DI501 Introduction to Data Informatics

Lecture 6

Clustering

Unsupervised Learning

- There is no known output, no teacher to instruct the learning algorithm.
  - We cannot compare the model output with the correct outputs.
- We just have the observations (a dataset that consists of instances with known features) and we extract knowledge from this dataset.
- Examples:
  - Dimensionality reduction
  - Clustering
  - Anomaly detection
  - …

Similarity

- Similarity underlies many data science methods and solutions to business problems.
- If two things (people, companies, products) are similar in some ways they often share other characteristics as well.
- We often group things by similarity or search for the right sort of similarity.
  - Supervised: we can create boundaries for grouping instances together that have similar values for their target variables.
  - Unsupervised: we can group similar items together into clusters

Cluster Analysis

- A cluster is a collection of data objects which are
  - Similar (or related) to one another within the same group (i.e., cluster)
  - Dissimilar (or unrelated) to the objects in other groups (i.e., clusters)
- Cluster analysis (or clustering, data segmentation, …)
  - Given a set of data points, partition them into a set of groups (i.e., clusters) which are as similar as possible
- Cluster analysis is unsupervised learning (i.e., no predefined classes)
  - This contrasts with classification (i.e., supervised learning)
- Typical ways to use/apply cluster analysis
  - As a stand-alone tool to get insight into data distribution, or
  - As a preprocessing (or intermediate) step for other algorithms

Cluster Analysis Applications

- A key intermediate step for other data mining tasks
  - Generating a compact summary of data for classification, pattern discovery, hypothesis generation and testing, etc.
  - Outlier detection: Outliers—those "far away" from any cluster
- Data summarization, compression, and reduction
  - Ex. Image processing: Vector quantization
- Collaborative filtering, recommendation systems, or customer segmentation
  - Find like-minded users or similar products
- Dynamic trend detection
  - Clustering stream data and detecting trends and patterns
- Multimedia data analysis, biological data analysis and social network analysis
  - Ex. Clustering images or video/audio clips, gene/protein sequences, etc.

Cluster Analysis

- A good clustering method will produce high quality clusters which should have
  - High intra-class similarity: Cohesive within clusters
  - Low inter-class similarity: Distinctive between clusters
- Similarity measure is critical for cluster analysis
- There is usually a separate quality function that measures the goodness of a cluster
- There exist many similarity measures and quality functions for different applications
- It is hard to define similar enough or good enough
  - The answer is typically highly subjective

Considerations for Cluster Analysis

- Partitioning criteria
  - Single level vs. hierarchical partitioning (often, multi-level hierarchical partitioning is desirable, e.g., grouping topical terms)
- Separation of clusters
  - Exclusive (e.g., one customer belongs to only one region) vs. nonexclusive (e.g., one document may belong to more than one class)
- Similarity measure
  - Distance-based (e.g., Euclidean, 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

- Quality
  - Ability to deal with different types of attributes: Numerical, categorical, text, multimedia, networks, and mixture of multiple types
  - Discovery of clusters with arbitrary shape
  - Ability to deal with noisy data
- Scalability
  - Clustering all the data instead of only on samples
  - High dimensionality
  - Incremental or stream clustering and insensitivity to input order
- Constraint-based clustering
  - User-given preferences or constraints; domain knowledge; user queries
- Interpretability and usability
  - The final generated clusters should be semantically meaningful and useful

Basic Steps in Cluster Analysis

- Feature selection
  - Select information concerning the task of interest
  - Minimal information redundancy
- Proximity measure
  - Similarity of two feature vectors
- Clustering criterion
  - Expressed via a cost function or some rules
- Clustering algorithms
  - Choice of algorithms
- Validation of the results
  - Validation test (also, clustering tendency test)
- Interpretation of the results
- Integration with applications

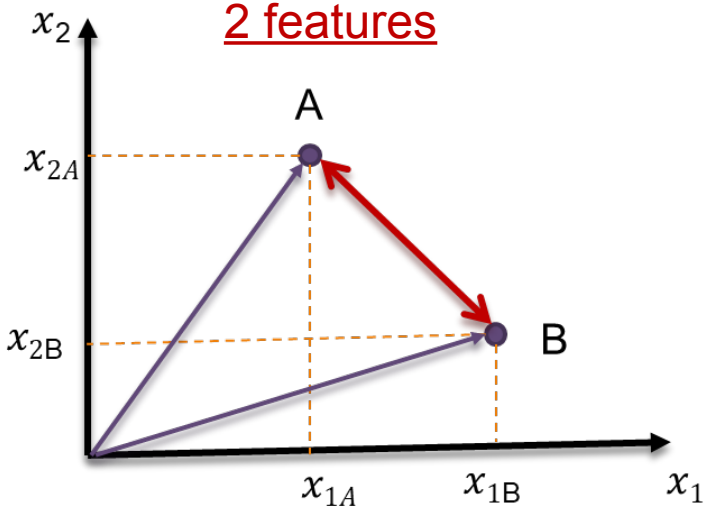
Dissimilarity/Similarity Metric

- Similarity is expressed in terms of a distance function, $d(A, B)$.
- 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.

Similarity and Distance

- We need a method for measuring similarity.
  - What does it mean that two customers are similar?
- Objects have multiple attributes and there is no single best method for reducing them to a single similarity measure.
- The most popular measure is Euclidian Distance (L2 norm):
  - We can represent each object as a feature vector.
  - The closer two objects are in the space defined by the features, to more similar they are.

2 features



$$d(\mathrm{A},\mathrm{B}) = \sqrt{(x_{1A} - x_{1B})^2 + (x_{2A} - x_{2B})^2}$$

n features

$$d(\mathrm{A},\mathrm{B}) = \sqrt{(x_{1A} - x_{1B})^2 + \ldots + (x_{nA} - x_{nB})^2}$$

Other Distance Functions

- Euclidian Distance (L2 norm): Sum of squares of pairwise distances.
  - $d(A,B) = \|A - B\|_2 = \sqrt{(x_{1A} - x_{1B})^2 + \ldots + (x_{nA} - x_{nB})^2}$
- Manhattan Distance (L1 norm): Sum of pairwise distances.
  - $d(A,B) = \|A - B\|_1 = \left|x_{1A} - x_{1B}\right| + \ldots + |x_{nA} - x_{nB}|$
- Cosine Distance: Particularly useful when you want to ignore differences in scale across instances
  - $d(A,B) = 1 - \frac{A \bullet B}{\|A\|_2 \bullet \|B\|_2}$
- Jaccard Distance: Treats two objects as sets of characteristics.
  - $d(A,B) = 1 - \frac{|A \cap B|}{|A \cup B|}$

Distance:

Mixed Data Types

- Most datasets contain mixed data (i.e., interval, ratio, ordinal, nominal data).
- For mixed data Gower Distance can be used.
- Gower distance can be computed as the average of partial dissimilarities across individuals.
  - Each partial dissimilarity ranges in [0 1]

$$S_{ij} = \frac{\sum_k \left(W_{ijk} \bullet S_{ijk} \right)}{\sum_k W_{ijk}}$$

  - Where, S_{ijk} is the contribution provided by the kth variable and $W_{ijk} = 1$ if the kth variable is valid, 0 otherwise.
    - For ordinal and continuous variables $S_{ijk} = \frac{\left| x_{ik} - x_{jk} \right|}{r_k}$ where r_k is the range of values for the kth variable.
    - For nominal variables, $S_{ijk} = 0$ if $x_{ij} = x_{ik}$, $S_{ijk} = 1$ otherwise.
    - For binary variables $S_{ijk} = 0$ if both are positive and $W_{ijk} = 1$ if at least one of them is positive. They are 0 otherwise.

Major Clustering Approaches - I

- 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

Major Clustering Approaches - II

- 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 Algorithms: Basic Concepts

- Partitioning method: Discovering the groupings in the data by optimizing a specific objective function and iteratively improving the quality of partitions
- K-partitioning method: Partitioning a dataset **D** of **n** objects into a set of **K** clusters so that an objective function is optimized (e.g., the sum of squared distances to c_i is minimized, where c_i is the centroid or medoid of cluster C_i)
- A typical objective function: Sum of Squared Errors (SSE)

$$SSE = \sum_{i=1}^{k} \sum_{p \in C_i} (d(p, c_i))^2$$

- Problem definition: Given **K**, find a partition of **K** clusters that optimizes the chosen partitioning criterion
  - Global optimal: Needs to exhaustively enumerate all partitions
  - Heuristic methods (i.e., greedy algorithms): K-Means, K-Medians, K-Medoids, etc.

The K-Means Clustering Method

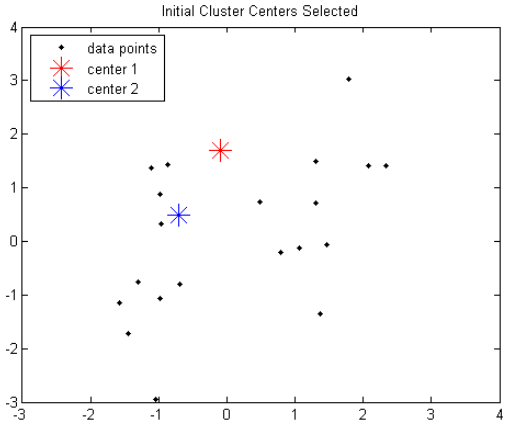
- K-Means (MacQueen'67, Lloyd'57/'82): Each cluster is represented by the center of the cluster
- Given **K**, the number of clusters, the K-Means clustering algorithm is outlined as follows:

1. Select **K** points as initial centroids
2. **Repeat**
   a) Form **K** clusters by assigning each point to its closest centroid
   b) Re-compute the centroids (i.e., mean point) of each cluster

   Until convergence criterion is satisfied

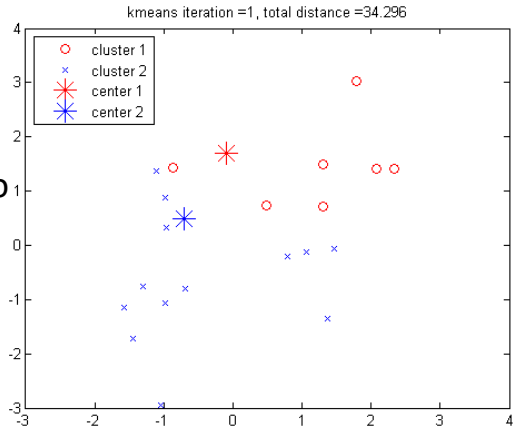
- Different kinds of measures can be used
  - Manhattan distance (L1 norm), Euclidean distance (L2 norm), Cosine similarity

Example: K-Means Clustering

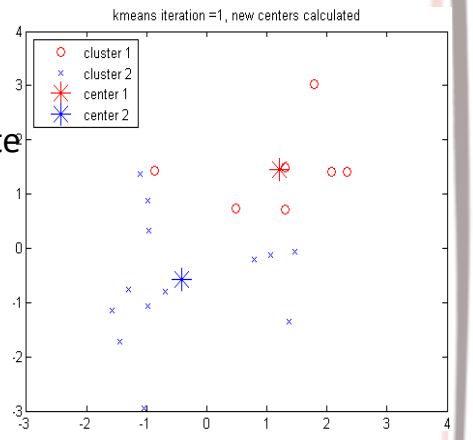


The original data points & randomly select $K = 2$ centroids

Assign points to clusters

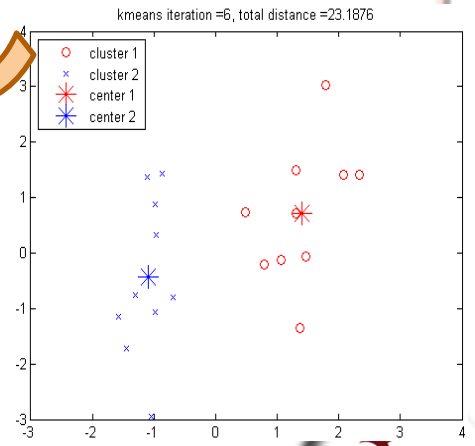


Recompute cluster centers



Redo point assignment

Recompute cluster centers and redo point assignment as necessary



Execution of the K-Means Clustering Algorithm

Select K points as initial centroids

Repeat

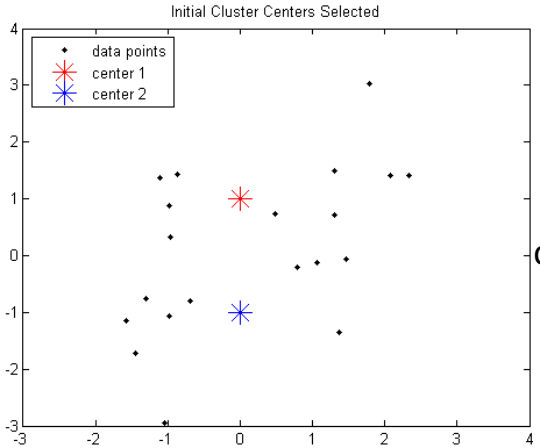
- Form K clusters by assigning each point to its closest centroid
- Re-compute the centroids (i.e., *mean point*) of each cluster

Until no change in cluster assignments

K-Means Method

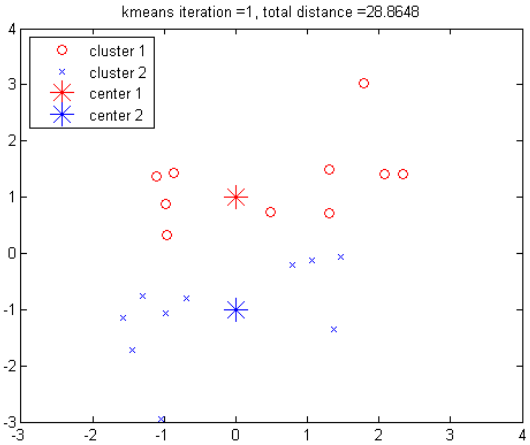
- Efficiency: $O(tKn)$ where n: # of objects, K: # of clusters, and t: # of iterations
  - Normally, K, $t << n$; thus, an efficient method
- K-means clustering often *terminates at a local optimal*
  - Initialization can be important to find high-quality clusters
- Need to specify *K,* the *number* of clusters, in advance
  - There are ways to automatically determine the "*best*" *K*
  - In practice, we often run a range of values and select the "*best*" *K* value
- Sensitive to noisy data and *outliers*
  - Variations: Using K-medians, K-medoids, etc.
- K-means is applicable only to objects in a continuous n-dimensional space
  - It is possible the K-modes for *categorical data*
- *It tends to produce more or less round clusters*
  - *It gives the same weight to all features.*
  - *Therefore, standardizing features before applying K-means is important.*
  - *Unequal variances of the features also lead to putting more weight on features with smaller variances.*

Poor Initialization May Lead to Poor Clustering

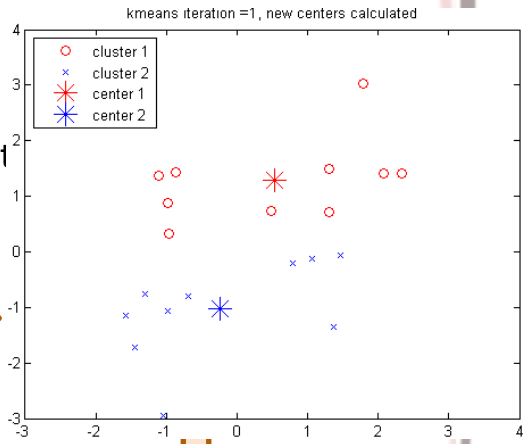


Another random selection of k centroids for the same data points

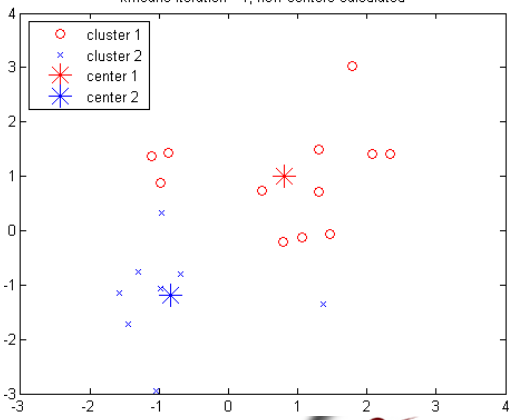
Assign points to clusters



Recompute cluster centers

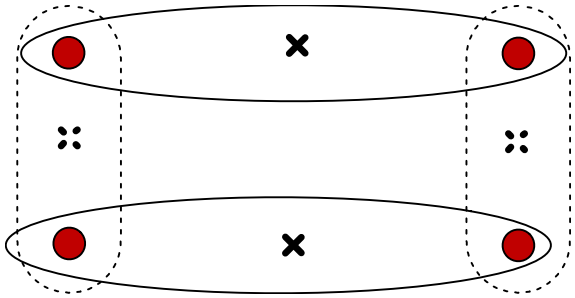


Redo point assignment



- Rerun of the K-Means using another random K seeds
- This run of K-Means generates a poor quality clustering

Initialization of K-Means

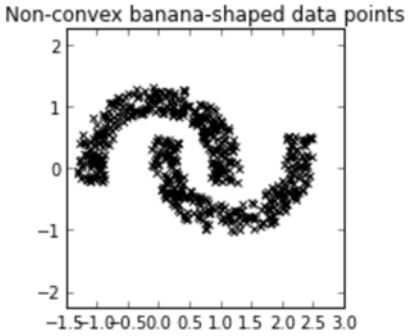


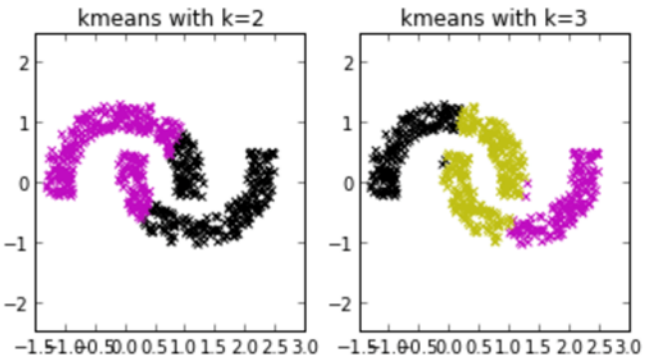
- Different initializations may generate rather different clustering results (some could be far from optimal)
- Original proposal (MacQueen'67): Select K seeds randomly
- Need to run the algorithm multiple times using different seeds

- There are many methods proposed for better initialization of *k* seeds
  - *K-Means++* (Arthur & Vassilvitskii'07):
    - The first centroid is selected at random
    - The next centroid selected is the one that is farthest from the currently selected (selection is based on a weighted probability score)
    - The selection continues until *K* centroids are obtained

Non-convex Shapes

- K-Means is not suitable to discover clusters with *non-convex shapes*
  - For example, it performs poorly on the banana shapes
  - Instead, we can use density-based clustering, kernel K-means, etc.





k-means performs poorly on the banana shapes

Variations of K-Means

- There are many variants of the K-Means method, varying in different aspects
  - Choosing better initial centroid estimates
    - K-means++, Intelligent K-Means, Genetic K-Means
  - Choosing different representative prototypes for the clusters
    - K-Medoids, K-Medians, K-Modes
  - Applying feature transformation techniques
    - Weighted K-Means, Kernel K-Means

K-Means: A Question

- In which of the following cases will K-Means clustering fail to give good results?
  a) Data points with round shapes nested within each other
  b) Data points with non-convex shapes
  c) Data points with outliers
  d) Data points with different densities

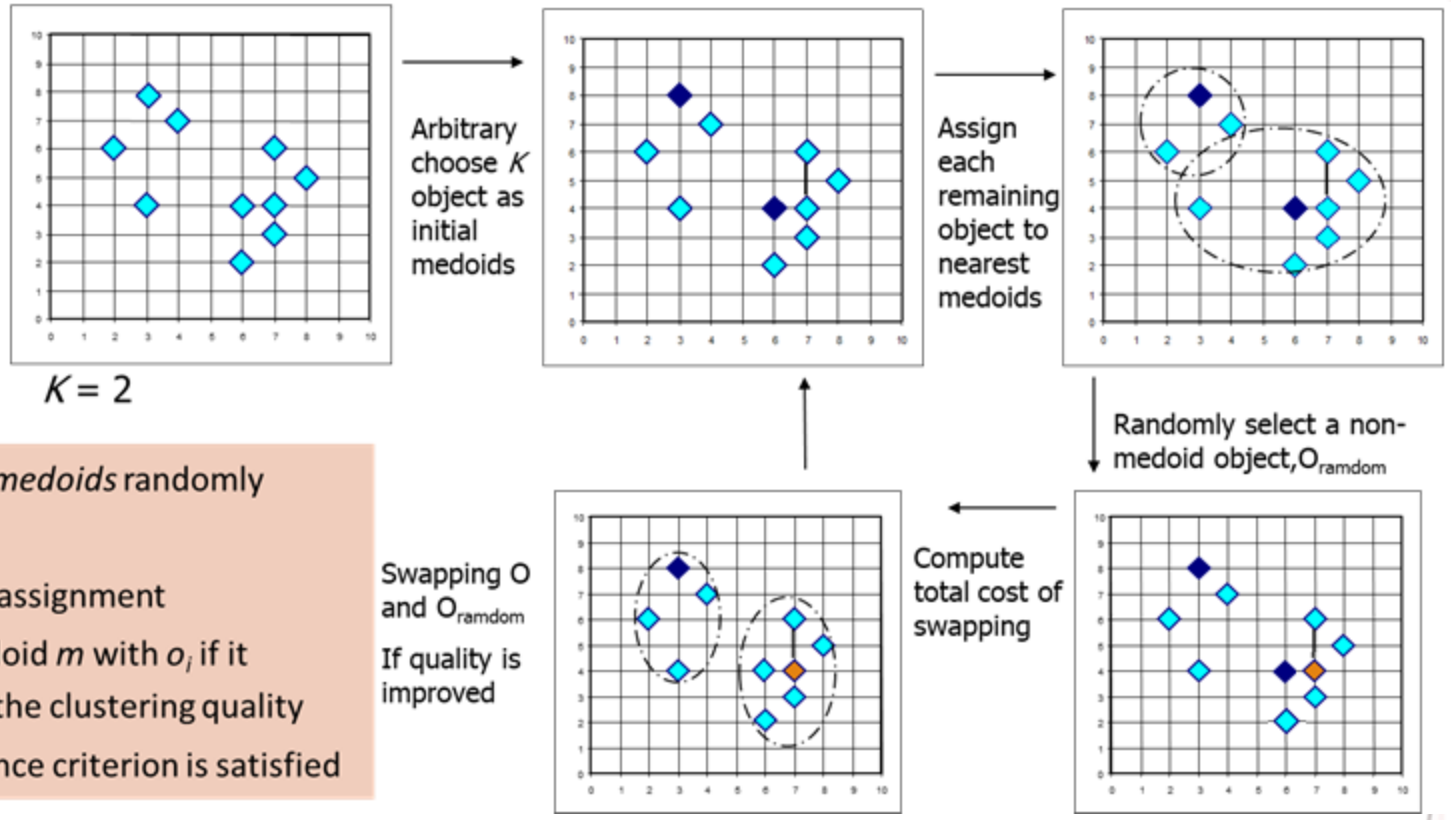
Handling Outliers:

From K-Means to K-Medoids

- 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
- The *K-Medoids* clustering algorithm:
    1. Select *K* points as the initial representative objects (i.e., as initial *K medoids*)
    2. **Repeat**
        a. Assigning each point to the cluster with the closest medoid
        b. Randomly select a non-representative object o_i
        c. Compute the total cost *S* of swapping the medoid *m* with o_i
        d. If $S < 0$, then swap *m* with o_i to form the new set of medoids

    Until convergence criterion is satisfied

PAM: A Typical K-Medoids Algorithm



Select initial *K medoids* randomly

Repeat

Object re-assignment

Swap medoid *m* with o_i if it improves the clustering quality

Until convergence criterion is satisfied

K-Medoids Clustering

- *K-Medoids* Clustering: Find *representative* objects (medoids) in clusters
- *PAM* (Partitioning Around Medoids: Kaufmann & Rousseeuw 1987)
  - 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 sum of the squared errors (SSE) of the resulting clustering
  - *PAM* works effectively for small data sets but does not scale well for large data sets (due to the computational complexity)
  - Computational complexity: PAM: $O(K(n - K)^2)$ (quite expensive!)
- Efficiency improvements on PAM
  - *CLARA* (Kaufmann & Rousseeuw, 1990):
    - PAM on samples; $O(Ks^2 + K(n - K))$, s is the sample size
  - *CLARANS* (Ng & Han, 1994): Randomized re-sampling, ensuring efficiency + quality

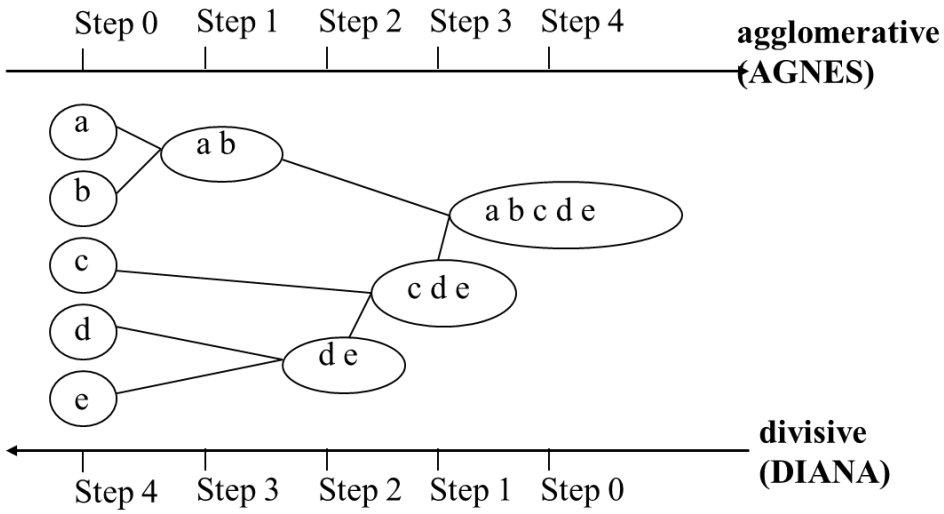
K-Medians:
Handling Outliers by Computing Medians

- Medians are less sensitive to outliers than means
  - Think of the median salary vs. mean salary of a large firm when adding a few top executives!
- K-Medians: Instead of taking the mean value of the object in a cluster as a reference point, medians are used.
- L1-norm is used as the distance measure.
- The criterion function for the K-Medians algorithm: $S = \sum_{k=1}^{K} \sum_{x_i \in C_k} \left\| x_i - c_k \right\|_1$
- The *K-Medians* clustering algorithm:
  - Select *K* points as the initial representative objects (i.e., as initial *K medians*)
  - **Repeat**
    - Assign every point to its nearest median
    - Re-compute the median using the median of each individual feature
  - **Until** convergence criterion is satisfied

Hierarchical Clustering:

Basic Concepts

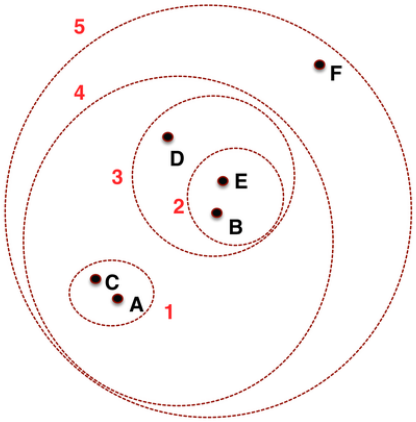
- Hierarchical clustering
  - Generate a clustering hierarchy (drawn as a dendrogram)
  - Not required to specify K, the number of clusters
  - More deterministic
  - No iterative refinement

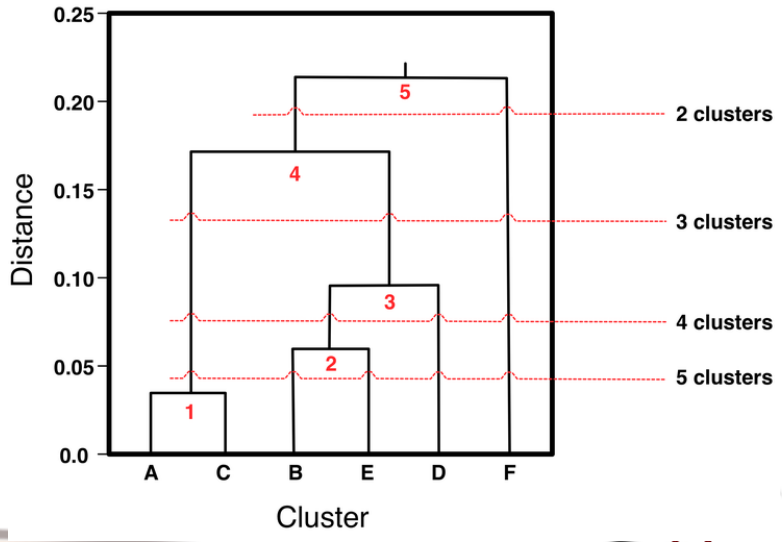


- Two categories of algorithms:
  - Agglomerative: Start with singleton clusters, continuously merge two clusters at a time to build a bottom-up hierarchy of clusters
  - Divisive: Start with a huge macro-cluster, split it continuously into two groups, generating a top-down hierarchy of clusters

Dendogram:
How Clusters are Merged

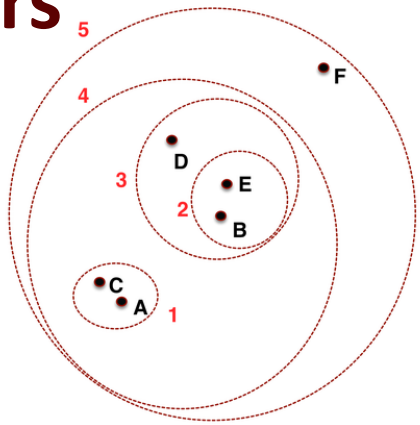
- Dendrogram: Decompose a set of data objects into a tree of clusters by multi-level nested partitioning
  - x-axis: individual objects
  - y-axis: distance between the clusters (linkage function)
- A clustering of the data objects is obtained by cutting the dendrogram at the desired level, then each connected component forms a cluster

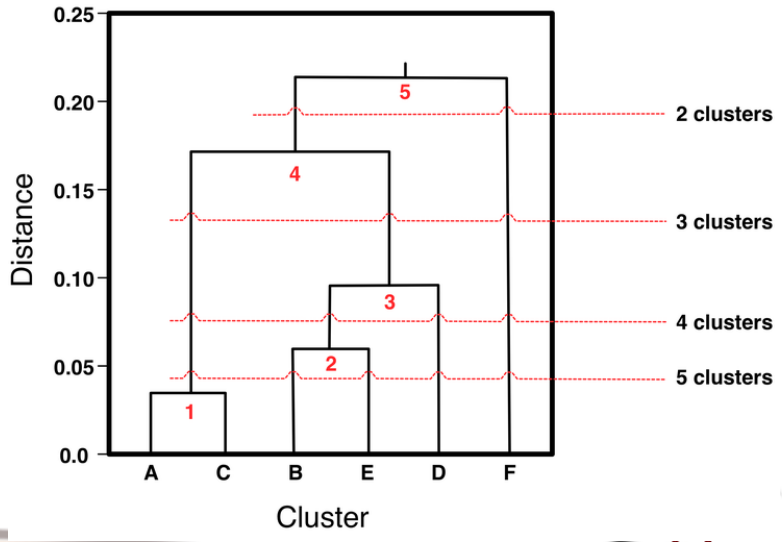




Dendogram:
Determining Number of Clusters

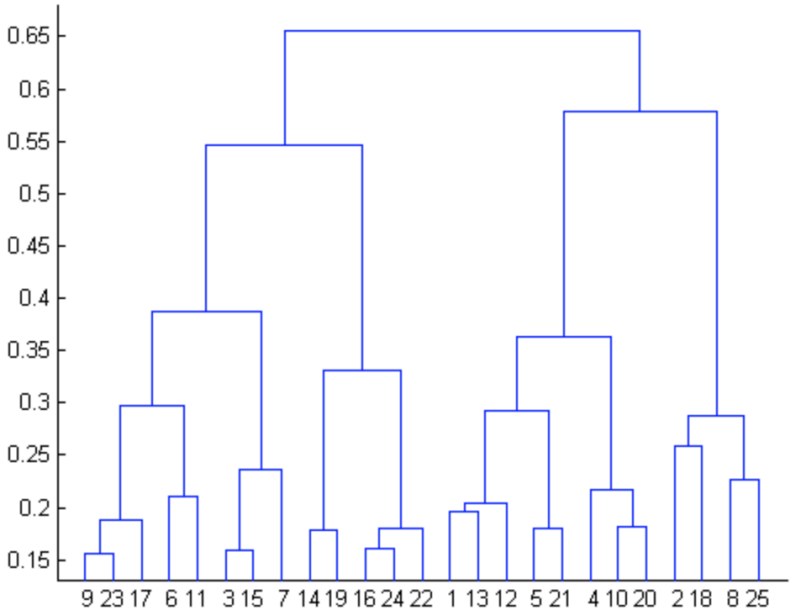
- Hierarchical clustering allows the analyst to see the groupings before deciding on the number of clusters.
- The dendogram can give an idea of where natural clusters may occur.
  - Ex. (right): 3 clusters may be a good choice as there is a relatively long distance between clusters 3 and 4.





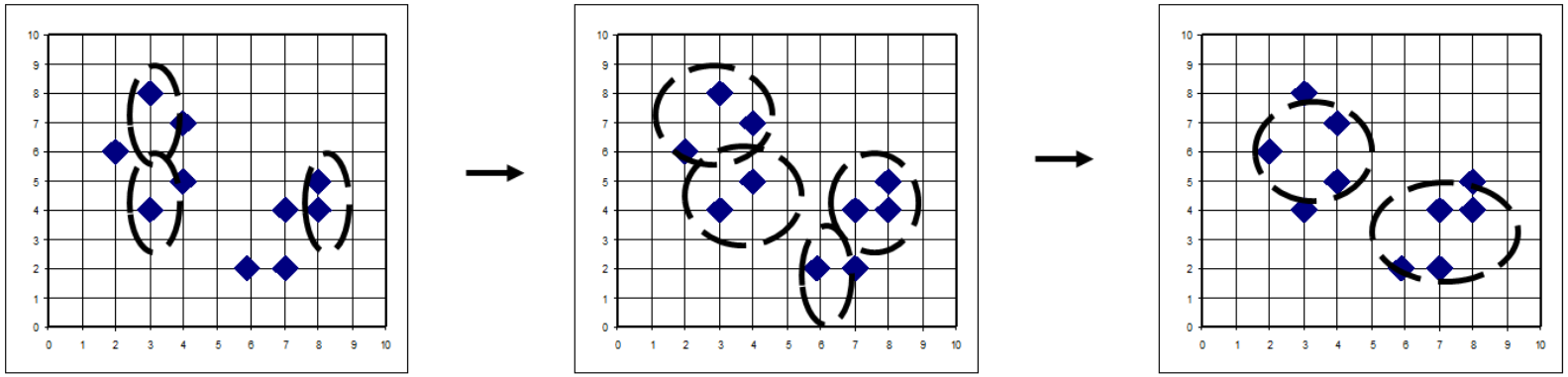
Example: Number of Clusters

- What is the most appropriate no. of clusters for the data points represented by the following dendrogram?



Agglomerative Clustering Algorithm

- AGNES (AGglomerative NESting) (Kaufmann and Rousseeuw, 1990)
  - Use the **single-link** method and the dissimilarity matrix
  - Continuously merge nodes that have the least dissimilarity
  - Eventually all nodes belong to the same cluster



- Agglomerative clustering varies on different similarity measures among clusters
  - Single Link (nearest neighbor), Complete Link (diameter), Average Link (group average), Centroid Link (centroid similarity).

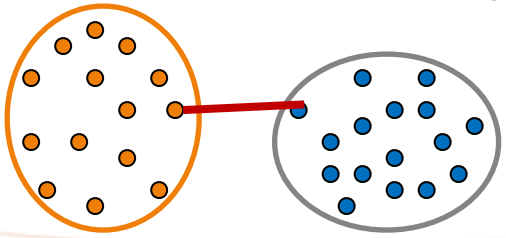
Distance Between Clusters:

Single Link vs. Complete Link

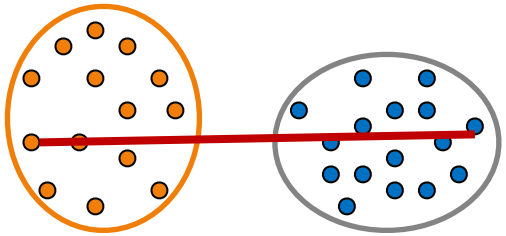
- Single link (nearest neighbor)
  - The similarity between two clusters is the similarity between their most similar (nearest neighbor) members
  - Local similarity-based: Emphasizing more on close regions, ignoring the overall structure of the cluster
  - Capable of clustering non-elliptical shaped group of objects
  - Sensitive to noise and outliers

- Complete link (diameter)
  - The similarity between two clusters is the similarity between their most dissimilar members
  - Merge two clusters to form one with the smallest diameter
  - Nonlocal in behavior, obtaining compact shaped clusters
  - Sensitive to outliers

Minimum distance: $dist_{min}(C_i, C_j) = \min_{p \in C_i, p' \in C_j} \{| p - p' |\}$



Maximum distance: $dist_{max}(C_i, C_j) = \max_{p \in C_i, p' \in C_j} \{| p - p' |\}$

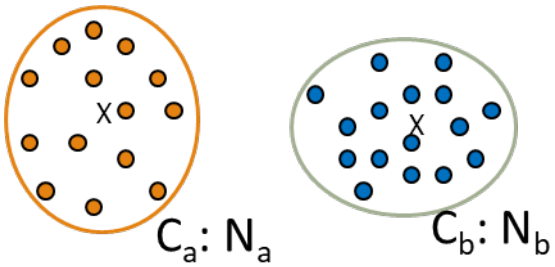


Distance Between Clusters:

Average Link vs. Centroid Link

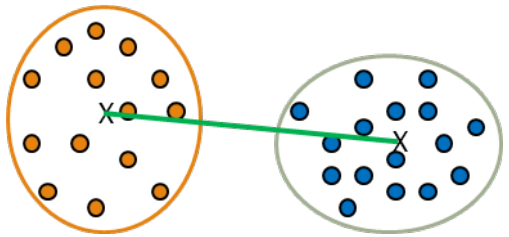
- Average link
  - The average distance between an element in one cluster and an element in the other (i.e., all pairs in two clusters)
  - Expensive to compute

$$\textbf{Average distance}: \quad dist_{avg}(C_i, C_j) = \frac{1}{n_i n_j} \sum_{p \in C_i, p' \in C_j} |p - p'|$$



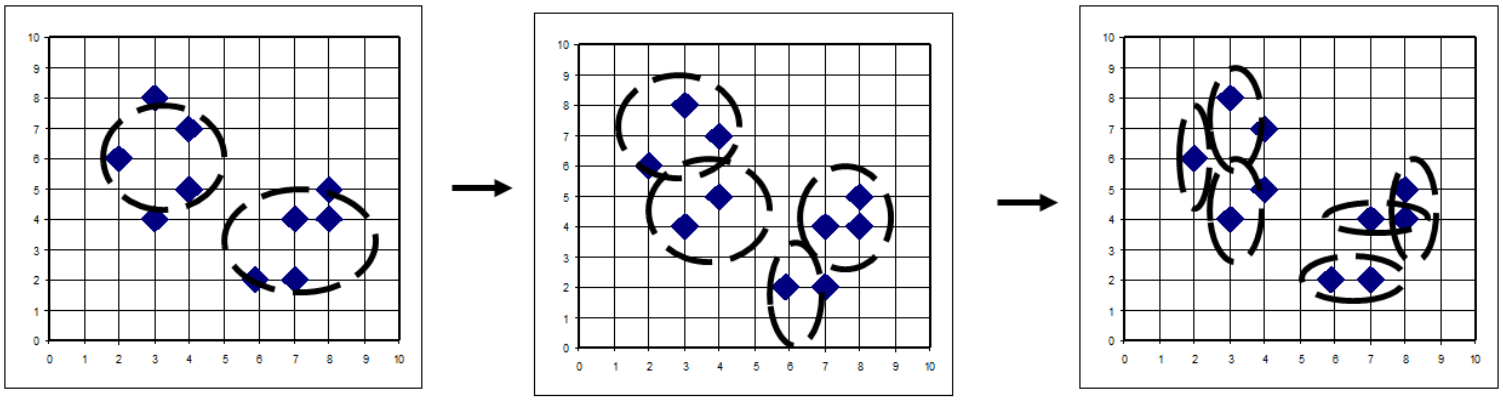
- Centroid Link
  - The distance between the centroids of two clusters

$$\textbf{Mean distance}: \quad dist_{mean}(C_i, C_j) = |m_i - m_j|$$



Divisive Clustering

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



- Divisive clustering is a top-down approach
  - The process starts at the root with all the points as one cluster
  - It recursively splits the higher level clusters to build the dendrogram
  - Can be considered as a global approach
  - More efficient when compared with agglomerative clustering

Divisive Clustering Algorithm

- Choosing which cluster to split
  - Check the sums of squared errors of the clusters and choose the one with the largest value
- Splitting criterion: Determining how to split
  - Look for greater reduction in the difference in the SSE criterion as a result of a split
  - For categorical data, Gini-index can be used
- Handling the noise
  - Use a threshold to determine the termination criterion (do not generate clusters that are too small because they contain mainly noises)

Extensions to Hierarchical Clustering

- Major weaknesses of hierarchical clustering methods
  - Can never undo what was done previously
  - Do not scale well
    - Time complexity of at least $O(n^2)$, where n is the number of total objects
- Other hierarchical clustering algorithms
  - BIRCH (1996): Use CF-tree (clustering feature tree) and incrementally adjust the quality of sub-clusters
  - CURE (1998): Represent a cluster using a set of well-scattered representative points
  - CHAMELEON (1999): Use graph partitioning methods on the K-nearest neighbor graph of the data

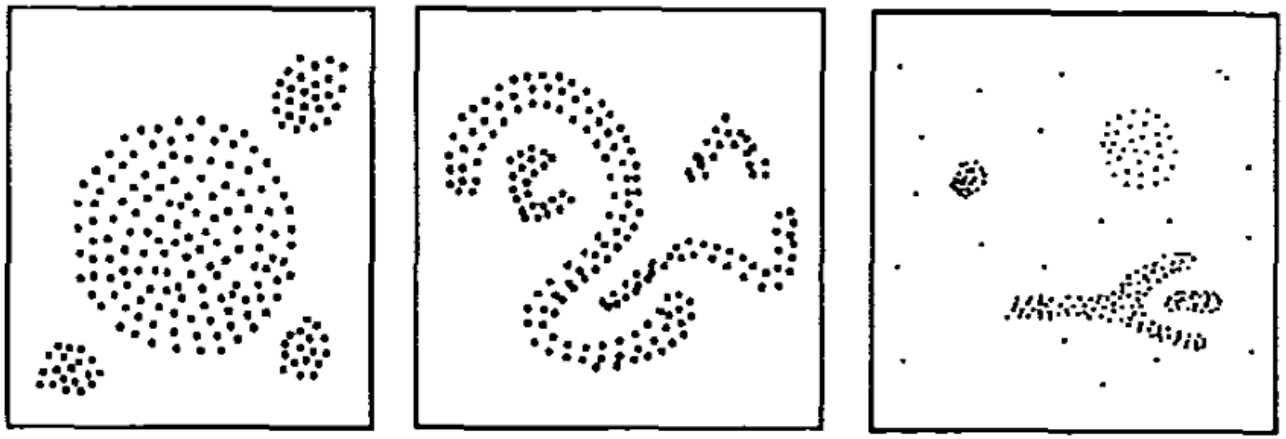
Density-Based Clustering Methods

- Clustering based on density (a local cluster criterion), such as density-connected points
- Major features:
  - Discover clusters of arbitrary shape
  - Handle noise
  - One scan (only examine the local region to justify density)
  - 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) (also, grid-based)

DBSCAN:

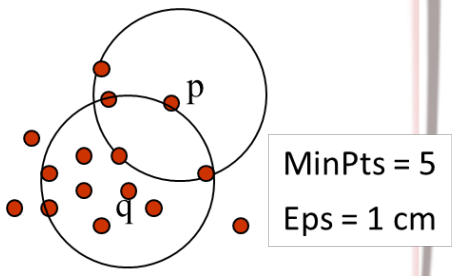
A Density-Based Spatial Clustering Algorithm

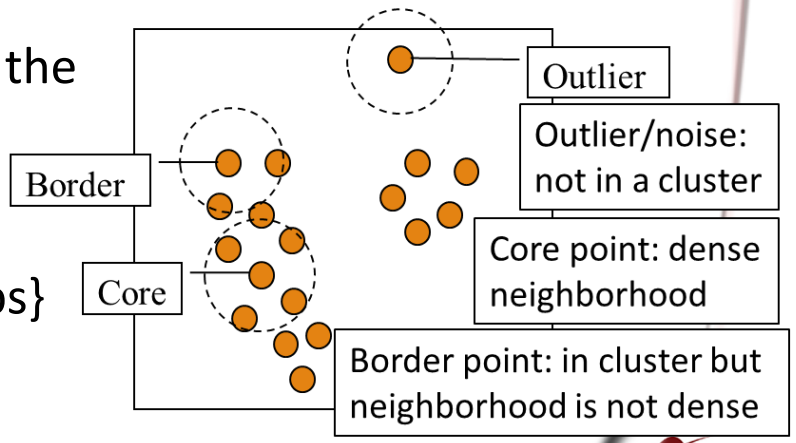
- Relies on a density-based notion of cluster: A cluster is defined as a maximal set of density-connected points
- Discovers clusters of arbitrary shape in spatial databases with noise



DBSCAN Basics

- DBSCAN: Density-Based Spatial Clustering of Applications with Noise
- A *density-based* notion of cluster
  - A *cluster* is defined as a maximal set of density-connected points
- Two parameters*:*
  - *Eps* (*ε*): Maximum radius of the neighborhood
  - *MinPts*: Minimum number of points in the Eps-neighborhood of a point
- The Eps(*ε*)-neighborhood of a point *q*:
  - $N_{Eps}(q)$: {p belongs to D | dist(p, q) ≤ Eps}

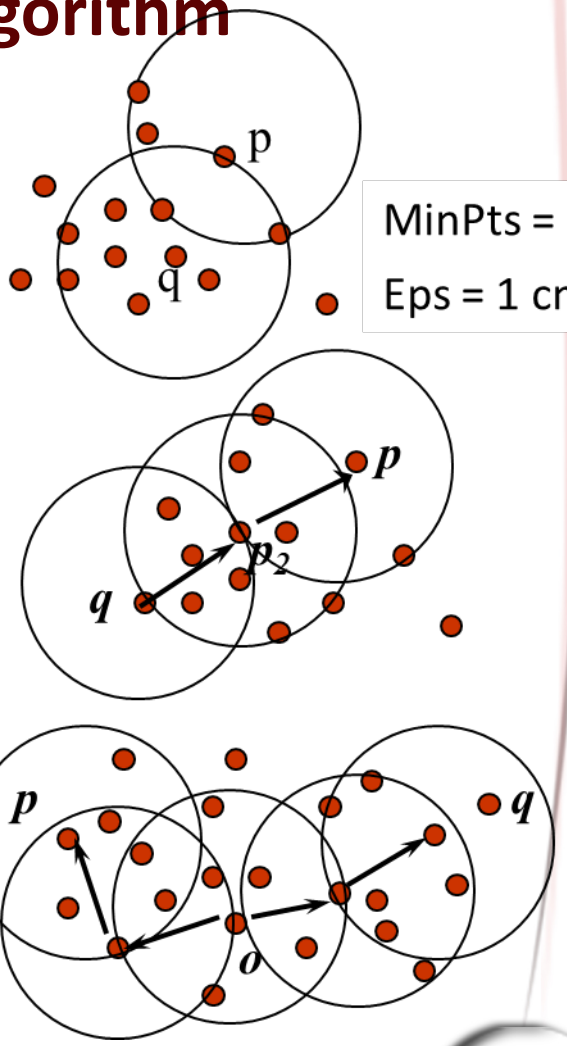




DBSCAN:

A Density-Based Spatial Clustering Algorithm

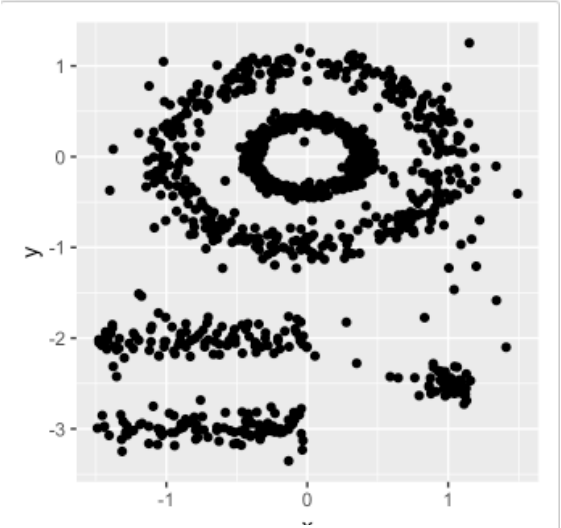
- **Directly density-reachable**:
  - A point *p* is directly density-reachable from a point *q* w.r.t. *Eps* (ε), *MinPts* if
    - *p* belongs to $N_{Eps}(q)$
    - **core point** condition: $|N_{Eps}(q)| \geq MinPts$
- **Density-reachable**:
  - A point *p* is density-reachable from a point *q* w.r.t. *Eps*, *MinPts* if there is a chain of points $p_1, \ldots, p_n, p_1 = q, p_n = p$ 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*



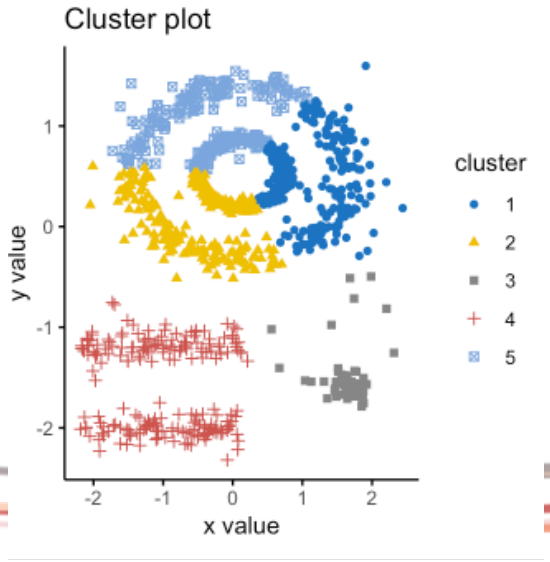
DBSCAN: The Algorithm

- Algorithm:
  - Arbitrarily select a point *p*
  - Retrieve all points density-reachable from *p* w.r.t. 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
- Computational complexity
  - If a spatial index is used, the computational complexity of DBSCAN is O(nlogn), where n is the number of database objects
  - Otherwise, the complexity is $O(n^2)$

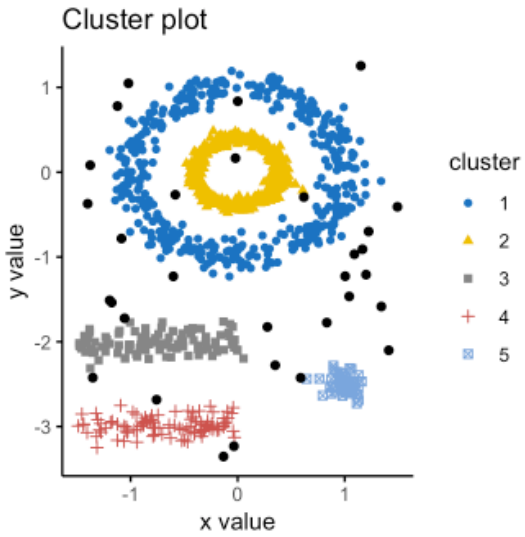
A Comparison: DBSCAN vs. K-Means



The plot contains 5 clusters and outliers, including:
- 2 oval clusters
- 2 linear clusters
- 1 compact cluster

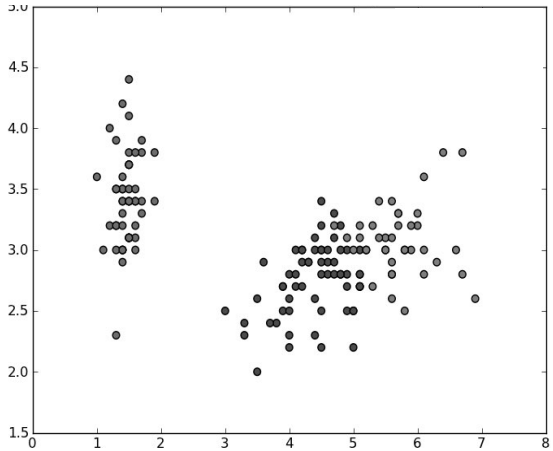


K-means

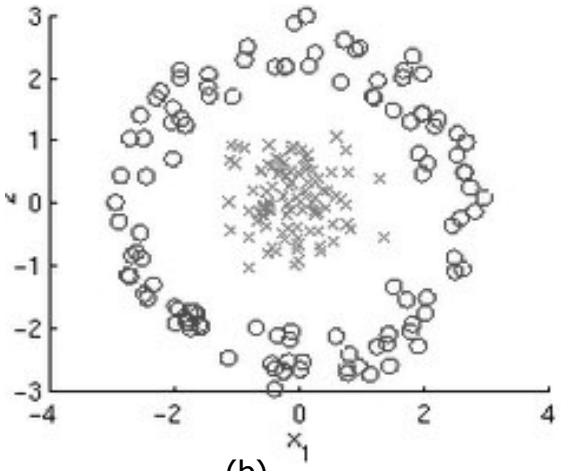


DB-scan

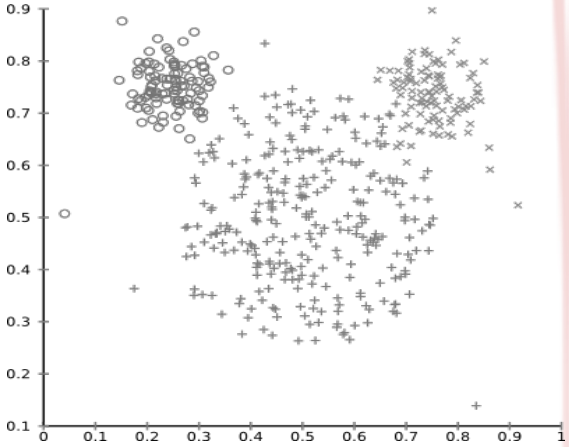
Example:



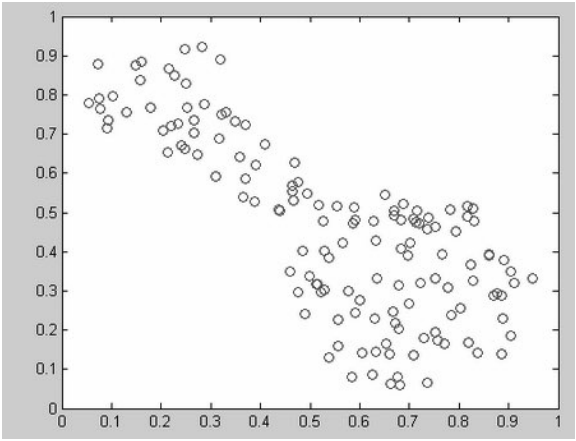
(a)



(b)



(c)



(d)

Figures show four data sets. The Figure (b) and (c) have class variables whereas Figure (a) and (b) have no class variables. If you would like to choose a clustering algorithm among K-Means and DBScan, which one/s would be the most practical and appropriate to use for the data sets? Why?

DBSCAN:
Sensitivity to Parameters

Figure 8. DBScan results for DS1 with MinPts at 4 and Eps at (a) 0.5 and (b) 0.4.

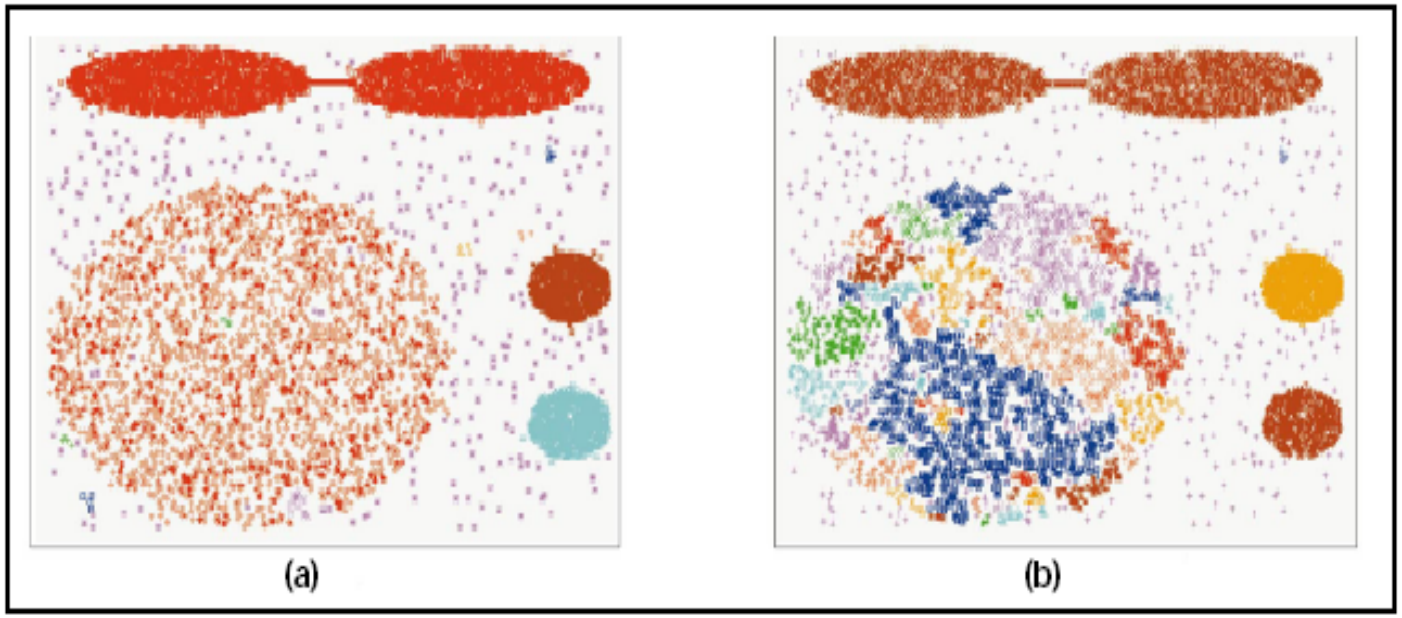
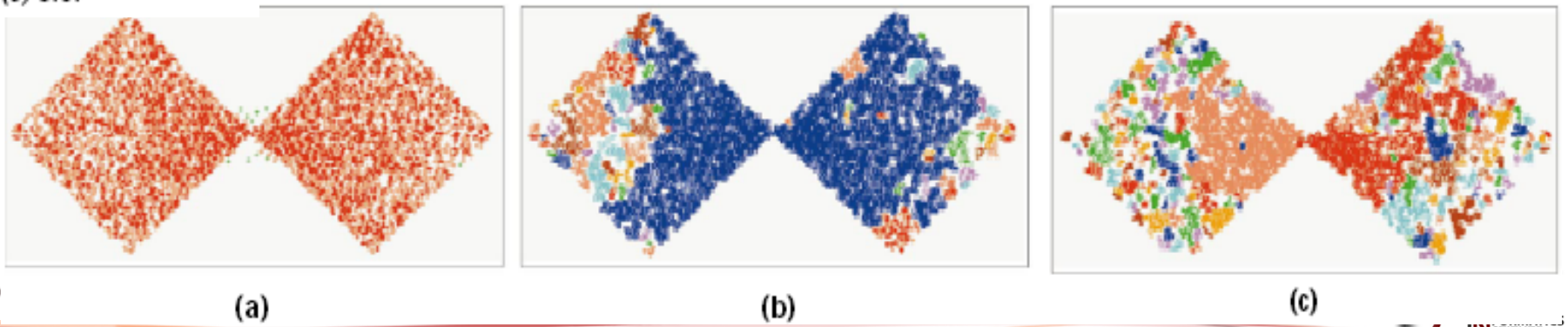
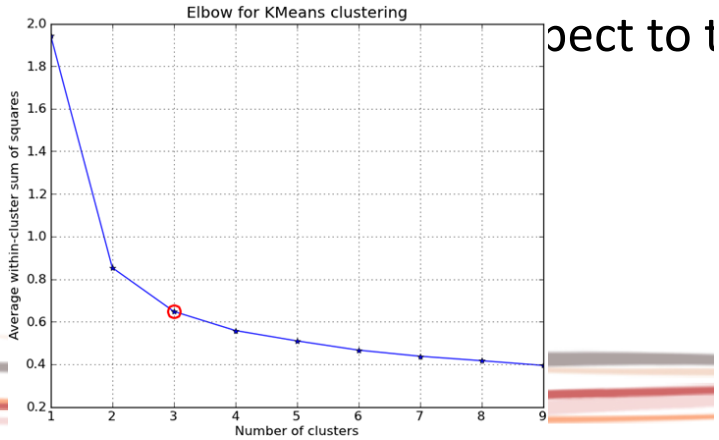


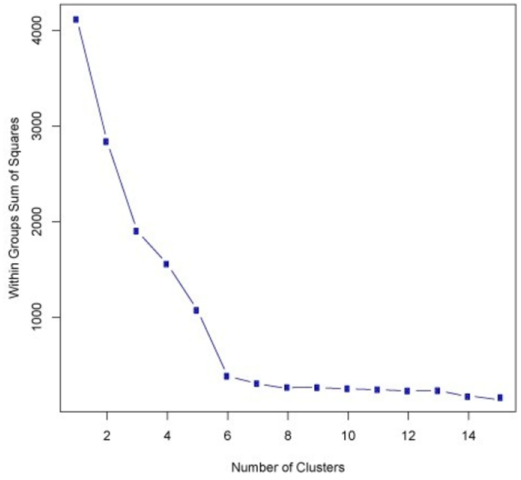
Figure 9. DBScan results for DS2 with MinPts at 4 and Eps at (a) 5.0, (b) 3.5, and (c) 3.0.



Determining the Number of Clusters

- Determining the number of clusters, K, is not easy.
  - It depends on the distribution's shape and scale as well as the resolution required by the user.
- There are many possible ways to estimate K:
  - Rule of thumb: $\mathrm{K} \approx sqrt\left(\frac{n}{2}\right)$ for a dataset of n points
    - e.g., n=200 -> K=10
  - Elbow method: Use the turning point in the curve of the sum of within





What is the best choice for K?

Determining the Number of Clusters

- Cross validation method:
  - Divide a given data set into *m* parts
  - Use *m* – 1 parts to obtain a clustering model
  - Use the remaining part to test the quality of the clustering
    - For example, for each point in the test set, find the closest centroid, and use the sum of squared distance between all points in the test set and the closest centroids to measure how well the model fits the test set
  - For any $K > 0$, repeat it *m* times, compare the overall quality measure w.r.t. different *K's*, and find # of clusters that fits the data the best

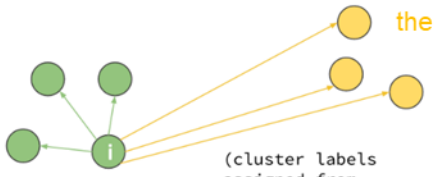
Determining the Number of Clusters

- Silhouette Analysis: Silhouette analysis can be used to study the separation distance between the resulting clusters.
  - Silhouette plot displays a measure (called silhouette coefficient) of how close each point in one cluster is to points in the neighboring clusters.
  - Thus, it provides a way to assess parameters like number of clusters visually.
  - Silhouette coefficient has a range of [-1, 1].
    - A value near +1 indicate that the sample is far away from the neighboring clusters.
    - A value of 0 indicates that the sample is on or very close to the decision boundary between two neighboring clusters.
    - Negative values indicate that those samples might have been assigned to the wrong cluster.

Silhouette Analysis

$$b(i) = \min_{k \neq i} \frac{\sum_{j \in C_k} d(i,j)}{|C_k|}$$

b(i): average distance between i and the nodes in the nearest cluster



(cluster labels assigned from ground-truth classes)

For a single point, i

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

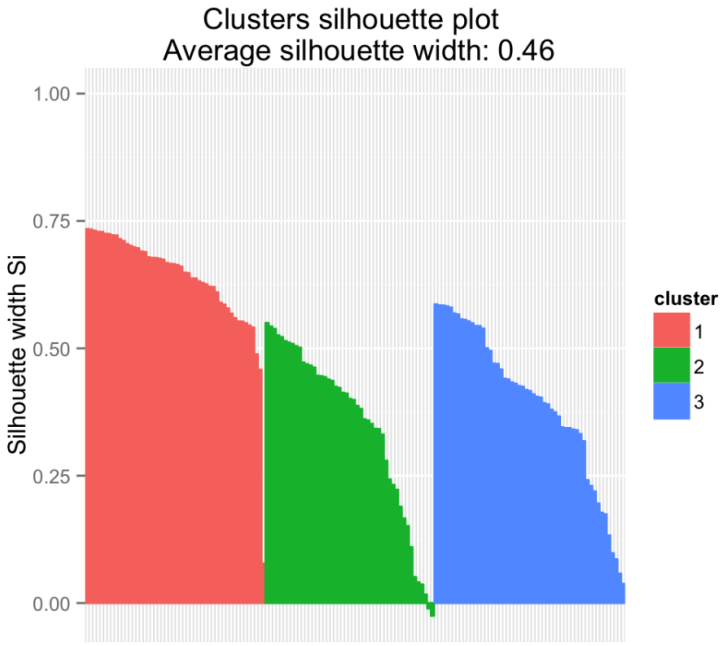
a(i): average distance between i and all of the other points in **its own cluster**

$$a(i) = \frac{\sum_{j \in C_i, i \neq j} d(i,j)}{|C_i| - 1}$$

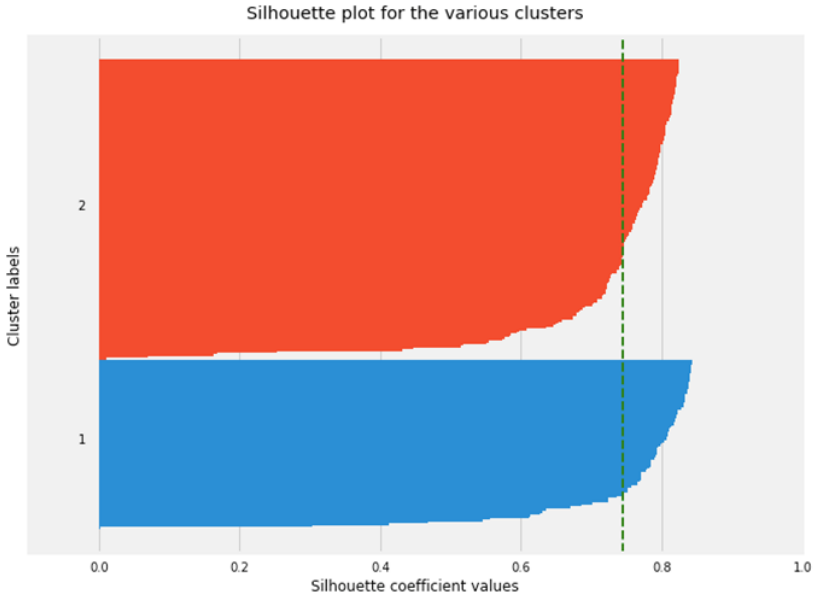
$$-1 \leq s(i) \leq 1$$

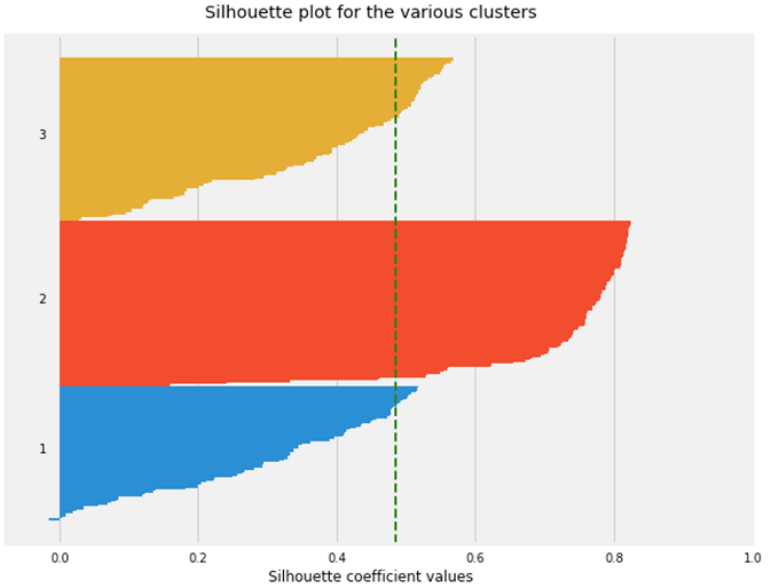
Sprawling, overlapped clusters

Tight, well-separated clusters

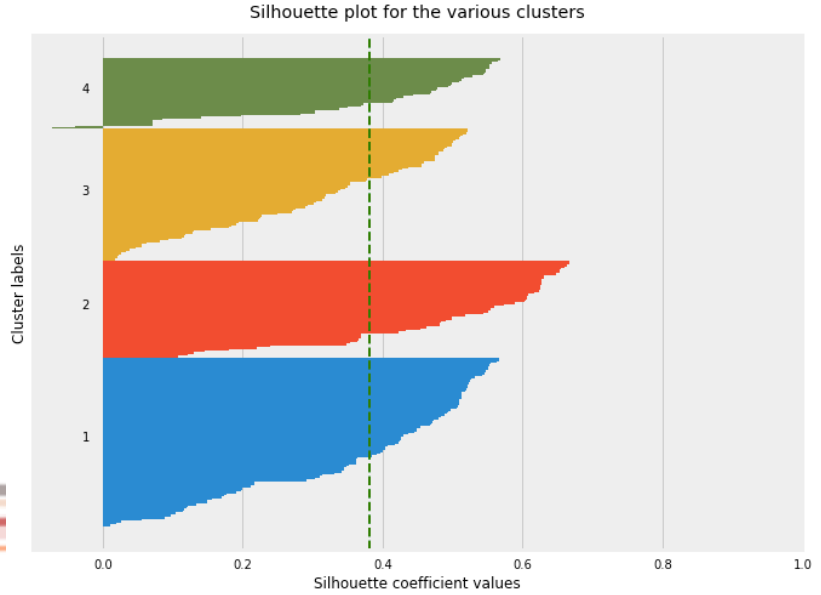


Silhouette Analysis: Example #1

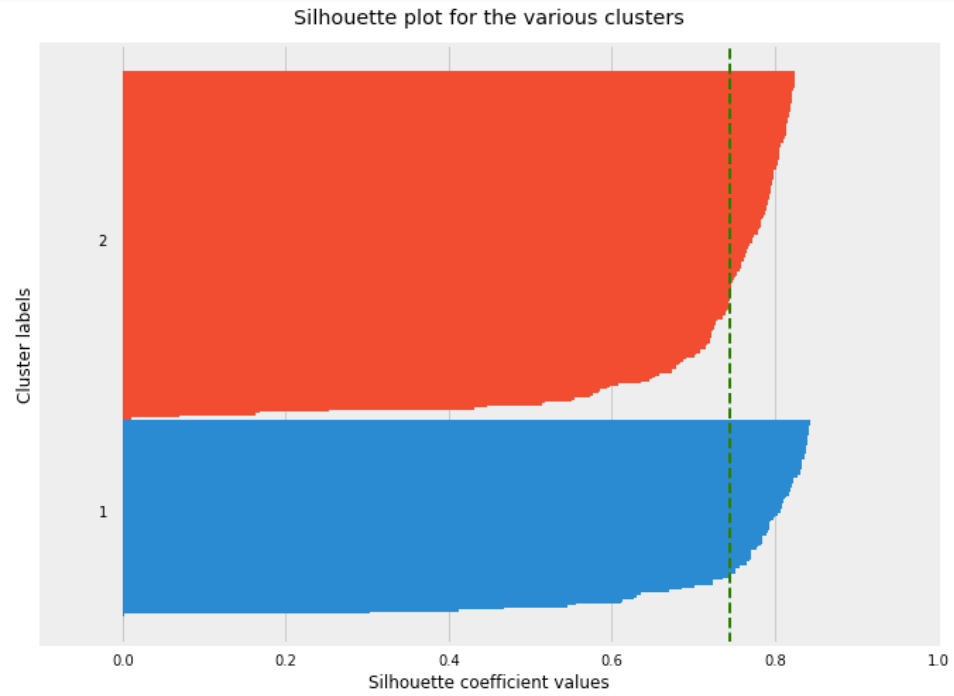


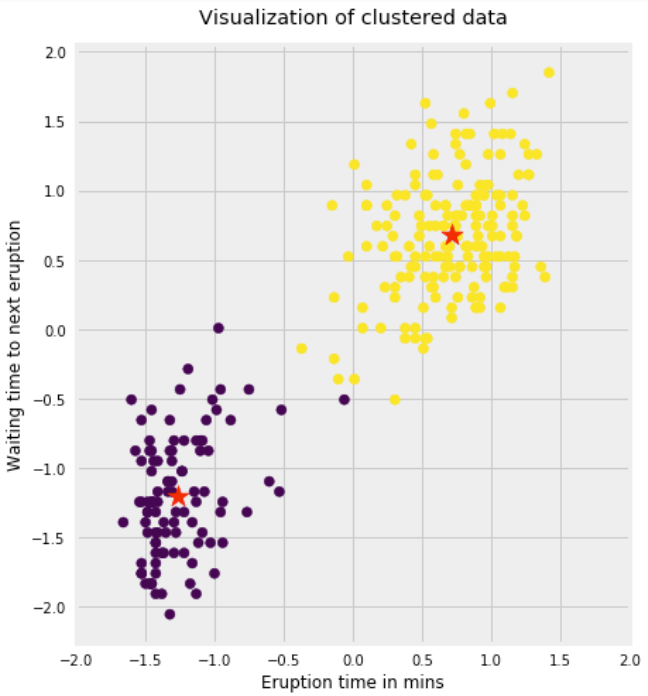


- Silhouette analysis results of K-Means for K=2, K=3 and K=4.
- Which K you would prefer? Why?

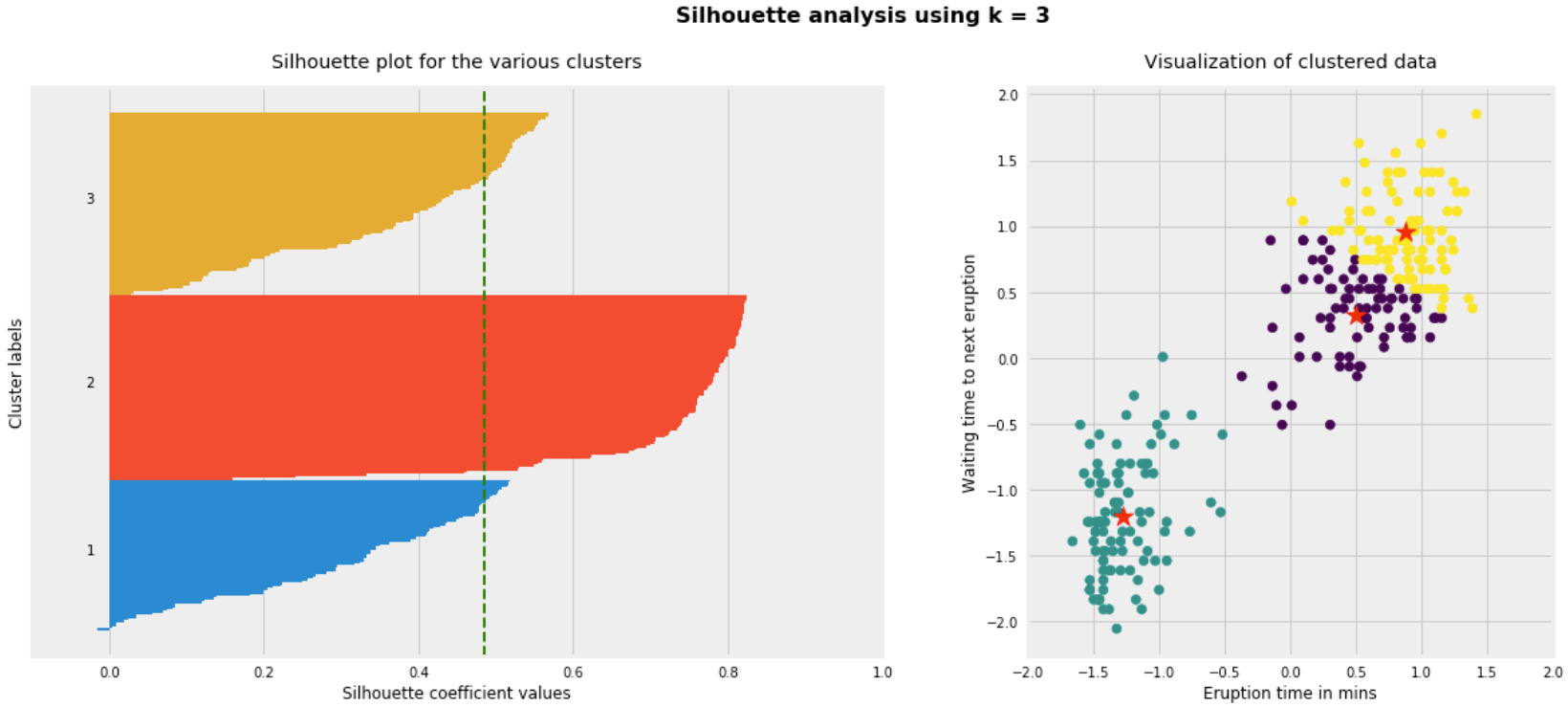


Silhouette Analysis: Example #1

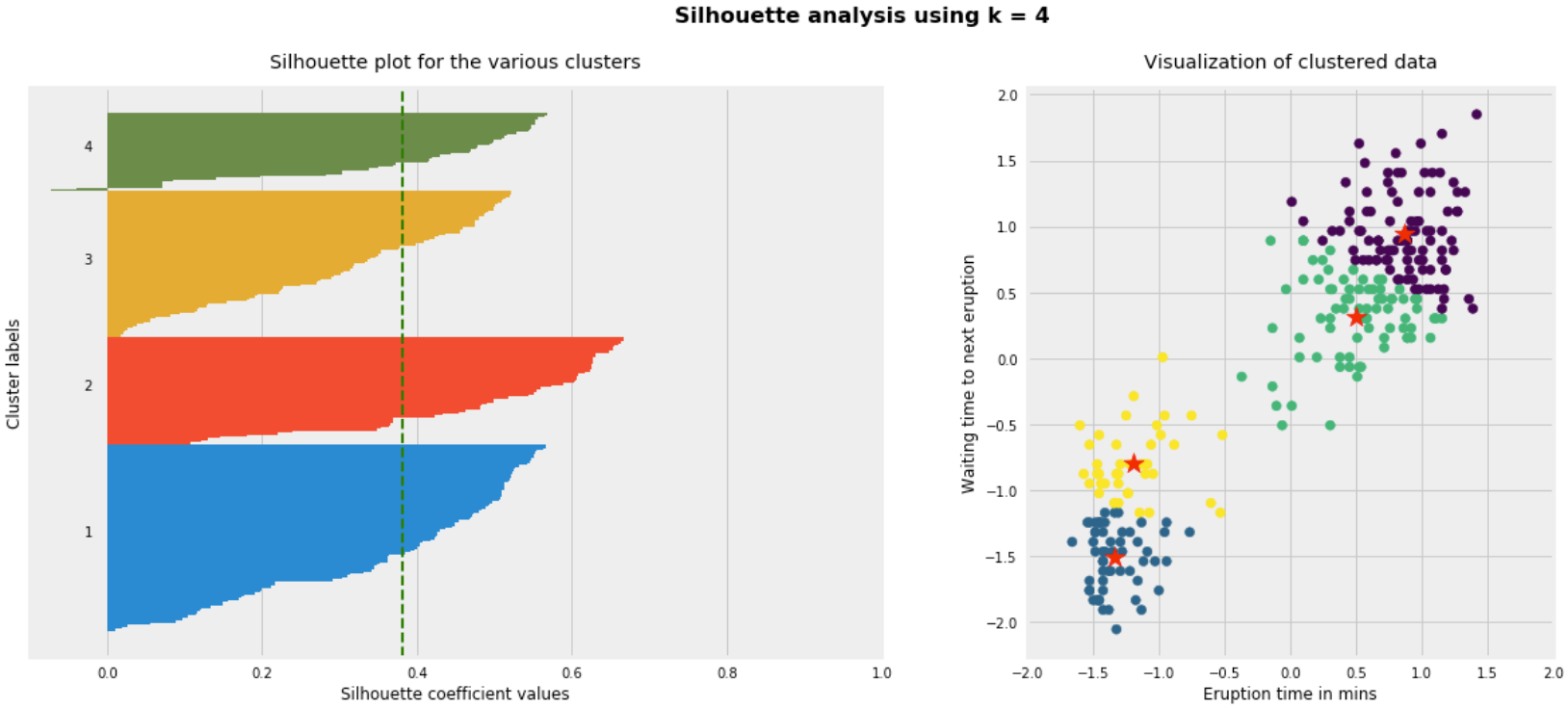




Silhouette Analysis: Example #1

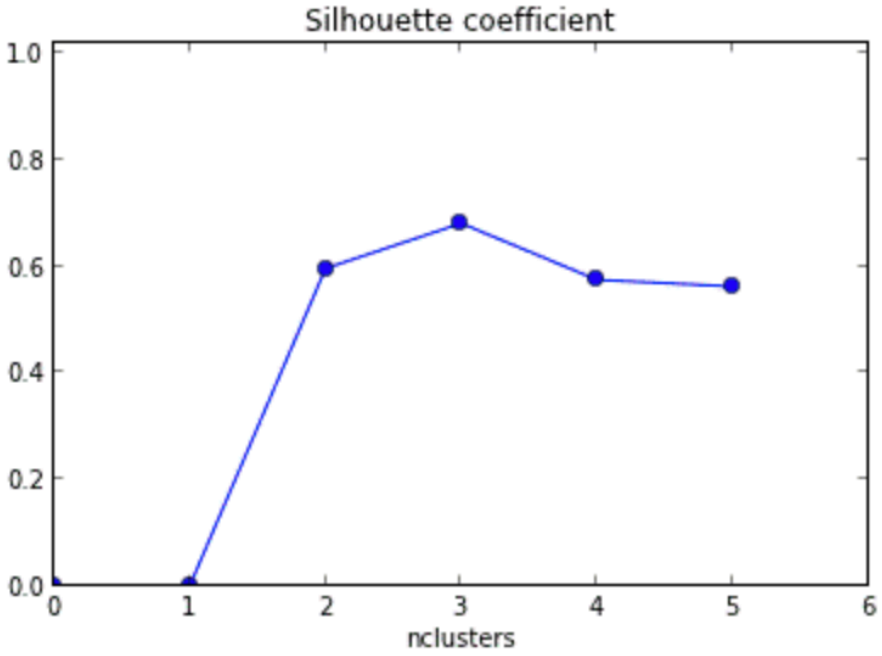


Silhouette Analysis: Example #1



Silhouette Analysis: Example #2

- What is the best choice of no. of clusters based on the following results?



Measuring Clustering Quality

- There is no commonly recognized best suitable measure in practice for evaluating the goodness of clustering results
- Three categories of measures:
  - External: Supervised, employ criteria not inherent to the dataset
    - Compare a clustering against prior or expert-specified knowledge (i.e., the ground truth) using certain clustering quality measure
  - Internal: Unsupervised, criteria derived from data itself
    - Evaluate the goodness of a clustering by considering how well the clusters are separated and how compact the clusters are, e.g., silhouette coefficient
  - Relative: Directly compare different clusterings, usually those obtained via different parameter settings for the same algorithm

Measuring Clustering Quality:

External Methods

- Given the ground truth *T*, *Q*(*C*, *T*) is the quality measure for a clustering *C*
- *Q*(*C*, *T*) is good if it satisfies the following four essential criteria
  - Cluster homogeneity: The purer, the better
  - Cluster completeness: Assign objects belonging to the same category in the ground truth to the same cluster
  - Rag bag better than alien: Putting a heterogeneous object into a pure cluster should be penalized more than putting it into a *rag bag* (i.e., "miscellaneous" or "other" category)
  - Small cluster preservation: Splitting a small category into pieces is more harmful than splitting a large category into pieces