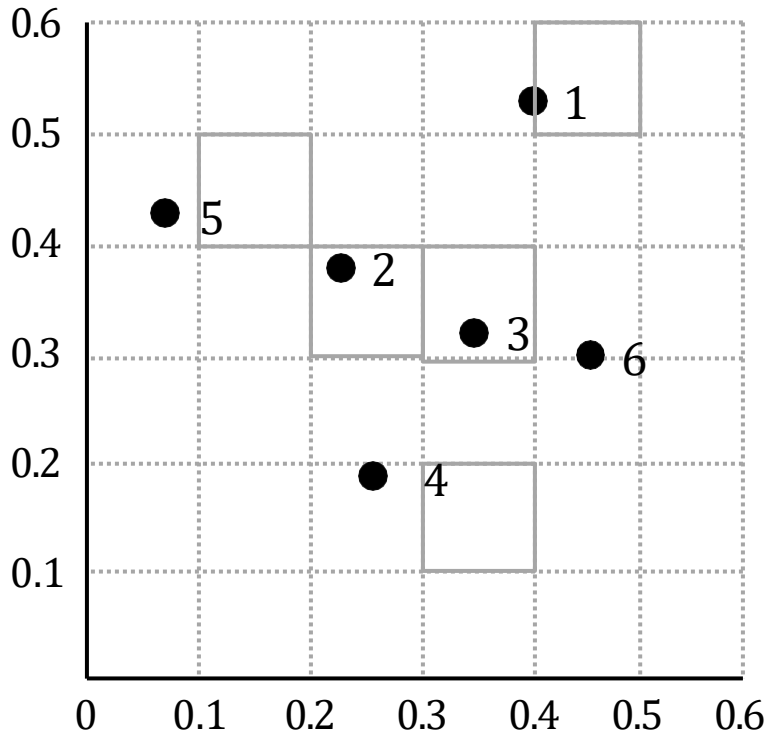
- 평균연결법(Average Linkage Method)

- Group Average Proximity
- Defines cluster proximity as **the average distance between two points,  $x$  and  $y$ ,** that are in different clusters,  $A$  and  $B$  (number of points in cluster  $j$  is  $n_j$ ):

$$d(A, B) = \sum_{x \in A, y \in B} d(x - y) n_A n_B$$



## Example

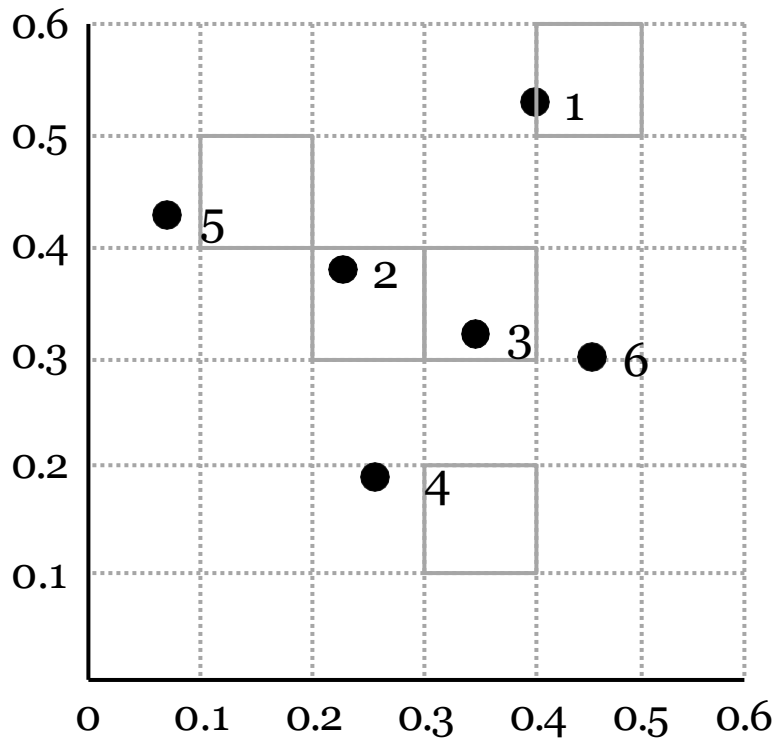


Point	$x$ Coordinate	$y$ Coordinate
p1	0.40	0.53
p2	0.22	0.38
p3	0.35	0.32
p4	0.26	0.19
p5	0.08	0.41
p6	0.45	0.30

# Euclidean Distance

39

Set of 6 Two-Dimensional Points



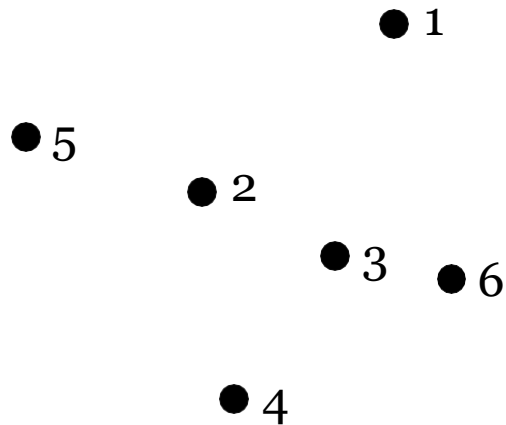
Euclidean Distance Matrix for 6 Points

	p1	p2	p3	p4	p5	p6
p1	0.00	0.24	0.22	0.37	0.34	0.23
p2	0.24	0.00	0.15	0.20	0.14	0.25
p3	0.22	0.15	0.00	0.15	0.28	0.11
p4	0.37	0.20	0.15	0.00	0.29	0.22
p5	0.34	0.14	0.28	0.29	0.00	0.39
p6	0.23	0.25	0.11	0.22	0.39	0.00

# Single Link Clustering(1)

40

Nested Cluster Diagram



Single Link Distance Matrix

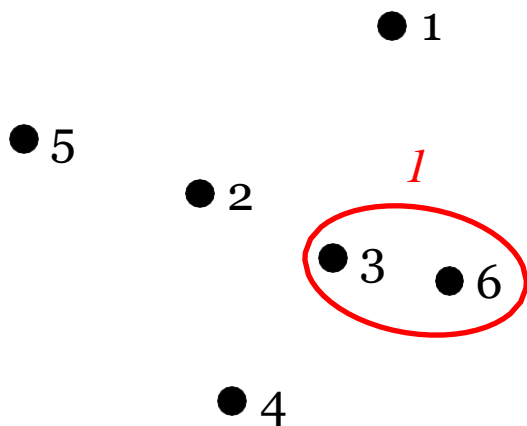
	1	2	3	4	5	6
1	0	0.24	0.22	0.37	0.34	0.23
2		0	0.15	0.20	0.14	0.25
3			0	0.15	0.28	0.11
4				0	0.29	0.22
5					0	0.39
6						0



## Single Link Clustering(2)

41

Nested Cluster Diagram



Single Link Distance Matrix

	1	2	3	4	5	6
1	0	0.24	<u>0.22</u>	0.37	0.34	<u>0.23</u>
2		0	<u>0.15</u>	0.20	0.14	<u>0.25</u>
3			0	<u>0.15</u>	<u>0.28</u>	<b>0.11</b>
4				0	<u>0.29</u>	<u>0.22</u>
5					0	<u>0.39</u>
6						0

Points 3 and 6 have the smallest single link proximity distance.

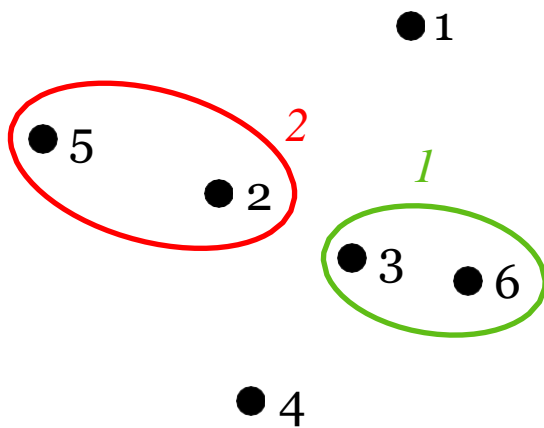
Merge these points into one cluster and update the distances to this new cluster.

For example, the distance from point 1 to this cluster is 0.22 (the distance to point 3).

# Single Link Clustering(3)

42

Nested Cluster Diagram



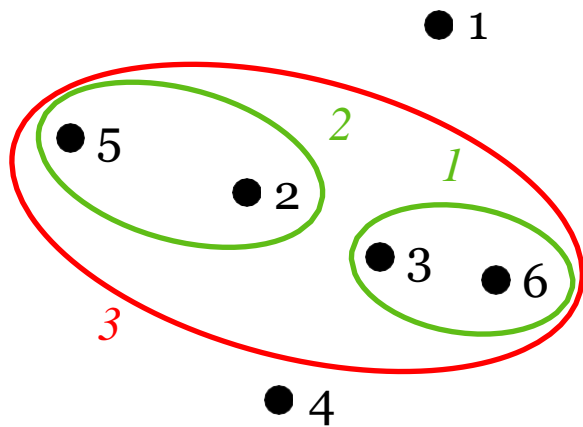
Single Link Distance Matrix

	1	2	4	5	3,6
1	0	<u>0.24</u>	0.37	<u>0.34</u>	0.22
2		0	<u>0.20</u>	0.14	0.15
4			0	<u>0.29</u>	0.15
5				0	0.28
3,6					0

# Single Link Clustering(4)

43

Nested Cluster Diagram



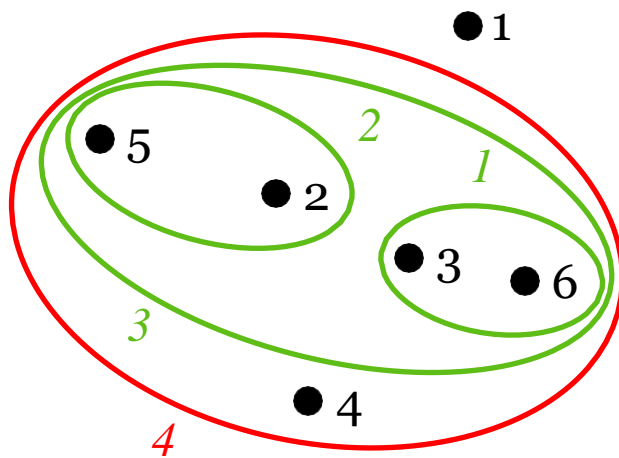
Single Link Distance Matrix

	1	4	2,5	3,6
1	0	0.37	<u>0.24</u>	<u>0.22</u>
4		0	<u>0.20</u>	<u>0.15</u>
2,5			0	0.15
3,6				0

# Single Link Clustering(5)

44

Nested Cluster Diagram



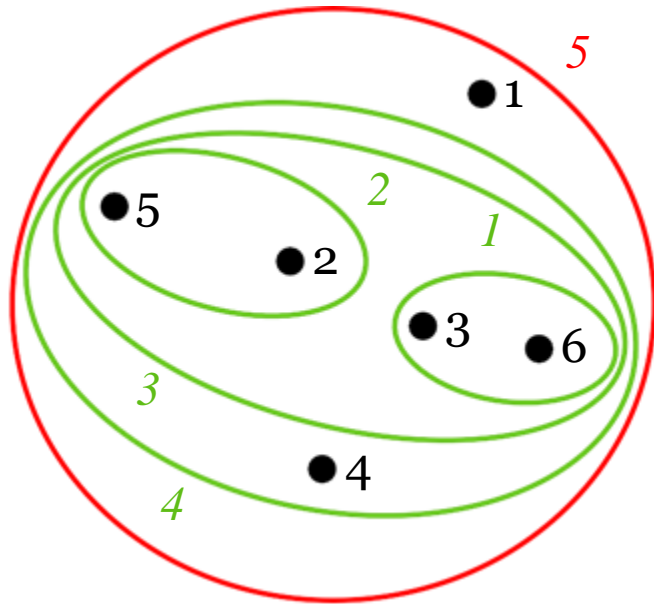
Single Link Distance Matrix

	1	4	2,5,3,6
1	0	<u>0.37</u>	<u>0.22</u>
4		0	0.15
2,5,3,6			0

# Single Link Clustering(6)

45

Nested Cluster Diagram

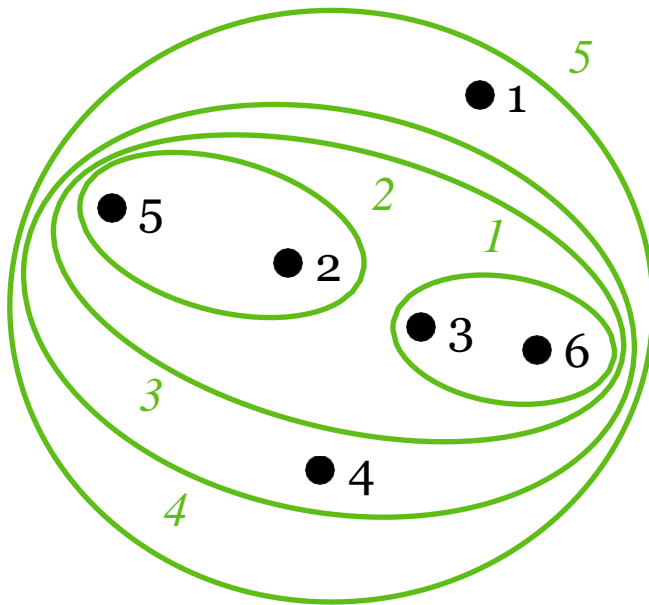


Single Link Distance Matrix

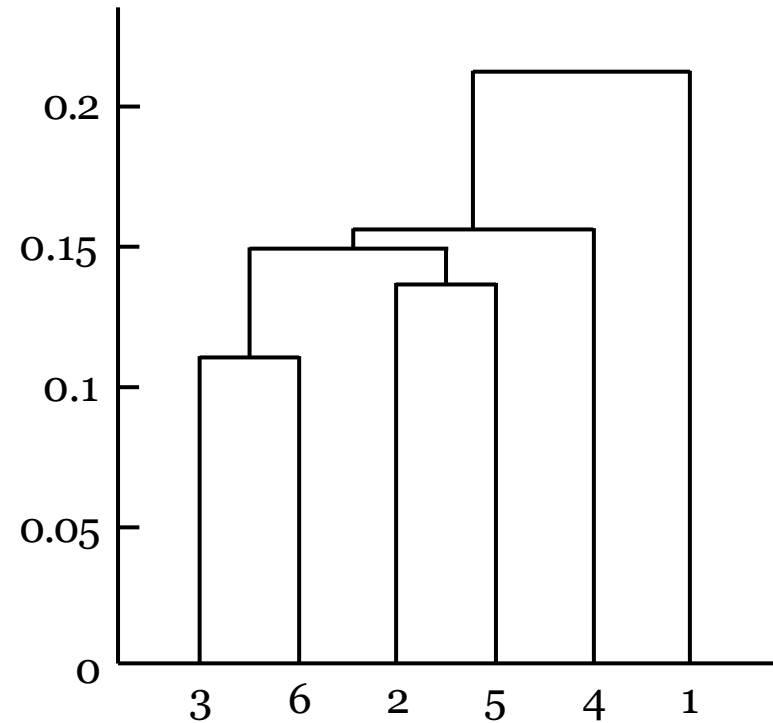
	1	4,2,5,3,6
1	0	0.22
2,5,3,6		0

And iterate until there is one all-inclusive cluster.

Nested Cluster Diagram



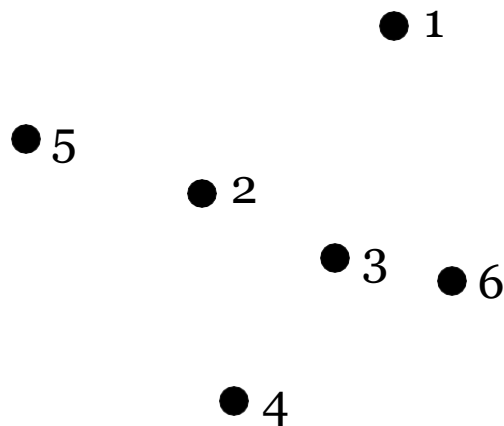
Hierarchical Tree Diagram



# Complete Link Clustering(1)

47

Nested Cluster Diagram



Complete Link Distance Matrix

	1	2	3	4	5	6
1	0	0.24	0.22	0.37	0.34	0.23
2		0	0.15	0.20	0.14	0.25
3			0	0.15	0.28	0.11
4				0	0.29	0.22
5					0	0.39
6						0

Points 3 and 6 have the smallest complete link proximity distance.

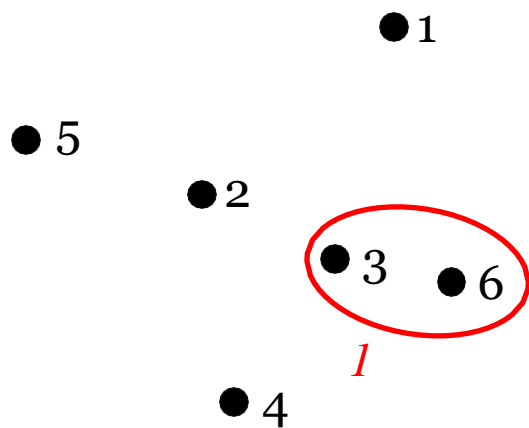
Merge these points into one cluster and update the distances to this new cluster.

For example, the distance from point 1 to this cluster is 0.23 (the distance to point 6).

## Complete Link Clustering(2)

48

Nested Cluster Diagram



Complete Link Distance Matrix

	1	2	3	4	5	6
1	0	0.24	<u>0.22</u>	0.37	0.34	<u>0.23</u>
2		0	<u>0.15</u>	0.20	0.14	<u>0.25</u>
3			0	<u>0.15</u>	<u>0.28</u>	0.11
4				0	0.29	<u>0.22</u>
5					0	<u>0.39</u>
6						0

Points 3 and 6 have the smallest complete link proximity distance.

Merge these points into one cluster and update the distances to this new cluster.

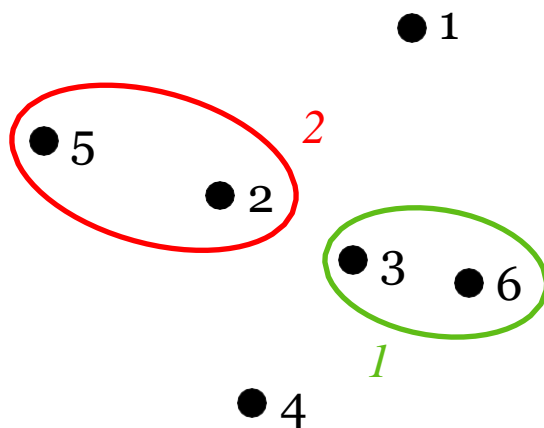
For example, the distance from point 1 to this cluster is 0.23 (the distance to point 6).



# Complete Link Clustering(3)

49

Nested Cluster Diagram



Complete Link Distance Matrix

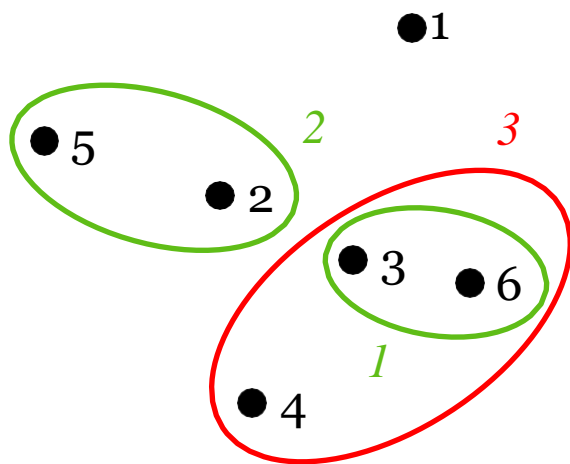
	1	2	4	5	3,6
1	0	<u>0.24</u>	0.37	<u>0.34</u>	0.23
2		0	<u>0.20</u>	0.14	0.25
4			0	<u>0.29</u>	0.22
5				0	0.39
3,6					0

Points 2 and 5 have the smallest complete link proximity distance.  
Merge these points into one cluster and update the distances to this new cluster.

## Complete Link Clustering(4)

50

Nested Cluster Diagram



And iterate...

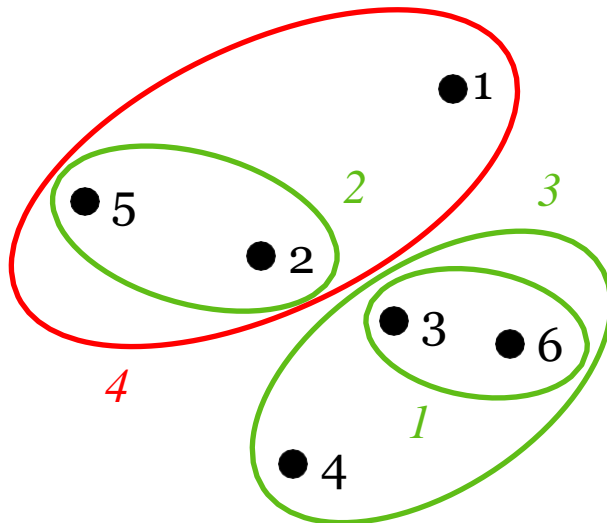
Complete Link Distance Matrix

	1	4	2,5	3,6
1	0	<u>0.37</u>	0.34	<u>0.23</u>
4		0	<u>0.29</u>	<b>0.22</b>
2,5			0	<u>0.39</u>
3,6				0

# Complete Link Clustering(5)

51

Nested Cluster Diagram



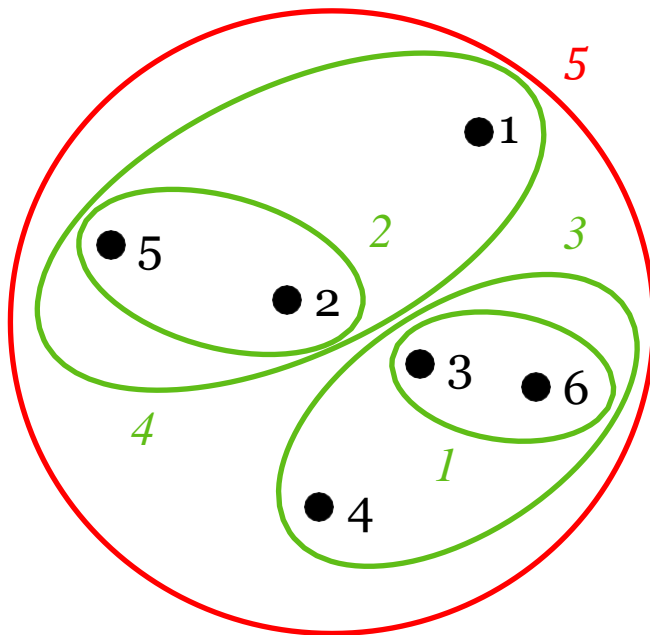
Complete Link Distance Matrix

	1	2,5	4,3,6
1	0	0.34	<u>0.37</u>
2,5		0	<u>0.39</u>
4,3,6			0

# Complete Link Clustering(6)

52

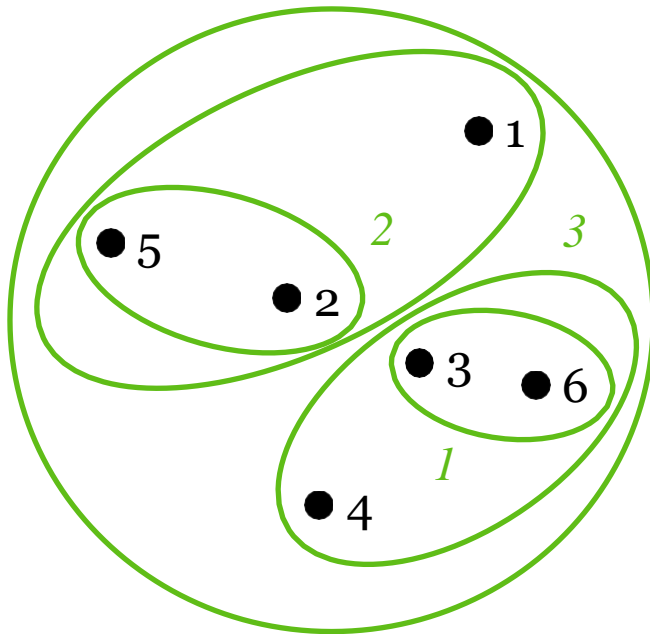
Nested Cluster Diagram



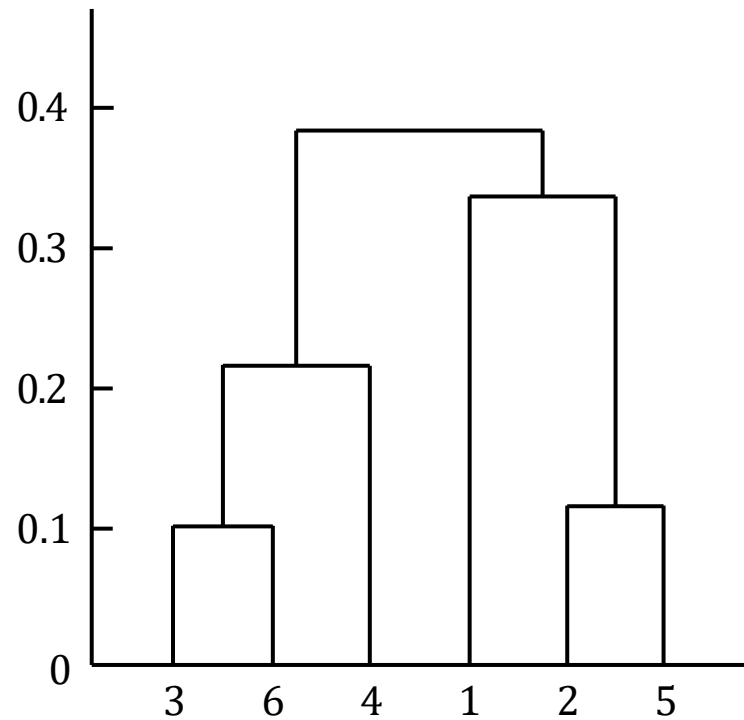
Complete Link Distance Matrix

	1,2,5	4,3,6
1,2,5	0	0.39
4,3,6		0

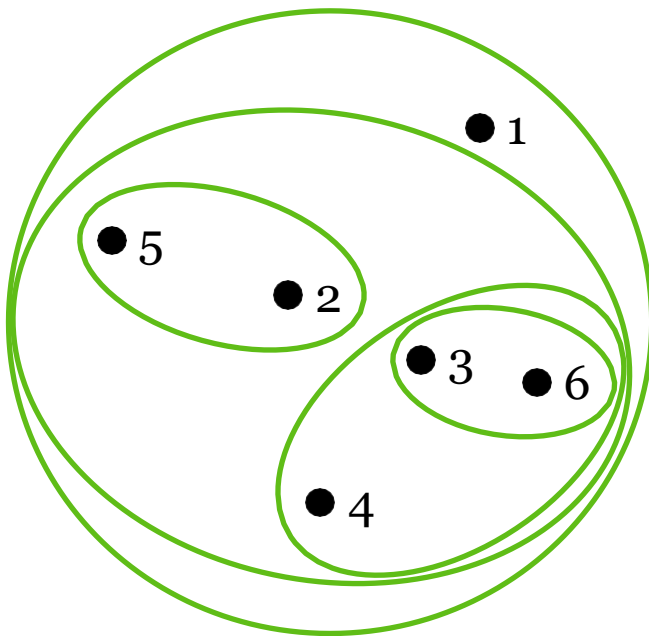
Nested Cluster Diagram



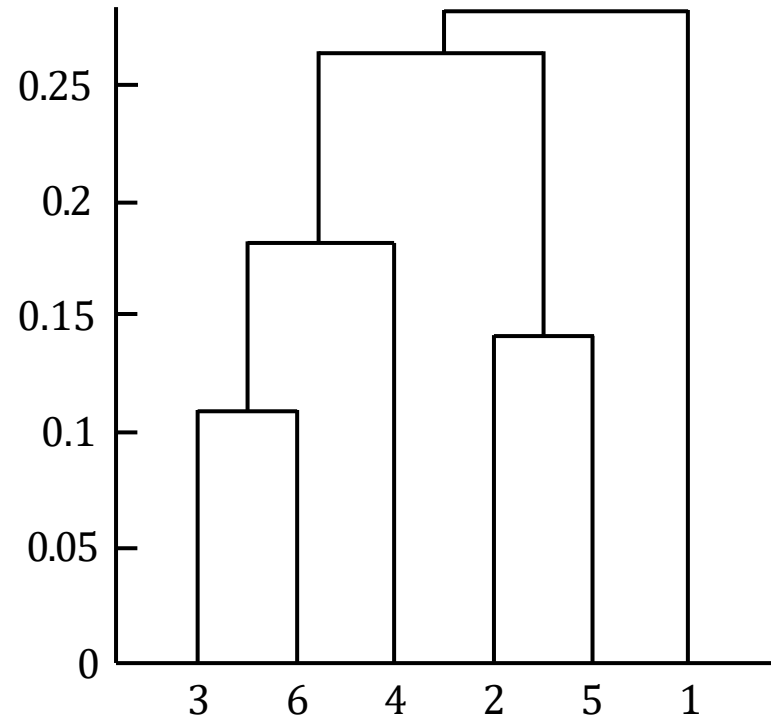
Hierarchical Tree Diagram



## Nested Cluster Diagram



## Hierarchical Tree Diagram



# Ward Linkage Method

55

```
!pip3 install mglearn
```

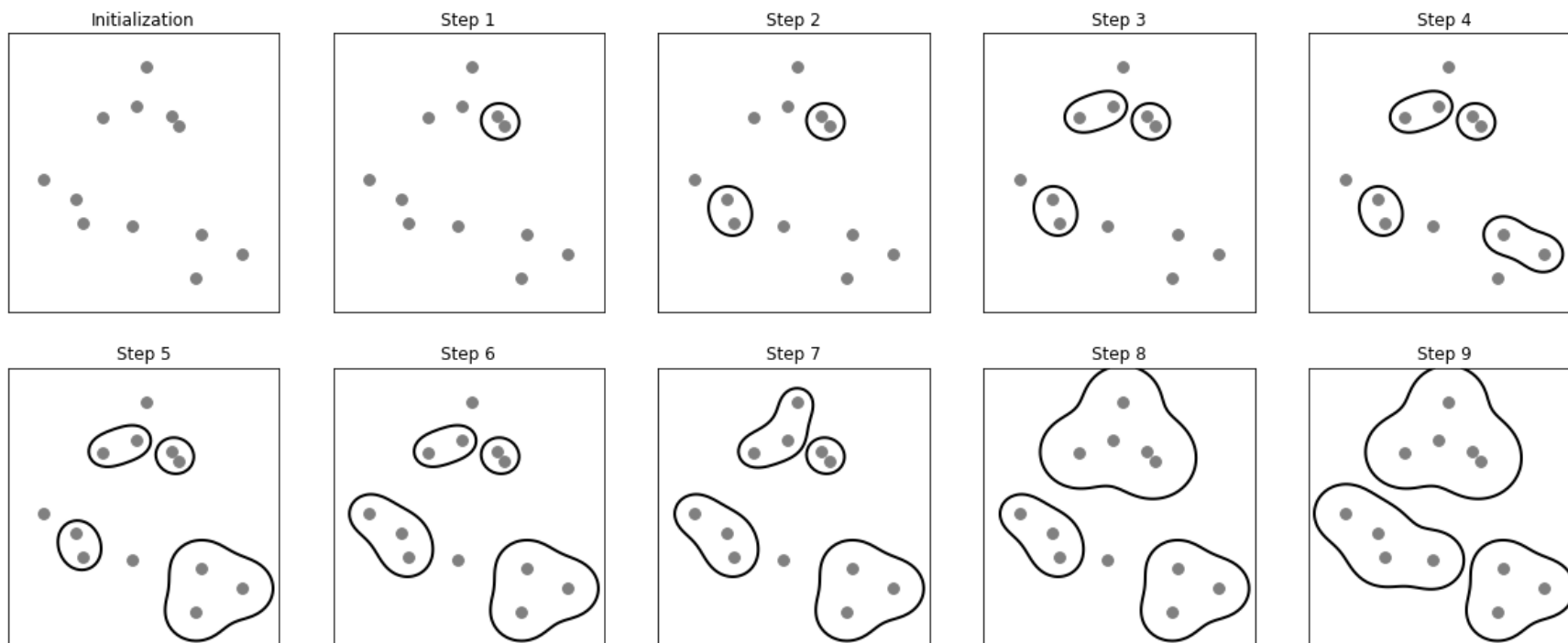
```
import mglearn as mglearn
```

```
# ward linkage method
```

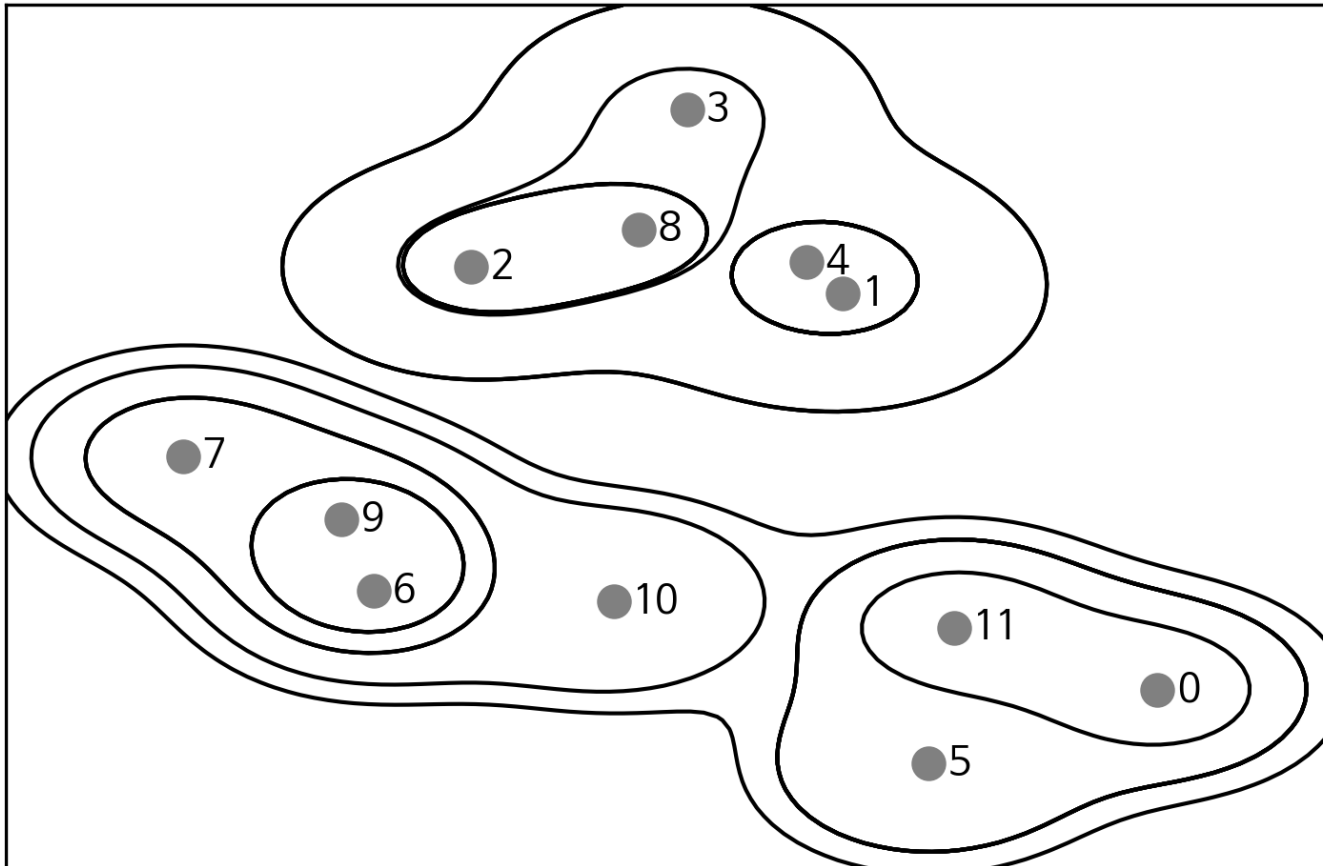
```
# 각 병합 단계에서 오차제곱합 (ESS) 의 증분이 최소가 되는 군집들을 병합
```

```
# 세개의 클러스터 찾기
```

```
mglearn.plots.plot_agglomerative_algorithm()
```



```
mglearn.plots.plot_agglomerative()
```





```
# ward 군집 함수와 덴드로그램 함수를 импорт
from scipy.cluster.hierarchy import dendrogram, ward

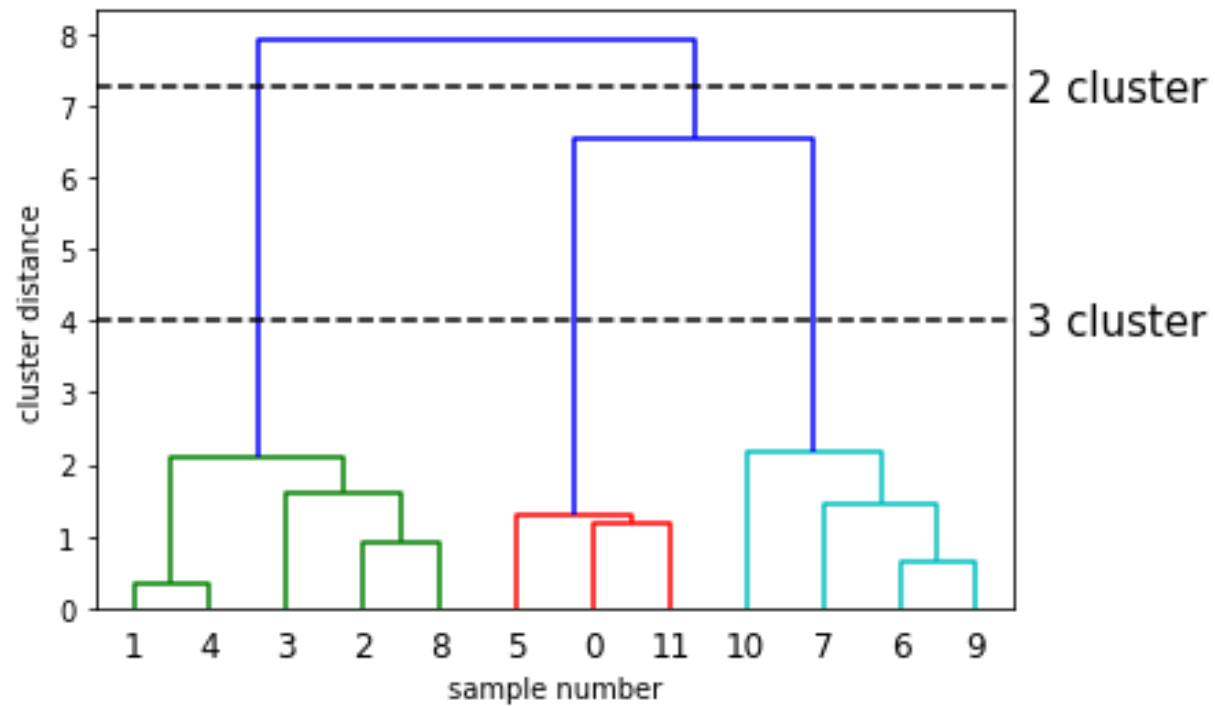
X, y = make_blobs(random_state=0, n_samples=12)

# 데이터 배열 X 에 ward 함수를 적용
# SciPy의 ward 함수는 병합 군집을 수행할 때 생성된 거리 정보가 담긴 배열을 리턴
linkage_array = ward(X)

# 클러스터 간의 거리 정보가 담긴 linkage_array를 사용해 덴드로그램을 그린다
dendrogram(linkage_array)

# 두 개와 세 개의 클러스터를 구분하는 커트라인 표시
ax = plt.gca()
bounds = ax.get_xbound()
ax.plot(bounds, [7.25, 7.25], '--', c='k')
ax.plot(bounds, [4, 4], '--', c='k')

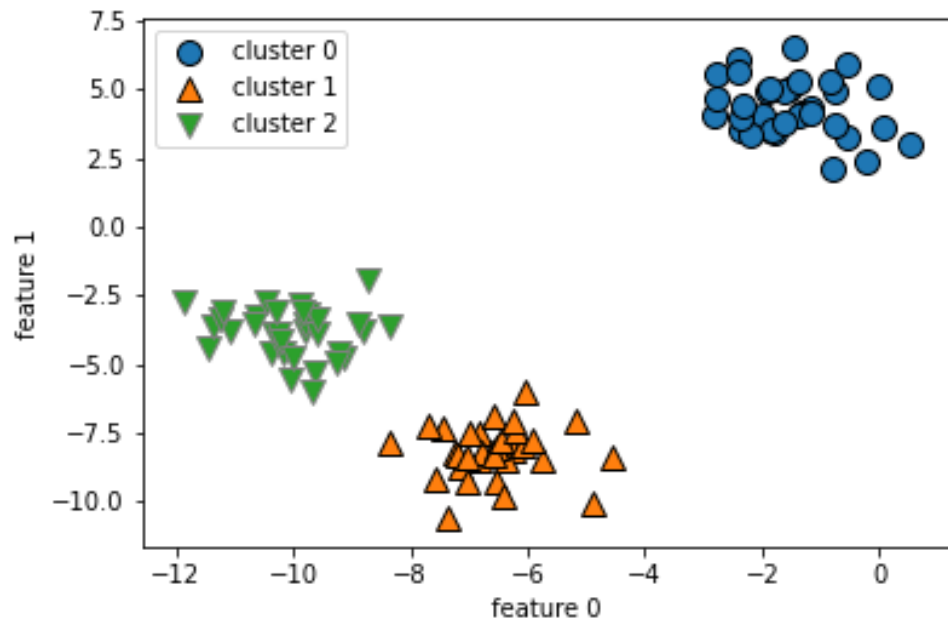
ax.text(bounds[1], 7.25, ' 2 cluster', va='center', fontdict={'size': 15})
ax.text(bounds[1], 4, ' 3 cluster', va='center', fontdict={'size': 15})
plt.xlabel("sample number")
plt.ylabel("cluster distance")
```




```
from sklearn.cluster import AgglomerativeClustering
from sklearn.datasets import make_blobs
import matplotlib.pyplot as plt
X, y = make_blobs(random_state=1)

agg = AgglomerativeClustering(n_clusters=3)
assignment = agg.fit_predict(X) # 클러스터 소속 정보

mglearn.discrete_scatter(X[:, 0], X[:, 1], assignment)
plt.legend(["cluster 0", "cluster 1", "cluster 2"], loc="best")
plt.xlabel("feature 0")
plt.ylabel("feature 1")
```

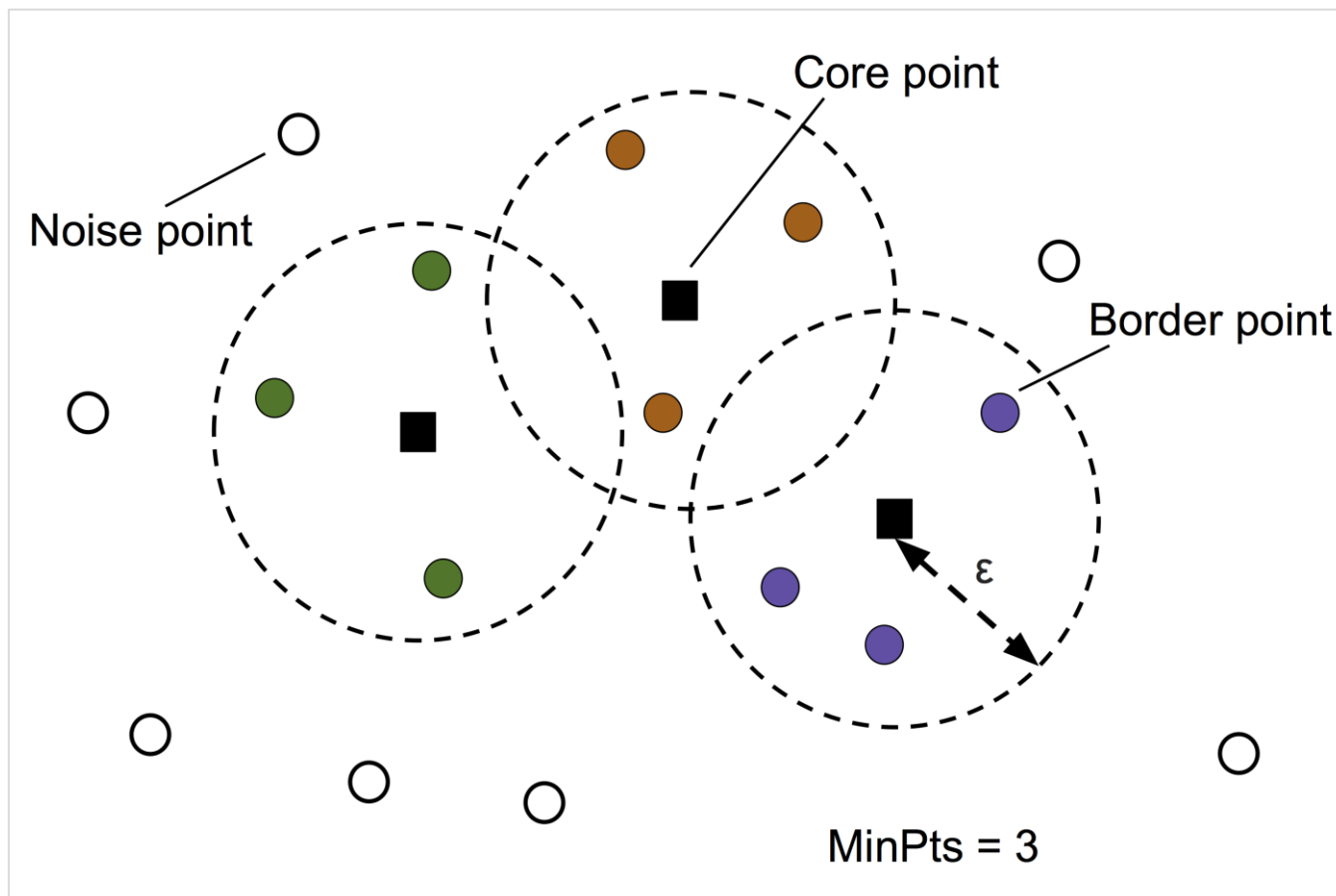




군집화(Clustering)란?  
K-평균 알고리즘(K-Means Algorithm)  
계층형 군집화(Hierarchical Clustering)  
**DBSCAN**

- DBSCAN : *Density – Based Spatial Clustering of Application with Noise*
- 아이디어 : 데이터의 밀집지역이 한 클러스터를 구성하며 비교적 비어 있는 지역을 경계로 다른 클러스터와 구분된다는 것
- 장점
  - 클러스터의 개수를 미리 지정할 필요가 없다
  - 복잡한 형상도 찾을 수 있으며, 어떤 클래스에도 속하지 않는 포인트(잡음, 이상치)를 구분할 수 있다.
  - 모든 클러스터가 충분히 밀집되어 있고 밀집되지 않은 지역과 잘 구분될 때 좋은 성능을 나타낸다.
  - k-means 보다는 다소 느리지만 비교적 큰 데이터셋에도 적용할 수 있다.
- 밀집도 : 특정 반경  $\epsilon$  내에 있는 샘플 개수로 정의
- 핵심샘플(*core point, core instance*)
  - $\epsilon$ -이웃 내에 적어도 지정된 개수(*MinPts, min\_samples*)의 샘플
  - 밀집된 지역에 있는 샘플
- 경계샘플(*border point*)
  - $\epsilon$ -이웃 내에 (*MinPts, min\_samples*)보다 이웃이 적지만 다른 핵심 샘플의 반경  $\epsilon$  안에 있는 샘플
- 잡음샘플(*noise point*) or 이상치
  - 핵심샘플과 경계샘플이 아닌 다른 모든 샘플

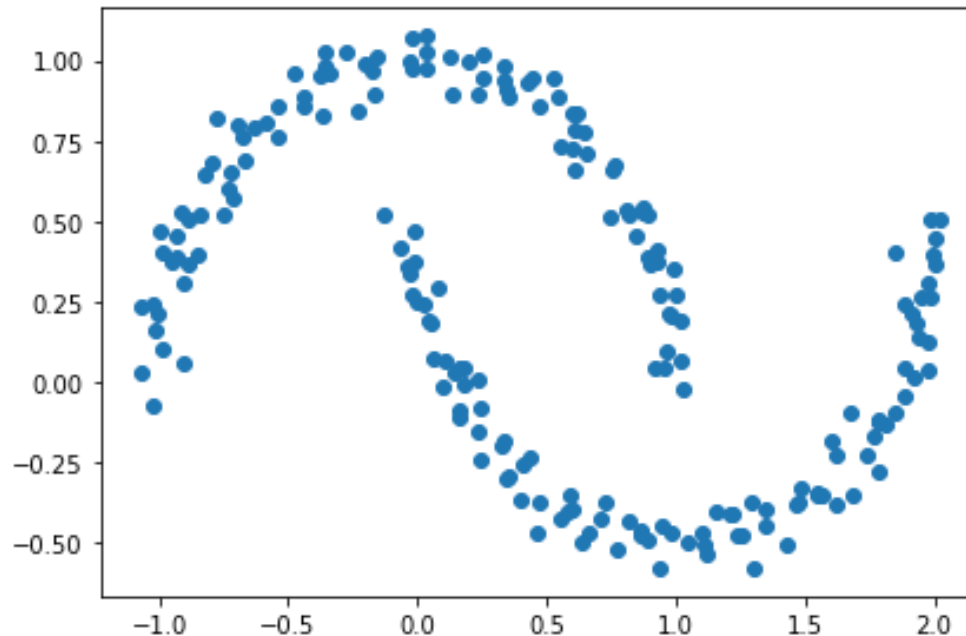
- 핵심샘플의 이웃에 있는 모든 샘플은 동일한 클러스터에 속한다.
- 이웃에는 다른 핵심 샘플이 포함될 수 있다.
- 따라서 핵심 샘플의 이웃의 이웃은 계속해서 하나의 클러스터를 형성한다



```
# 반달 모양 형태의 데이터셋 생성
```

```
from sklearn.datasets import make_moons  
import matplotlib.pyplot as plt
```

```
X, y = make_moons(n_samples=200, noise=0.05, random_state=0)  
plt.scatter(X[:, 0], X[:, 1])  
plt.tight_layout()  
plt.show()
```



```
from sklearn.cluster import KMeans
from sklearn.cluster import AgglomerativeClustering

f, (ax1, ax2) = plt.subplots(1, 2, figsize=(8, 3))

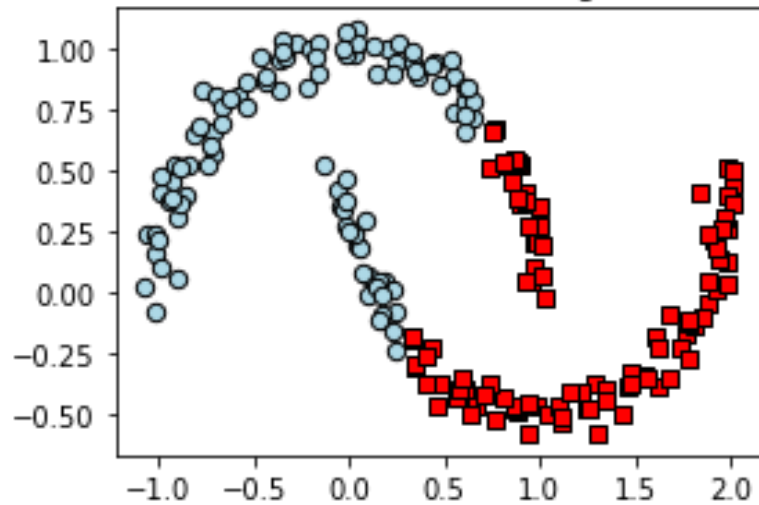
km = KMeans(n_clusters=2, random_state=0)
y_km = km.fit_predict(X)
ax1.scatter(X[y_km == 0, 0], X[y_km == 0, 1], edgecolor='black',
            c='lightblue', marker='o', s=40, label='cluster 1')
ax1.scatter(X[y_km == 1, 0], X[y_km == 1, 1], edgecolor='black',
            c='red', marker='s', s=40, label='cluster 2')
ax1.set_title('K-means clustering')

ac = AgglomerativeClustering(n_clusters=2, affinity='euclidean',
                             linkage='complete')
y_ac = ac.fit_predict(X)
ax2.scatter(X[y_ac == 0, 0], X[y_ac == 0, 1], c='lightblue',
            edgecolor='black', marker='o', s=40, label='cluster 1')
ax2.scatter(X[y_ac == 1, 0], X[y_ac == 1, 1], c='red', edgecolor='black',
            marker='s', s=40, label='cluster 2')
ax2.set_title('Agglomerative clustering')

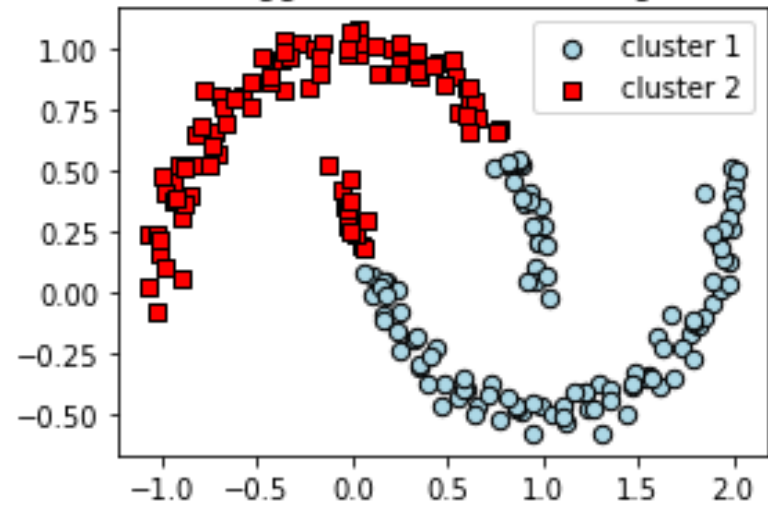
plt.legend()
plt.tight_layout()
plt.show()
```



K-means clustering



Agglomerative clustering



```
from sklearn.cluster import DBSCAN

db = DBSCAN(eps=0.2, min_samples=5, metric='euclidean')
y_db = db.fit_predict(X)
plt.scatter(X[y_db == 0, 0], X[y_db == 0, 1],
            c='lightblue', marker='o', s=40,
            edgecolor='black',
            label='cluster 1')
plt.scatter(X[y_db == 1, 0], X[y_db == 1, 1],
            c='red', marker='s', s=40,
            edgecolor='black',
            label='cluster 2')

plt.legend()
plt.tight_layout()
plt.show()
```

