Clustering: Part 1

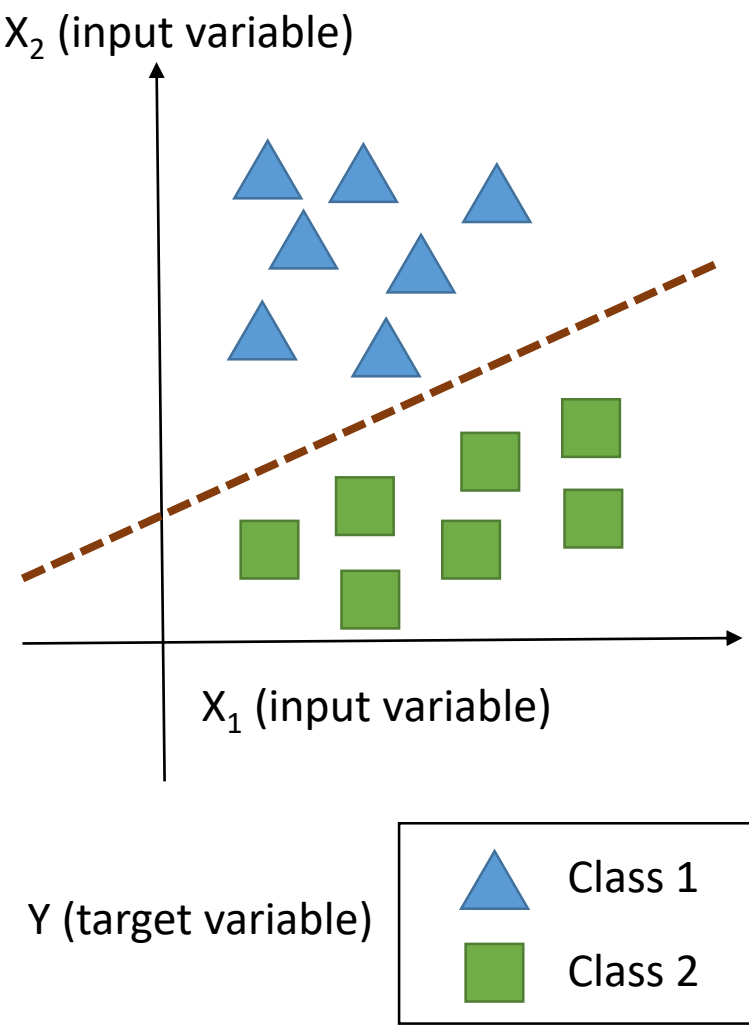
Taehoon Ko (taehoonko@snu.ac.kr)

지도학습 vs. 비지도학습

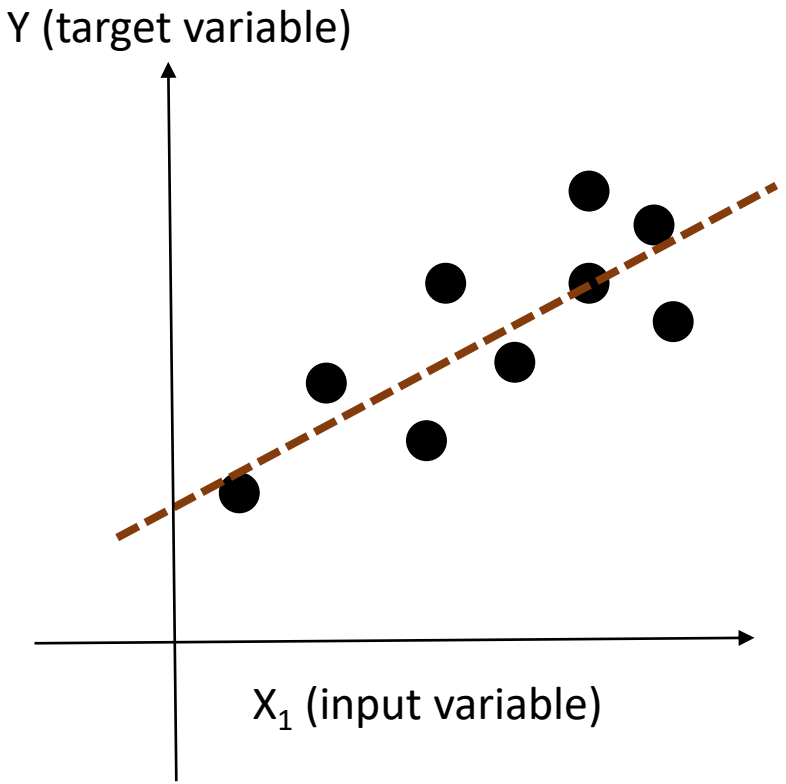
- **Supervised learning (지도학습)**
  - **정답을 알고 있는**, 즉 **타겟변수가 있는** 포인트들을 학습
  - 예) 엄마가 옆에서 저 동물이 개 혹은 고양이인지 알려줄 때 이루어지는 학습
- **Unsupervised learning (비지도학습)**
  - **정답을 모르는**, 즉 **타겟변수가 없는** 포인트들을 학습
  - 예) 그 누구도 저 동물이 개 혹은 고양이인지 알려주지 않을 때 이루어지는 학습

Supervised learning (지도학습)

분류 (Classification)



회귀 (Regression)



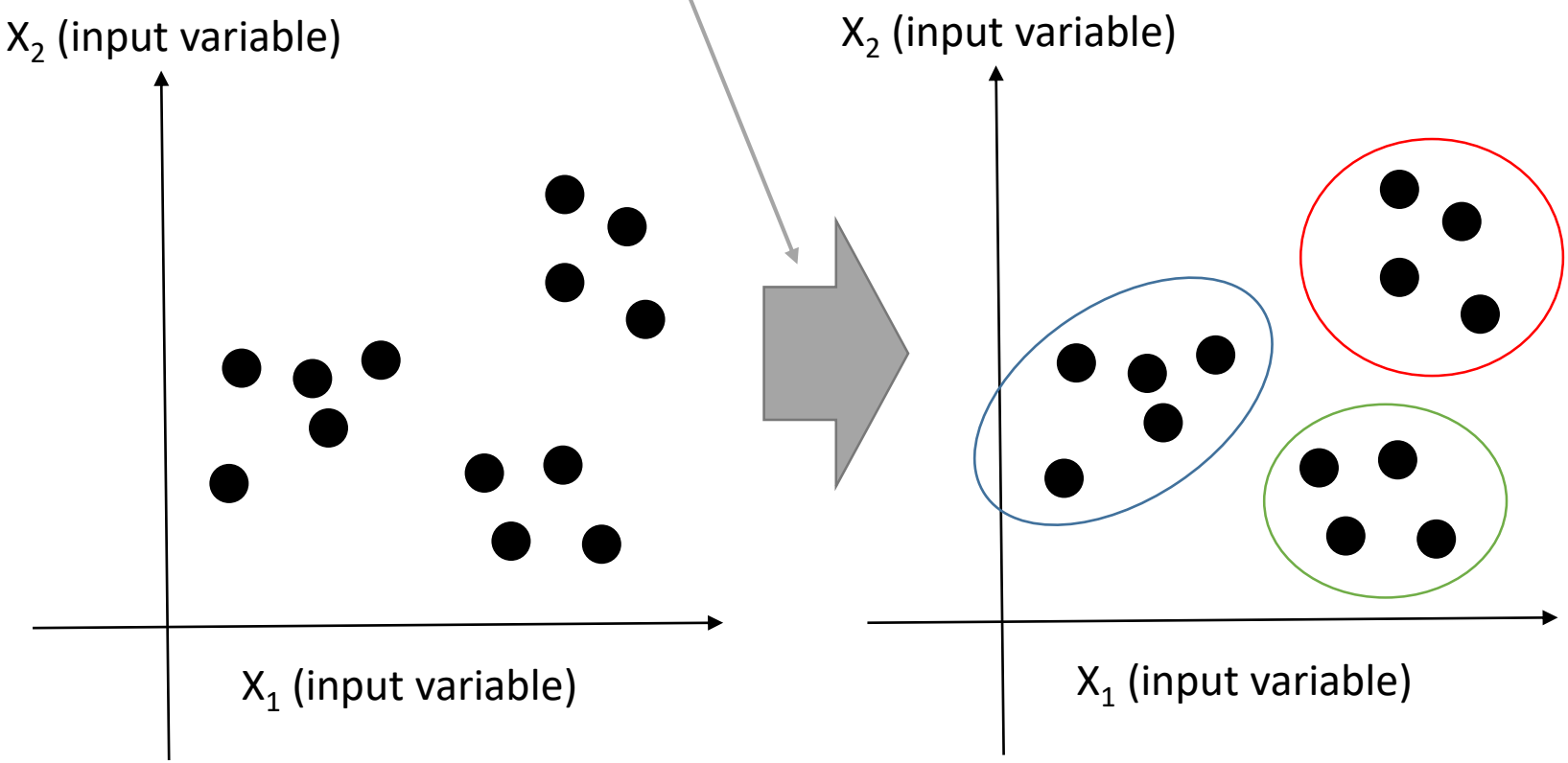
Unsupervised learning: Clustering

Learning algorithm

Find an optimal structure of clusters.

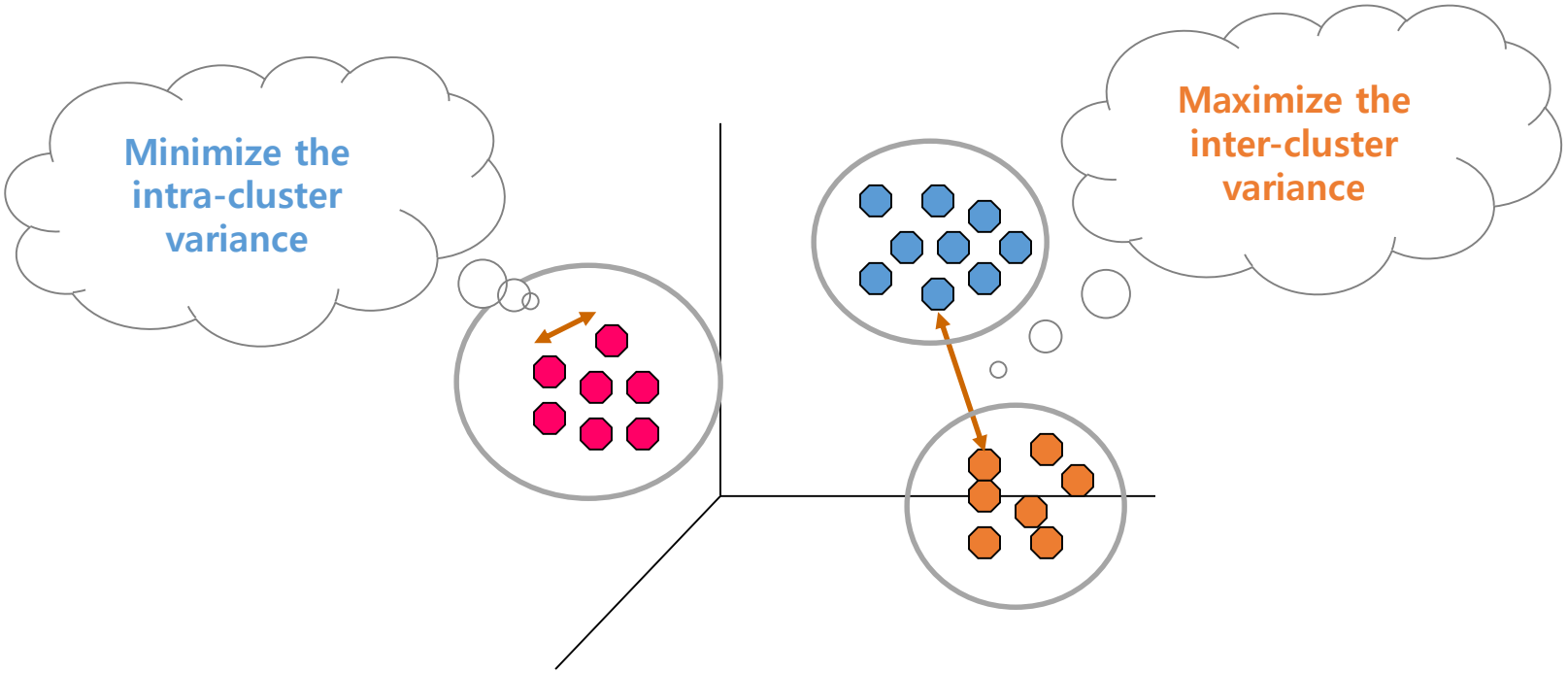
Model or ***Structure of clusters***

Cluster IDs for each point



Clustering (군집화)

- **군집화는 데이터에서 비슷한 객체들을 하나의 그룹으로 묶는 것**
  - 각 객체들이 어떤 군집으로 할당되어야 하는가에 대한 정보(y)가 없기 때문에 unsupervised 알고리즘에 해당
  - 그러므로 군집화 방법들은 각 객체들의 유사도(거리) 정보를 기반으로 작동
  - Find **groups of objects** such that the **objects in a group will be similar (or related) to one another** and **different from (or unrelated to) the objects in other groups**



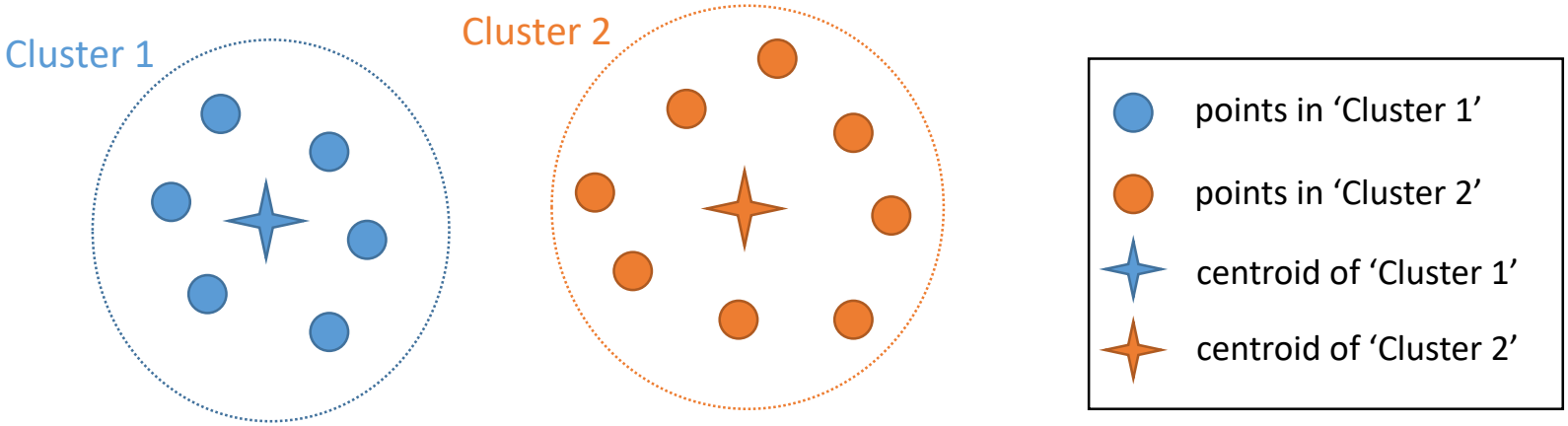
Clustering algorithms

- **k-means clustering**
- **(Agglomerative) hierarchical clustering**
- **Others**
  - Density-based spatial clustering of applications with noise (DBSCAN)
  - Gaussian mixture model
  - Self-organizing map (SOM)

k-means clustering

k-means clustering

- **분할 군집화 (Partitional clustering) 방법 중 하나**
  - 하나의 포인트는 반드시 하나의 군집에만 소속됨.
- **군집의 생성 형태**
  - 각 군집은 하나의 중심을 갖고 있으며, 이를 centroid라고 부름.
  - 각 포인트는 가장 가까운 centroid에 해당하는 군집에 소속됨.
- **k?**
  - 군집의 개수. 이를 학습할 시 반드시 사용자가 정의해야 함.



k-means clustering

- **유사도**
  - n개의 데이터 X에 대하여 두 데이터 x_i, x_j간에 정의되는 임의의 거리 $d(x_i, x_j)$
    - 유클리디언, 코싸인 등 벡터에서 정의되는 모든 거리 척도
- **알고리즘**
  - 목표: 각 포인트와 가까이 있는 centroid와의 거리제곱이 최소가 되는 군집화 구조를 찾는 것
  - 오른쪽과 같이 근사적인 방법으로 군집화 수행

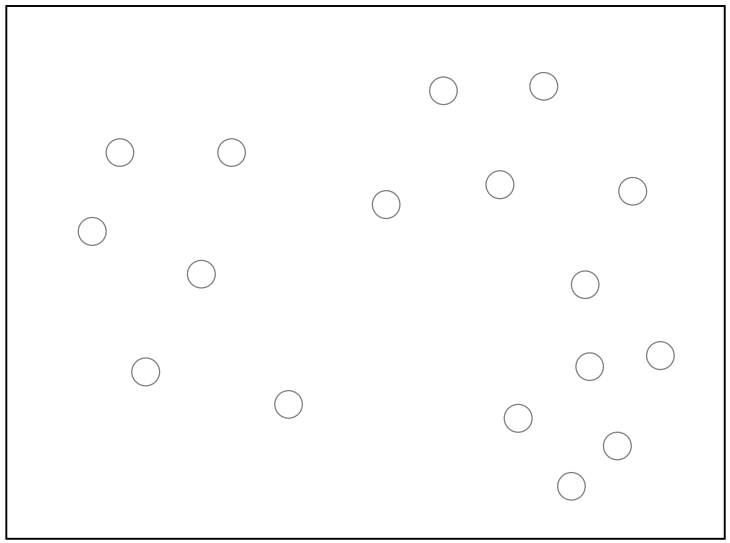
<목적함수>

$$\arg\min_{\mathbf{C}} \sum_{i=1}^{K} \sum_{\mathbf{x}_j \in C_i} ||\mathbf{x}_j - \mathbf{c}_i||^2$$

<(근사적) 해결 방법>

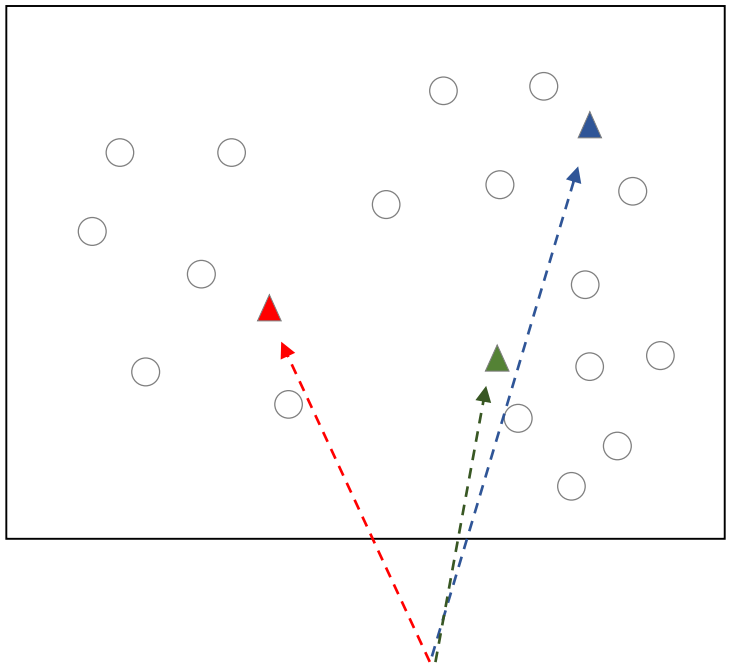
1: Select K points as the initial centroids.
2: **repeat**
3: Form K clusters by assigning all points to the closest centroid.
4: Recompute the centroid of each cluster.
5: **until** The centroids don't change

k-means clustering



0. Data

k-means clustering

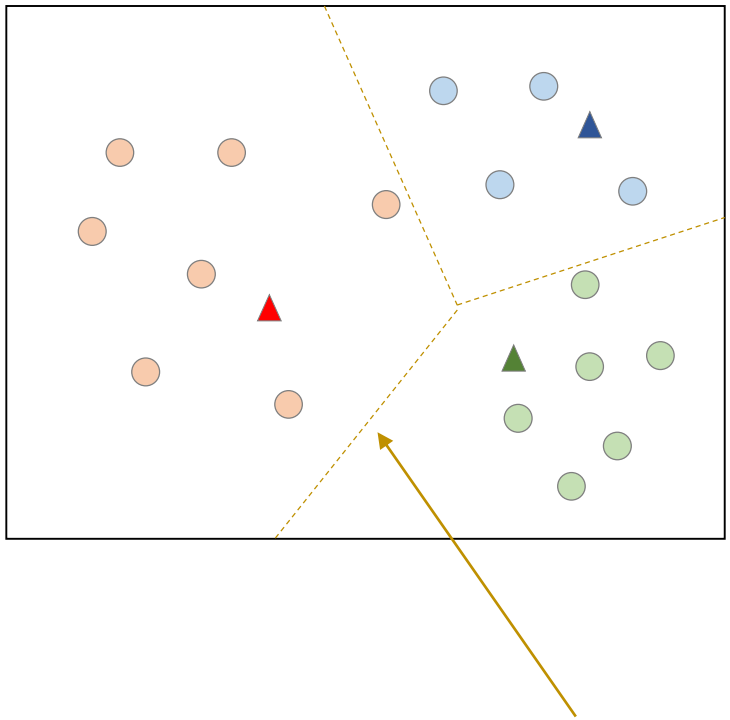


1. Initialize

k=3 이라 가정하면 (데이터 입력 공간에서)
3개의 점을 임의로 선택

데이터에는 존재하지 않는 가상의 centroids

k-means clustering



1. Initialize

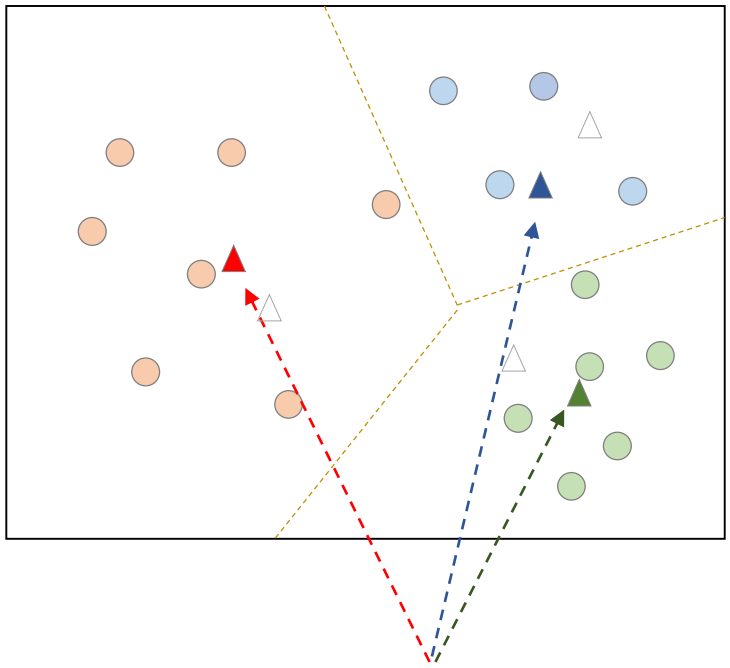
k=3 이라 가정하면 3개의 점을 임의로 선택

2. Assign (epoch=0)

모든 점을 *k*개의 centroid 중 가장 가까운 점의 색깔(label)로 할당

*k*개의 centroids에 의하여 분할된 공간의 경계면으로,
Voronoi partition, Voronoi diagram이라 부름

k-means clustering



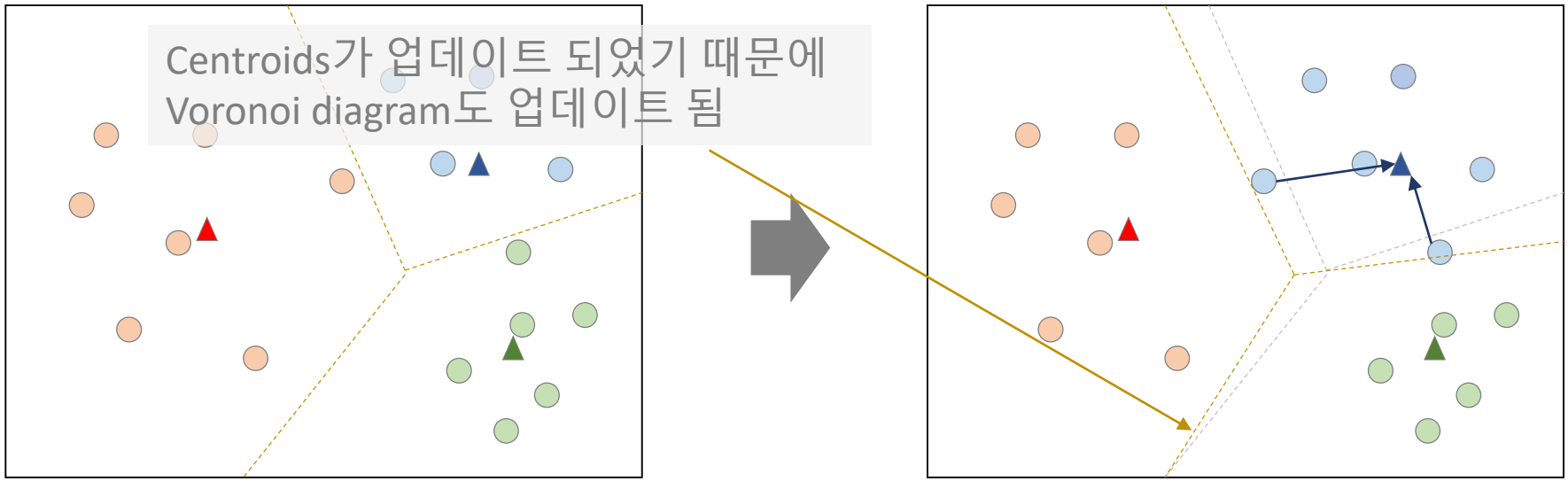
1. Initialize
k=3 이라 가정하면 3개의 점을 임의로 선택

2. Assign (epoch=0)
모든 점을 *k*개의 centroid 중 가장 가까운 점의 색깔(label)로 할당

3. Update centroid (epoch=0)
같은 색깔(label) 점들의 평균값을 가상의 centroids로 설정 ➔ centroid의 업데이트

데이터에는 존재하지 않는 가상의 centroids

k-means clustering



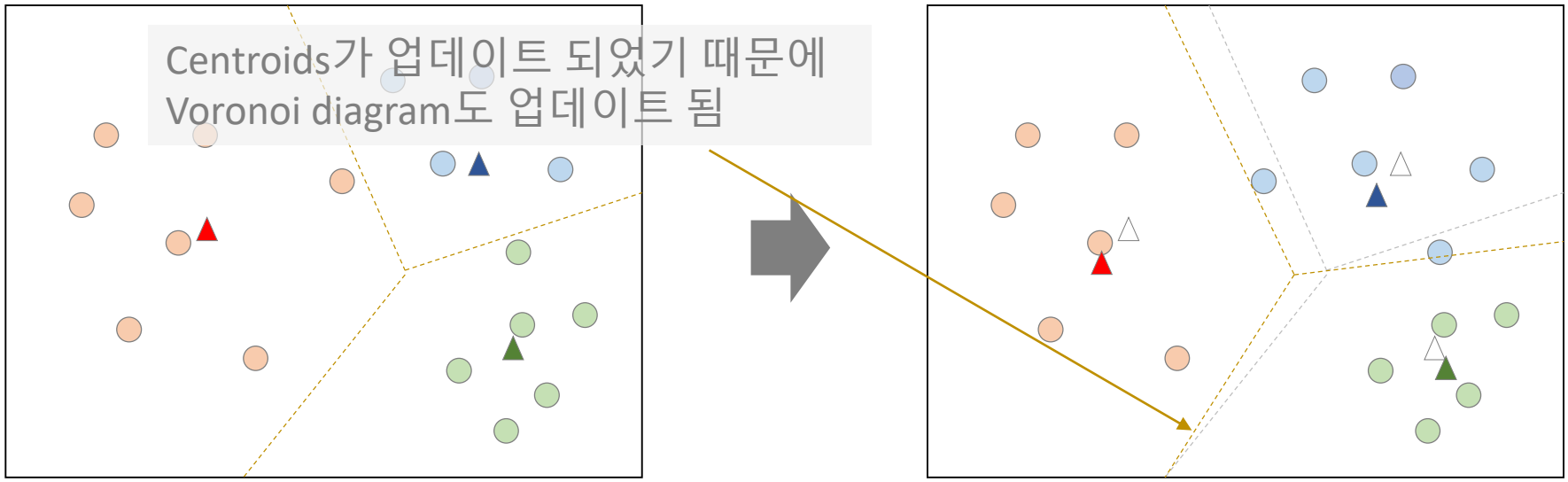
1. Initialize

k=3 이라 가정하면 3개의 점을 임의로 선택

2. Assign (epoch=1)

모든 점을 업데이트 된 centroids 중 가장 가까운 점으로 할당

k-means clustering



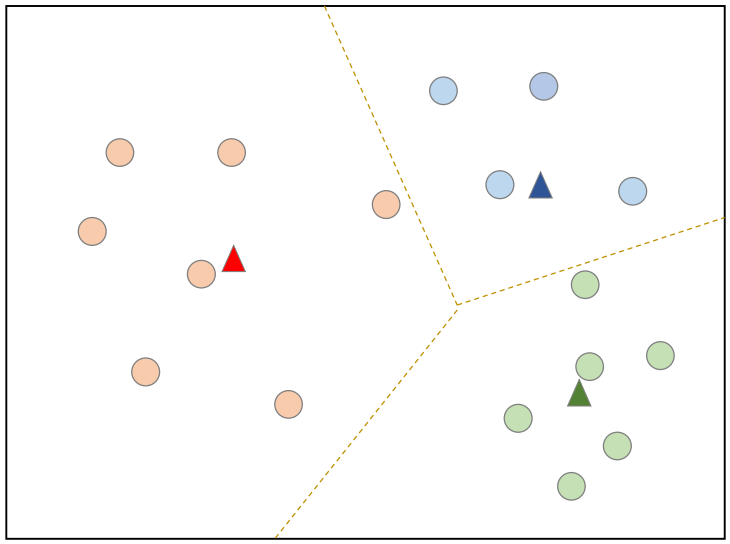
1. Initialize
k=3 이라 가정하면 3개의 점을 임의로 선택

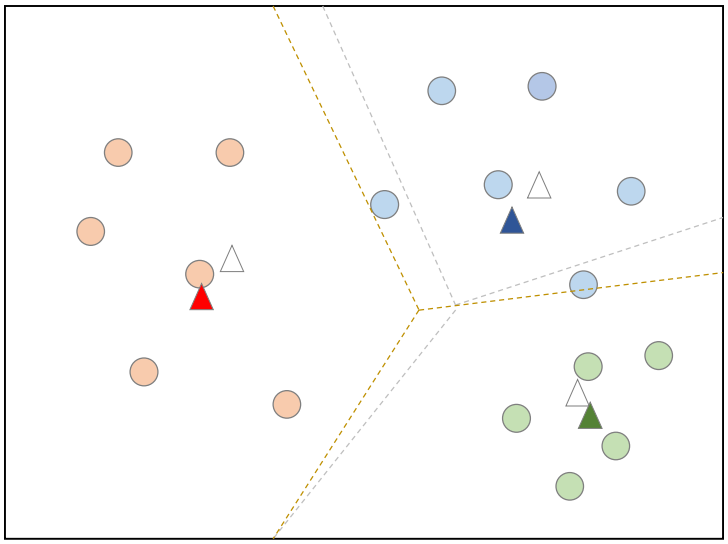
2. Assign (epoch=1)
모든 점을 업데이트 된 centroids 중 가장 가까운 점으로 할당

3. Update centroid (epoch=1)
색깔이 바뀐 점이 있기 때문에 Centroid를 다시 업데이트

알고리즘이 종료 될 때까지 2, 3을 반복

k-means clustering



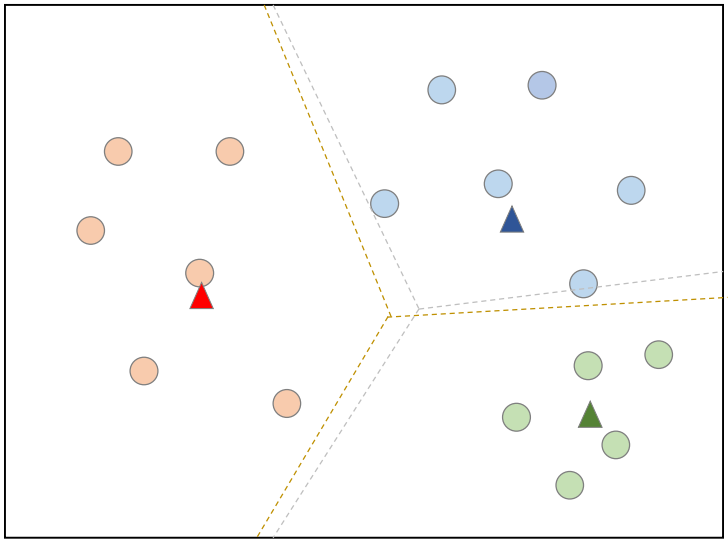


1. Initialize

k=3 이라 가정하면 3개의 점을 임의로 선택

2. Assign (epoch=2)

모든 점을 가장 가까운 centroids로 할당하여도 색깔이 변하지 않으므로 알고리즘 종료



k-means clustering

- ***k*-means 알고리즘은 다음의 단점이 있다고 알려짐**

1. Sensitive results from Initial points
   - 초기 centroids에 의하여 군집화 결과가 달라짐

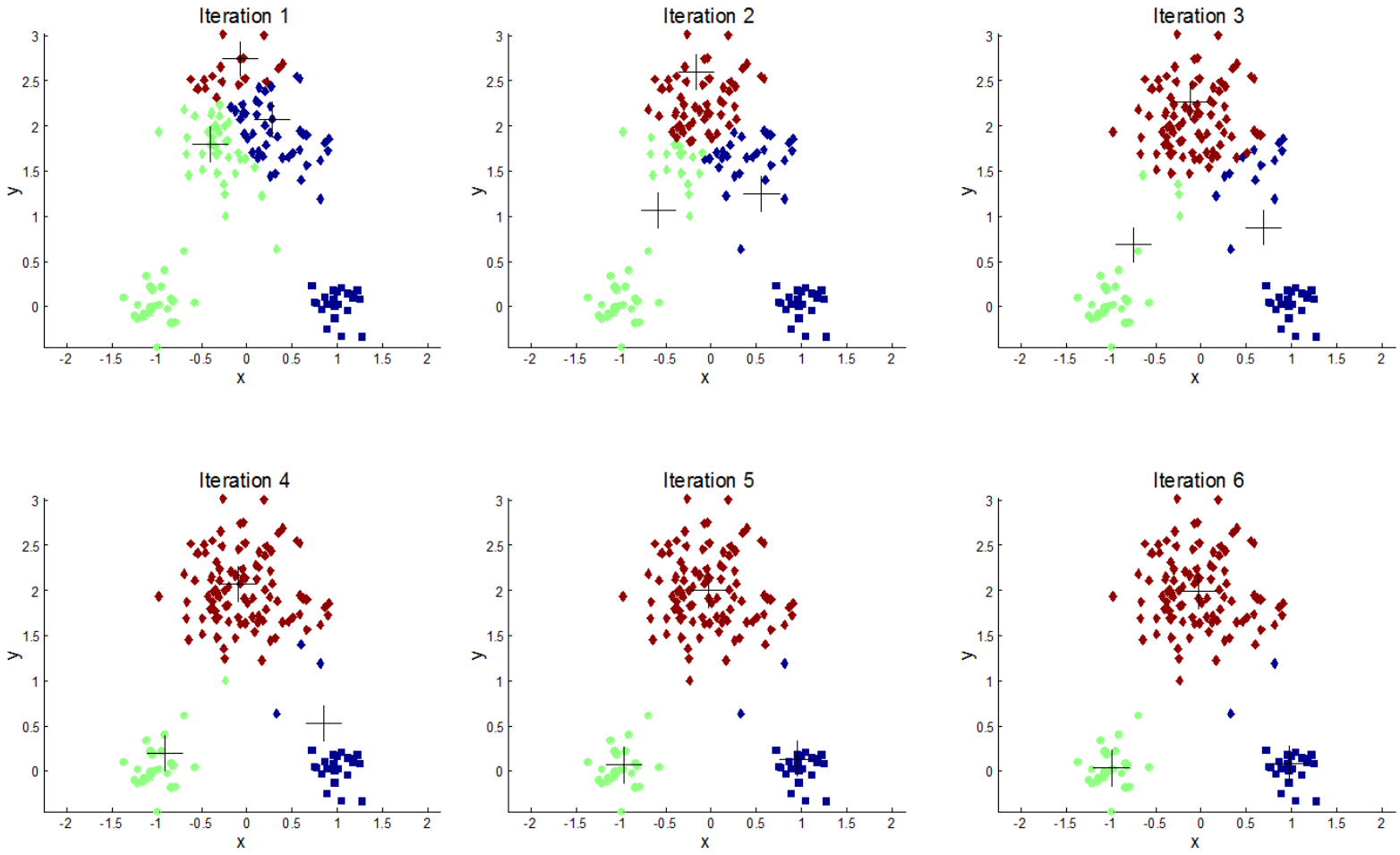
2. Ball-shaped clusters
   - 군집의 모양은 centroid를 중심으로 한 구형으로 제한됨 (Voronoi diagram)

3. Sensitive to noise points
   - Centroids와 노이즈와의 거리가 멀 경우, 노이즈에 의해 잘못된 centroid가 학습됨

k-means clustering

1. Sensitive results from Initial points
   - 초기 centroids에 의하여 군집화 결과가 달라짐

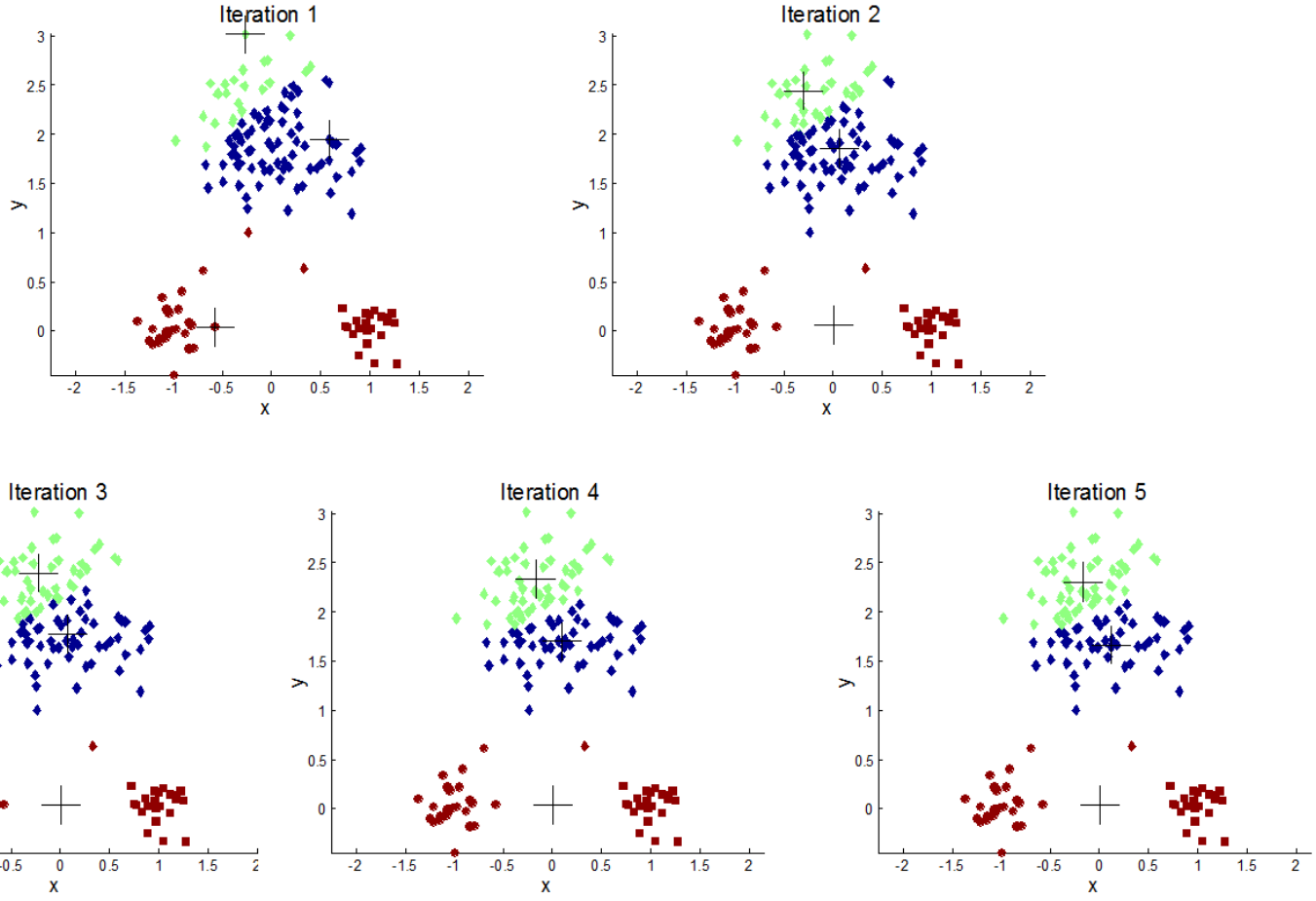
<Desirable centroid initialization>



k-means clustering

1. Sensitive results from Initial points
   - 초기 centroids에 의하여 군집화 결과가 달라짐

<Undesirable centroid initialization>



k-means clustering

1. Sensitive results from Initial points
   - 해결방법 1) 군집 중심 초기화를 반복 수행하여 best results를 return
     : sklearn.cluster.Kmeans 의 **n_init**
   - 해결방법 2) 군집 중심 초기화를 단순 랜덤이 아닌 기법을 사용
     : sklearn.cluster.Kmeans 의 **init='k-means++'**

[참고] k-means++
- centroid들을 데이터 공간 상에 퍼지게 생성하는 것을 목표로 함.
- 가까운 중심과의 거리 분포를 이용하여 랜덤한 centroid를 생성
- 초기화하는 데에 시간이 소요되지만, k-means clustering이 빨리 학습됨.

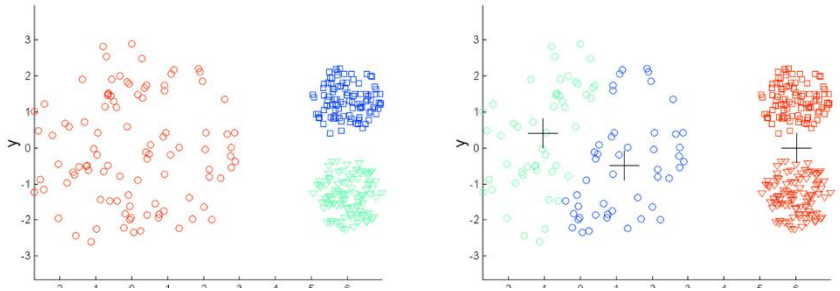
sklearn.cluster.KMeans

class sklearn.cluster. KMeans (*n_clusters=8, init='k-means++', n_init=10, max_iter=300, tol=0.0001, precompute_distances='auto', verbose=0, random_state=None, copy_x=True, n_jobs=1, algorithm='auto'*) [source]

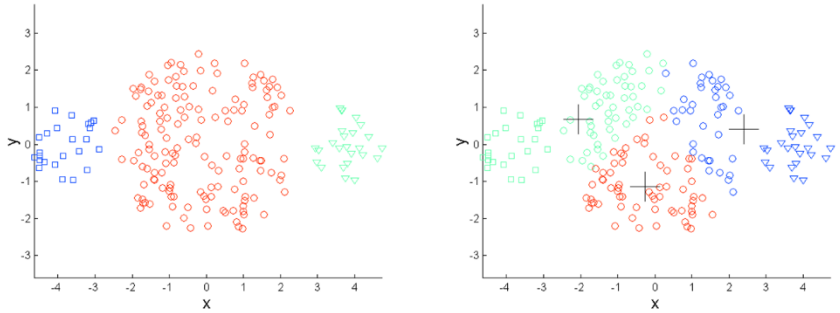
http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

k-means clustering

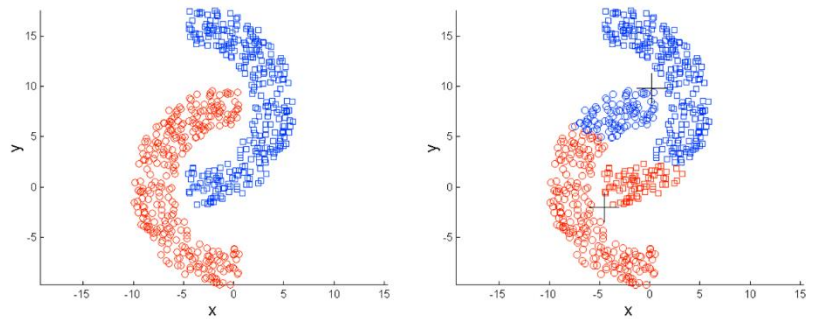
2. Ball-shaped clusters
   - 군집의 모양은 centroid를 중심으로 한 구형으로 제한됨
   - 군집들의 크기가 균일하지 못하거나 특이한 형태의 군집 형태는 생성 어려움



Cannot cope with different sizes



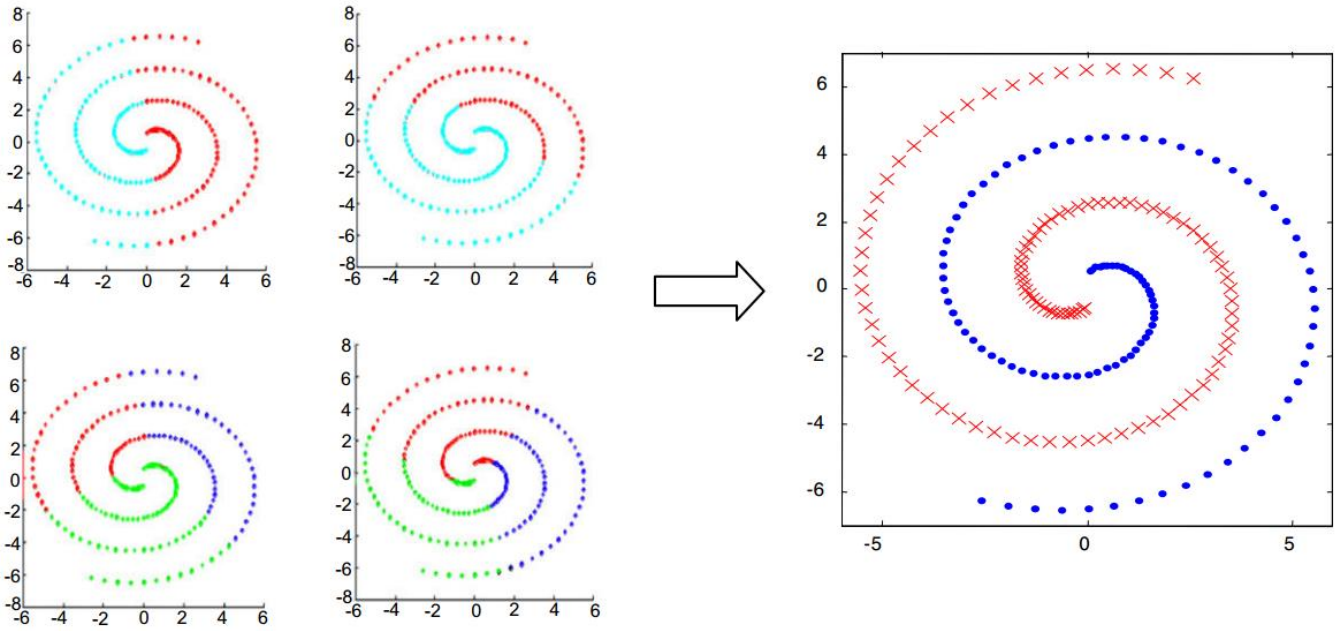
Cannot cope with different densities



Cannot cope with non-globular shapes

k-means clustering

- **(참고) Clustering ensemble**
  - 여러 번의 클러스터링 결과를 이용하여 데이터간의 co-occurrence 횟수를 similarity matrix로 이용하여 최종적인 군집화를 수행
  - 데이터의 representation을 co-occurrence vector로 바꾸는 의미이기도 함

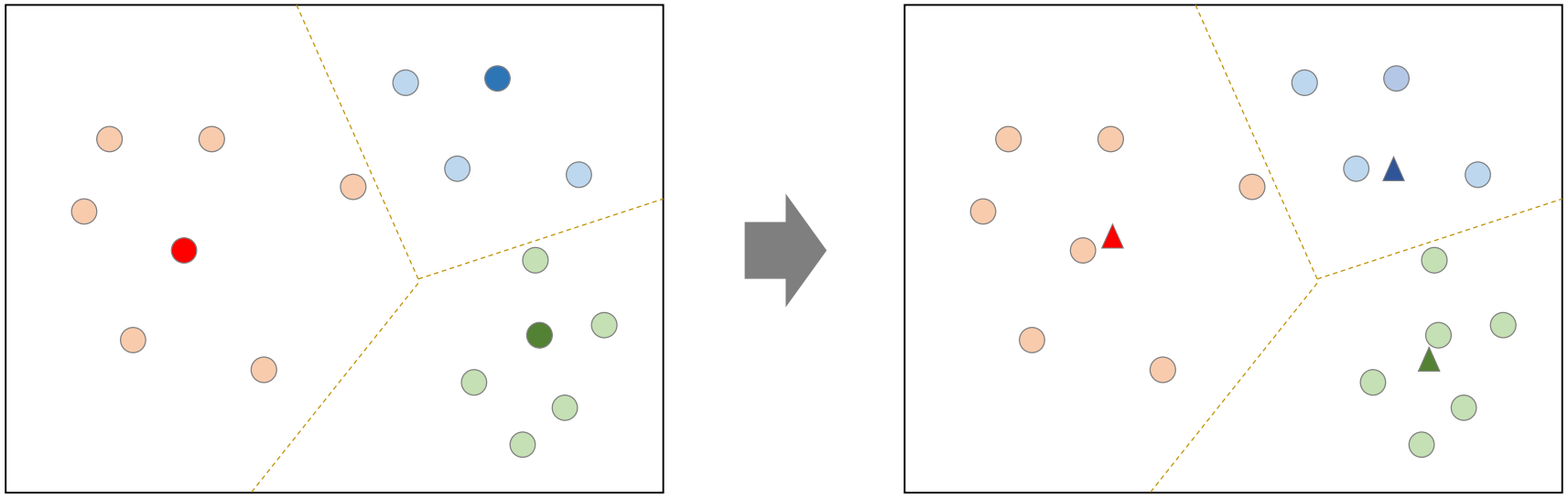


Jain, A. K. (2010). Data clustering: 50 years beyond k-means. Pattern recognition letters, 31(8), 651-666. ISO 690

k-means clustering

3. Sensitive to noise points

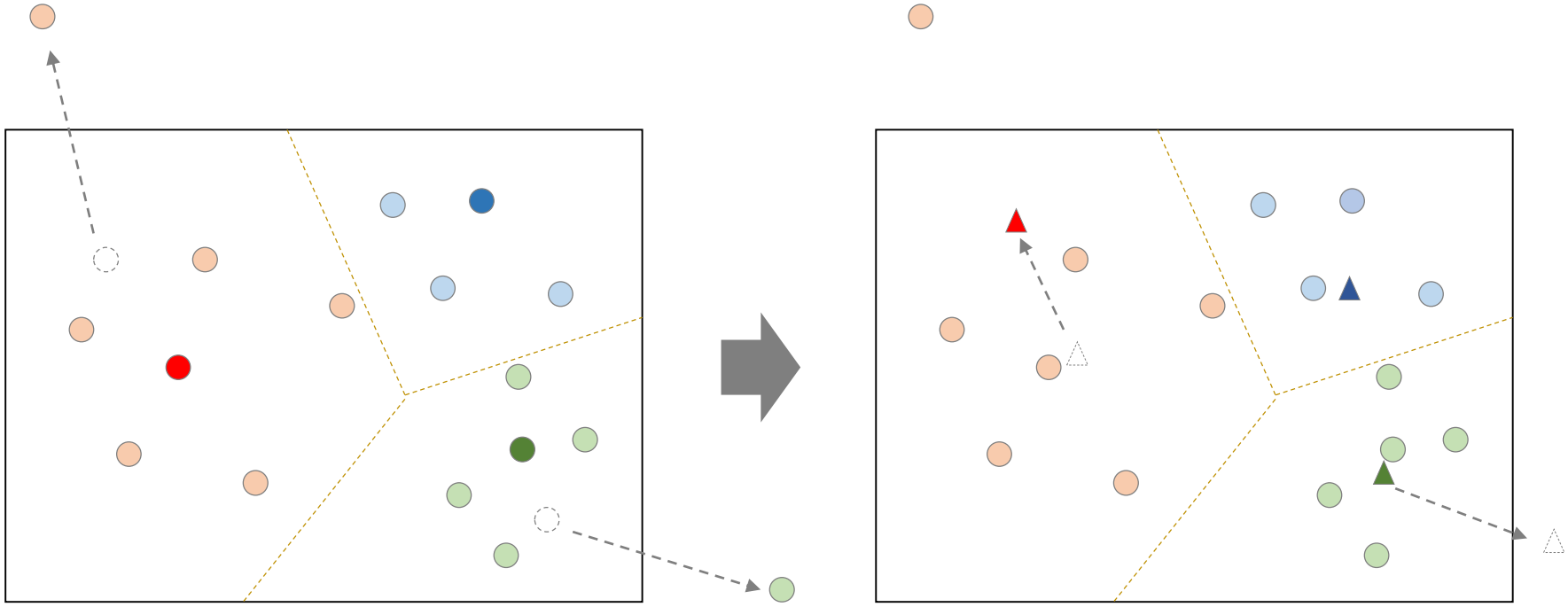
- 모든 점을 반드시 한 개 이상의 군집으로 assign해야 하기 때문에 노이즈 역시 가장 가까운 (하지만 의미적으로는 전혀 가깝지 않은) centroid에 할당이 됨



k-means clustering

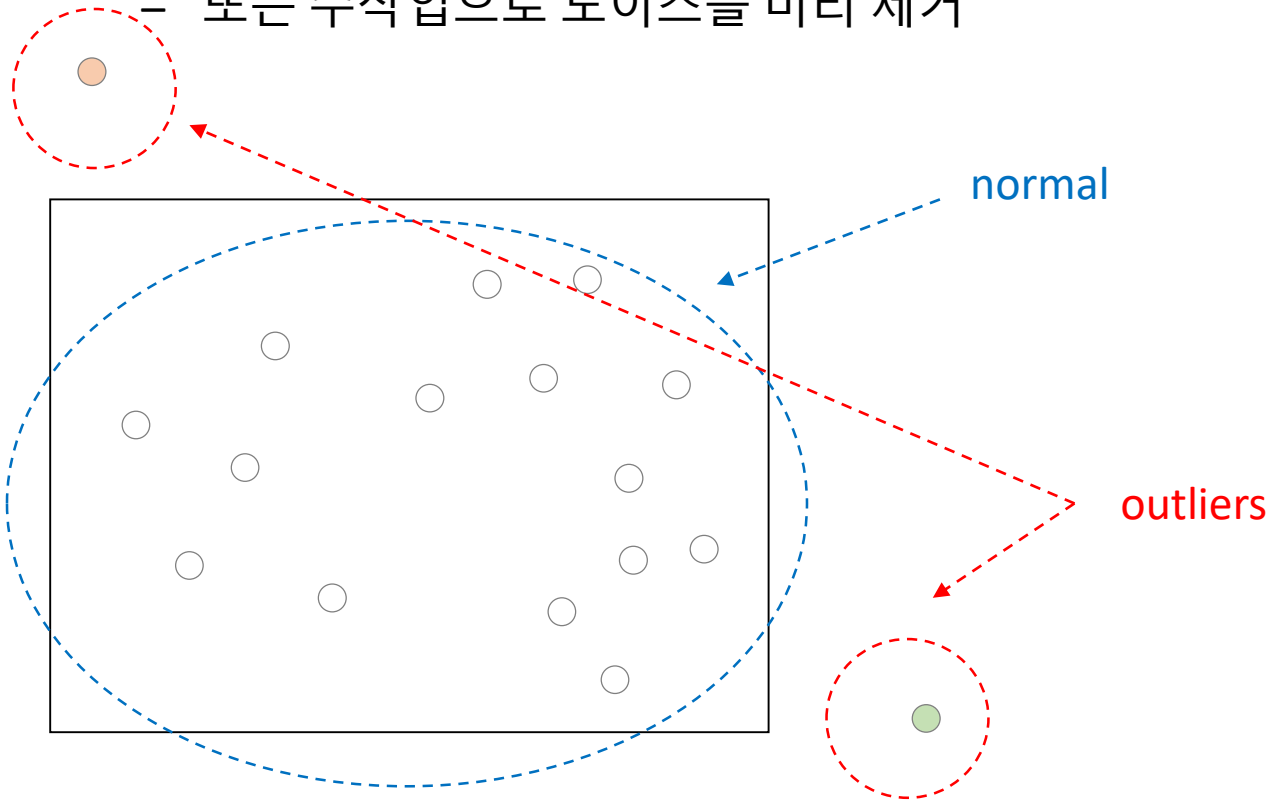
3. Sensitive to noise points

– 몇 개의 노이즈값에 의하여 centroids가 크게 흔들리고, 다음 단계에서 군집의 모양을 제대로 잡지 못함



k-means clustering

3. Sensitive to noise points
   - LOF와 같은 outlier detection 알고리즘으로 데이터의 노이즈를 미리 제거
   - 또는 수작업으로 노이즈를 미리 제거



k-means clustering

- **장점**
  - 계산이 쉬움: 다른 군집화 알고리즘에 비해 복잡도가 낮음.
  - 구현이 쉽고 다양한 언어와 플랫폼에서 제공되는 알고리즘

- **단점**
  - 노이즈에 매우 민감함.
  - 군집의 개수를 사전에 정의해야 함.
  - (앞서 언급한) 몇 가지 상황에서는 최적의 군집 구조를 찾기 어려움.

Evaluation metrics for clustering

How to evaluate clustering results?

- **Sadly, there is no good way.**
  - 지도학습은 정답이 있기 때문에 y와 $\hat{y}$를 비교한 평가지표를 생성하기 용이하나, 비지도학습은 정답이 없음.
  - 군집화의 구조가 얼마나 잘 되어 있는가를 평가할 수 밖에 없는데 아직 이 중에 만족할만한 좋은 방법은 없다.

- **So,**
  - 학술 영역에서는 일부러 정답이 있는 데이터에서 정답이 없다고 가정한 후 군집화 알고리즘을 적용 ==> 이후 정답과 군집화 결과를 비교
  - (~~아쉬운 대로~~) 몇 가지 사용 가능한 지표는 존재함.

정답을 모를 때 사용할 수 있는 평가지표

- **Sum of squared distance** for each point to it's assigned centroid
  - In scikit-learn, ‘inertia’
  - 군집의 수가 많아지면 값이 작아지는 경향
  - 변수의 수가 많아지면 값이 커지는 경향
- **Silhouette score**
  - 각 포인트에 silhouette value를 계산한 후, 이들의 평균을 점수로 활용
  - silhouette value for i-th point

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

$$-1 \leq s(i) \leq 1$$

$a(i)$: average distance between i and all other data assigned the same cluster (**mean intra-cluster distance**)

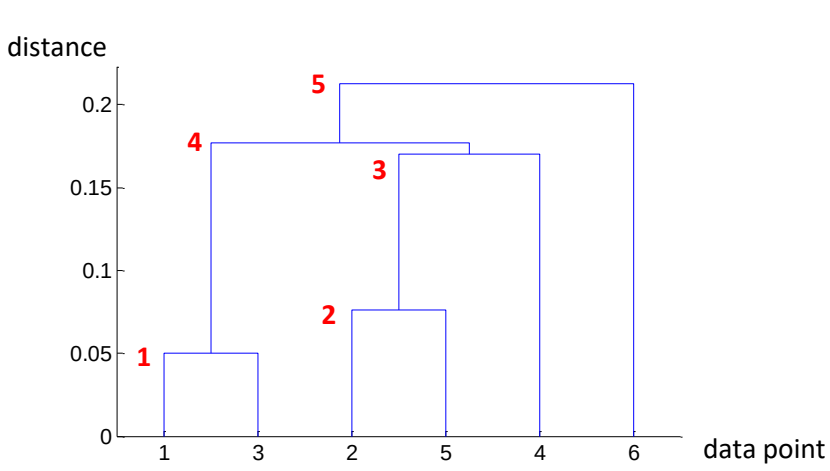
$b(i)$: average distance of i to all points in the nearest cluster that i is not a part of (**mean nearest-cluster distance**)

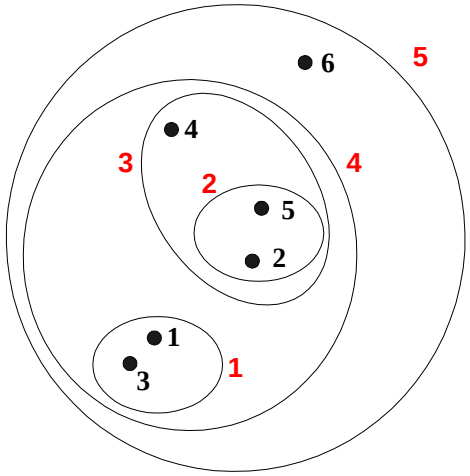
(Agglomerative) Hierarchical clustering

(Agglomerative) hierarchical clustering

- **계층적 군집화 방법**
  - 앞서 분할 군집화 방법은 각 군집이 서로 겹치지 않는 형태로 데이터를 분할
  - 계층적 군집화는 나무 형태의 군집 구조를 생성
    - 초기에는 모든 포인트를 개별적 군집으로 시작
    - 군집을 순차적으로 통합하여 전체 구조를 완성

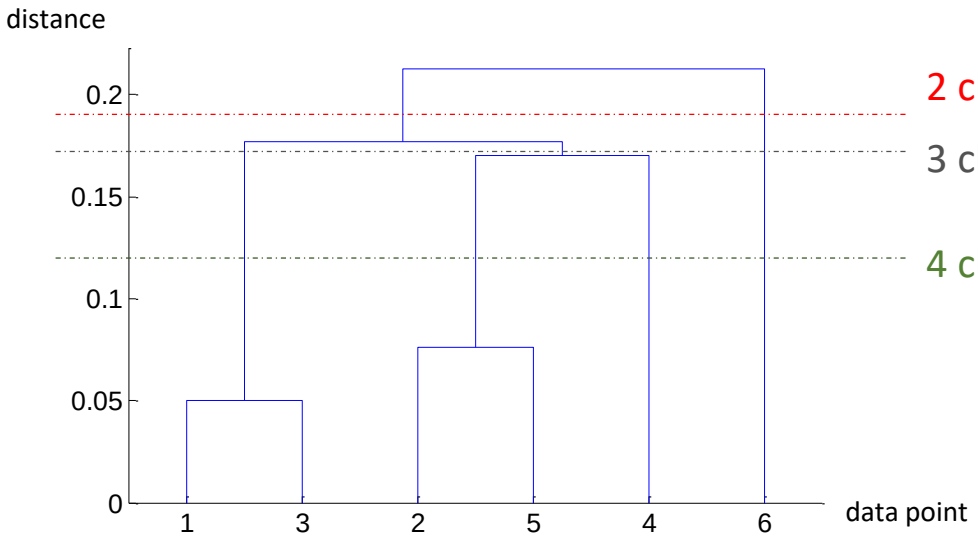
<Dendrogram>





(Agglomerative) hierarchical clustering

- **계층적 군집화 방법**
  - Dendrogram에서 군집의 개수를 맞춰 edge를 cutting하는 것을 고려할 수 있음



2 clusters: (1, 3, 2, 5, 4) / (6)

3 clusters: (1, 3) / (2, 5, 4) / (6)

4 clusters: (1, 3) / (2, 5) / (4) / (6)

Hierarchical clustering

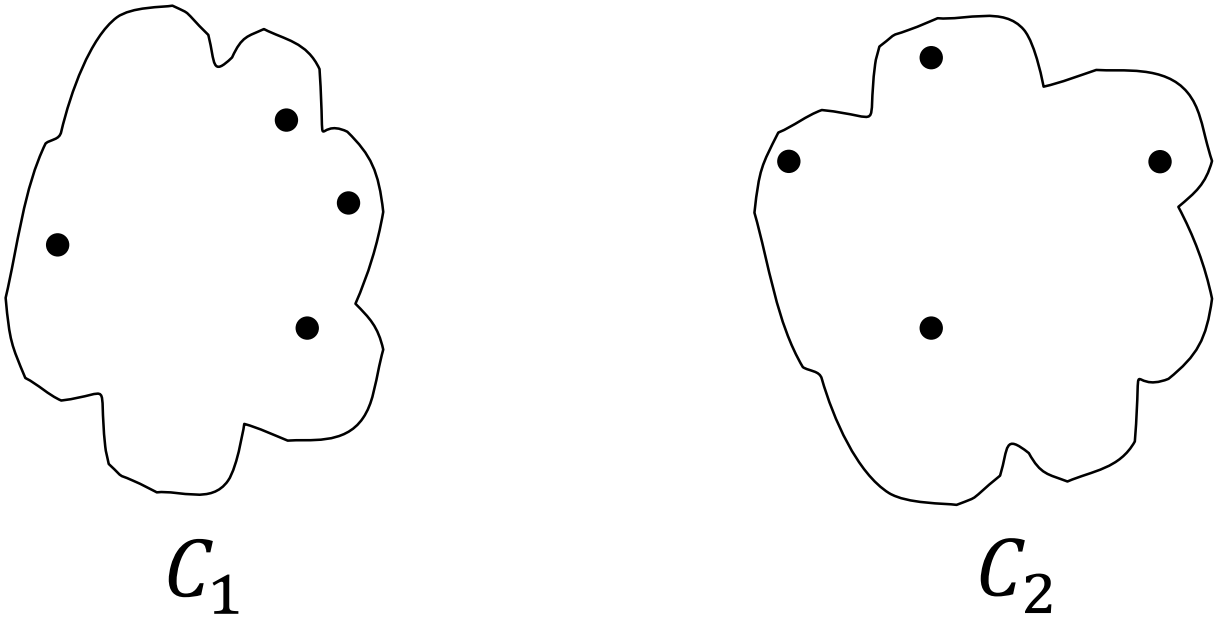
- **유사도**
  - n개의 데이터 X에 대하여 두 데이터 x_i, x_j간에 정의되는 임의의 거리 $d(x_i, x_j)$
    - 그룹 간의 거리 $d(C_i, C_j)$를 기반으로 정의 (min, max, average등)
    - 하나의 그룹 C_i는 1개 이상의 데이터로 이뤄짐
      (1개의 데이터도 그룹으로 정의 됨)
    - 다양한 방식: single linkage, complete linkage, average linkage, centroid linkage, ward linkage

- **군집화의 방식**
  - 군집의 수는 정하지 않으며, 거리가 가장 가까운 점들을 하나의 집합으로 묶어감

Hierarchical clustering

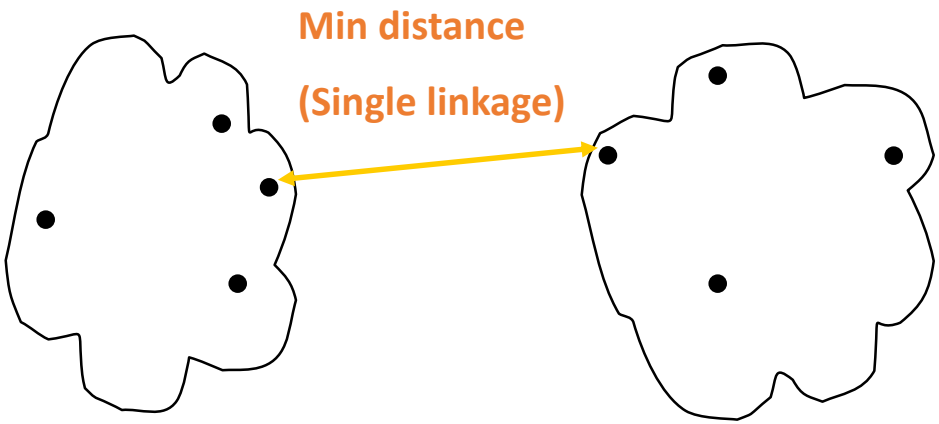
- **유사도 정의 방법**
  - 그룹(C_1)과 그룹(C_2) 사이의 거리를 어떻게 계산할 것인가?

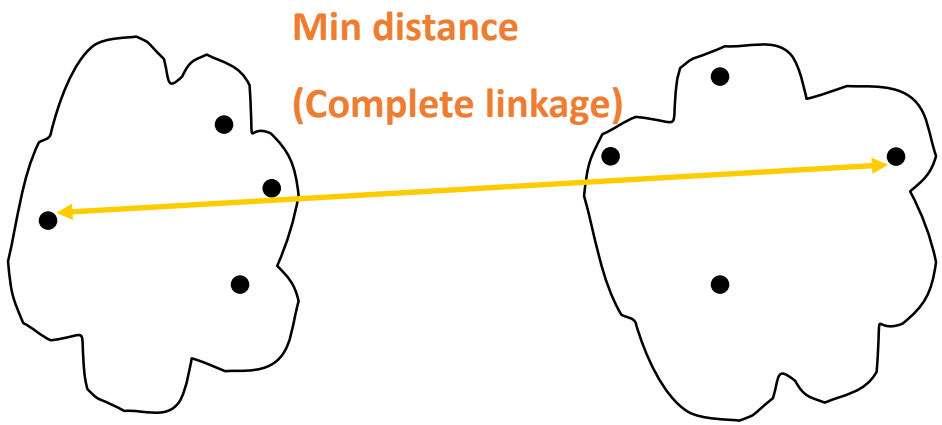
$$d(C_1, C_2)=?$$

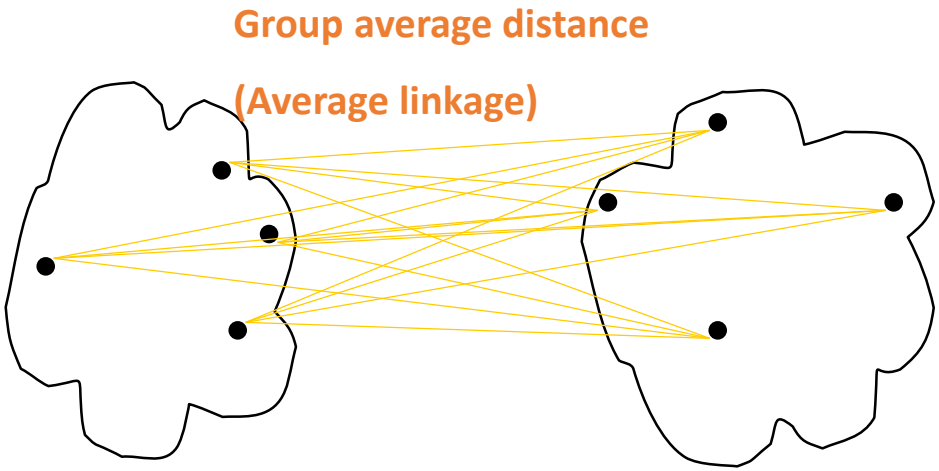


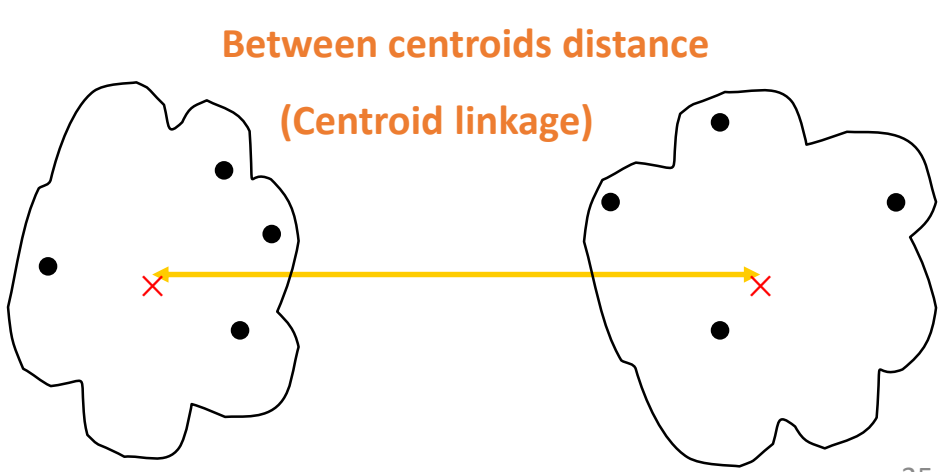
Hierarchical clustering

- 유사도 정의 방법



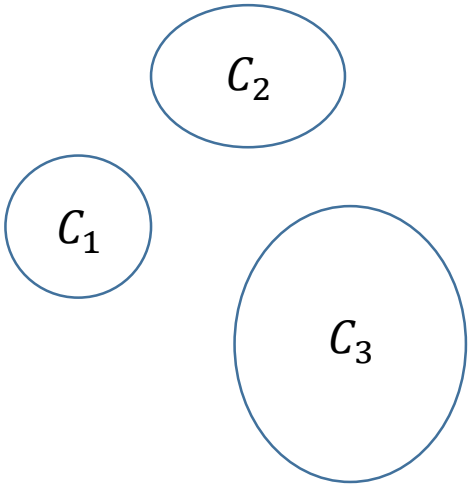






Hierarchical clustering

- **유사도 정의 방법: Ward's method / Ward linkage**
  - 군집을 합치면 군집의 분산 (중심과의 거리 제곱합) 이 커지게 될 것.
  - 분산의 증가량이 가장 작은 방향으로 군집화



$SS(C_1) = 100$ $SS(C_2) = 150$ $SS(C_3) = 250$

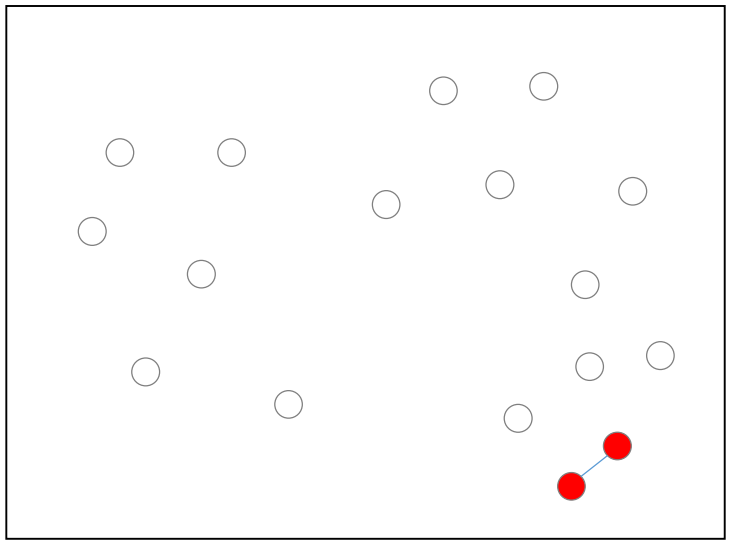
$SS(C_1 \cup C_2) = 300$ $SS(C_1 \cup C_2) - SS(C_1) - SS(C_2) = \mathbf{50}$

$SS(C_1 \cup C_3) = 500$ $SS(C_1 \cup C_3) - SS(C_1) - SS(C_3) = 150$

$SS(C_2 \cup C_3) = 600$ $SS(C_2 \cup C_3) - SS(C_2) - SS(C_3) = 200$

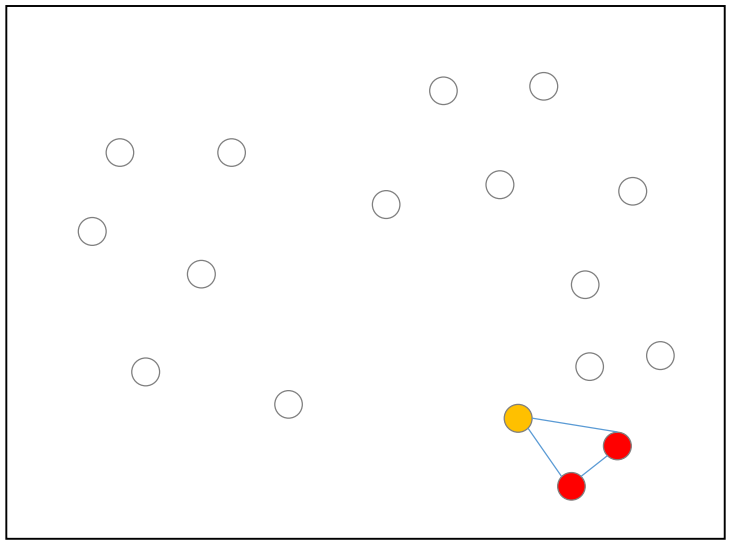
C_1과 C_2를 결합

(Agglomerative) hierarchical clustering



Iter = 1
가장 가까운 두 점을 연결

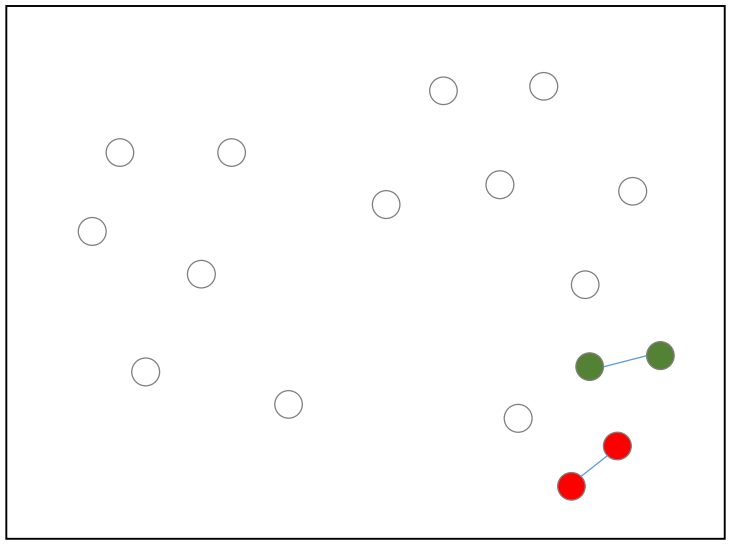
(Agglomerative) hierarchical clustering



Iter = 1
가장 가까운 두 점을 연결

Iter = 2
$d(C_i, C_j)$를 $d(x_p, x_q)$의 평균으로 정의한다면
두 빨간색점들과의 거리 평균이 다른 점들보다
가까우므로 주황색 점이 연결
(completed linkage)

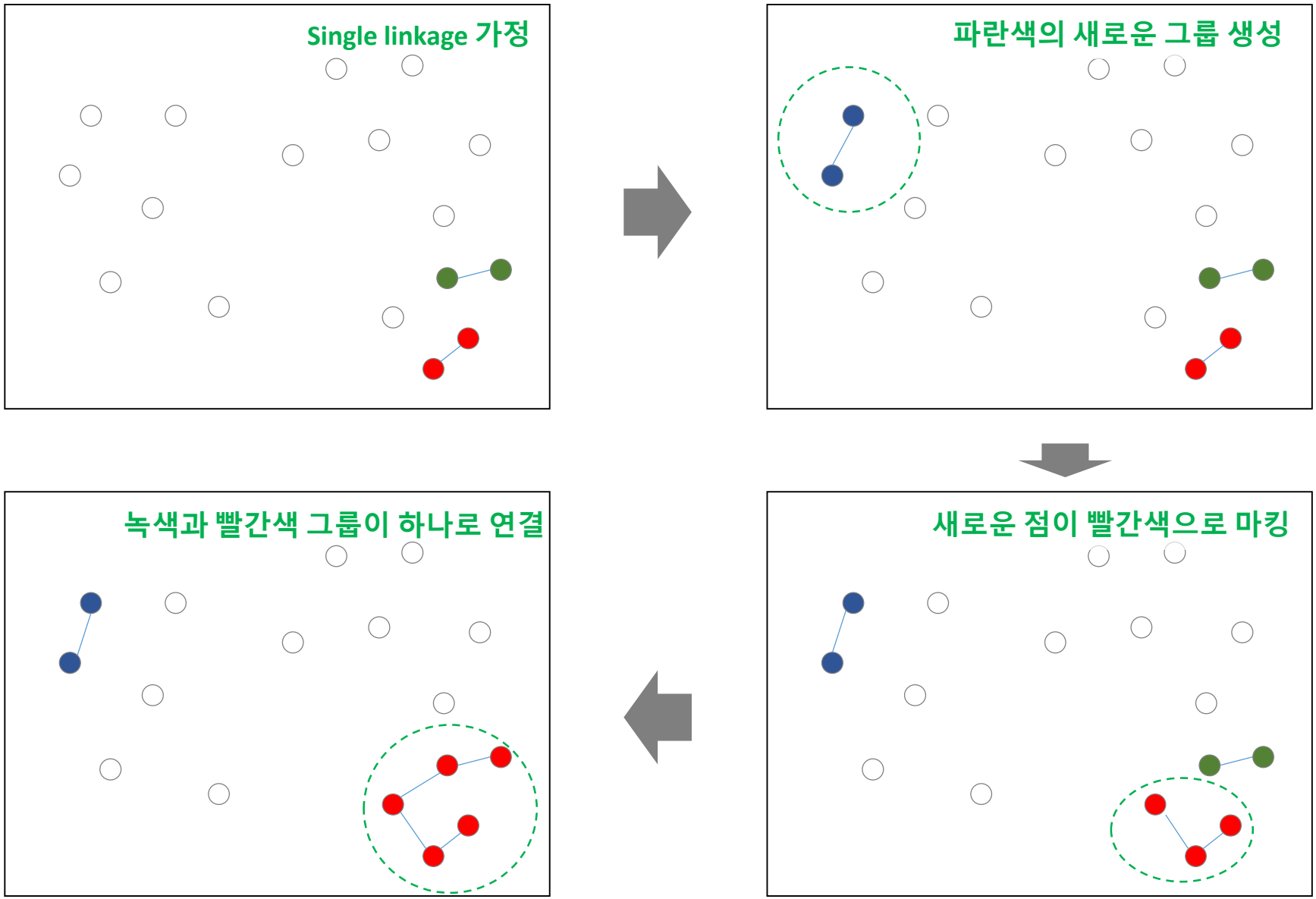
(Agglomerative) hierarchical clustering



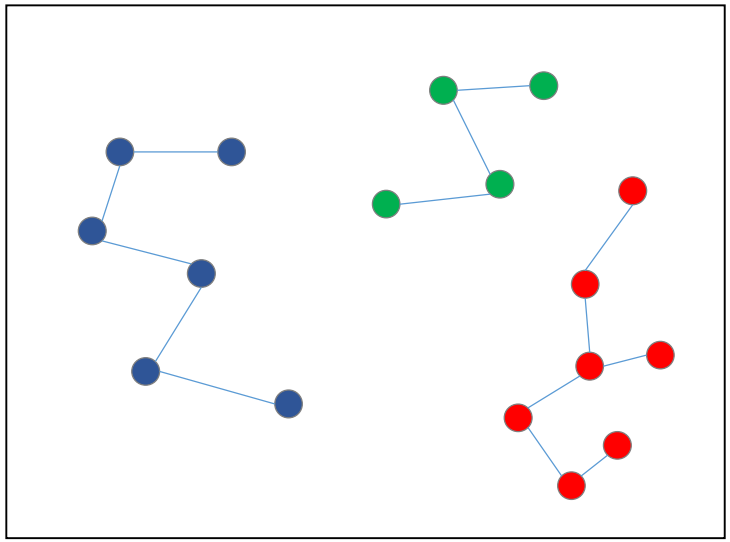
Iter = 1
가장 가까운 두 점을 연결

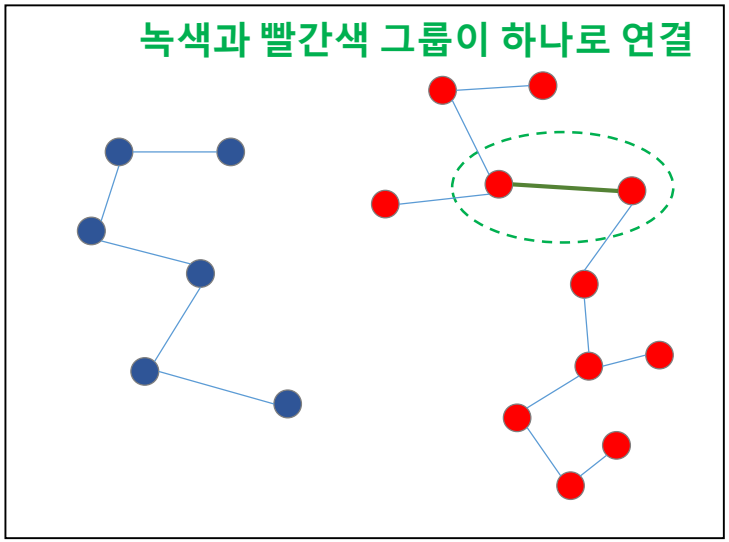
Iter = 2
$d(C_i, C_j)$를 $d(x_p, x_q)$의 min으로 정의한다면
녹색의 점이 하나로 연결
(single linkage)

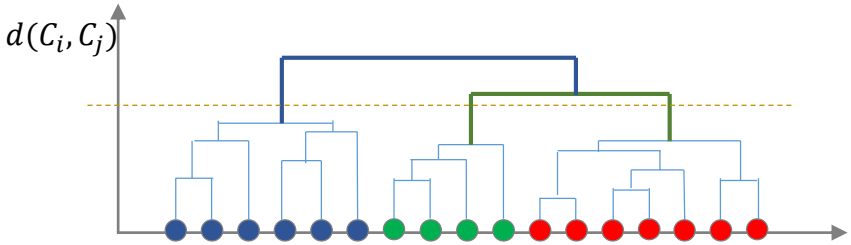
(Agglomerative) hierarchical clustering

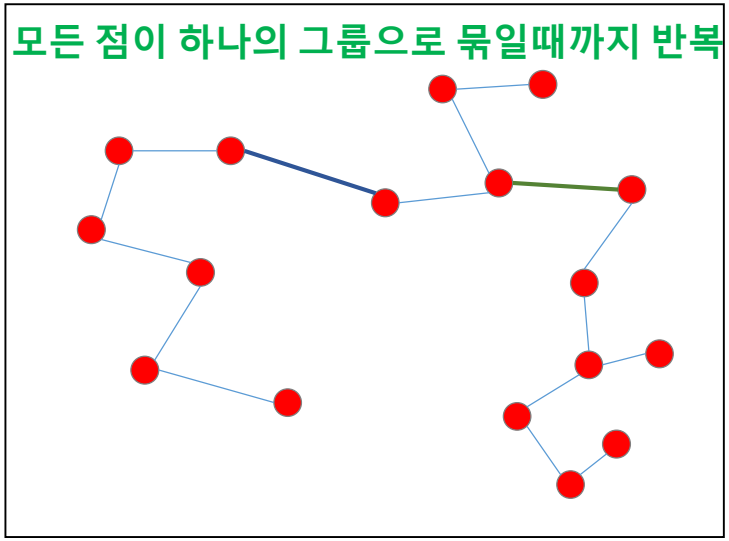


(Agglomerative) hierarchical clustering





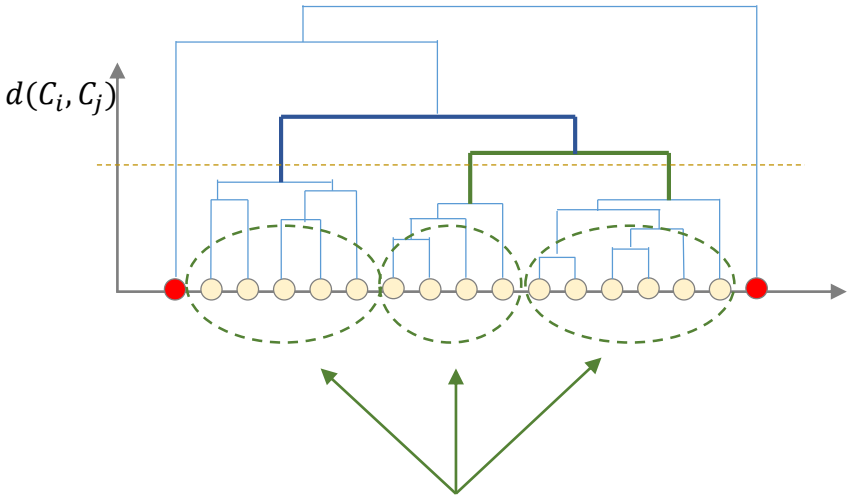




- Dendrogram은 링크가 생성되는 과정을 시각화한것
- 노란선의 distance로 cut한다는 것은 파란/녹색의 링크를 추가하지 않고 3개의 군집으로 묶겠다는 의미

(Agglomerative) hierarchical clustering

- **Outliers를 알아서 걸러줄 수 있음**
  - Single linkage는 가장 가까운 점들을 하나씩 이어나가는 구조이기 때문에, 다른 점들이 큰 군집으로 묶여갈 때 까지 다른 점들과 잘 묶이지 않는 점이 outliers



다른 점들은 큰 3개의 그룹으로 묶이지만,
붉은색 점들은 마지막에 큰 군집으로 묶임

(Agglomerative) hierarchical clustering

- **장점**
  - 군집 구조 정보가 매우 풍부함.
  - 군집화의 결과가 stable하게 생성됨.
  - 임의의 모양을 갖는 군집을 생성할 수 있음.
- **단점**
  - 계산 복잡도가 높음. 메모리 확보가 필요함.
    - 데이터의 개수가 N개라고 할 때, 모든 점들간의 거리를 계산해야 하기 때문에 $O(N^2)$ 계산 공간과 비용이 필요
- **이슈**
  - 파이썬에서는…
    - scikit-learn에 있는 AgglomerativeClustering은 기능이 약하고, scipy.cluster.hierarchy가 더 많은 기능을 제공
  - Is it possible to cut dynamically a hierarchical cluster tree?

DBSCAN

Density-based Clustering

- **Major features of density-based clustering**
  - It discovers clusters of arbitrary shape.
  - It can handle noise automatically.
- **Several interesting studies :**
  - DBSCAN (Ester et al., 1996), GDBSCAN (Sander et al., 1998), DENCLUE (Hinneburg & D.Keim, 1998)

DBSCAN

- **Density-Based Spatial Clustering of Applications with Noise**
  - 모든 점이 반드시 그룹에 속하지 않는다고 가정 (노이즈)

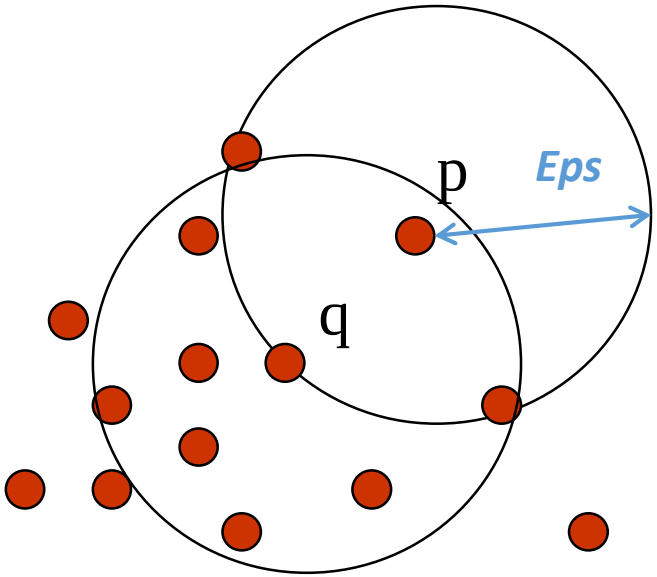
- **유사도**
  - n개의 데이터 X에 대하여 두 데이터 x_i, x_j간에 정의되는 임의의 거리 $d(x_i, x_j)$

- **그룹화의 방식**
  - Threshold 이상의 밀도를 지닌 점들을 모두 이어나가는 방식

Density Concepts

The algorithm counts the number of neighborhoods of each point.

- **Maximum radius of the neighborhood**
  - Eps
- **Eps-neighborhood**
  - $N_{Eps}(p) = \{q \text{ belongs to } D \mid dist(p,q) <= Eps\}$



$N_{Eps}(p) = 3$

$N_{Eps}(q) = 8$

Density Concepts

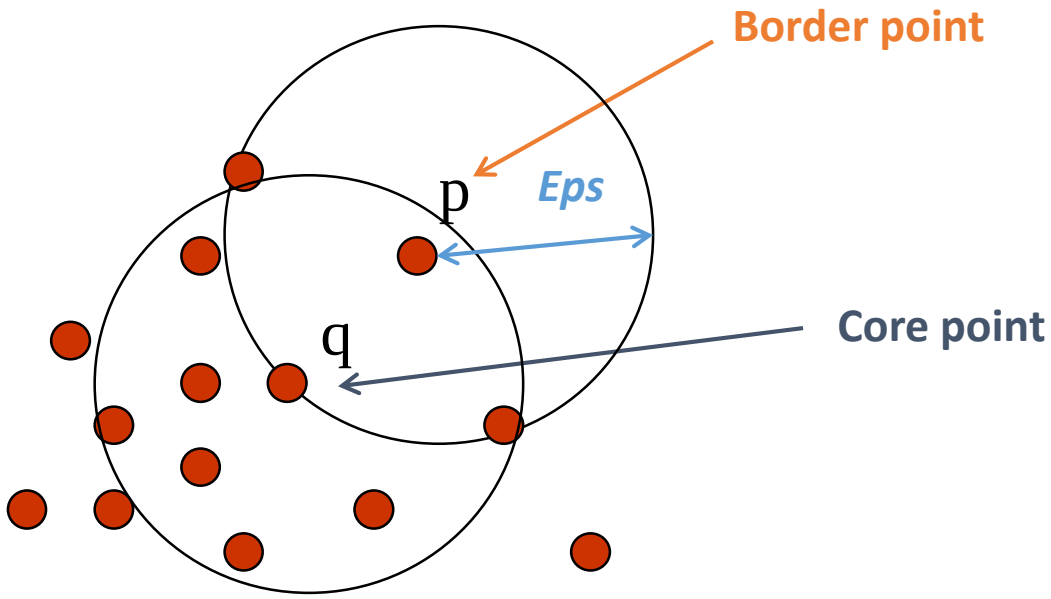
Each point is decided whether it is a "core point" or "border point".

- **Core point :**
  - Object with at least MinPts objects within a radius 'Eps-neighborhood'
  - MinPts : Minimum number of points in an Eps-neighborhood of that point
- **Border point :**
  - Object that on the border of a cluster

$N_{Eps}(p) = 3$

$N_{Eps}(q) = 8$

MinPts = 5



Density Concepts

- **Directly density-reachable :**
  - A point *p* is directly density-reachable from a point *q* w.r.t. *Eps* and *MinPts* if

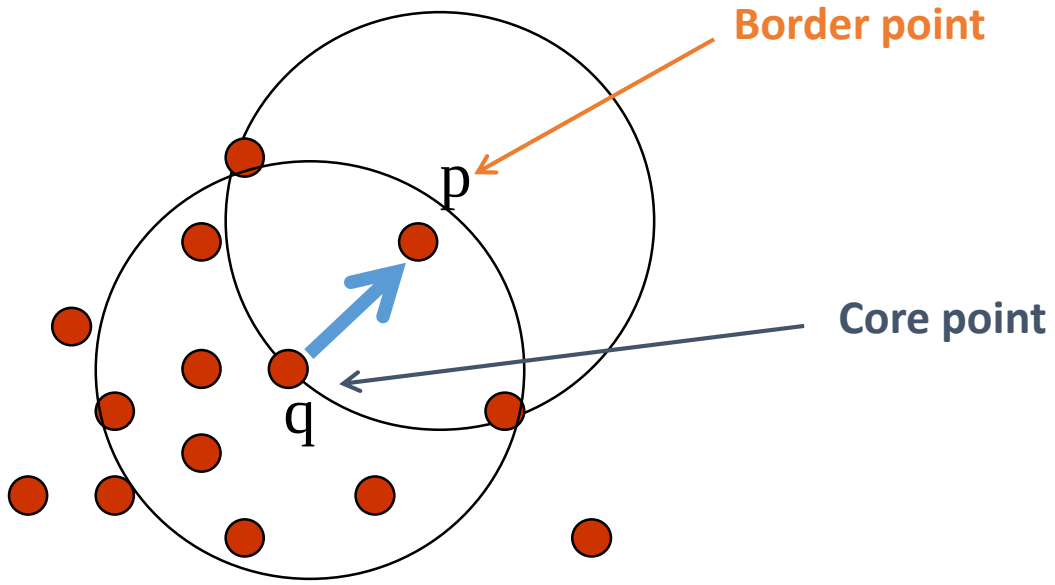
    1) *p* belongs to $N_{Eps}(q)$

    2) $N_{Eps}(q) >= MinPts$ (core point condition)

$N_{Eps}(p) = 3$

$N_{Eps}(q) = 8$

MinPts = 5



Density Concepts

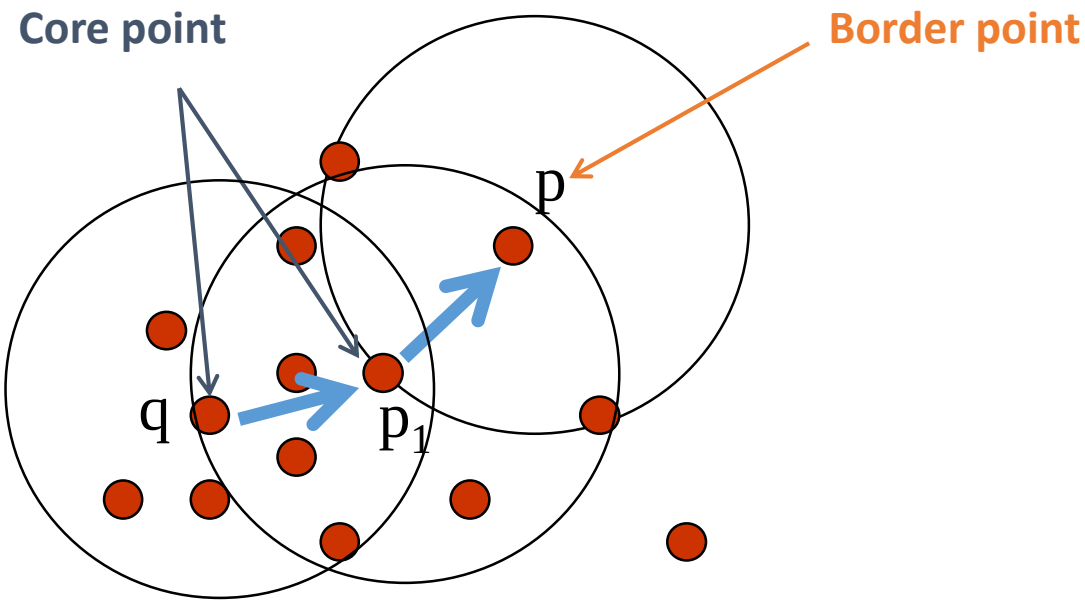
- **Density-reachable :**
  - A point *p* is "density-reachable" from a point *q* wrt *Eps*, *MinPts* if there is a chain of points $p_1, \ldots, p_n$, $p_1 = q$, $p_n = p$ such that p_{i+1} is directly density-reachable from p_i
  - Sequence of points that are density-reachable from each previous point

$N_{Eps}(p) = 3$

$N_{Eps}(p_1) = 8$

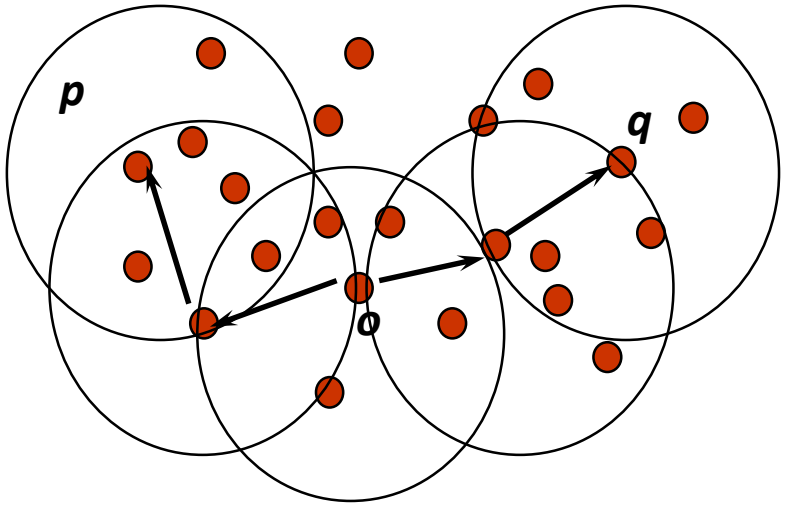
$N_{Eps}(q) = 8$

MinPts = 5



Density Concepts

- **Density-connected :**
  - A point *p* is “density-connected” to a point *q* wrt. *Eps*, *MinPts* if there is a point *o* such that both, *p* and *q* are density-reachable from *o* wrt. *Eps* and *MinPts*.

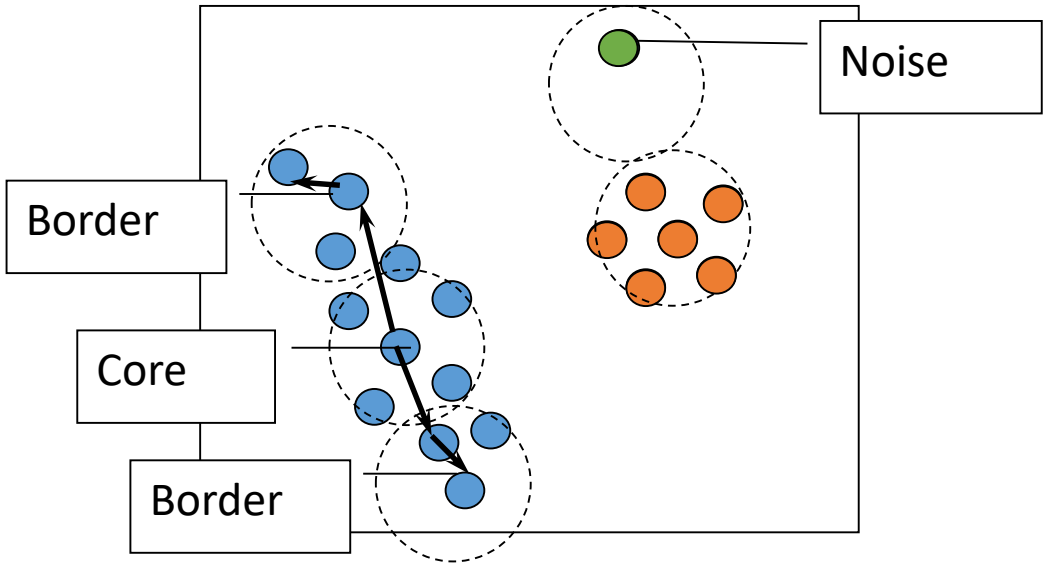


DBSCAN

- **DBSCAN**

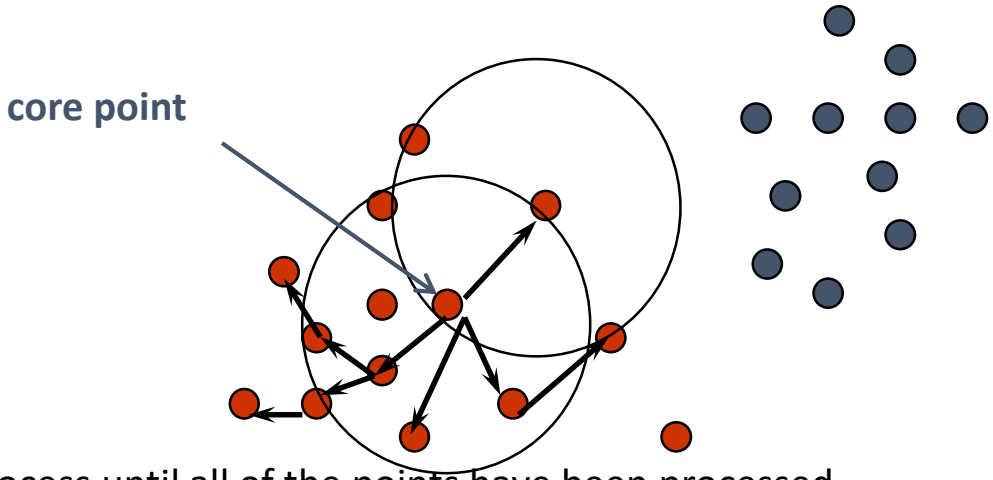
  (Density Based Spatial Clustering of Applications with Noise)

  - Relies on a *density-based* notion of cluster: A *cluster* is defined as a maximal set of density-connected points.
  - Discovers clusters of arbitrary shape in spatial databases with noise



DBSCAN

- **DBSCAN : Algorithm**
  - Arbitrary select a point *p*
  - Retrieve all points density-reachable from *p* wrt *Eps* and *MinPts*.
  - If *p* is a core point, a cluster is formed.
  - If *p* is a border point, no points are density-reachable from *p* and DBSCAN visits the next point of the database.



  - Continue the process until all of the points have been processed.

DBSCAN

- **Heuristics to determine *Eps* and *MinPts***
  - *k*-dist(p_i) : distance from the *k*th nearest neighbor to a data point p_i
  - Calculating *k*-dist(p_i) of each point
  - Sorting the points in descending order of their *k*-dist(p_i) values

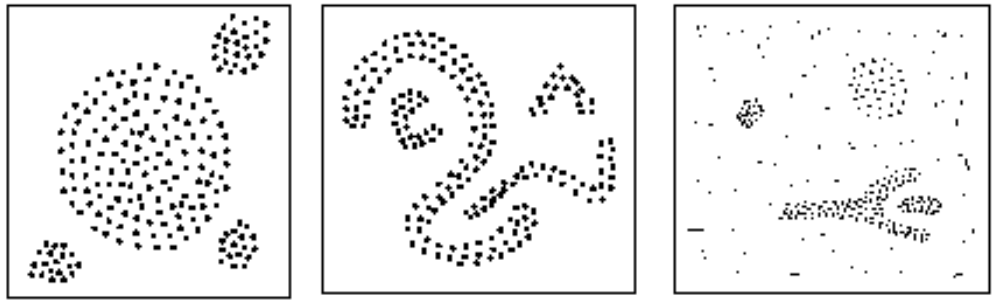
Set ***MinPts*** to *k*.

Set ***Eps*** to this 4-dist value.

DBSCAN

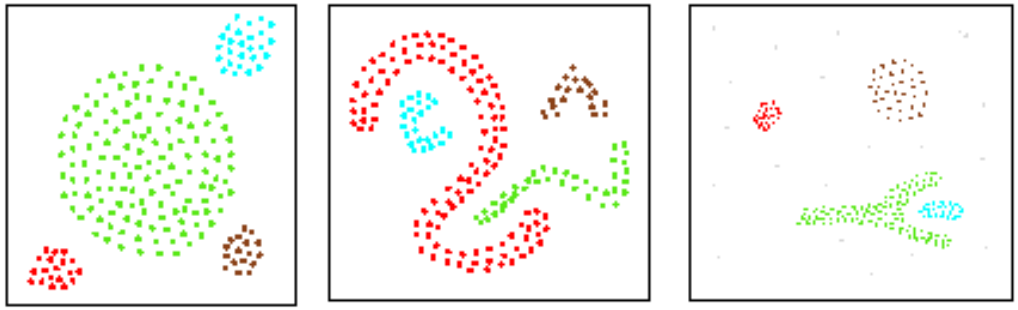
- **DBSCAN vs. Other clustering techniques**

Data sets

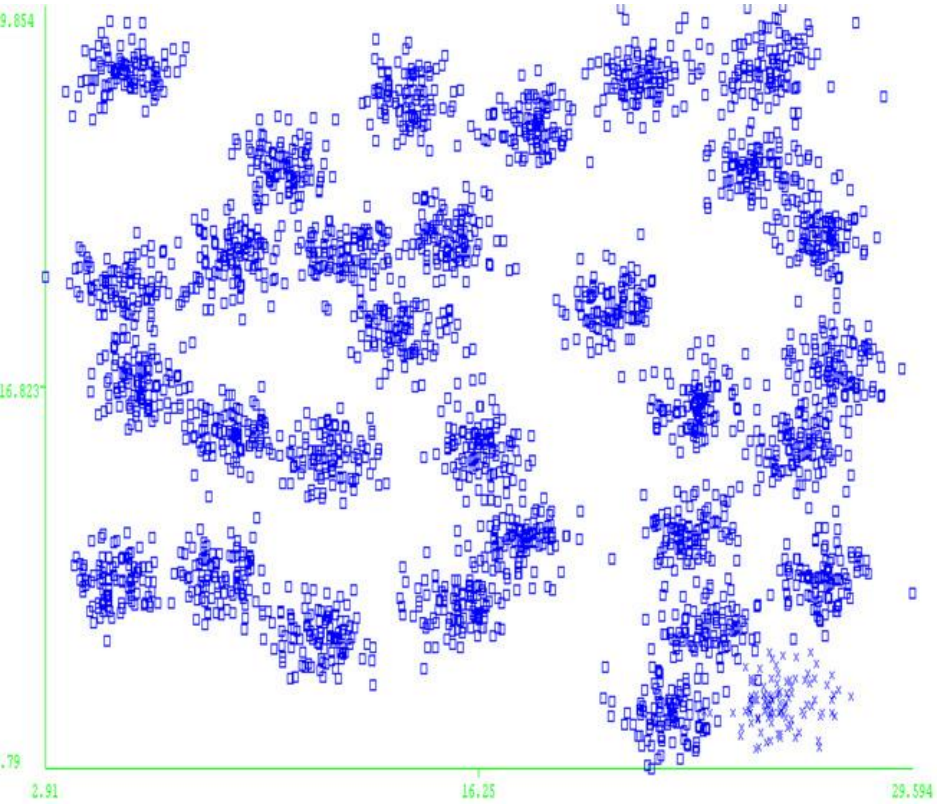


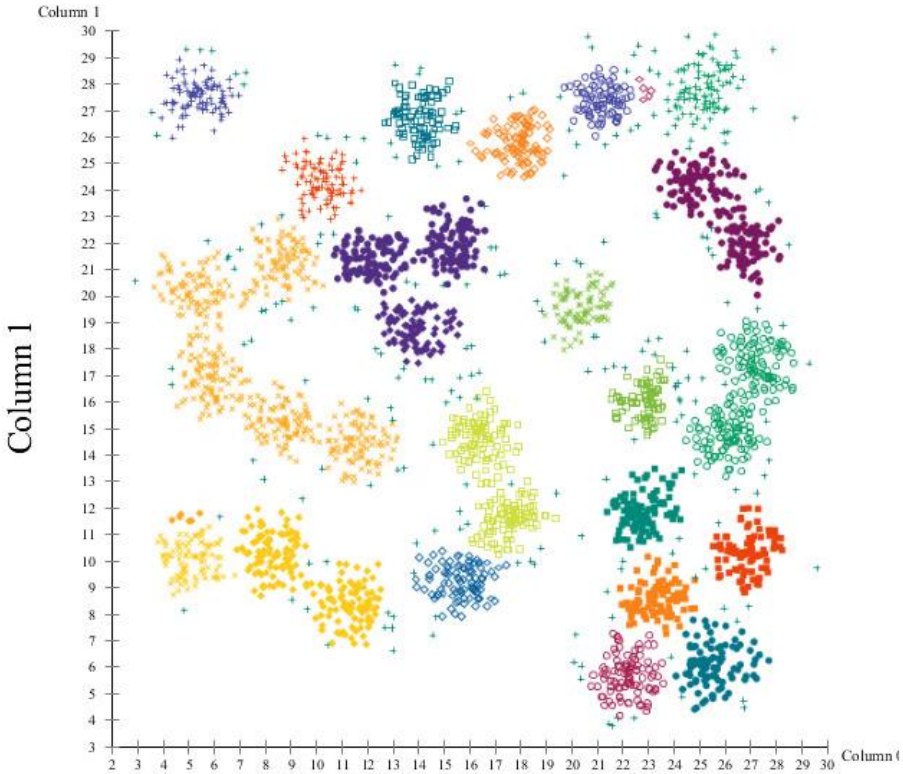
Algorithms
(Non-convex shapes)

DBSCAN
(Arbitrary shapes)



DBSCAN





DBSCAN

- **k-means clustering vs. DBSCAN**

Name	Complexity	Noise, Outliers	Input Parameters	Geometry
k-means clustering	*O(n)*	No	Number of clusters	Non-convex shapes
DBSCAN	***O(n²)***	**Yes**	1. *Eps* 2. *MinPts* **(2 parameters)**	**Arbitrary shapes**

DBSCAN

- **장점**
  - 노이즈에 매우 둔감한 군집화 가능
  - 임의의 모양을 갖는 군집을 생성할 수 있음
- **단점**
  - Parameters에 따라 결과가 민감하게 작동
    - 군집을 결정하는 밀도값 threshold에 의하여 데이터에서의 노이즈 비율이 예민하게 변함
  - 높은 계산 비용
    - DBSCAN은 모든 점들간의 거리를 한 번 이상 계산해야하기 때문에 $O(N^2)$의 계산 비용 필요

Clustering Overview: Issues

- How many clusters are optimal?

How many clusters?

Six Clusters

Two Clusters

Four Clusters

Clustering Overview: Issues

- **How to evaluate the clustering results?**
  - There is no rule of thumb.

External	• Rand Statistic • Jaccard Coefficient • Folks and Mallows index • (Normalized) Hurbert Γ statistic
Internal	• Cophenetic Correlation Coefficient • Sum of Squared error (SSE) • Cohesion and separation
Relative	• Dunn family of indices • Davies-Bouldin (DB) index • Semi-partial R-squared • SD validity index • Silhouette