



# CLUSTERING

**Courtesy:**  
**Jiawei Han, Micheline Kamber, and Jian Pei**  
**Anjali Jivani**



# WHAT IS CLUSTERING?

---

- Cluster: A collection of data objects
  - similar (or related) to one another within the same group
  - dissimilar (or unrelated) to the objects in other groups
- Cluster analysis (or *clustering*, *data segmentation*, ...)
  - Finding similarities between data according to the characteristics found in the data and grouping similar data objects into clusters
- **Unsupervised learning**: no predefined classes (i.e., *learning by observations* vs. learning by examples: supervised)
- Typical applications
  - As a **stand-alone tool** to get insight into data distribution
  - As a **preprocessing step** for other algorithms

# CLUSTERING APPLICATIONS



- Biology: taxonomy of living things: kingdom, phylum, class, order, family, genus and species
- Information retrieval: document clustering
- Land use: Identification of areas of similar land use in an earth observation database
- Marketing: Help marketers discover distinct groups in their customer bases, and then use this knowledge to develop targeted marketing programs
- City-planning: Identifying groups of houses according to their house type, value, and geographical location
- Earth-quake studies: Observed earth quake epicenters should be clustered along continent faults
- Climate: understanding earth climate, find patterns of atmospheric and ocean
- Economic Science: market research

# CLUSTERING AS PRE-PROCESSING



## ➤ SUMMARIZATION:

- Preprocessing for regression, PCA, classification, and association analysis

## ➤ COMPRESSION:

- Image processing: vector quantization

## ➤ FINDING K-NEAREST NEIGHBORS

- Localizing search to one or a small number of clusters

## ➤ OUTLIER DETECTION

- Outliers are often viewed as those “far away” from any cluster

# GOOD QUALITY CLUSTERING

- A GOOD CLUSTERING METHOD WILL PRODUCE HIGH QUALITY CLUSTERS
  - high intra-class similarity: **cohesive** within clusters
  - low inter-class similarity: **distinctive** between clusters
- THE QUALITY OF A CLUSTERING METHOD DEPENDS ON
  - the similarity measure used by the method
  - its implementation, and
  - Its ability to discover some or all of the hidden patterns

# MEASURING QUALITY OF CLUSTERS

## ➤ Dissimilarity/Similarity metric

- Similarity is expressed in terms of a distance function, typically metric:  $d(i, j)$
- The definitions of **distance functions** are usually rather different for interval-scaled, boolean, categorical, ordinal ratio, and vector variables
- Weights should be associated with different variables based on applications and data semantics

## ➤ Quality of clustering:

- There is usually a separate “quality” function that measures the “goodness” of a cluster.
- It is hard to define “similar enough” or “good enough”
  - ❖ The answer is typically highly subjective

# CONSIDERATIONS FOR CLUSTERING



- Partitioning criteria
  - Single level vs. hierarchical partitioning (often, multi-level hierarchical partitioning is desirable)
- Separation of clusters
  - Exclusive (e.g., one customer belongs to only one region) vs. non-exclusive (e.g., one document may belong to more than one cluster)
- Similarity measure
  - Distance-based (e.g., Euclidian, road network, vector) vs. connectivity-based (e.g., density or contiguity)
- Clustering space
  - Full space (often when low dimensional) vs. subspaces (often in high-dimensional clustering)

# REQUIREMENTS AND CHALLENGES

---

- Scalability
  - Clustering all the data instead of only on samples
- Ability to deal with different types of attributes
  - Numerical, binary, categorical, ordinal, linked, and mixture of these
- Constraint-based clustering
  - User may give inputs on constraints
  - Use domain knowledge to determine input parameters
- Interpretability and usability
- Others
  - Discovery of clusters with arbitrary shape
  - Ability to deal with noisy data
  - Incremental clustering and insensitivity to input order
  - High dimensionality



# CLUSTERING APPROACHES

- Partitioning approach:
  - Construct various partitions and then evaluate them by some criterion, e.g., minimizing the sum of square errors
  - Typical methods: k-means, k-medoids, CLARANS
- Hierarchical approach:
  - Create a hierarchical decomposition of the set of data (or objects) using some criterion
  - Typical methods: Diana, Agnes, BIRCH, CAMELEON
- Density-based approach:
  - Based on connectivity and density functions
  - Typical methods: DBSCAN, OPTICS, DenClue
- Grid-based approach:
  - based on a multiple-level granularity structure
  - Typical methods: STING, WaveCluster, CLIQUE

# CLUSTERING APPROACHES

- Model-based:
  - A model is hypothesized for each of the clusters and tries to find the best fit of that model to each other
  - Typical methods: EM, SOM, COBWEB
- Frequent pattern-based:
  - Based on the analysis of frequent patterns
  - Typical methods: p-Cluster
- User-guided or constraint-based:
  - Clustering by considering user-specified or application-specific constraints
  - Typical methods: COD (obstacles), constrained clustering
- Link-based clustering:
  - Objects are often linked together in various ways
  - Massive links can be used to cluster objects: SimRank, LinkClus

# PARTITIONING METHODS

- Partitioning method: Partitioning a database  $D$  of  $n$  objects into a set of  $k$  clusters, such that the sum of squared distances is minimized (where  $c_i$  is the centroid or medoid of cluster  $C_i$ )

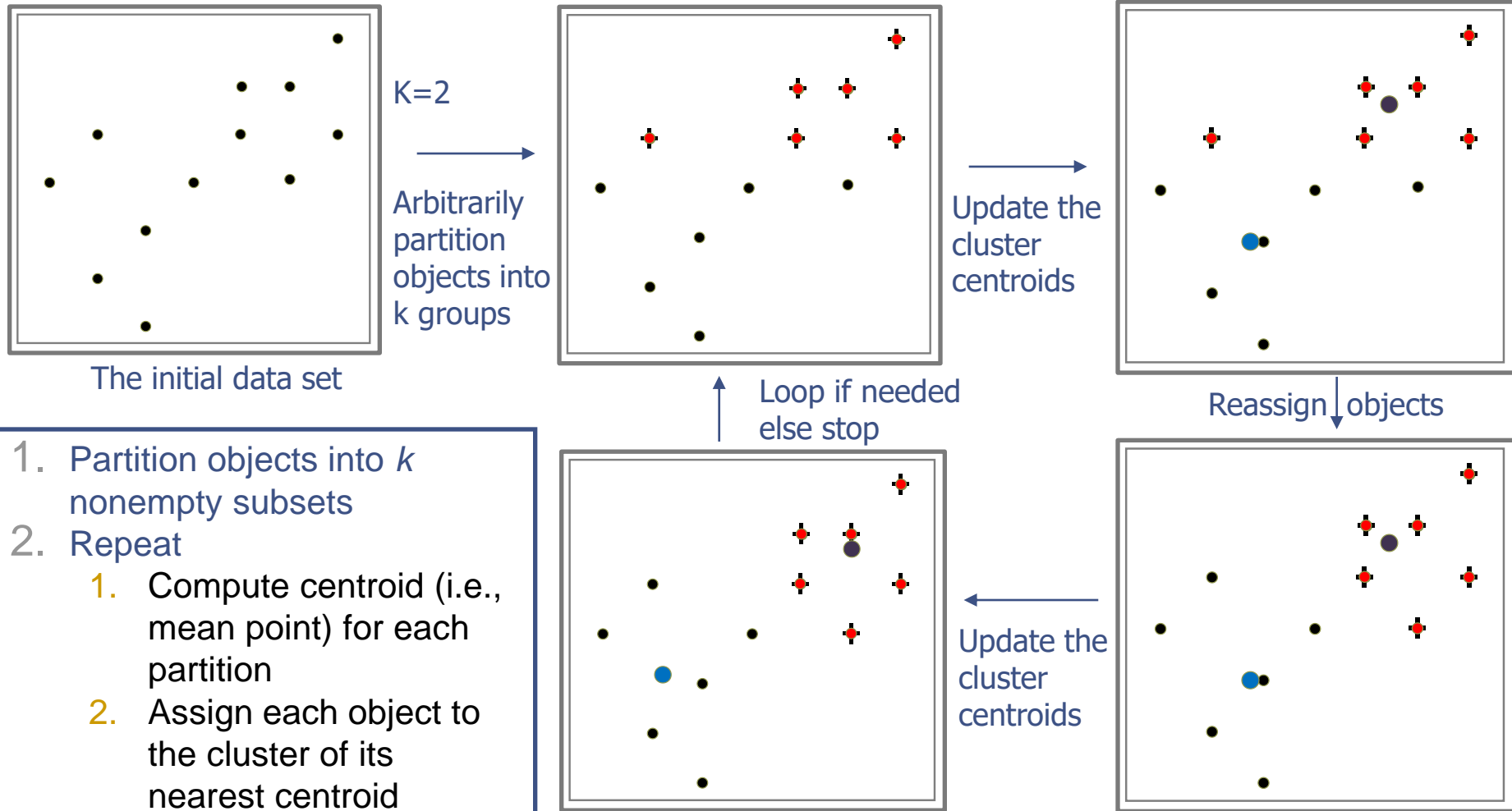
$$E = \sum_{i=1}^k \sum_{p \in C_i} (p - c_i)^2$$

- Given  $k$ , find a partition of  $k$  clusters that optimizes the chosen partitioning criterion
  - Global optimal: exhaustively enumerate all partitions
  - Heuristic methods: *k-means* and *k-medoids* algorithms
  - *k-means* (MacQueen'67, Lloyd'57/'82): Each cluster is represented by the center of the cluster
  - *k-medoids* or PAM (Partition around medoids) (Kaufman & Rousseeuw'87): Each cluster is represented by one of the objects in the cluster

# THE K-MEANS CLUSTERING ALGORITHM

- **Given  $k$ , the k-means algorithm is implemented in four steps:**
1. Partition objects into  $k$  nonempty subsets
  2. Compute seed points as the centroids of the clusters of the current partitioning (the centroid is the center, i.e., *mean point*, of the cluster)
  3. Assign each object to the cluster with the nearest seed point
  4. Go back to Step 2, stop when the assignment does not change

# K-MEANS CLUSTERING EXAMPLE

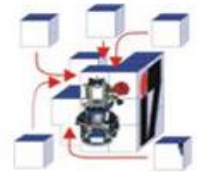


1. Partition objects into  $k$  nonempty subsets
2. Repeat
  1. Compute centroid (i.e., mean point) for each partition
  2. Assign each object to the cluster of its nearest centroid
3. Until no change

# THE SSE

## Minimizing the Sum of Squared Error (SSE)

### *k*-means



- The algorithm aims at minimizing an objective function, in this case a squared error function

$$J = \sum_{j=1}^k \sum_{i=1}^n \|x_i^{(j)} - c_j\|^2$$
$$E = \sum_{i=1}^k \sum_{p \in C_i} (p - c_i)^2$$

- Where  $\|x_i^{(j)} - c_j\|^2$  is a chosen distance measure between a data point  $x_i^{(j)}$  and the cluster centre  $c_j$
- The objective function is an indicator of the distance of the  $n$  data points from their respective cluster centers

# ADVANTAGES & LIMITATIONS OF K-MEANS

- Strength: *Efficient*:  $O(tkn)$ , where  $n$  is # objects,  $k$  is # clusters, and  $t$  is # iterations. Normally,  $k, t \ll n$ .
  - Comparing: PAM:  $O(k(n-k)^2)$ , CLARA:  $O(ks^2 + k(n-k))$
- Comment: Often terminates at a *local optimal*.
- Weakness
  - Applicable only to objects in a continuous  $n$ -dimensional space
    - ❖ Using the k-modes method for categorical data
    - ❖ In comparison, k-medoids can be applied to a wide range of data
  - Need to specify  $k$ , the *number* of clusters, in advance (there are ways to automatically determine the best  $k$  (see Hastie et al., 2009))
  - Sensitive to noisy data and *outliers (cannot detect noise points)*
  - Not suitable to discover clusters with *non-convex shapes*
  - Initial values of cluster centroids or the partitioning can affect the final clusters

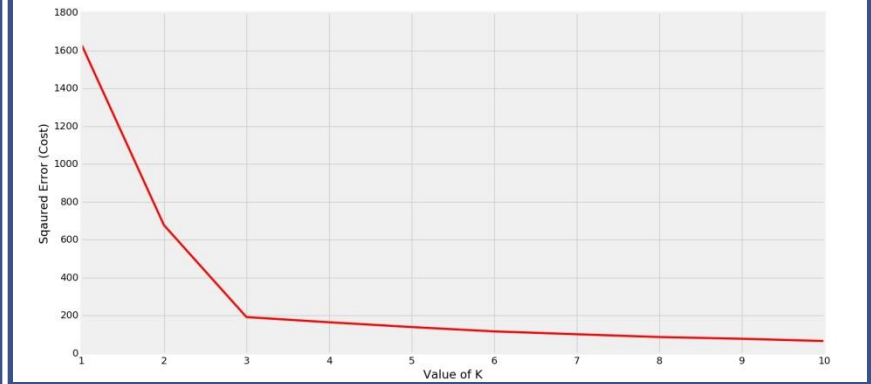
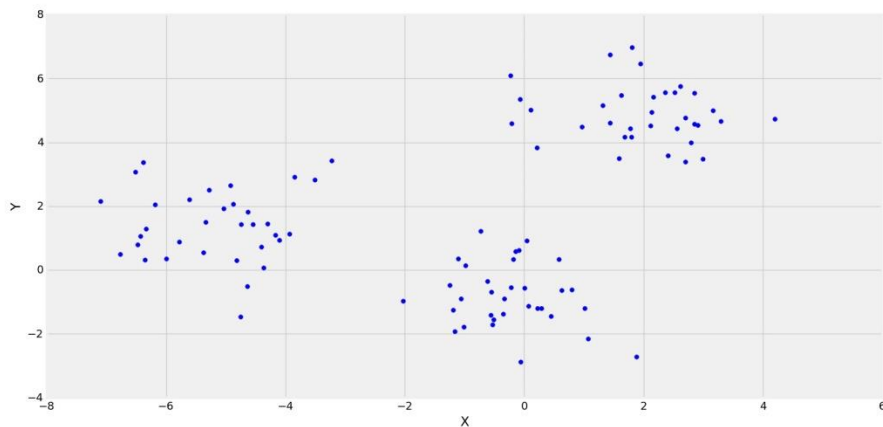
# DETERMINING THE VALUE OF K

➤ There are two popular methods:

**1. The Elbow method**

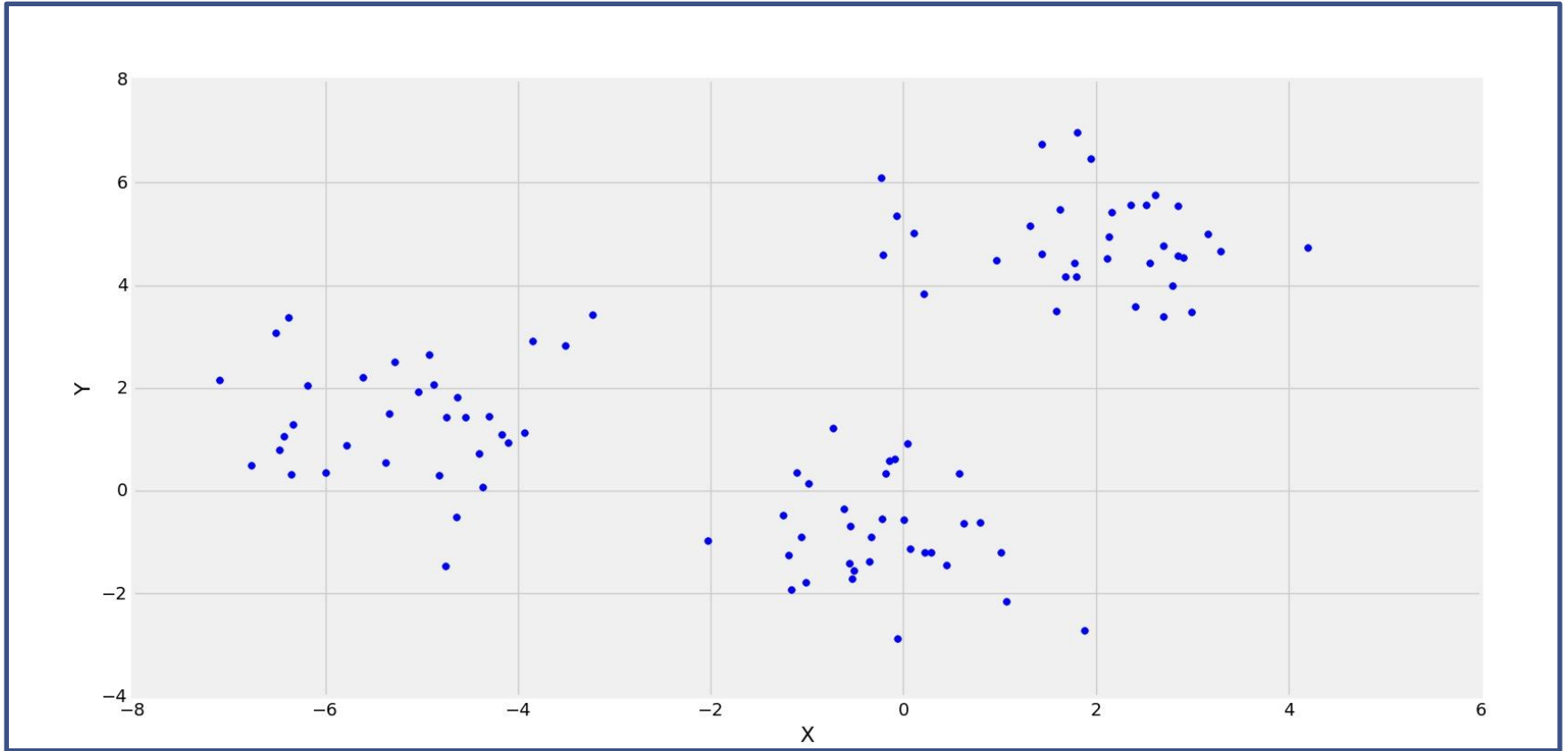
**2. The Silhouette method**

➤ **The Elbow method:** The basic idea behind this method is that it plots the various values of cost with changing  $k$ . As the value of  $K$  increases, there will be fewer elements in the cluster. So average distortion will decrease. The lesser number of elements means closer to the centroid. So, the point where this distortion declines the most is the **elbow point**.

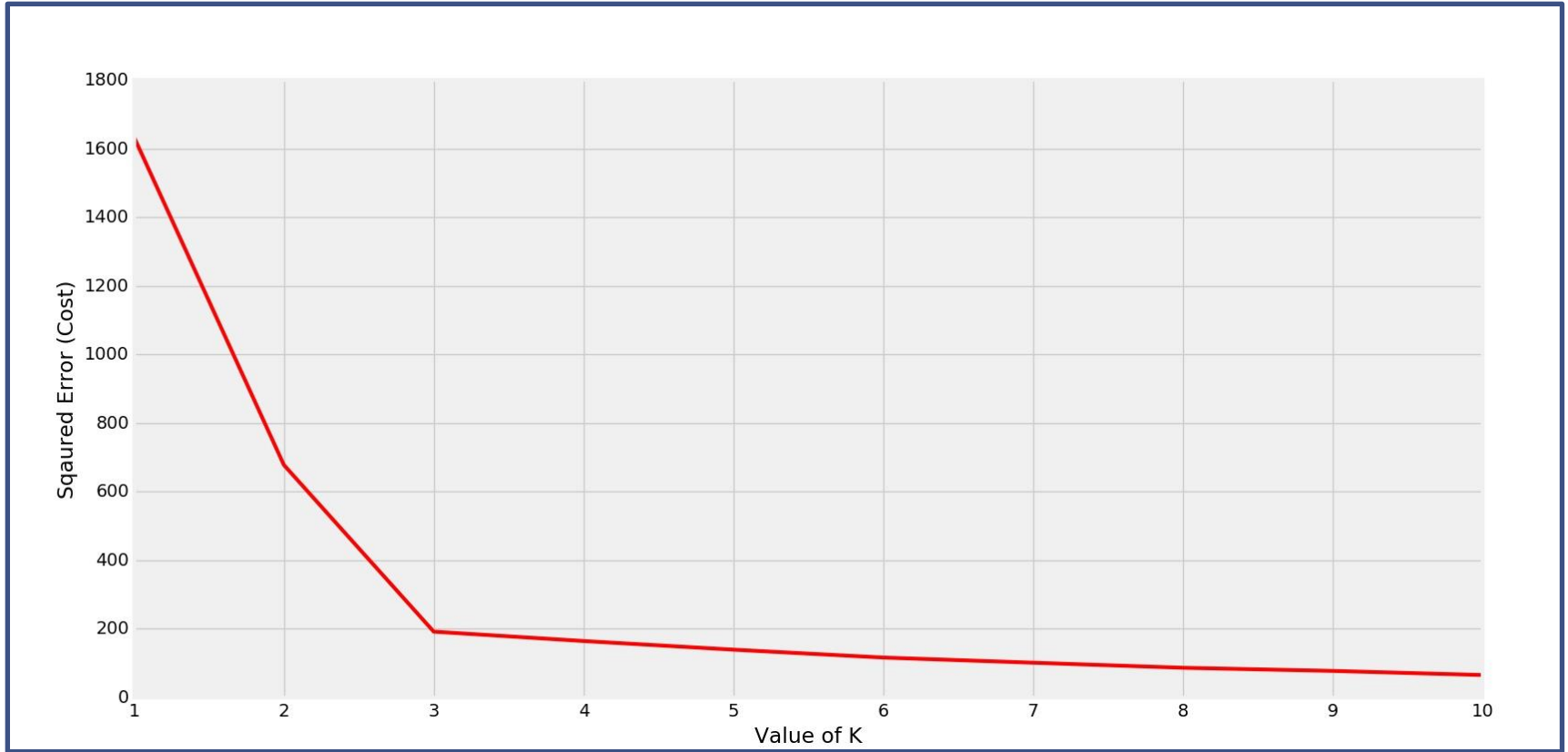




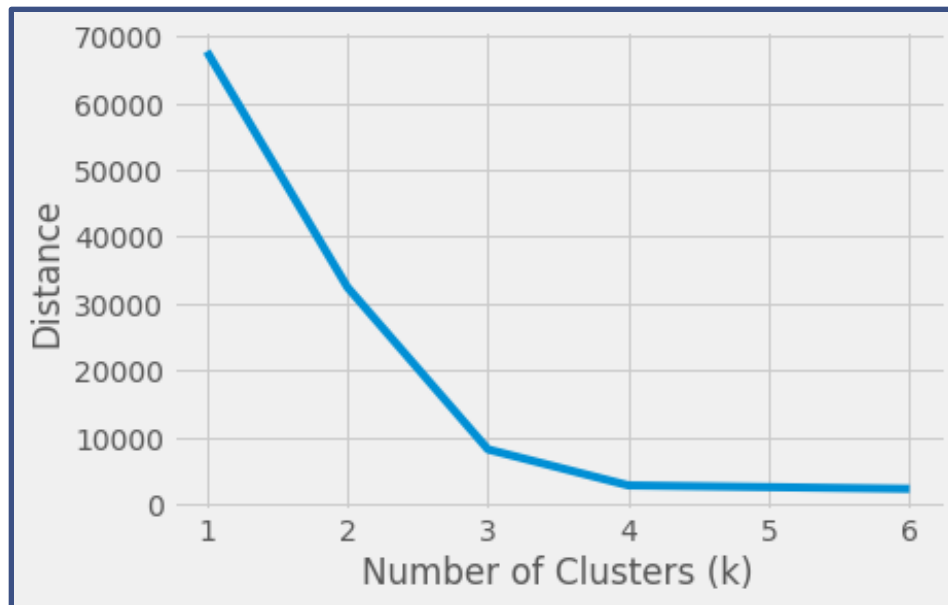
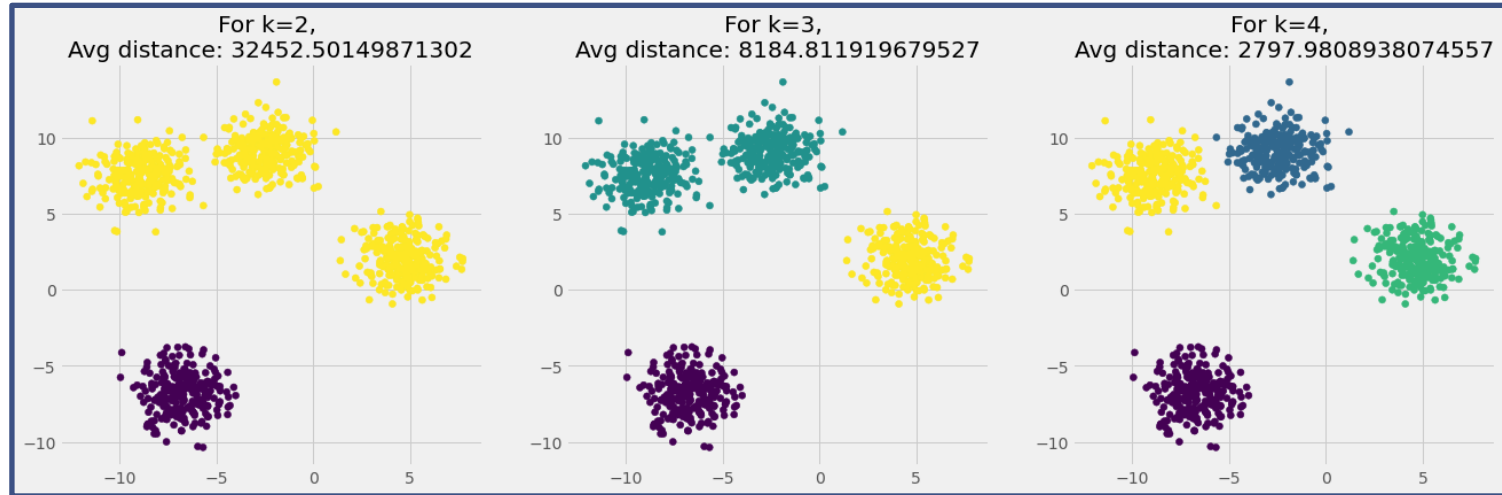
# THE ELBOW METHOD



# THE ELBOW METHOD

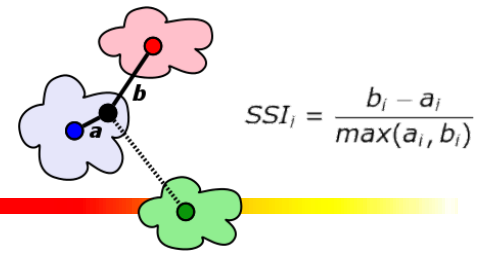


# THE ELBOW METHOD



- In the plot there is a sharp fall of average distance at **k=2, 3, and 4**. Here comes a confusion to pick the best value of k. In the plot observe the clusters formed for **k=2, 3, and 4** with their average distance.
- This data is 2-D, so it's easy to visualize and pick the best value of k, which is k=4. For higher-dimensional data, we can employ the **Silhouette Method** to find the best k, which is a **better alternative to Elbow Method**.

# THE SILHOUETTE METHOD



- **The Silhouette method:** The silhouette method computes silhouette coefficients of each point that measure how much a point is similar to its own cluster (cohesion) compared to other clusters (separation).
- The value of the silhouette ranges between  $[-1, 1]$ , where a high value indicates that the object is well matched to its own cluster and poorly matched to neighbouring clusters.
- If most objects have a high value, then the clustering configuration is appropriate.
- If many points have a low or negative value, then the clustering configuration may have too many or too few clusters.
- Clusters are formed for different values of  $k$ . And for each value of  $k$  the Silhouette score is found.

# THE SILHOUETTE METHOD



## The Silhouette Coefficient

- $a(o)$ : average distance between object  $o$  and the objects in its cluster  $A$
- $b(o)$ : average distance between object  $o$  and the objects in its "second closest" cluster  $B$
- The silhouette of  $o$  is then defined as 
$$s(o) = \frac{b(o) - a(o)}{\max\{a(o), b(o)\}}$$
- measures how good the assignment of  $o$  to its cluster is
  - $s(o) = -1$ : bad, on average closer to members of  $B$
  - $s(o) = 0$ : in-between  $A$  and  $B$
  - $s(o) = 1$ : good assignment of  $o$  to its cluster  $A$
- Silhouette Coefficient  $s_c$  of a clustering: average silhouette of all objects
  - $0.7 < s_c \leq 1.0$  strong structure,  $0.5 < s_c \leq 0.7$  medium structure
  - $0.25 < s_c \leq 0.5$  weak structure,  $s_c \leq 0.25$  no structure

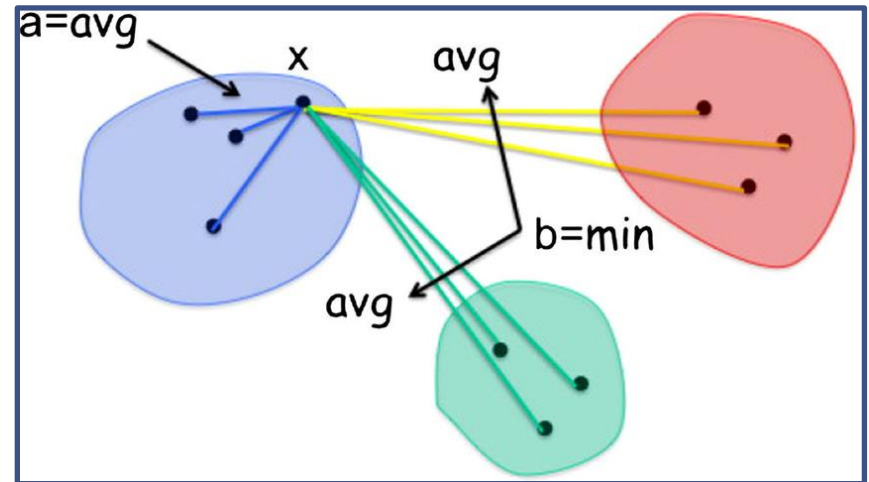
# THE SILHOUETTE METHOD

- Computing **Silhouette Score**:
- Steps to find the silhouette coefficient of an  $i^{\text{th}}$  point:

1. Compute  $a(i)$ : The average distance of that point with all other points in the same clusters.
2. Compute  $b(i)$ : The average distance of that point with all the points in the closest cluster to its cluster.
3. Compute  $s(i)$  — silhouette coefficient of  $i^{\text{th}}$  point using below mentioned formula.

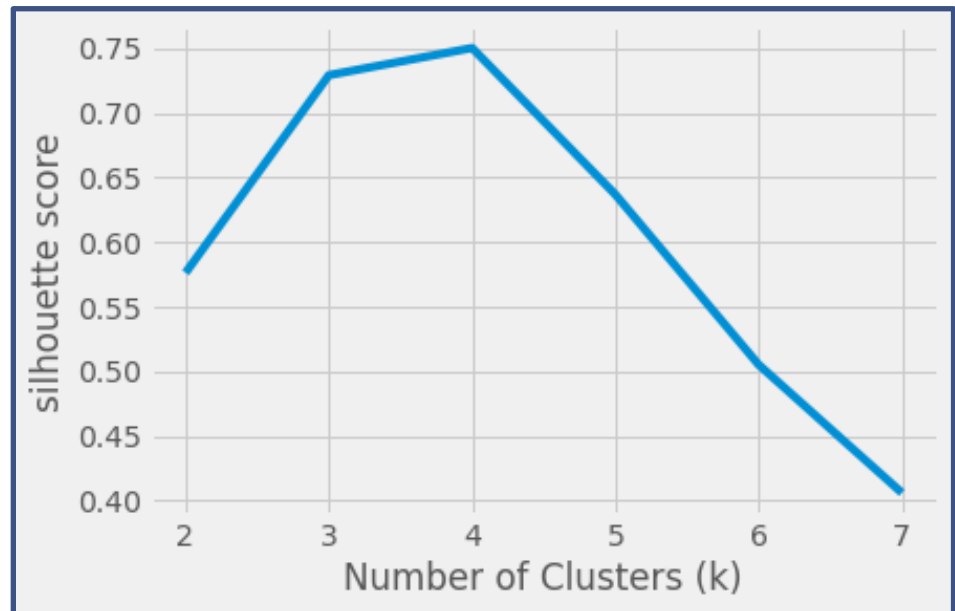
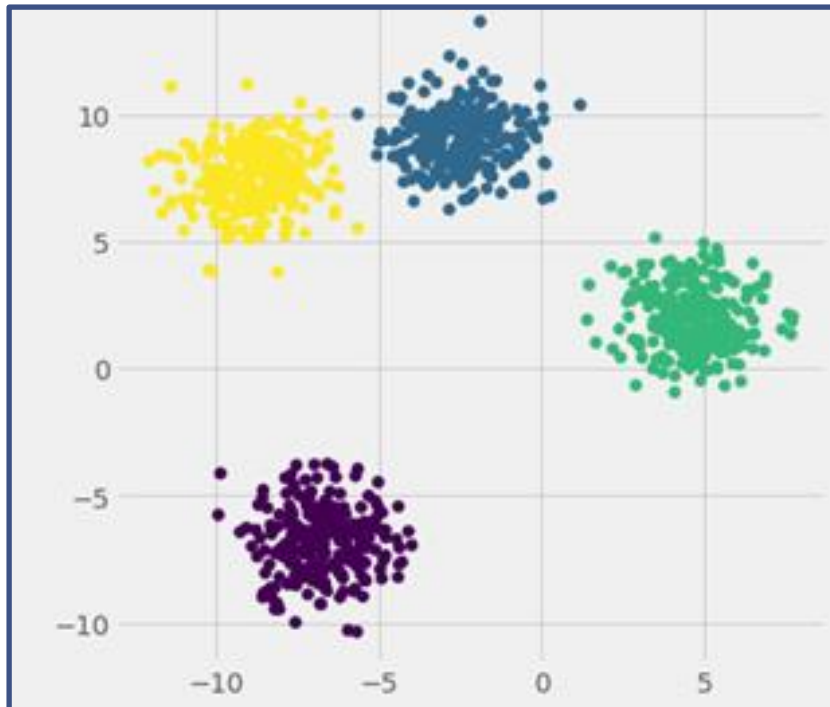
$$s(i) = \frac{b(i) - a(i)}{\max(b(i), a(i))}$$

4. After computing the silhouette coefficient for each point, average it out to get the silhouette score.

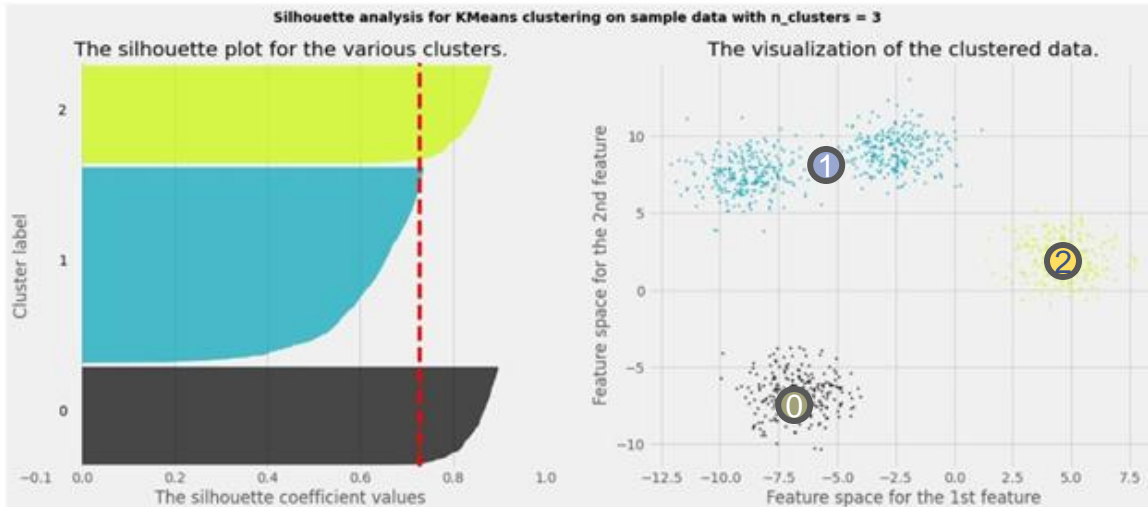
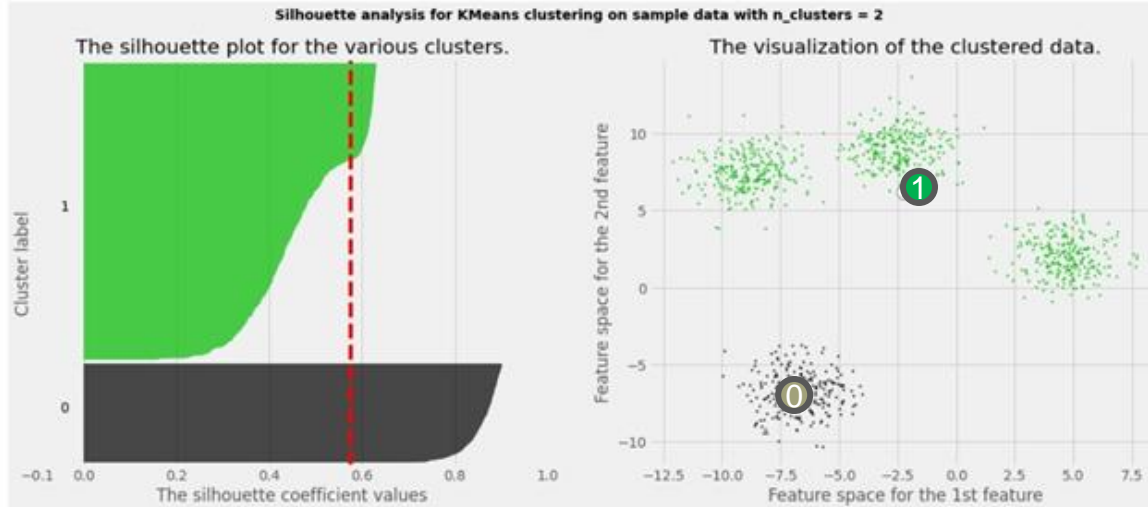


# THE SILHOUETTE METHOD

- Find the optimal value of 'k' using Silhouette Analysis:
- Similar to the previous Elbow method, we pick a range of candidate values of k (number of clusters), then train K-Means clustering for each of the values of k.
- For each k-Means clustering model represent the silhouette coefficients in a plot.



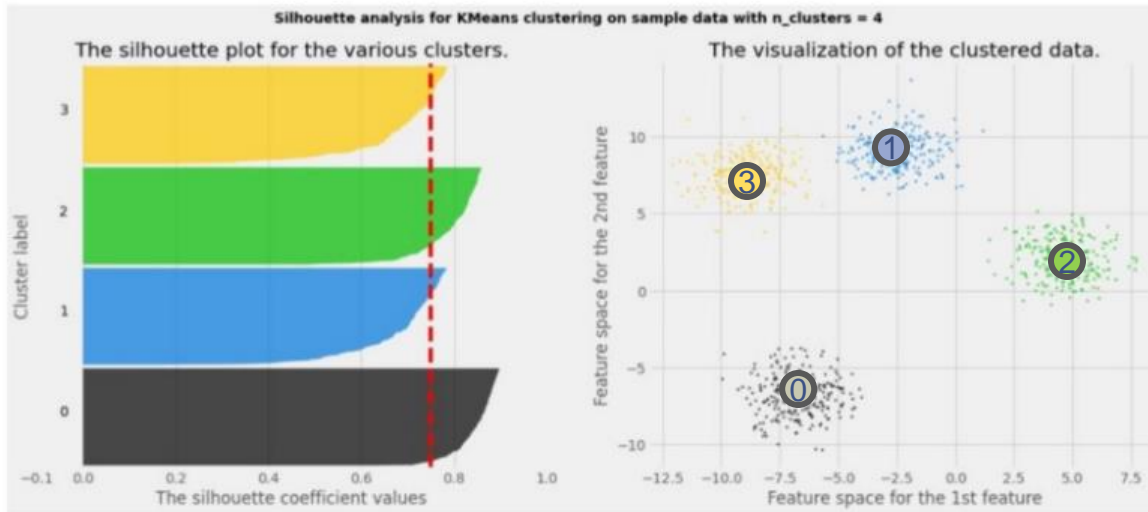
# THE SILHOUETTE METHOD



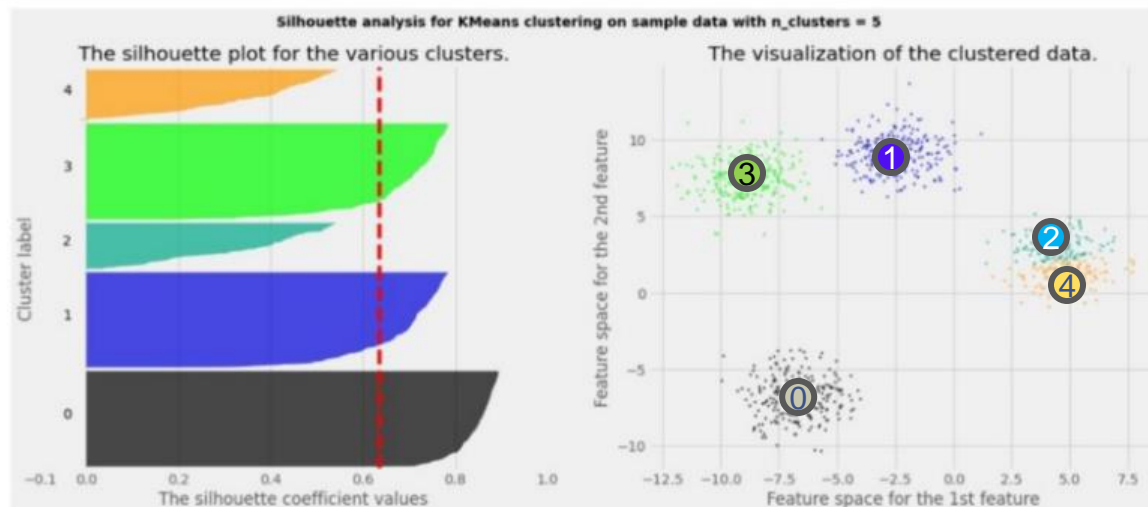
- The red dotted line is the silhouette score.
- The thickness of the silhouette plot for the cluster with  $cluster\_label=1$  when  $n\_clusters=2$ , is bigger in size owing to the grouping of the 3 sub-clusters into one big cluster.  
**Sc = 0.59**
- The silhouette plot shows that the  $n\_cluster$  value of **3** is a bad pick, as all the points in the cluster with  $cluster\_label=1$  are below-average silhouette scores.  
**Sc = 0.72**



# THE SILHOUETTE METHOD

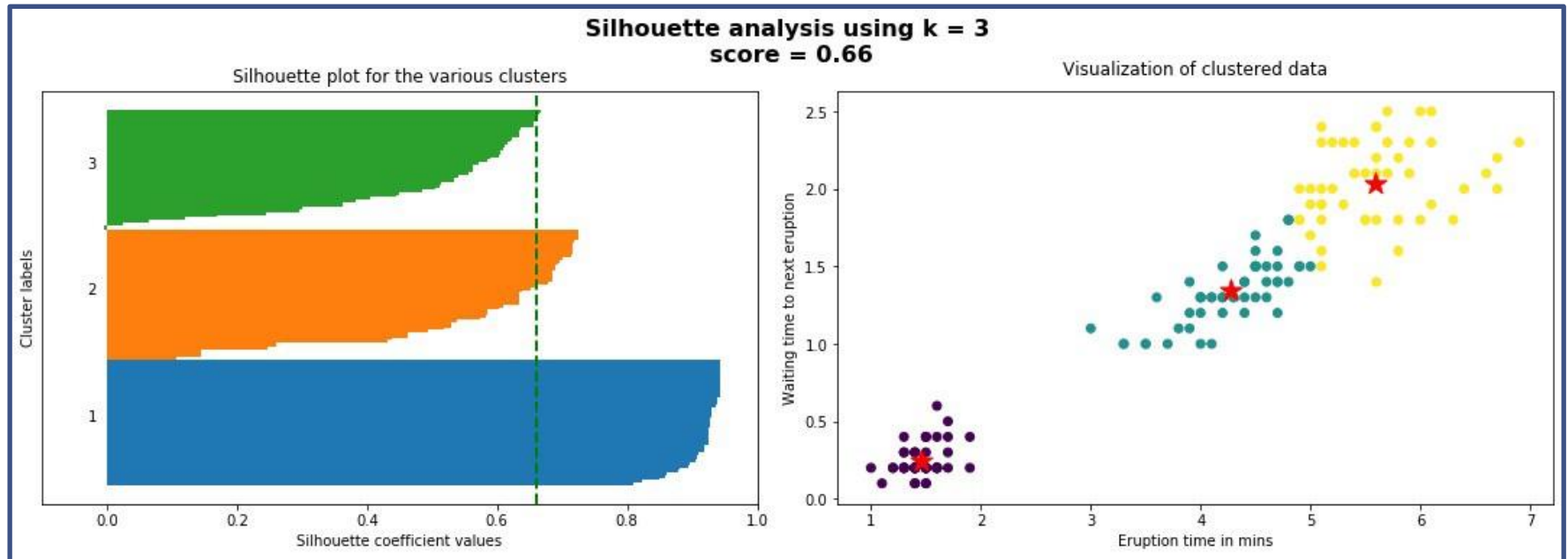


- For  $n\_clusters=4$ , all the plots are more or less of similar thickness and hence are of similar sizes, and can be considered as **best 'k'**.  
**Sc = 0.78.**



- The silhouette plot shows that the  $n\_cluster$  value of **5** is a bad pick, as all the points in the cluster with  $cluster\_label=2$  and 4 are below-average silhouette scores. **Sc = 0.63**

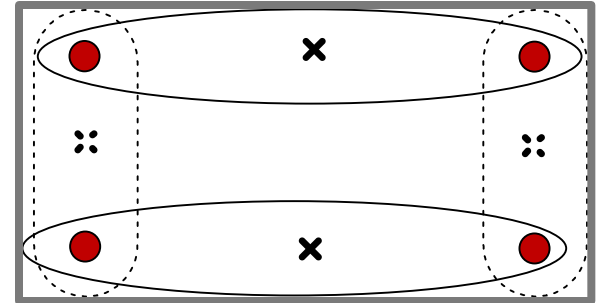
# THE SILHOUETTE METHOD



# VARIATIONS IN K-MEANS

➤ Most of the variants of the *k-means* which differ in

- Selection of the initial *k* means
- Dissimilarity calculations
- Strategies to calculate cluster means

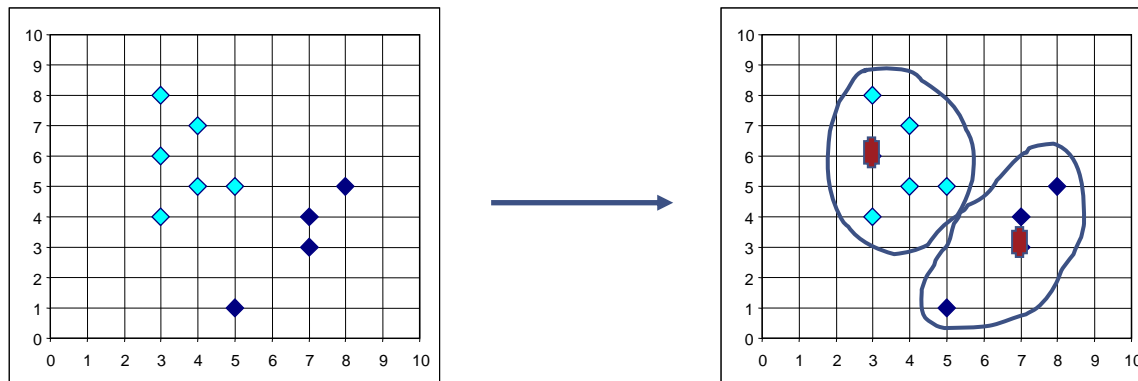


➤ Handling categorical data: *k-modes*

- Replacing means of clusters with modes
- Using new dissimilarity measures to deal with categorical objects
- Using a frequency-based method to update modes of clusters
- A mixture of categorical and numerical data: *k-prototype* method

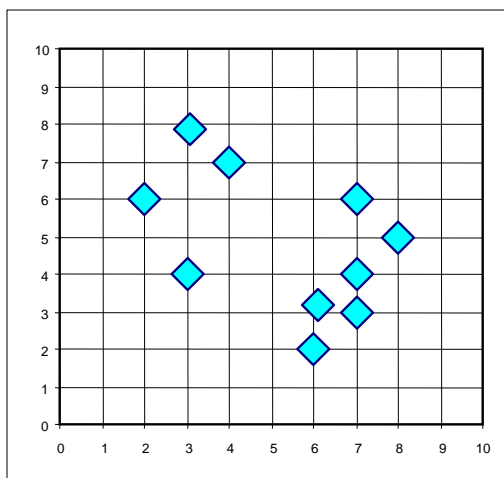
# PROBLEM WITH K-MEANS

- The k-means algorithm is sensitive to outliers !
  - Since an object with an extremely large value may substantially distort the distribution of the data
- K-Medoids: Instead of taking the **mean** value of the object in a cluster as a reference point, **medoids** can be used, which is the **most centrally located** object in a cluster



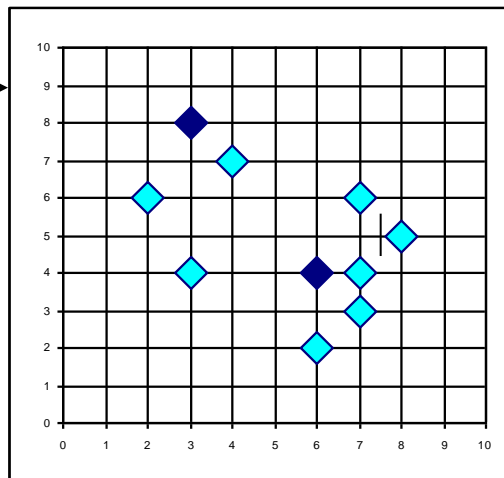
# PAM: A typical k-medoids algorithm

## Partitioning Around Medoids

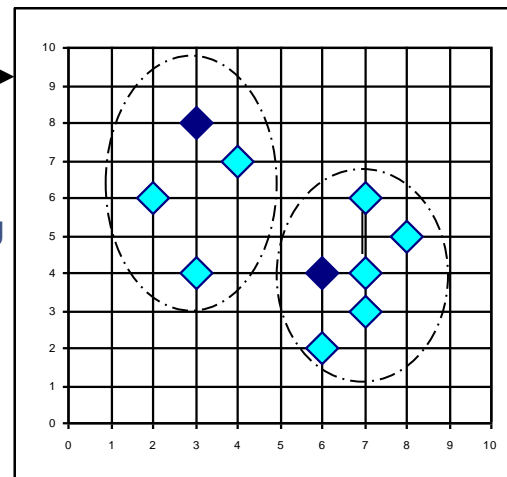


K=2

Arbitrary  
choose k  
objects as  
initial  
medoids

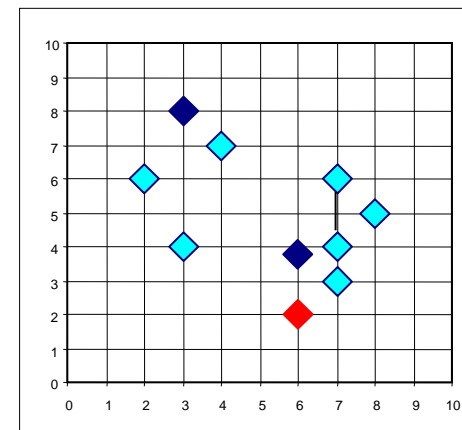


Assign  
each  
remaining  
object to  
nearest  
medoids

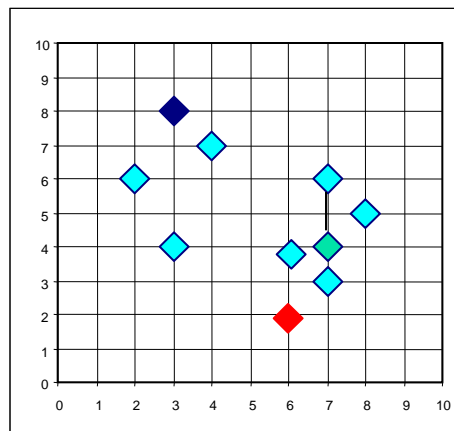


Total Cost = 20

Randomly select a  
nonmedoid object,  $O_{\text{random}}$



Compute  
total cost  
of  
swapping



Total Cost = 26

Swapping  $O$   
and  $O_{\text{random}}$   
If quality is  
improved.

**Do loop  
Until no change**

# PAM (Partitioning Around Medoids)

- **K-Medoids Clustering:** Find *representative* objects (medoids) in clusters
  - **PAM (Partitioning Around Medoids)**, Kaufmann & Rousseeuw 1987)
    1. Starts from an initial set of medoids and iteratively replaces one of the medoids by one of the non-medoids if it improves the total distance of the resulting clustering
    2. PAM works effectively for small data sets, but does not scale well for large data sets (due to the computational complexity)
- **Efficiency improvement on PAM**
  - **CLARA (Clustering LARge Applications)**(Kaufmann & Rousseeuw, 1990): PAM on samples
  - **CLARANS (Clustering Large Applications based on RANdomized Search)**(Ng & Han, 1994): Randomized re-sampling

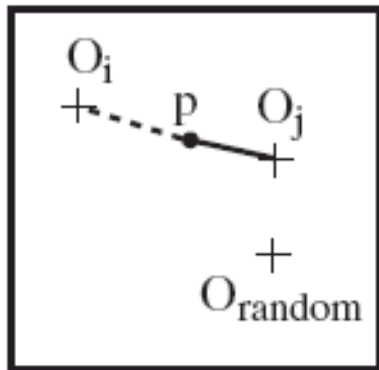
# PAM



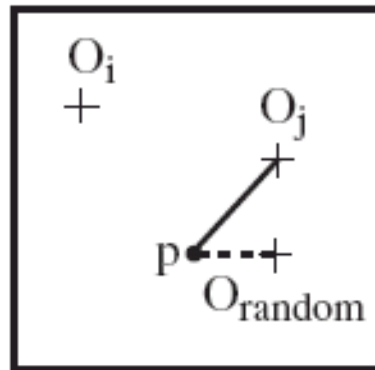
- PAM (Kaufman and Rousseeuw, 1987), built in Splus
- Use real object to represent the cluster
  1. Select  **$k$**  representative objects arbitrarily
  2. For each pair of non-selected object  **$h$**  and selected object  **$i$** , calculate the total swapping cost  **$TC_{ih}$**
  3. For each pair of  **$i$**  and  **$h$** ,
    1. If  $TC_{ih} < 0$ ,  **$i$**  is replaced by  **$h$**
    2. Then assign each non-selected object to the most similar representative object
  4. Repeat steps 2-3 until there is no change

# PAM

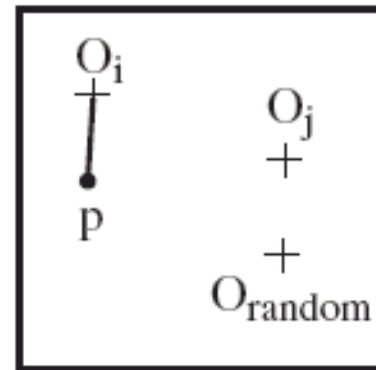
Case 1:  $p$  currently belongs to  $o_j$ . If  $o_j$  is replaced by  $o_{\text{random}}$  as a representative object and  $p$  is the closest to one of the other representative object  $o_i$ , then  $p$  is reassigned to  $o_i$



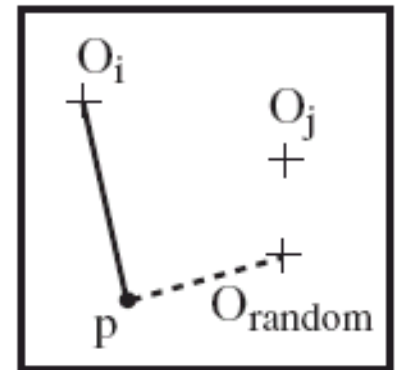
1. Reassigned to  $O_i$



2. Reassigned to  $O_{\text{random}}$



3. No change



4. Reassigned to  $O_{\text{random}}$

- data object
- + cluster center
- before swapping
- after swapping



# LIMITATIONS OF PAM

- Pam is more robust than k-means in the presence of noise and outliers because a medoid is less influenced by outliers or other extreme values than a mean
- Pam works efficiently for small data sets but does not **scale well** for large data sets.
  - $O(k(n-k)^2)$  for each iteration

where  $n$  is # of data,  $k$  is # of clusters
- Sampling-based method
  - CLARA(Clustering LARge Applications)

# CLARA(Clustering LARge Applications)

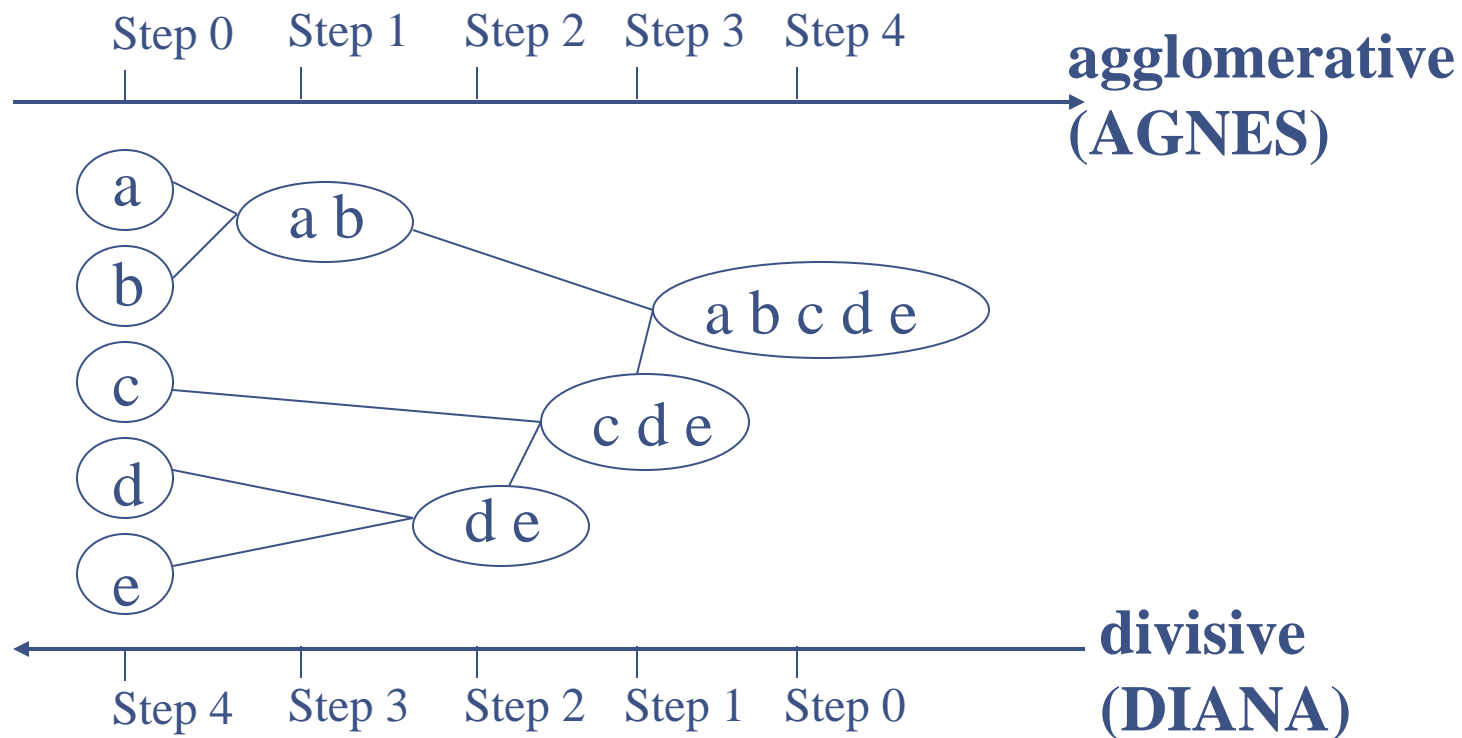
- CLARA (Kaufmann and Rousseeuw in 1990)
  - Built in statistical analysis packages, such as SPlus
  - It draws *multiple samples* of the data set, applies *PAM* on each sample, and gives the best clustering as the output
- Strength: deals with larger data sets than *PAM*
- Weakness:
  - Efficiency depends on the sample size
  - A good clustering based on samples will not necessarily represent a good clustering of the whole data set if the sample is biased

# CLARANS

- *CLARANS* (A Clustering Algorithm based on Randomized Search) (Ng and Han'94)
  1. Draws sample of neighbors dynamically
  2. The clustering process can be presented as searching a graph where every node is a potential solution, that is, a set of  $k$  medoids
  3. If the local optimum is found, *it* starts with new randomly selected node in search for a new local optimum
- Advantages: More efficient and scalable than both *PAM* and *CLARA*
- Further improvement: Focusing techniques and spatial access structures (Ester et al.'95)

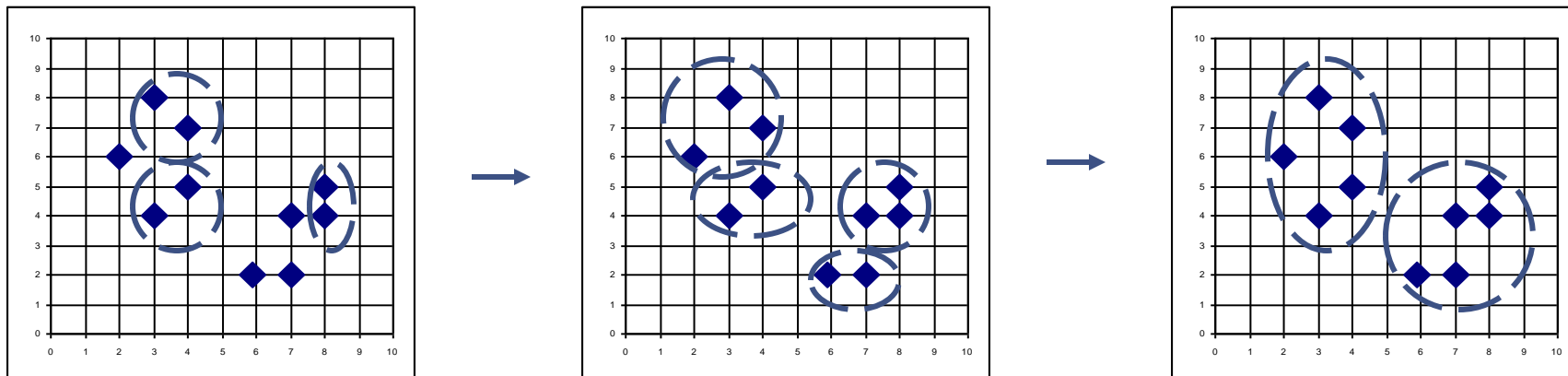
# HIERARCHICAL CLUSTERING

- Use distance matrix as clustering criteria. This method does not require the number of clusters  $k$  as an input, but needs a termination condition



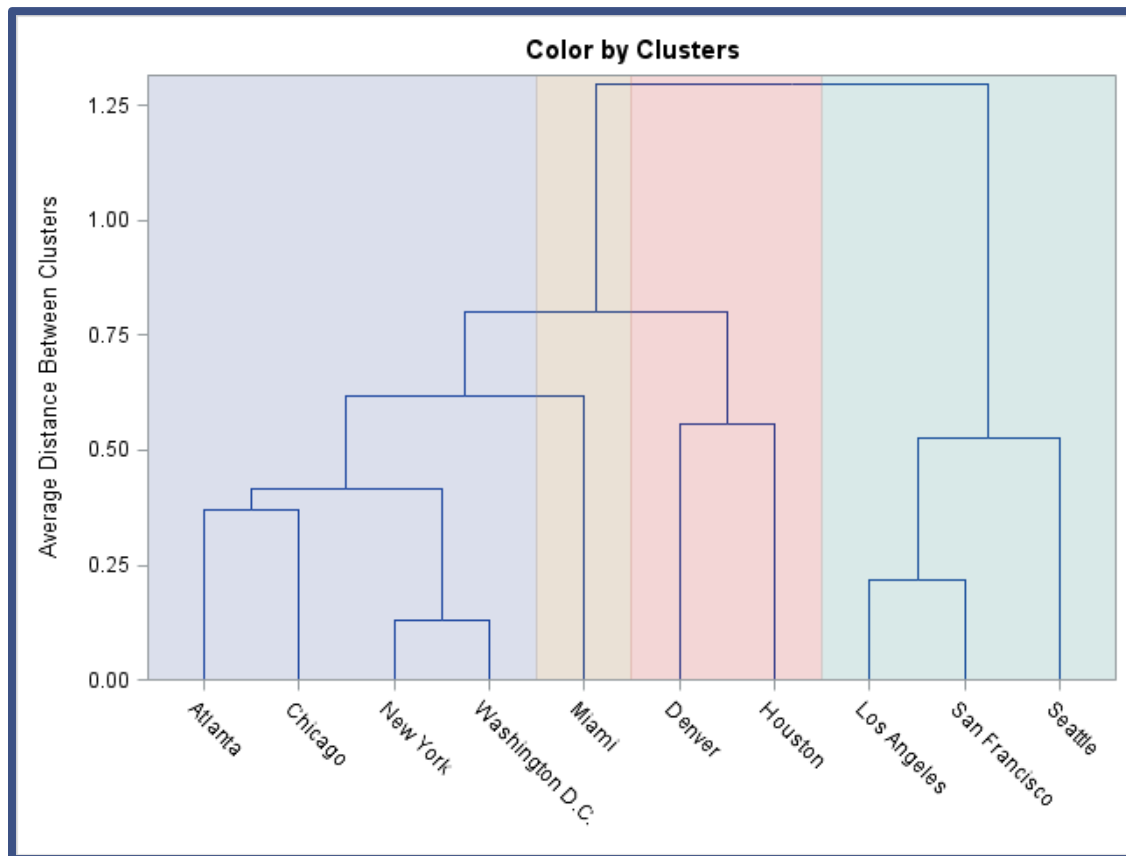
# AGNES (AGGLOMERATIVE NESTING)

- Introduced by Kaufmann and Rousseeuw (1990)
- Implemented in statistical packages, e.g., Splus
- Uses the single-link method and the dissimilarity matrix
- Merge nodes that have the least dissimilarity (most similar)
- Go on in a non-descending fashion
- Eventually all nodes belong to the same cluster



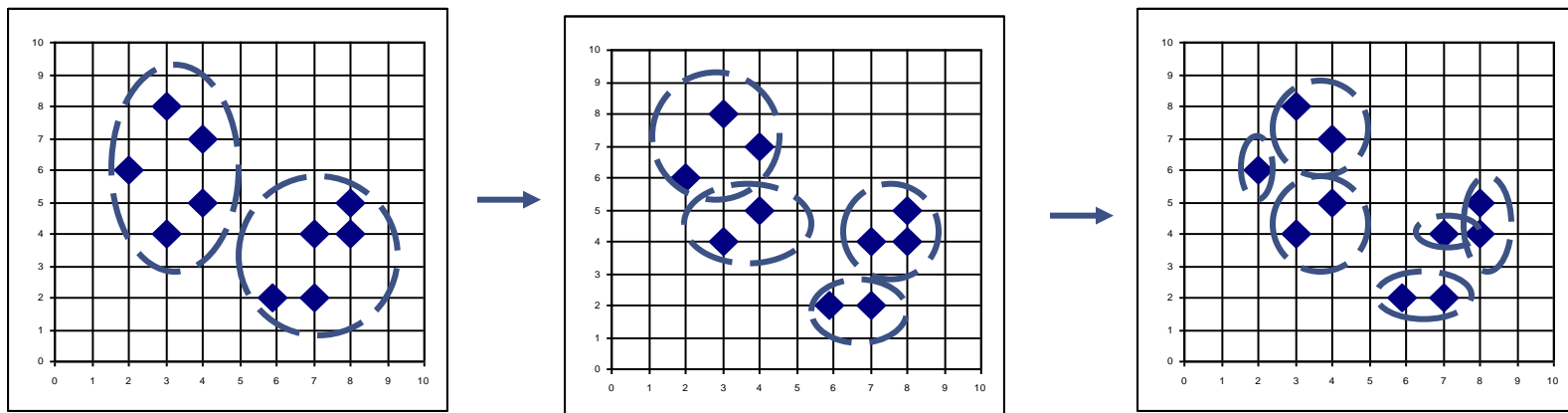
# DENDROGRAMS

- Decompose data objects into a several levels of nested partitioning (tree of clusters), called a dendrogram
- A clustering of the data objects is obtained by cutting the dendrogram at the desired level, then each connected component forms a cluster



# DIANA (DIVISIVE ANALYSIS)

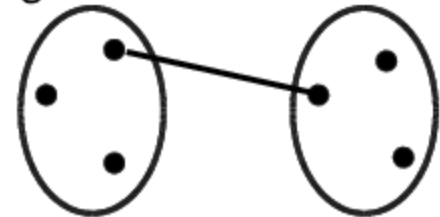
- Introduced by Kaufmann and Rousseeuw (1990)
- Implemented in statistical analysis packages, e.g., Splus
- Inverse order of AGNES
- Eventually each node forms a cluster on its own



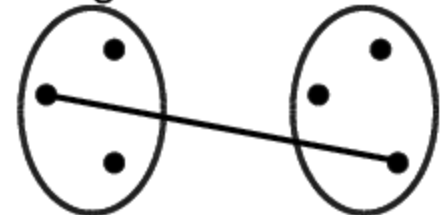
# DISTANCE BETWEEN CLUSTERS

- **Single linkage:** smallest distance between an element in one cluster and an element in the other, i.e.,  
 $\text{dist}(K_i, K_j) = \min(t_{ip}, t_{jq})$
- **Complete linkage:** largest distance between an element in one cluster and an element in the other, i.e.,  
 $\text{dist}(K_i, K_j) = \max(t_{ip}, t_{jq})$
- **Average linkage:** avg distance between an element in one cluster and an element in the other, i.e.,  
 $\text{dist}(K_i, K_j) = \text{avg}(t_{ip}, t_{jq})$

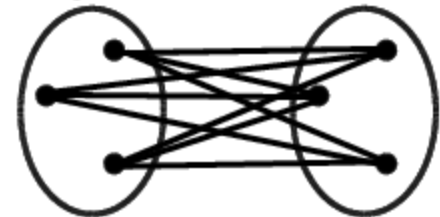
Single Linkage



Complete Linkage



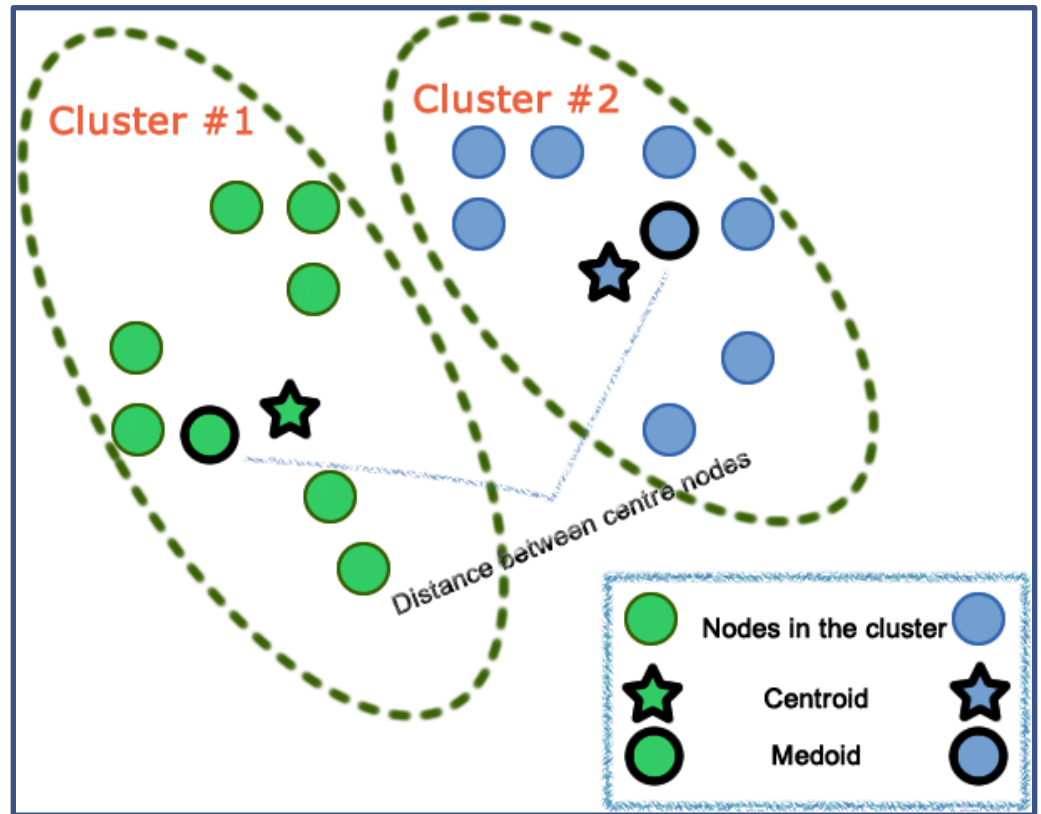
Average Linkage





# DISTANCE BETWEEN CLUSTERS

- **Centroid:** distance between the centroids of two clusters, i.e.,  
 $\text{dist}(K_i, K_j) = \text{dist}(C_i, C_j)$
- **Medoid:** distance between the medoids of two clusters, i.e.,  
 $\text{dist}(K_i, K_j) = \text{dist}(M_i, M_j)$ 
  - Medoid: a chosen, centrally located object in the cluster



# CLUSTER MEASURES

Centroid: the “middle” of a cluster.

$$C_m = \frac{\sum_{i=1}^N (t_{ip})}{N}$$

Radius: square root of average distance from any point of the cluster to its centroid.

$$R_m = \sqrt{\frac{\sum_{i=1}^N (t_{ip} - c_m)^2}{N}}$$

Diameter: square root of average mean squared distance between all pairs of points in the cluster.

$$D_m = \sqrt{\frac{\sum_{i=1}^N \sum_{i=1}^N (t_{ip} - t_{iq})^2}{N(N-1)}}$$

# DENSITY BASED CLUSTERING METHODS

- Clustering based on density (local cluster criterion), such as density-connected points
- Major features:
  - Discover clusters of arbitrary shape
  - Handle noise
  - One scan
  - Need density parameters as termination condition
- Several interesting studies:
  - DBSCAN: Ester, et al. (KDD'96)
  - OPTICS: Ankerst, et al (SIGMOD'99).
  - DENCLUE: Hinneburg & D. Keim (KDD'98)
  - CLIQUE: Agrawal, et al. (SIGMOD'98) (more grid-based)

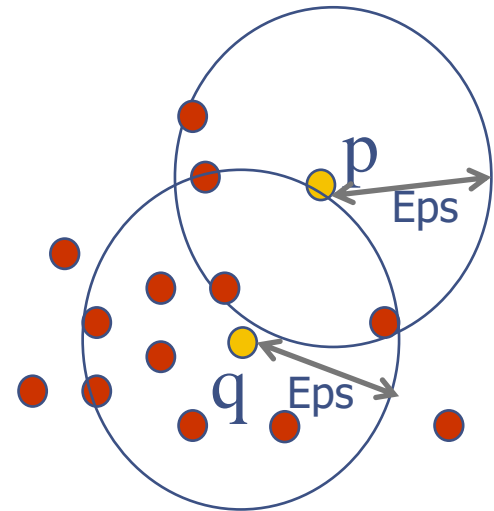
# DBSCAN Basics

DBSCAN: **D**ensity-**B**ased **S**patial **C**lustering of **A**pplications with **N**oise

➤ Two parameters:

- **Eps** : A distance measure that will be used to locate the points in the neighborhood of any point. Maximum radius of the neighbourhood
- **MinPts** : The minimum number of points (a threshold) clustered together for a region to be considered dense. Minimum number of points in an Eps neighbourhood of that point.  $N_{Eps}(p)$  is the neighbourhood of p. D is the dataset.

$N_{Eps}(p)$ : {q belongs to D |  $\text{dist}(p,q) \leq \text{Eps}$ }



**MinPts = 5**

**Eps = 1 cm**

# DBSCAN Basics

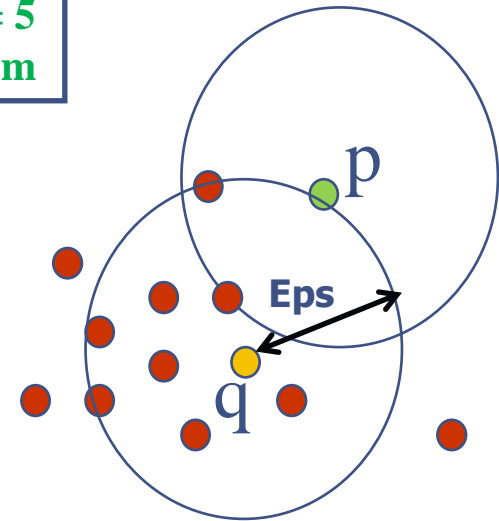
➤ **Directly density-reachable:** A point  $p$  is **directly density-reachable** from a point  $q$  w.r.t.  $Eps$ ,  $MinPts$  if

- $p$  belongs to  $N_{Eps}(q)$
- core point condition:

$$|N_{Eps}(q)| \geq MinPts$$

i.e.  $q$  is a core point

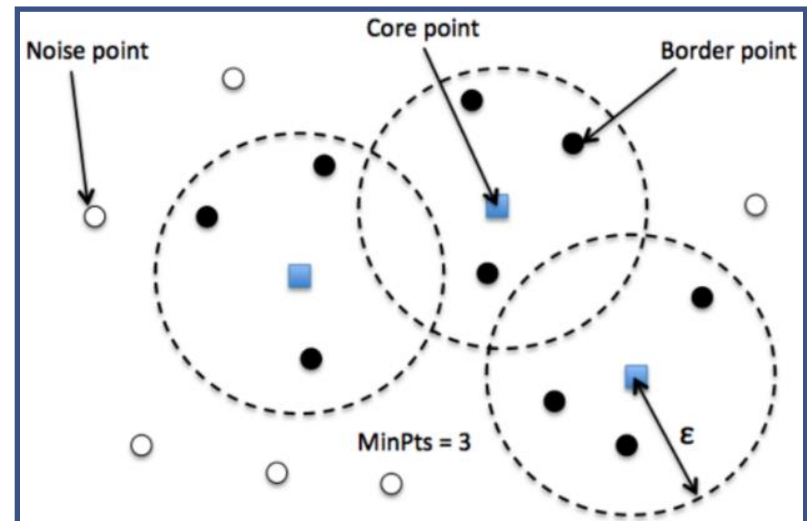
$MinPts = 5$   
 $Eps = 1 \text{ cm}$



➤ **Core** — This is a point that has at least  $MinPts$  number of points within distance  $Eps$  from itself.

➤ **Border** — This is a point that has at least one Core point at a distance  $Eps$ .

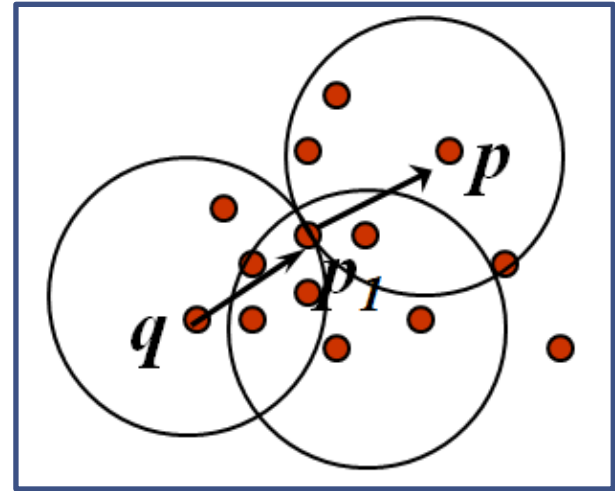
➤ **Noise** — This is a point that is neither a Core nor a Border. And it has less than  $MinPts$  points within distance  $Eps$  from itself.



# Density Reachable & Density Connected

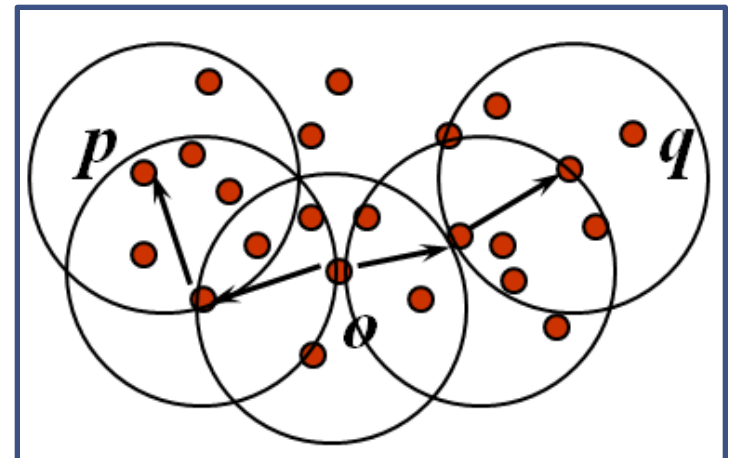
## ➤ Density-reachable:

- A point  $p$  is **density-reachable** from a point  $q$  w.r.t.  $Eps$ ,  $MinPts$  if there is a chain of points  $p_1, \dots, p_n$  and,  $p_1 = q$  and  $p_n = p$  such that  $p_{i+1}$  is directly density-reachable from  $p_i$



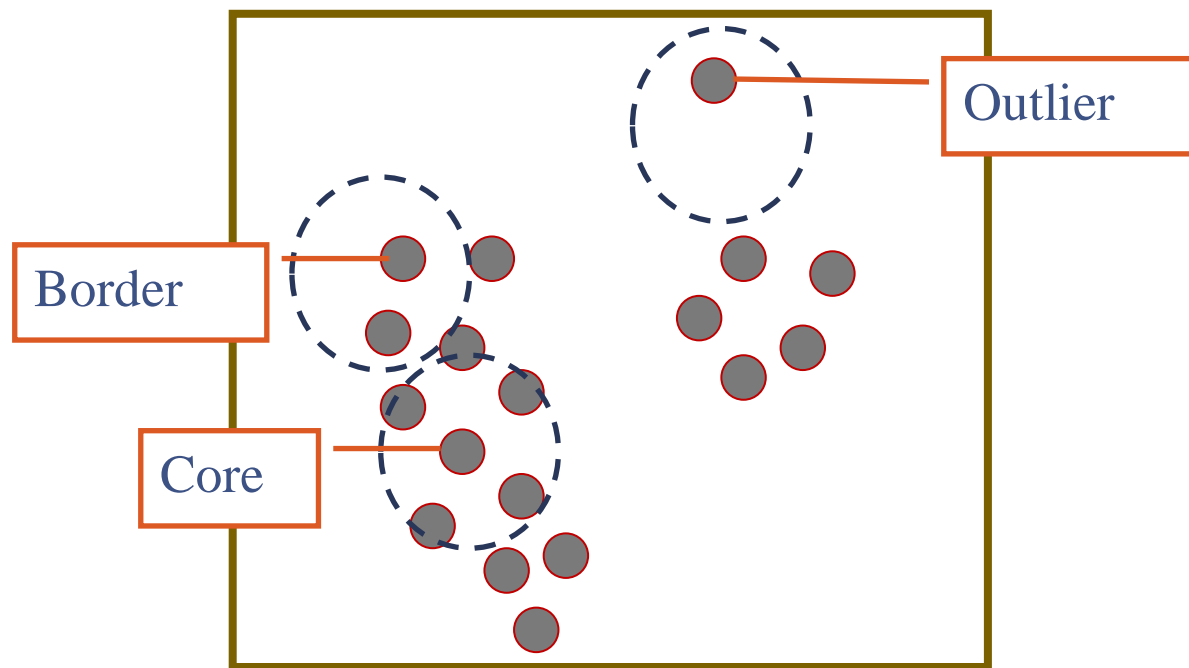
## ➤ Density-connected

- A point  $p$  is **density-connected** to a point  $q$  w.r.t.  $Eps$ ,  $MinPts$  if there is a point  $o$  such that both,  $p$  and  $q$  are density-reachable from  $o$  w.r.t.  $Eps$  and  $MinPts$



# DBSCAN

- 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



Eps = 1cm

MinPts = 5

# DBSCAN ALGORITHM

---

1. Arbitrary select a point  $p$
2. Retrieve all points density-reachable from  $p$  w.r.t.  $Eps$  and  $MinPts$
3. If  $p$  is a core point, a cluster is formed
4. If  $p$  is a border point and no points are density-reachable from  $p$  then DBSCAN visits the next point of the database
5. Continue the process until all of the points have been processed



# DBSCAN: Advantages & Limitations

## ➤ Advantages of DBSCAN:

- Discovers clusters of arbitrary shape
- Detect noise points
- Do not need to mention the number of clusters to be created

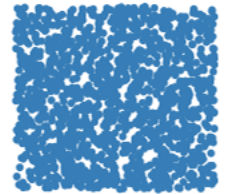
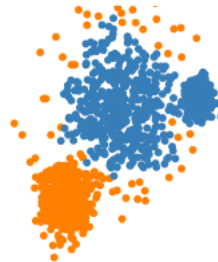
## ➤ Limitations of DBSCAN:

- Need to specify MinPts and Eps. Will not generate correct clusters if density within the data differs
- The issue of transitivity can exist i.e. if point A is similar to B and B is similar to C, then A and C will also belong to the same cluster. There is every possibility that there is nothing common between A and C.

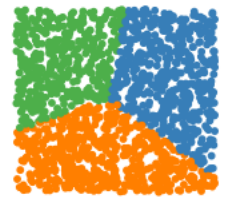
Objects	Hair	Eyes	Height (feet)	Weight (kg)	Music	Class
A	Black	Blue	5	45	Y	MCA-II
B	Black	Green	5	65	Y	BE-IV
C	Red	Green	6	65	N	BE-IV

# K-Means and DBSCAN

DBSCAN



k-means



# Parameter Estimation

- **minPts:** As a rule of thumb, a minimum minPts can be derived from the number of dimensions  $D$  in the data set, as **minPts  $\geq D + 1$** . The low value **minPts = 1** does not make sense, as then every point on its own will already be a cluster. With **minPts  $\leq 2$** , the result will be the same as of hierarchical clustering with the single link metric, with the dendrogram cut at height  $\epsilon$ . Therefore, minPts must be chosen at least 3. However, larger values are usually better for data sets with noise and will yield more significant clusters. As a rule of thumb, **minPts = 2·dim** can be used, but it may be necessary to choose larger values for very large data, for noisy data or for data that contains many duplicates.
- **$\epsilon$ :** The value for  $\epsilon$  can then be chosen by using a k-distance graph, plotting the distance to the **k = minPts-1** nearest neighbor ordered from the largest to the smallest value. Good values of  $\epsilon$  are where this plot shows an “elbow”: if  $\epsilon$  is chosen much too small, a large part of the data will not be clustered; whereas for a too high value of  $\epsilon$ , clusters will merge and the majority of objects will be in the same cluster. In general, small values of  $\epsilon$  are preferable, and as a rule of thumb, only a small fraction of points should be within this distance of each other.
- **Distance function:** The choice of distance function is tightly linked to the choice of  $\epsilon$ , and has a major impact on the outcomes. In general, it will be necessary to first identify a reasonable measure of similarity for the data set, before the parameter  $\epsilon$  can be chosen. There is no estimation for this parameter, but the distance functions need to be chosen appropriately for the data set.

# ASSESSMENT METRICS FOR CLUSTERING

There are three popular methods to assess clustering:

➤ The Davies-Bouldin Index

The DB Index is calculated by the following formula:

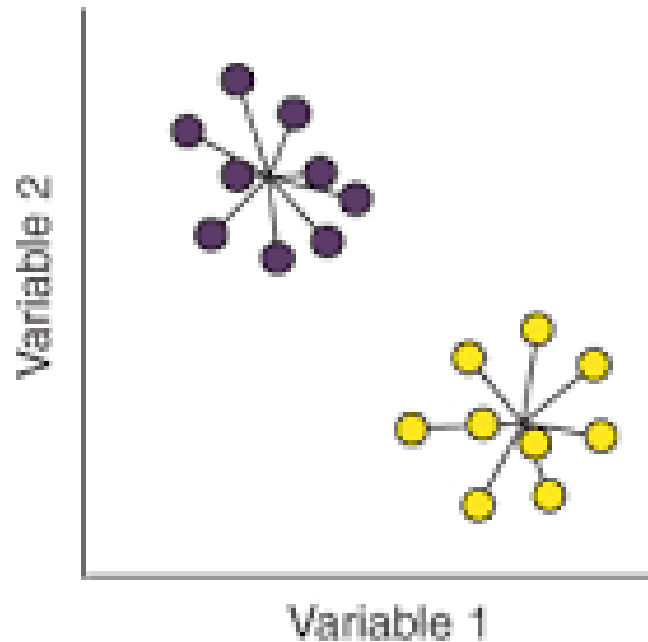
$$DB = \frac{1}{n} \sum_{i=1}^n \max_{j \neq i} \left( \frac{\sigma_i + \sigma_j}{d(c_i, c_j)} \right)$$

where,  $n$  is the number of clusters and  $\sigma_i$  is the average distance of all points in cluster  $i$  from the cluster centroid  $c_i$ .

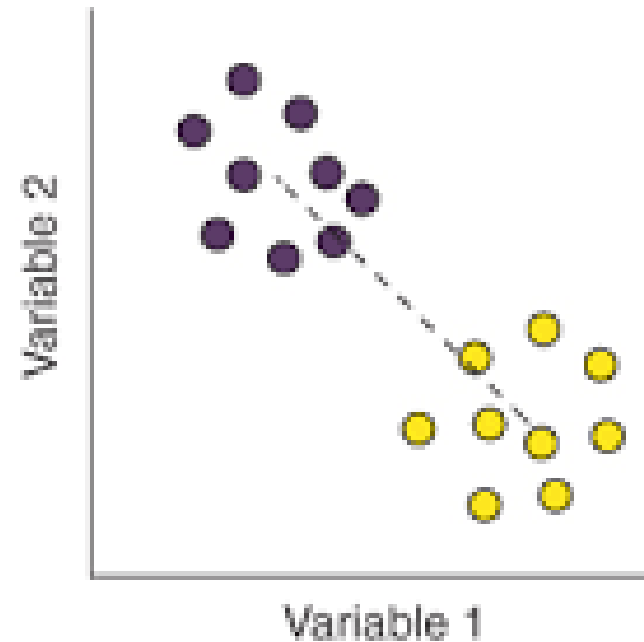
- The DB index captures the intuition that clusters that are:
  1. well-spaced from each other and
  2. themselves very dense are likely a 'good' clustering.
- This is because the measure's 'max' statement repeatedly selects the values where the average point is farthest away from its centroid, and where the centroids are closest together.
- As the DB index shrinks, the clustering is considered 'better'.

# DAVIS BOULDIN INDEX

Intraccluster variance



Distance between centroids

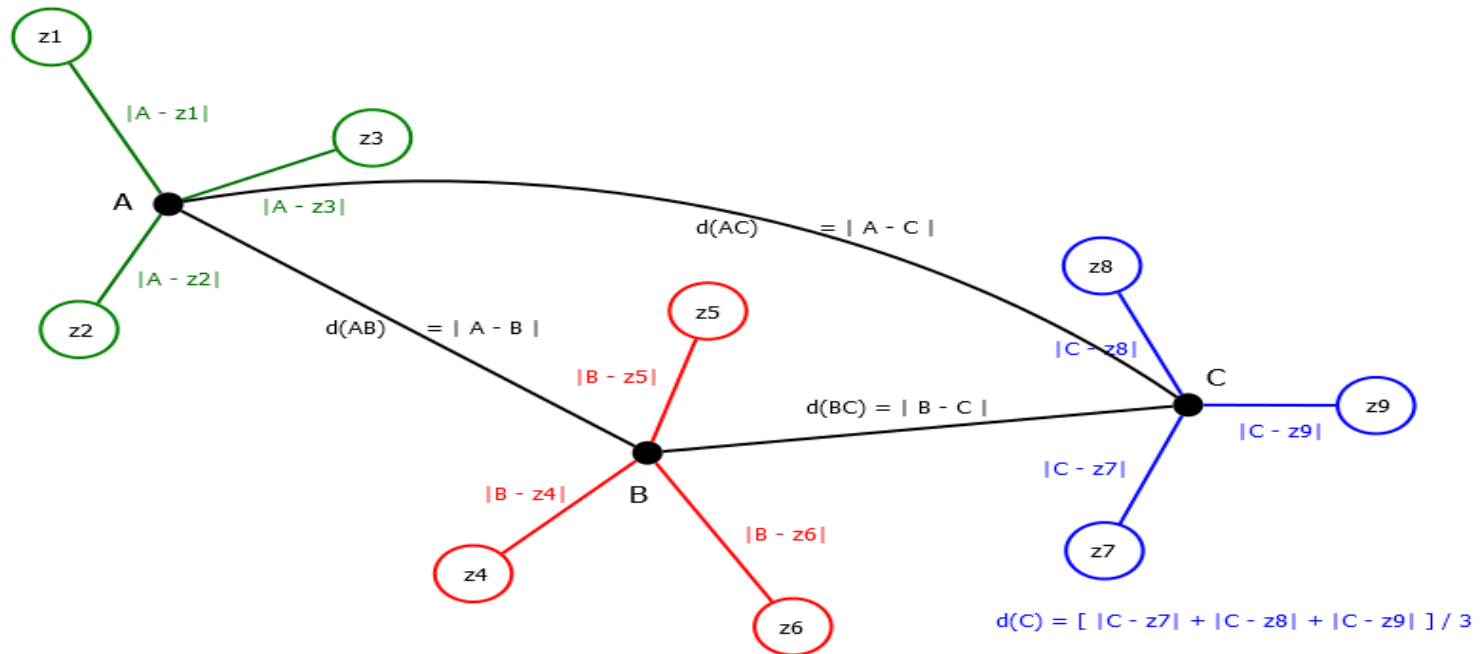


# DAVIS BOULDIN INDEX

## Davies-Bouldin Index

$$DB = \text{MAX} \left[ \begin{aligned} &(d(A) + d(B)) / d(AB), \\ &(d(B) + d(C)) / d(BC), \\ &(d(A) + d(C)) / d(AC) \end{aligned} \right]$$

$$d(A) = [ |A - z1| + |A - z2| + |A - z3| ] / 3$$



$$d(B) = [ |B - z4| + |B - z5| + |B - z6| ] / 3$$

# THE DUNN INDEX

## ➤ The Dunn Index

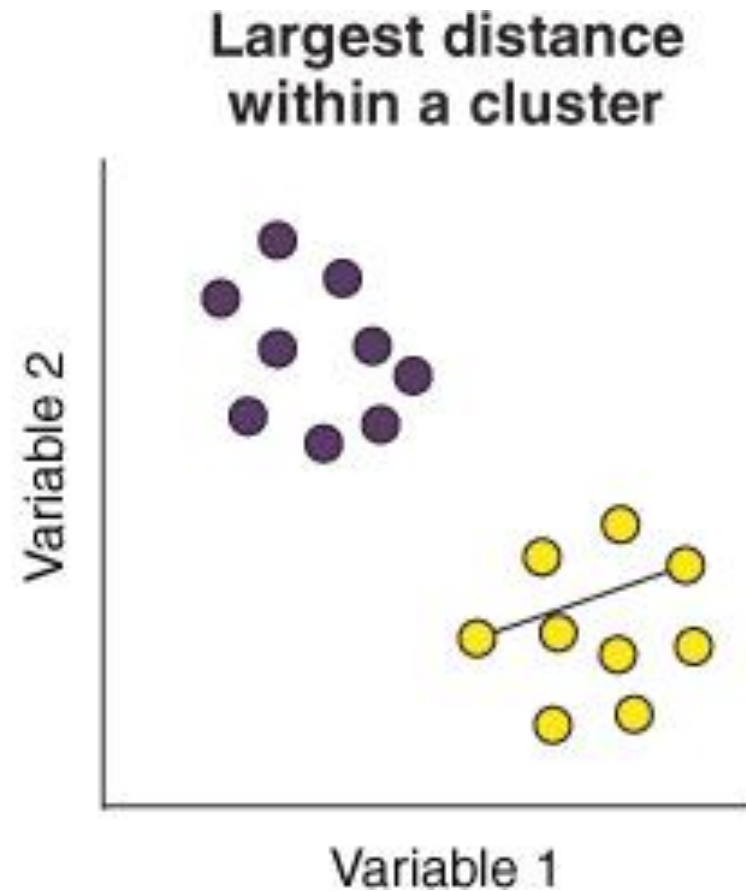
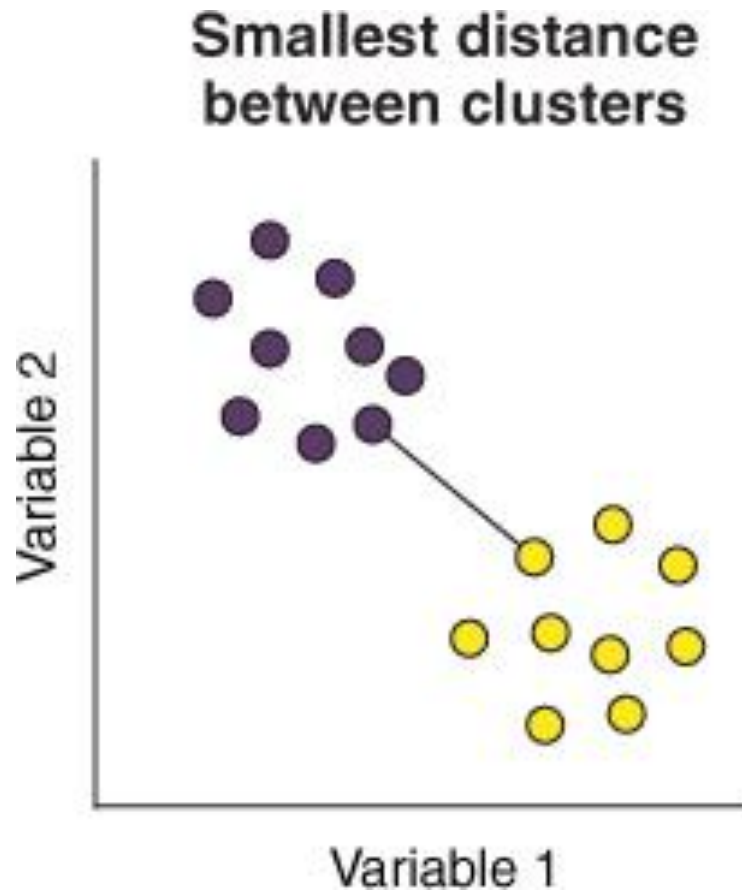
The D Index is calculated by the following formula:

$$D = \frac{\min_{1 \leq i < j \leq n} d(i, j)}{\max_{1 \leq k \leq n} d'(k)}$$

where  $i, j$  and  $k$  are each indices for clusters,  $d$  measures the inter-cluster distance and  $d'$  measures the intra-cluster difference.

- The Dunn Index captures the same idea as the DB Index: it gets better when clusters are well-spaced and dense. But the Dunn Index increases as performance improves.
- While the DB index considers the dispersion and separation of all clusters, the Dunn Index only considers the worst cases in the clustering: the clusters that are closest together and the single most dispersed cluster.
- Depending on your application, the change in objective may introduce unexpected problems.

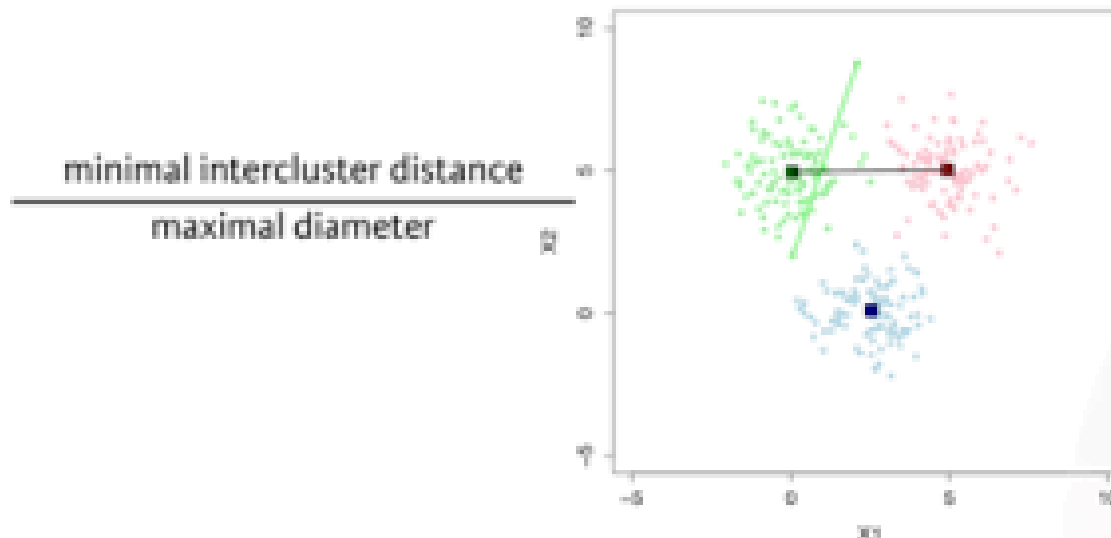
# THE DUNN INDEX





# THE DUNN INDEX

## Dunn's index



$$\text{Dunn Index} = \frac{\min(\text{Inter cluster distance})}{\max(\text{Intra cluster distance})}$$

Clusters are compact

# THE SILHOUETTE COEFFICIENT

## ➤ The Silhouette Coefficient

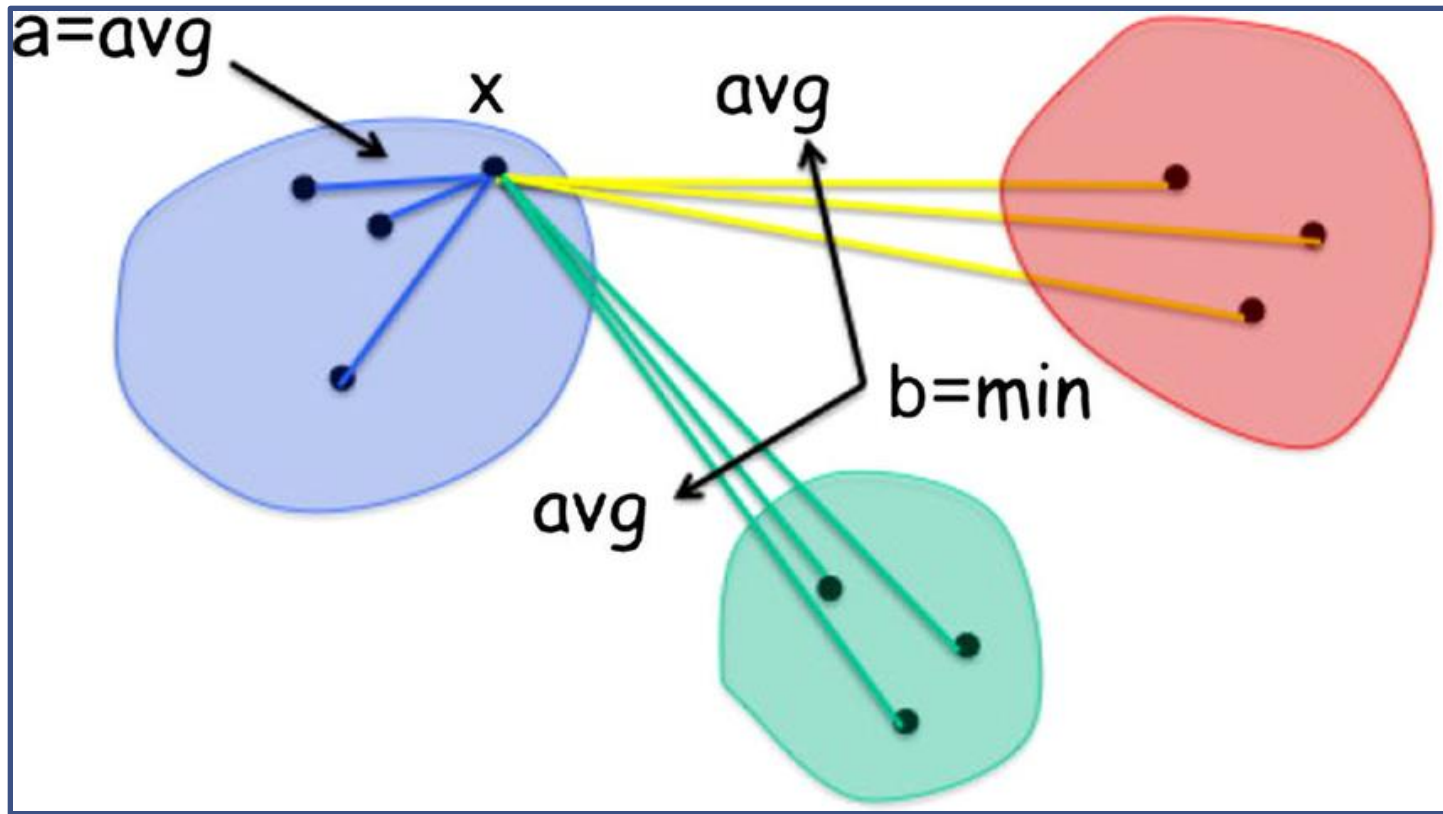
The Silhouette Coefficient is calculated by the following formula:

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

where  $a(i)$  is the average distance of point  $i$  from all other points in its cluster and  $b(i)$  is the smallest average distance of  $i$  to all points in any other cluster.

- To clarify,  $b(i)$  is found by measuring the average distance of  $i$  from every point in cluster A, the average distance of  $i$  from every point in cluster B, and taking the smallest resulting value.
- The Silhouette Coefficient tells us how well-assigned each individual point is. If  $S(i)$  is close to 0, it is right at the inflection point between two clusters.
- If it is closer to -1, then we would have been better off assigning it to the other cluster.
- If  $S(i)$  is close to 1, then the point is well-assigned and can be interpreted as belonging to an 'appropriate' cluster.
- The Silhouette Coefficient is a very intuitive and sophisticated measure of distance.
- Its downfall is that it can be extremely expensive to compute on all  $n$  points. This is because we must compute the distance of  $i$  from all other  $n - 1$  points for each  $i$ , which leads to a complexity of  $O(n^2)$ .

# THE SILHOUETTE COEFFICIENT



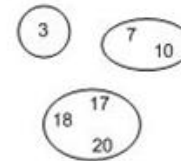
# Summary

- **Cluster analysis** groups objects based on their **similarity** and has wide applications
- Measure of similarity can be computed for **various types of data**
- Clustering algorithms can be **categorized** into partitioning methods, hierarchical methods, density-based methods, grid-based methods, and model-based methods
- **K-means** and **K-medoids** algorithms are popular partitioning-based clustering algorithms
- **Birch** and **Chameleon** are interesting hierarchical clustering algorithms, and there are also probabilistic hierarchical clustering algorithms
- **DBSCAN**, **OPTICS**, and **DENCLU** are interesting density-based algorithms
- **STING** and **CLIQUE** are grid-based methods, where CLIQUE is also a subspace clustering algorithm
- Quality of clustering results can be evaluated in various ways

END

# DAVIS BOULDIN INDEX

- For eg: for the clusters shown



- Compute  $R_{ij} = \frac{\text{var}(C_i) + \text{var}(C_j)}{\|c_i - c_j\|}$

$$\text{var} = \frac{\sum (x_i - \bar{x})^2}{n - 1}$$


- $\text{var}(C_1)=0, \text{var}(C_2)=4.5, \text{var}(C_3)=2.33$
- Centroid is simply the mean here, so  $c_1=3, c_2=8.5, c_3=18.33$
- So,  $R_{12}=1, R_{13}=0.152, R_{23}=0.797$

- Now, compute  $R_i = \max_{j=1, \dots, k, i \neq j} R_{ij}$

- $R_1=1$  (max of  $R_{12}$  and  $R_{13}$ );  $R_2=1$  (max of  $R_{21}$  and  $R_{23}$ );  $R_3=0.797$  (max of  $R_{31}$  and  $R_{32}$ )

- Finally, compute  $DB = \frac{1}{k} \sum_{i=1}^k R_i$

- DB=0.932



c1 = 3  
c2 = 7, 10  
c3 = 20, 18, 17  
m1 = 3  
m2 = 8.5  
m3 = 18.33  
var(c1) = 0

var(c2) = ((8.5-7)^2 + (10-8.5)^2) / (2-1) = 4.5

var(c3) = ((20-18.33)^2 + (18-18.33)^2 + (17-18.33)^2) / (3-1) = 2.33

R12 = (0+4.5) / (8.5 - 3) = 0.81

R13 = 0.15

R23 = 0.69

$$S^2 = \frac{\sum (x_i - \bar{x})^2}{n - 1}$$

# DAVIS BOULDIN INDEX

