# *COMP5121*
# Data Mining and Data Warehousing Applications

**Week 7: Cluster Analysis – Basic Concepts and Methods**
(Chapter 10 in textbook)

Dr. Fengmei Jin

- Email: fengmei.jin@polyu.edu.hk
- Office: PQ747 (+852 3400 3327)
- Consultation Hours: 2.30-4.30 pm every Thursday

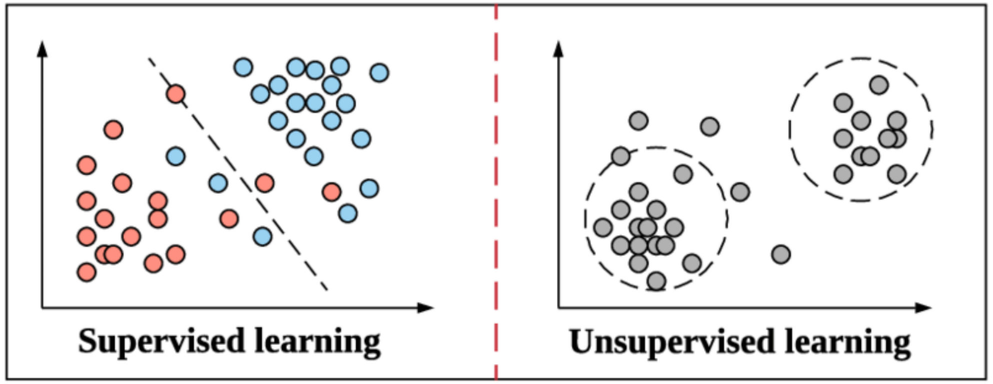
# Outline

- Cluster Analysis: An Introduction
- Partitioning Methods: K-means
- Hierarchical Methods: AGNES
- Density-based Methods: DBSCAN
- Evaluation of Clustering

# What Is Cluster Analysis?

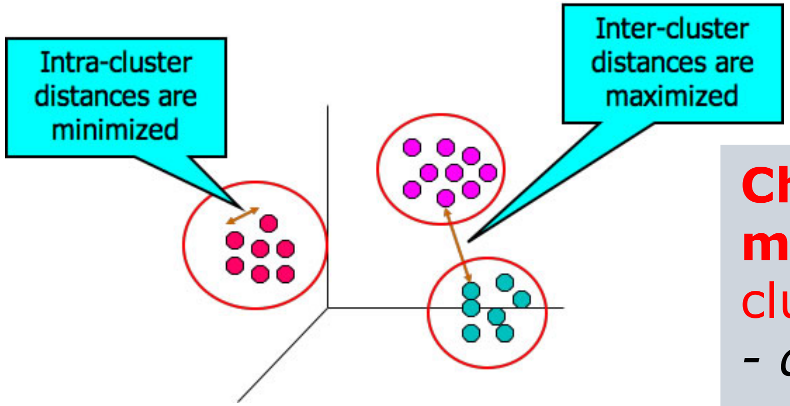
- **Cluster**: a collection of data objects that are –
  - Similar (or related) to one another within the same group / cluster
  - Dissimilar (or unrelated) to the objects in other groups / clusters

- Cluster analysis (also called clustering or data segmentation)
  - The process of partitioning data points into a set of groups where members of each group are as similar as possible to each other.
  - Unsupervised: no predefined classes

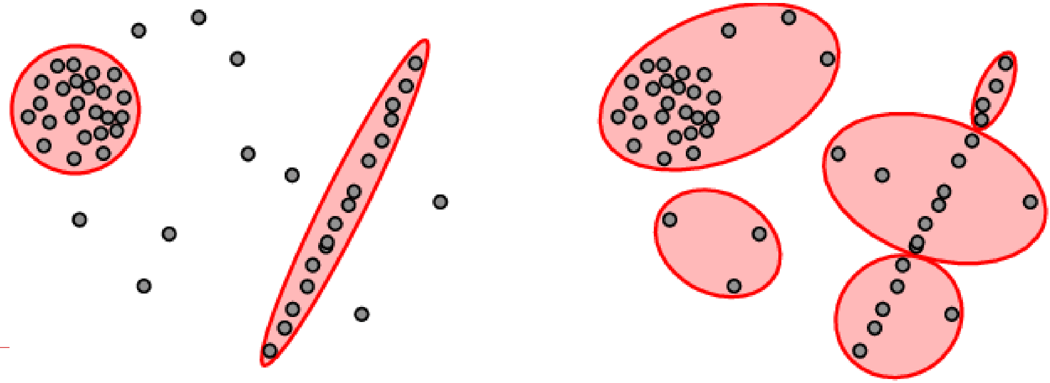


# What Is ‘Good’ Clustering?

- A good clustering method produces high-quality clusters:
  - **High intra-cluster similarity**: Cohesiveness within clusters
  - **Low inter-cluster similarity**: Distinctiveness between clusters
- Measuring clustering quality:
  - Use a separate quality function to evaluate clusters’ goodness
  - Hard to define “similar enough” or “good enough” – subjective



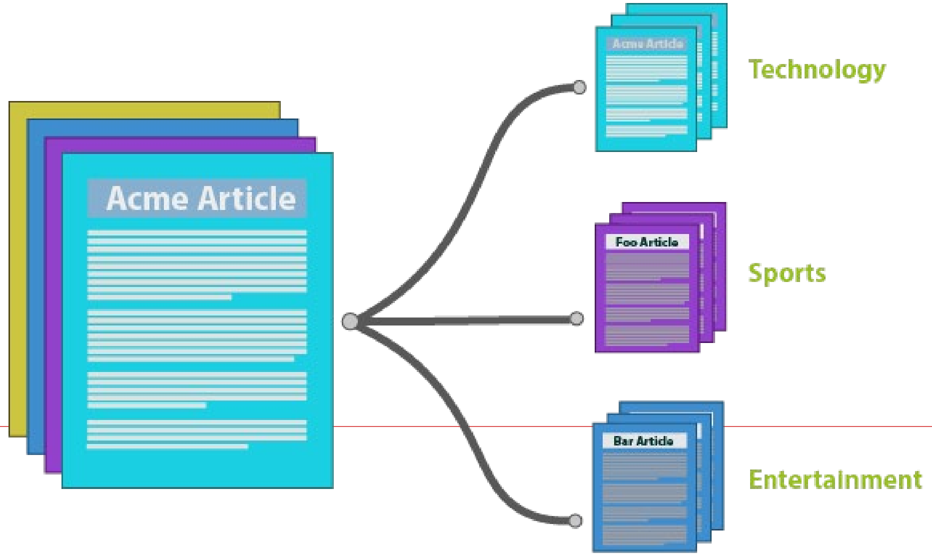

**Choosing a similarity measure** is critical for clustering!!
*- depends on application context (e.g., Euclidean for spatial, cosine for text)*

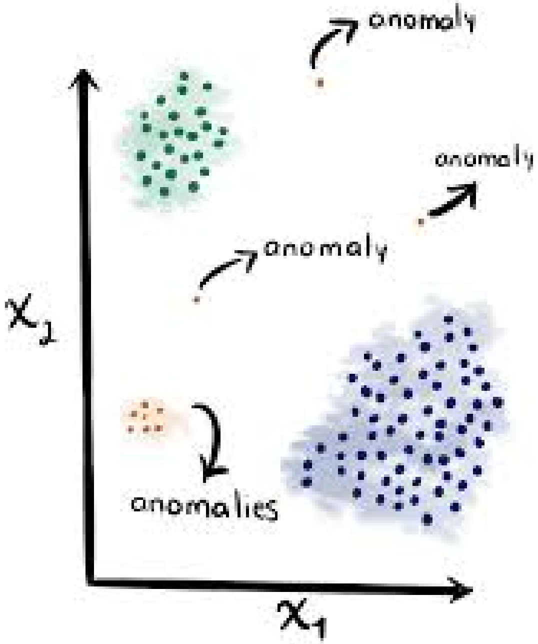


(a) ‘good’ (b) ‘bad’

# Cluster Analysis: Applications (I)

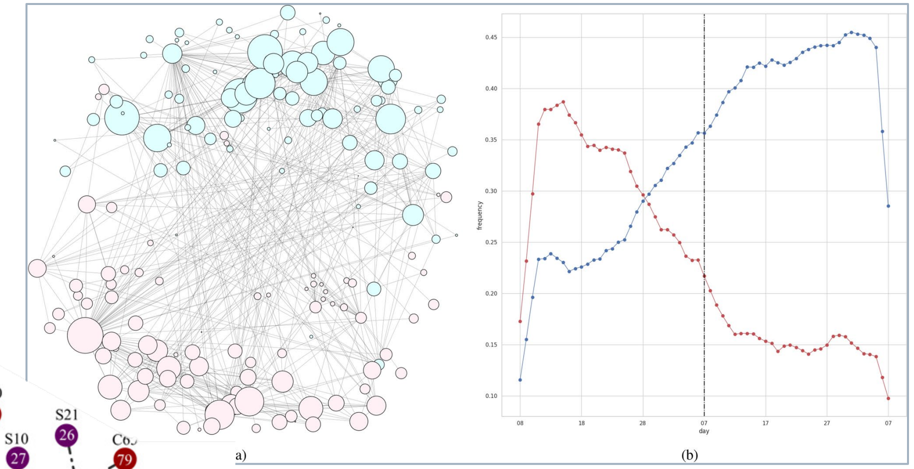
- A key intermediate step (preprocessing) for *other mining tasks*
  - Generating a compact summary of data
    - Organizing articles into topics for text classification or pattern discovery.
  - Anomaly detection: those "far away" from any cluster
  - Data compression and dimensionality reduction
    - e.g., image processing (via vector quantization)

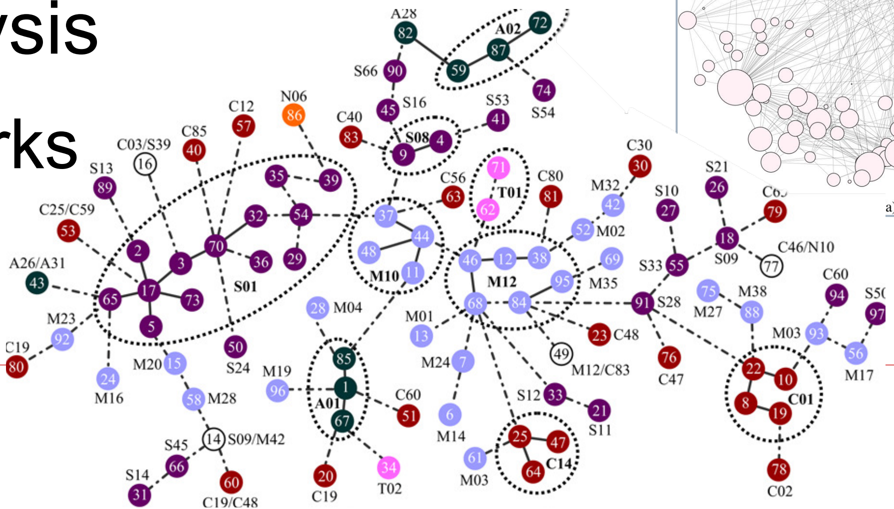


# Cluster Analysis: Applications (II)

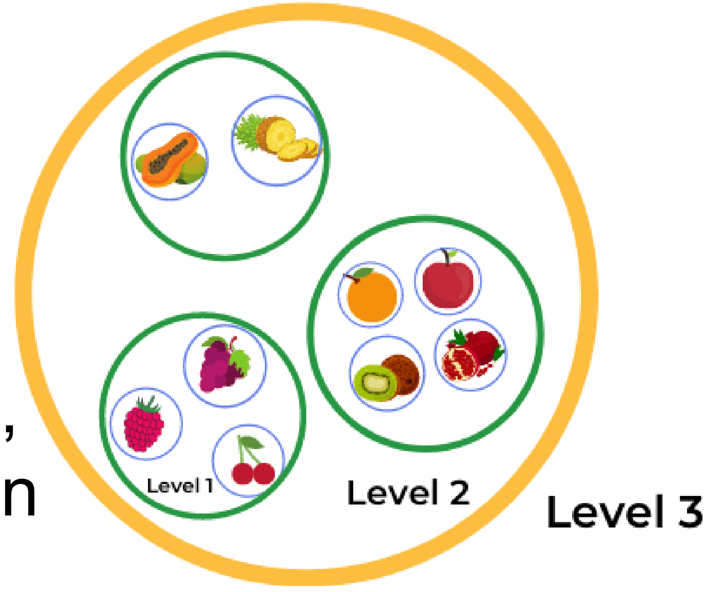
- Recommender systems
  - e.g., find *like-minded* users or similar products
- Dynamic trend detection
  - e.g., cluster and detect trends/patterns in social networks
- Multimedia data analysis
  - e.g., cluster images or video clips
- Biological data analysis
  - e.g., protein networks
- …



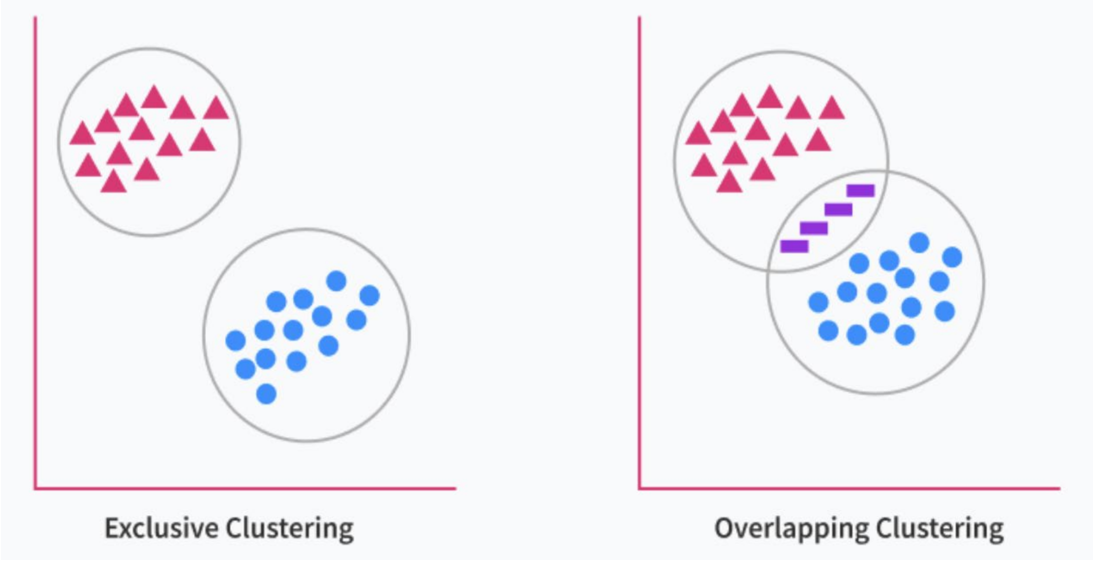
# Key Considerations for Clustering (I)

- **Partitioning**: Single level *vs.* Hierarchical
  - Multi-level clustering, where clusters are nested, are useful for applications like taxonomy creation or organizing topical terms




- **Separation of clusters**
  - Exclusive (a data point belongs to only one cluster, e.g., a customer assigned to a single region)

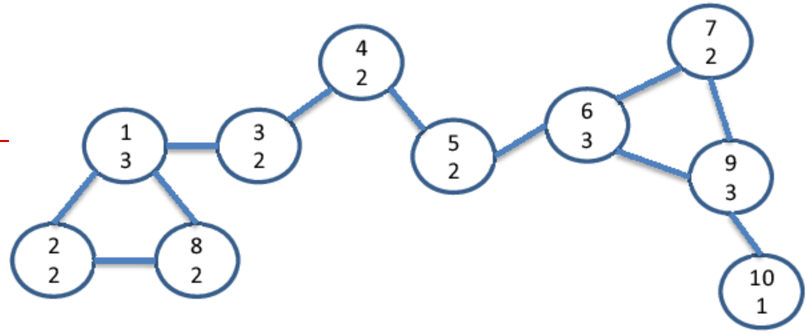
  *vs.* Non-exclusive (e.g., an article categorized into multiple topics)

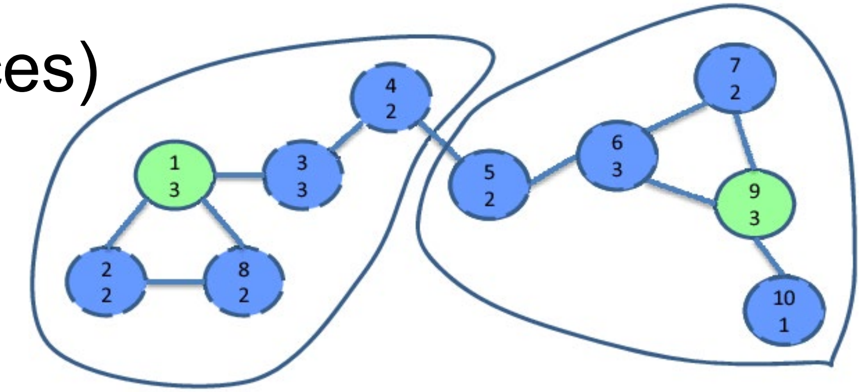


# Key Considerations for Clustering (II)

- **Similarity measures**
  - Distance-based, suitable for structured, continuous data (e.g., Euclidean, road network, vector spaces)

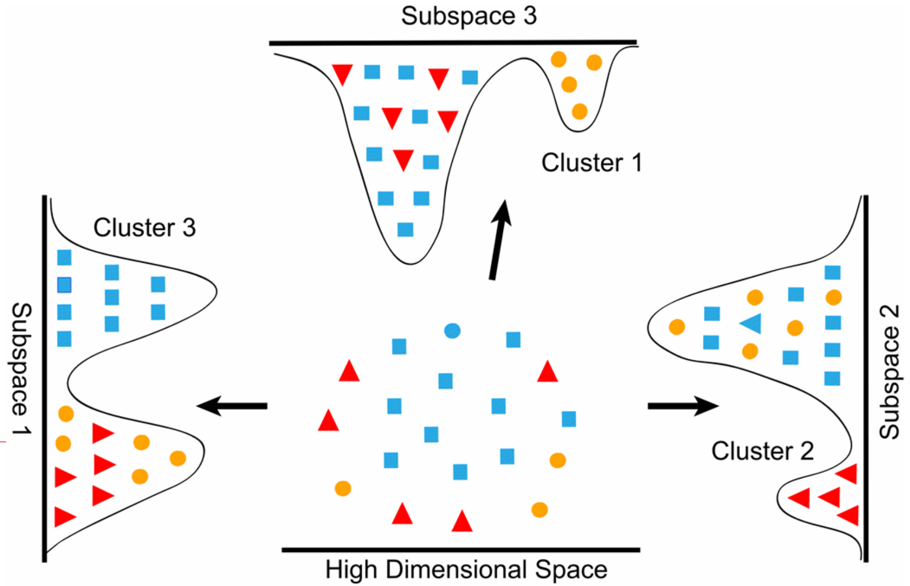
  *vs.* Connectivity-based, data with complex shapes or clusters defined by proximity (e.g., DBSCAN)





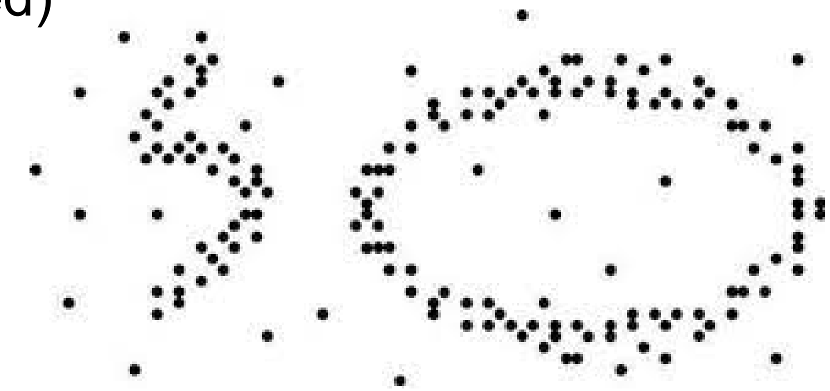
- **Clustering space**
  - Full space (for low-dimensional data)

  *vs.* Subspace (necessary for high-dimensional data)



# Requirements and Challenges (I)

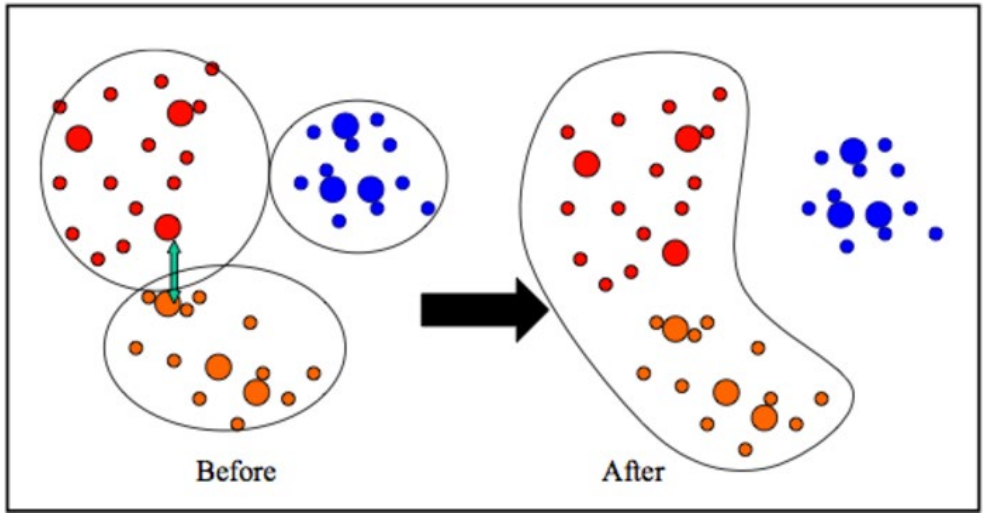
- **Quality**
  - Support for different types of attributes
    - numeric, nominal, text, multimedia, networks, or even mixed types
  - Discovery of clusters with arbitrary shapes
    - irregularly shaped clusters (e.g., circular, elongated)
  - Ability to deal with noisy data



- **Scalability**
  - High dimensionality
  - Clustering all the data instead of small samples
  - Incremental or stream clustering and insensitivity to input order

# Requirements and Challenges (II)

- **Constraint-based clustering**
  - User-given preferences / constraints
  - Leverages domain knowledge to improve clustering results
  - Customizes clustering results based on specific user-defined queries
    - e.g., cluster products that share similar features but prioritize high-selling items



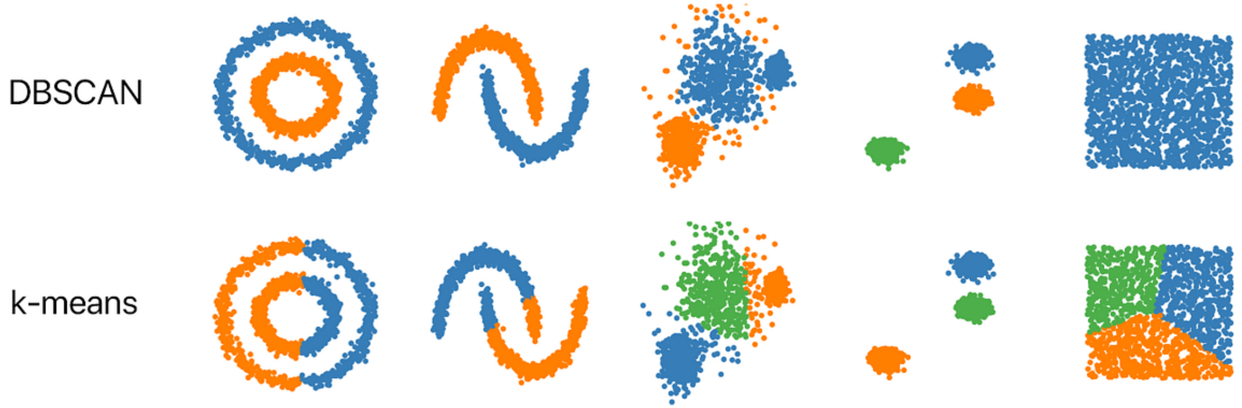

- **Interpretability and Usability**
  - Clustering results should be easy to understand for non-expert users.
  - Outputs should directly address user needs and allow actionable insights.

# Cluster Analysis: A Multi-Dimensional Categorization

- **Technique-Centered**
  - Distance-based: effective for compact, well-separated clusters
  - Density-based and grid-based: clusters of arbitrary shapes and handle noise
  - Probabilistic and generative models
  - High-dimensional clustering: useful for data with many irrelevant dimensions
  - Other scalable techniques for cluster analysis
- **Data Type-Centered**
  - numeric data, categorical data, text, multimedia data, time-series data, sequences, stream data, networked data, uncertain data, …
- **Additional Insight-Centered**
  - Visual insights, semi-supervised, ensemble-based, validation-based

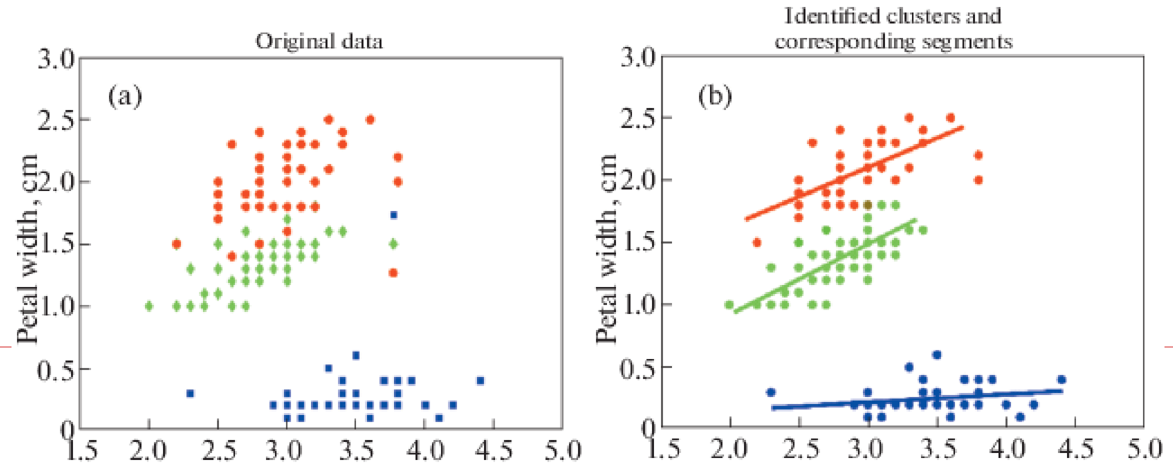
# 1) Typical Clustering Methodologies (I)

- **Distance-based methods**
  - Partitioning algorithms: K-Means, K-Medians, K-Medoids
  - Hierarchical algorithms: agglomerative *vs.* divisive methods
- **Density-based and Grid-based methods**
  - Density-based: Identify clusters as dense regions of points (arbitrary shapes) separated by sparse regions
  - Grid-based: Form clusters by grouping adjacent dense grid cells



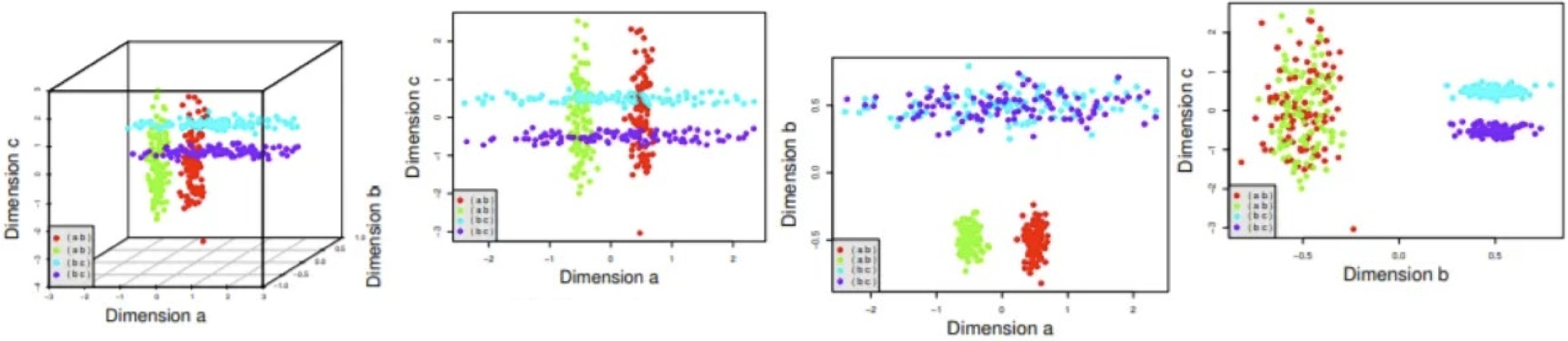
# 1) Typical Clustering Methodologies (II)

- **Probabilistic and Generative models**: Modeling clustering as a generative process
  1) Assume a specific form of the generative model
     - e.g., Hidden Markov Model (HMM) for sequential data
  2) Estimate model parameters using the EM algorithm
     - finding parameters that maximize the likelihood of the observed data
  3) Assign data points to clusters based on generative probabilities



# 1) Typical Clustering Methodologies (III)

- **High-dimensional clustering**
  - Subspace clustering:
    - Find clusters in subsets of dimensions (subspaces) rather than full space
    - e.g., bottom-up, top-down, correlation-based, …
  - Dimensionality reduction
    - Reduce dimensions to simplify data while retaining meaningful structure



# 2) Clustering Different Types of Data (I)

- **Numerical data:** *points in a multi-dimensional space*
  - Most early clustering algorithms were designed for such data.
- **Nominal data**
  - Discrete data without natural order (e.g., gender, race, zip codes, product categories, market-basket)
- **Text data**: Popular in social media, Web, documents
  - High-dimensional, sparse, represented as word frequencies
  - Methods: K-means and hierarchical clustering with vectorized data, such as topic modeling, co-clustering, etc.

# 2) Clustering Different Types of Data (II)

- **Multimedia data**: Image, audio, video
  - Multi-modal, often combined with text data
  - Containing both behavioral and contextual attributes
    - A pixel of images: intensity/color (behavior) and position (context)
  - Methods: deep learning models like CNNs, RNNs, etc.

- **Time-series data**: Sensor readings, stock prices, GPS tracking
  - Data are temporally dependent
    - Time: contextual attribute; Value: behavioral attribute
  - Correlation-based online analysis & Shape-based offline analysis

# 3) User Insights and Interactions in Clustering

- **Visual insights**: "*One picture is worth a thousand words.*"
  - Visualizations help humans intuitively understand cluster structures.
- **Semi-supervised insights**: Passing user's intention to system
  - User-seeding: A user provides several labeled examples, approximately representing initial categories of interest
- **Multi-view and ensemble-based insights**
  - Multi-view clustering: Multiple clusters represent different perspectives
  - Multiple clustering results can be ensembled for a more robust solution.
- **Validation-based insights**: Evaluating quality of generated clusters
  - May use case studies, specific measures, or pre-existing labels

k-means, k-medoids, k-medians, k-modes, …

# PARTITIONING-BASED CLUSTERING METHODS

# Partitioning Algorithms: Basic Concepts

- **Partitioning method**
  - Discover groupings in the data by optimizing a specific objective function and iteratively improving the quality of partitions
- ***K*-partitioning method**
  - Objective: Divide a dataset $\boldsymbol{D}$ of $n$ objects into a set of $K$ clusters, so that an objective function is optimized (e.g., minimizing the sum of distances within clusters)
  - Typical objective function: **Sum of Squared Errors (SSE)**

$$SSE(C) = \sum_{k=1}^{K} \sum_{x_{i \in C_k}} \| x_i - c_k \|^2$$

where $c_k$ is the centroid or medoid of cluster $C_k$

# Partitioning Algorithms: Basic Concepts

- To find the best partition of $K$ clusters that optimizes the chosen partitioning criterion –
  - **Global optimal**: exhaustively enumerate all possible partitions
  - **Heuristic methods** (i.e., greedy algorithms): approximation
    - $K$-Means, $K$-Medians, $K$-Medoids, etc.

- Different kinds of measures can be used:
  - Manhattan / Euclidean distance
  - Cosine similarity
  - …

# The $K$-Means Clustering Algorithm

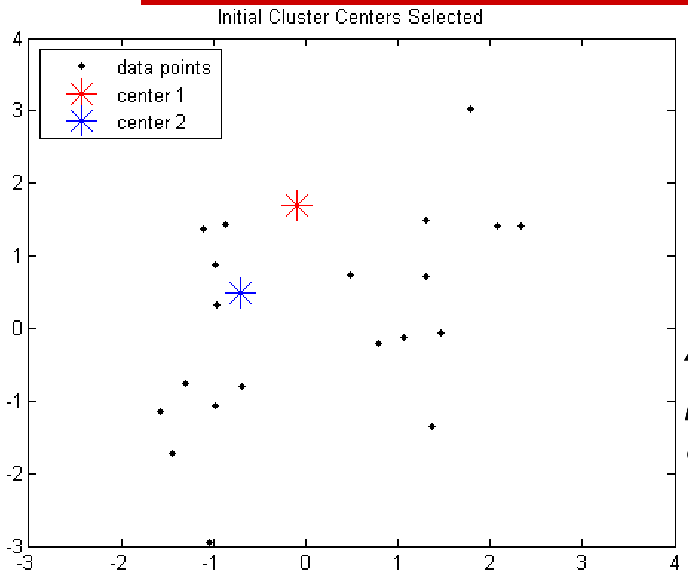
- **Idea**: each cluster is represented by the centroid, which is the mean position of all data points in the cluster
  - *It may not correspond to an actual data point in the dataset!*

- Given $K$, the number of clusters, the $K$-Means clustering algorithm is outlined as follows:

**Initialization**: Select $K$ data 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

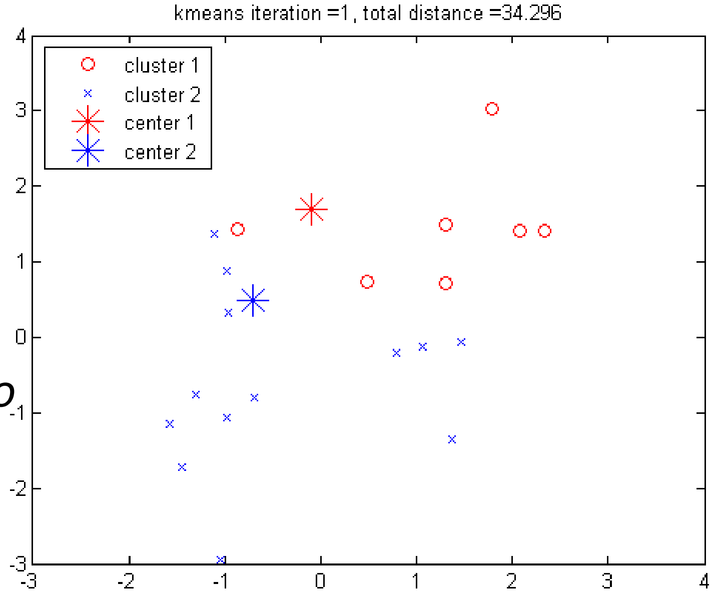
**Until** centroids no longer change or convergence criterion is met

# Example: $K$-Means Clustering

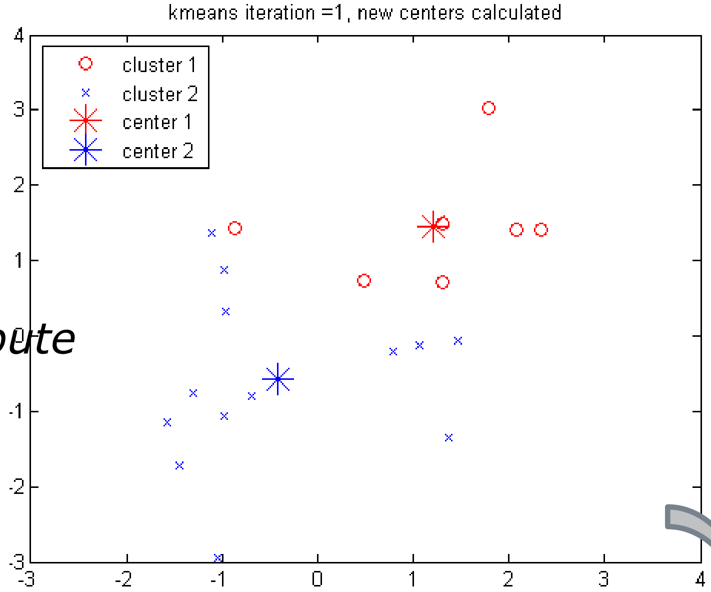



The original data points &
randomly select $K = 2$ centroids

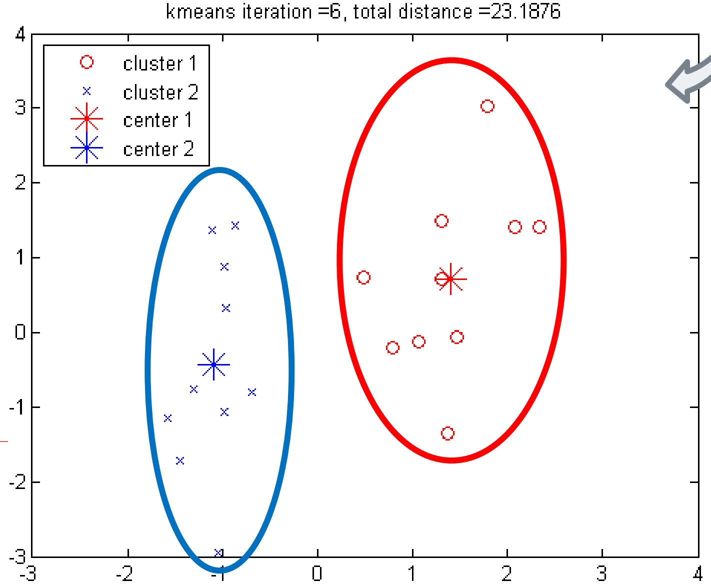
*Assign points to clusters*




*Re-compute cluster centers*




*Iteratively redo point assignment*




## *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** convergence criterion is satisfied

# Discussion on $K$-Means Clustering (I)

- **Efficiency**: $O(tKn)$
  - Input terms: $t = \#$ iterations, $K = \#$ clusters, $n = \#$ objects
  - Normally, $K, t \ll n$, thus, an efficient method.

- **Limitations (I)**
  - Need to specify $K$ in advance
    - There are ways to automatically determine the '*best*' $K$.
    - In practice, one often runs a range of values and selected the '*best*'.
  - Only for objects in a continuous data space: $K$-modes for nominal data

# Discussion on $K$-Means Clustering

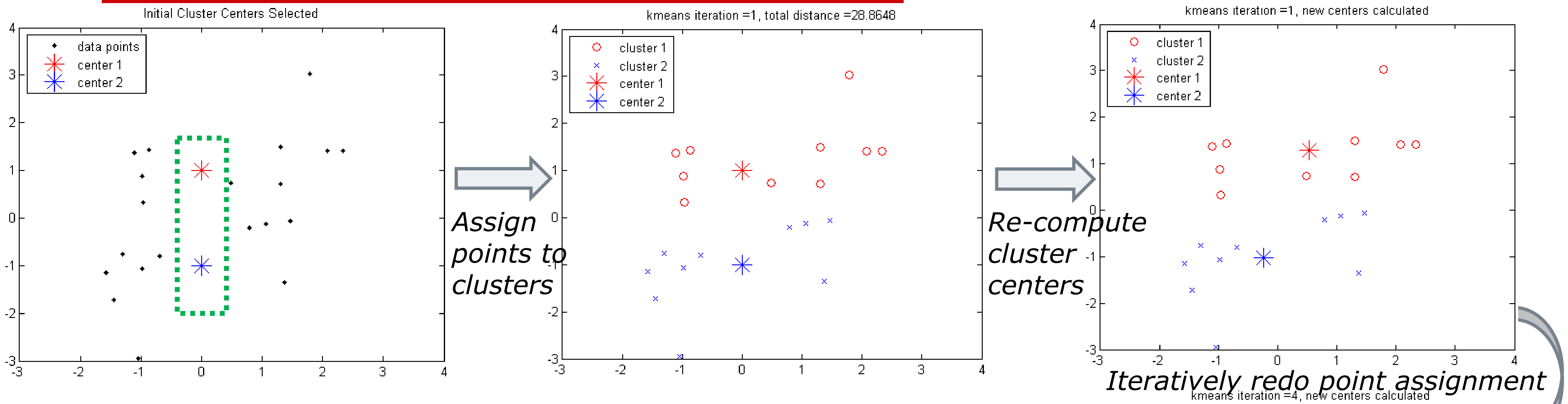
## Limitations (II)

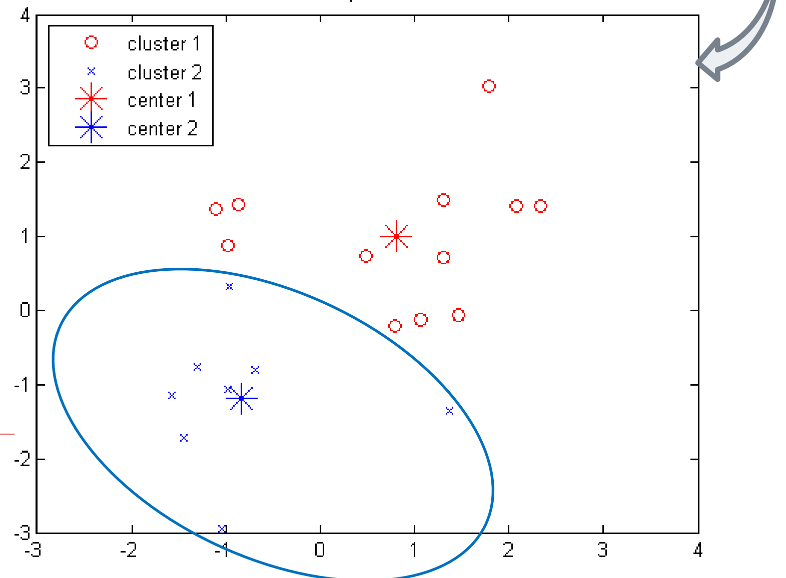
- $K$-means clustering often terminates at a local optimum.
  - Poor initialization can lead to suboptimal clusters.
- Sensitive to noisy data and outliers (extreme values)

## Variations of $K$-Means

- Choosing better initial centroid estimates
  - e.g., $K$-Means++, Intelligent $K$-Means, Genetic $K$-Means
- Choosing different representative prototypes for the clusters
  - e.g., $K$-Medoids, $K$-Medians, $K$-Modes
- Applying feature transformation techniques
  - e.g., Weighted $K$-Means, Kernel $K$-Means

# Example: Poor Initialization May Lead to Poor Clustering




- This run of $K$-Means generates poor-quality clusters.
- Re-run of $K$-*Means* using another random $K$ seeds.

# Problem 1: Initialization of $K$-Means

- Different initializations may generate very different clustering results. Some could be far from optimal!
- **Original proposal** (MacQueen'67): Select $K$ seeds randomly
  - Require running the algorithm multiple times with different seeds
- Improved methods for better initialization of $K$ seeds

**Variation: $K$-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 all $K$ centroids are chosen

# Problem 2: Handling Outliers by Medoids

- The $K$-Means algorithm is sensitive to outliers!
  - An object with an extremely large value may substantially distort the distribution of the data.

- Variation 1: From $K$-Means to **$K$-Medoids**
  - Robustness to outliers
    - $K$-medoids choose actual data points, which are the most centrally located object in a cluster, as centers (called medoids)
  - Greater interpretability of the cluster centers
    - $K$-means takes the calculated mean value of objects in a cluster as a reference point, while $K$-medoids are real data points.

# Variation 1: The $K$-Medoids Clustering Algorithm

- Select $K$ points as the initial representative objects (i.e., as initial $K$ medoids)
- **Repeat**
  - Assigning each point to the cluster with its closest medoid
  - **Update Step**: within each cluster (the current medoid is $m$)
    - For each "non-medoid" object $o_i$ in this cluster, compute the total cost of swapping the current medoid $m$ with $o_i$
    - Determine which swap would improve the clustering quality
  - Form new set of medoids if any cluster's medoid was changed
- **Until** convergence criterion is satisfied

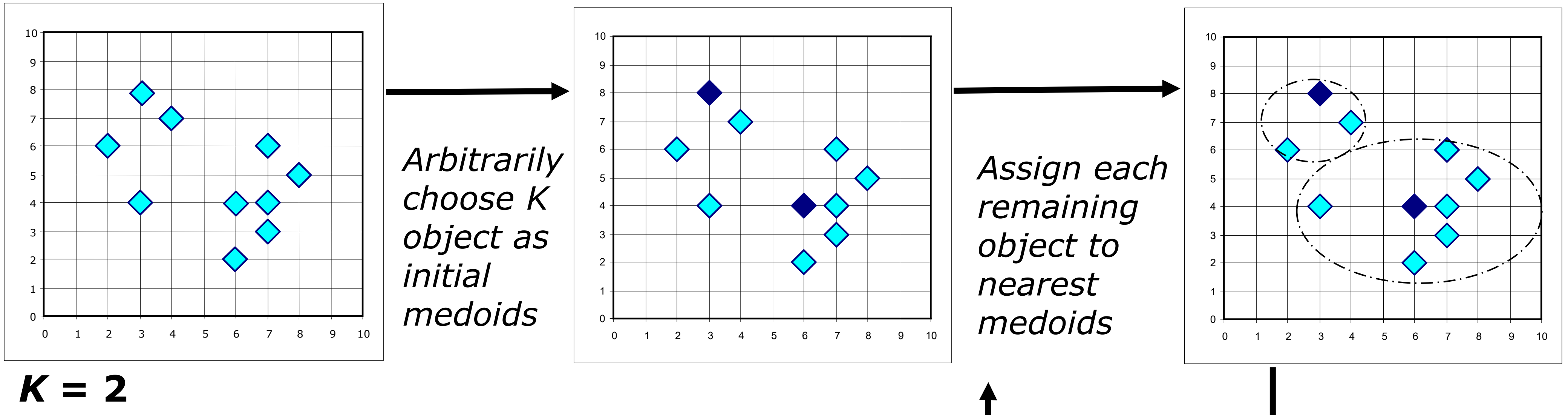
# Discussion on $K$-Medoids Clustering

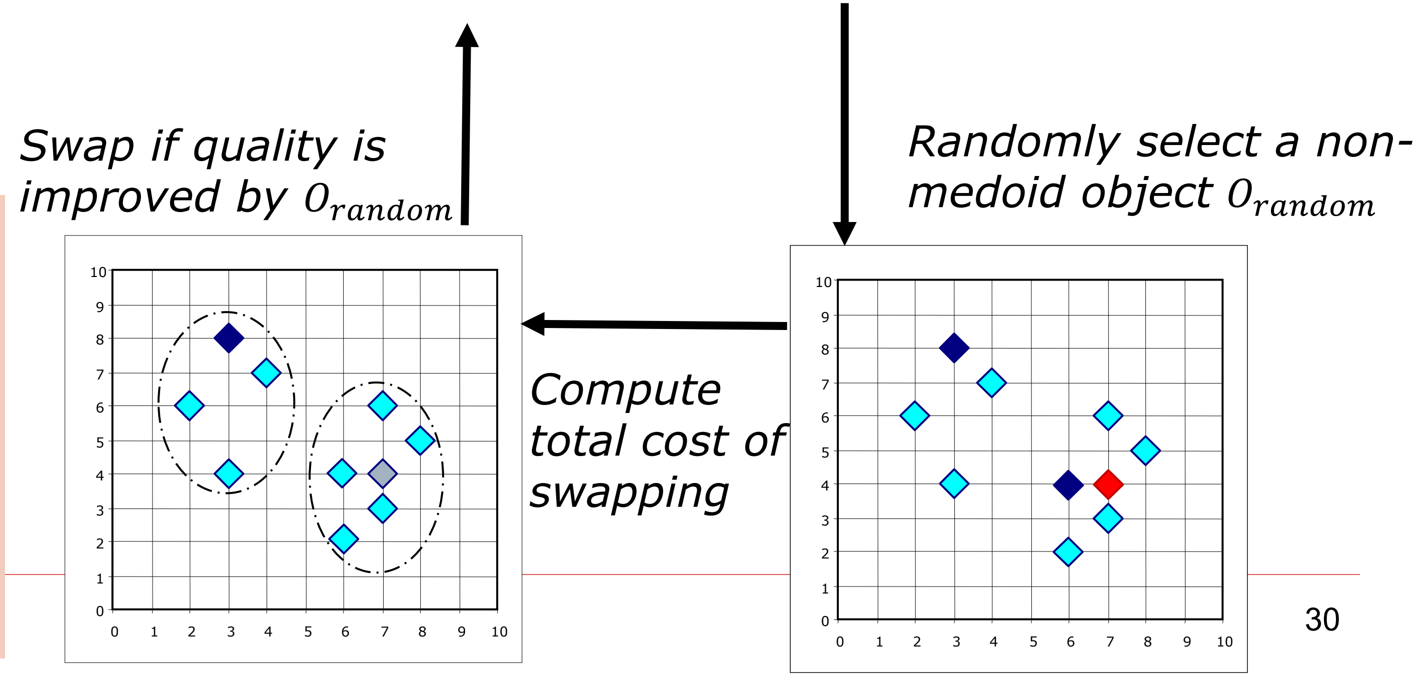
- $K$-Medoids Clustering
  - Find the most representative objects (medoids) in clusters
- **PAM** (Partitioning Around Medoids: *Kaufmann, et al.,1987*)

> - 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

  - Efficient for small data, but hard to scale well for large data
    - Computational complexity of **PAM**: $O(K(n - K)^2)$ – expensive!

# 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

# Problem 2: Handling Outliers by Medians

- Medians are less sensitive to outliers than means
  - e.g., median salary *vs.* mean salary of a large company when adding a few top executives!

- Variation 2: ***K*-Medians**
  - Instead of taking the mean value of the object in a cluster as a reference point, medians are used ($L_1$-norm as the measure)
  - The criterion function for the $K$-Medians algorithm:

$$S = \sum_{k=1}^{K} \sum_{x_{i \in C_k}} | x_{ij} - med_{kj} |$$

# Variation 2: 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

# Variation 3: $K$-Modes for Clustering Nominal Data

- $K$-Means cannot handle non-numerical (categorical) data
  - Mapping categorical value to 1 / 0 cannot generate quality clusters for high-dimensional data
- **$K$-Modes**: An extension to $K$-Means by replacing means of clusters with modes
  - Different similarity measures between categorical objects can be used. It may be frequency-based.
  - Algorithm is still based on iterative object cluster assignment and centroid update.

Agglomerative vs Divisive

# HIERARCHICAL-BASED CLUSTERING

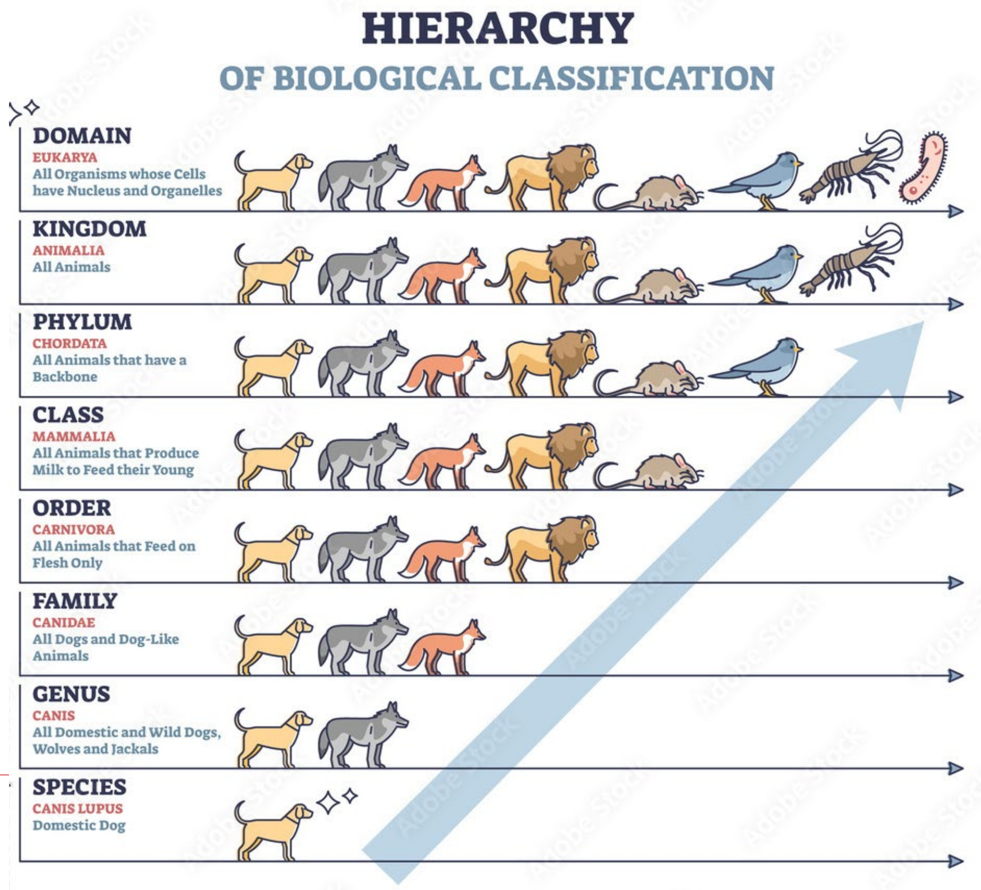
# Hierarchical-based clustering

- To group data objects into a hierarchy or “tree” of clusters
  - Very useful for data summarization and visualization as it provides a nested structure of clusters

**Employees can be grouped hierarchically:**

- Top-level: {Executives, Managers, Staff}
- Subgroups: Staff = {Senior Officers, Officers, Trainees}

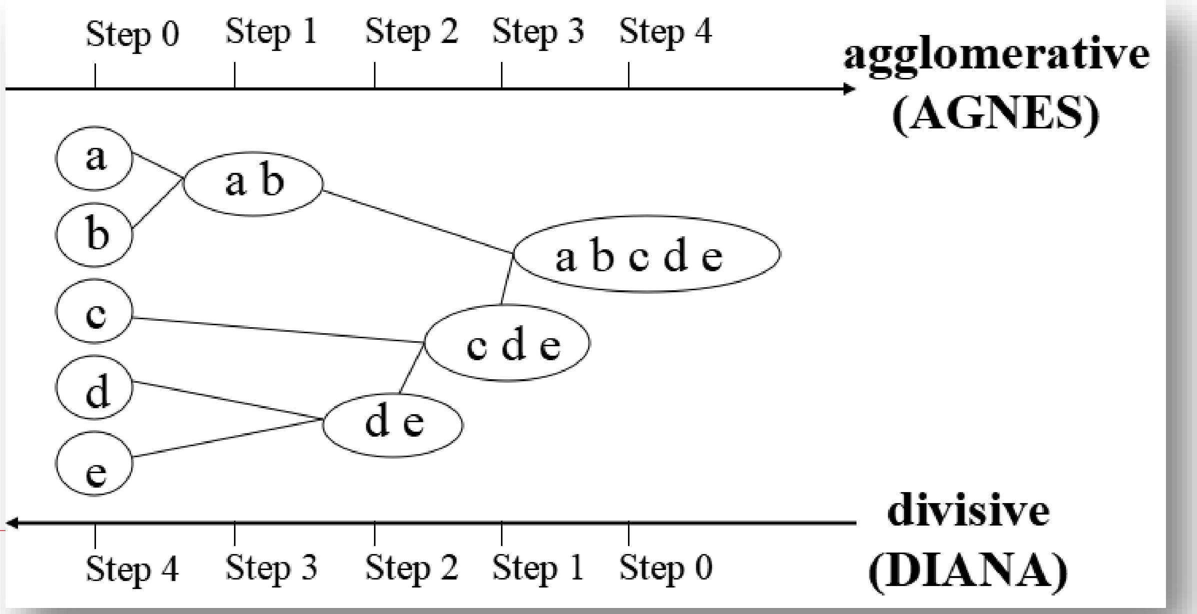
In the study of evolution, **group animals** according to their biological features to uncover evolutionary paths – a hierarchy of species



# Why Hierarchical Clustering?

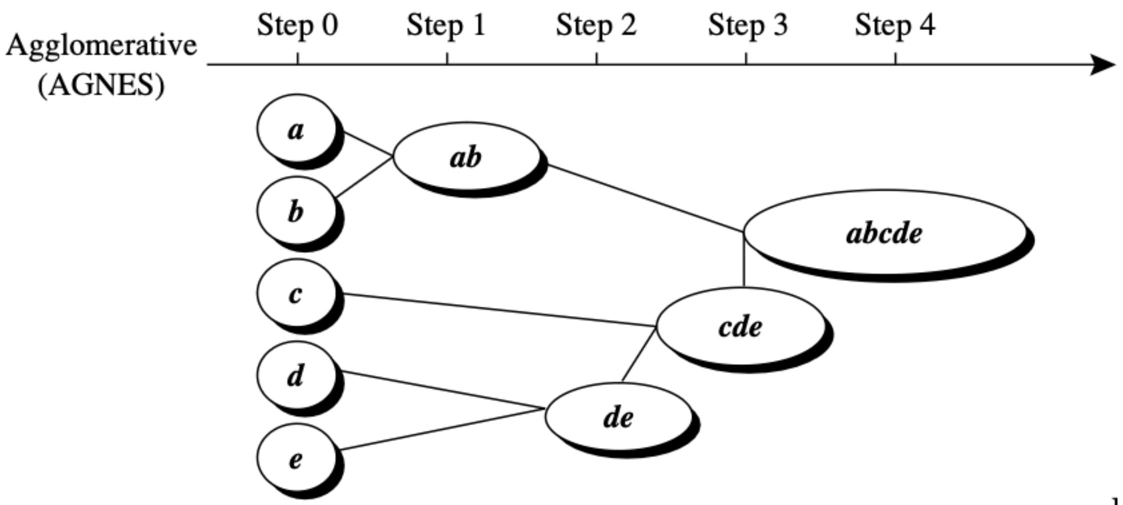
- No need to specify # clusters in advance
- Provide more insights into data structure (relationships)
- Flexible linkage criteria based on data characteristics
- Handle any type of distance metric

- Two types of methods:
  - **Agglomerative**: bottom-up
  - **Divisive**: top-down



# Agglomerative Hierarchical Clustering

- Bottom-up strategy
  - Start by letting every object form its own cluster and iteratively merges clusters into larger and larger clusters
    - For **merging**, it finds the two clusters that are closest to each other (according to some similarity measures) as a new one.
  - Stop until all the objects are in a single cluster (the hierarchy's root), or certain termination conditions are satisfied.



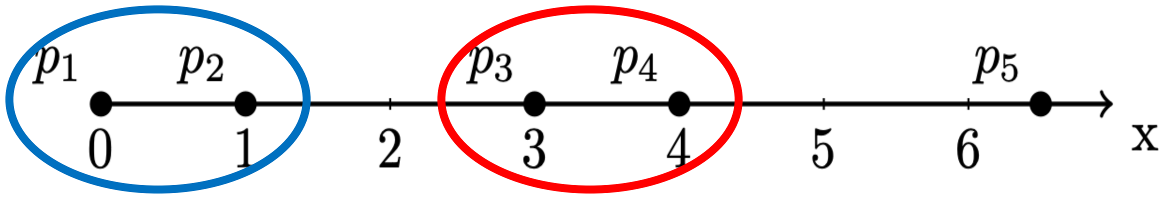
# Distance Between Clusters

- Single linkage (between the nearest points of two clusters)
- Complete linkage (between the farthest points of two clusters)
- Average linkage (between all points)
- Centroid linkage (mean)

**Single linkage:**
$dist(C_1, C_2) = 2, dist(C_2, C_3) = 2.5 \rightarrow$ Merge $C_1$ and $C_2$

**Complete linkage:**
$dist(C_1, C_2) = 4, dist(C_2, C_3) = 3.5 \rightarrow$ Merge $C_2$ and $C_3$

- Example:
  - $C_1 = \{p_1, p_2\}$
  - $C_2 = \{p_3, p_4\}$
  - $C_3 = \{p_5\}$



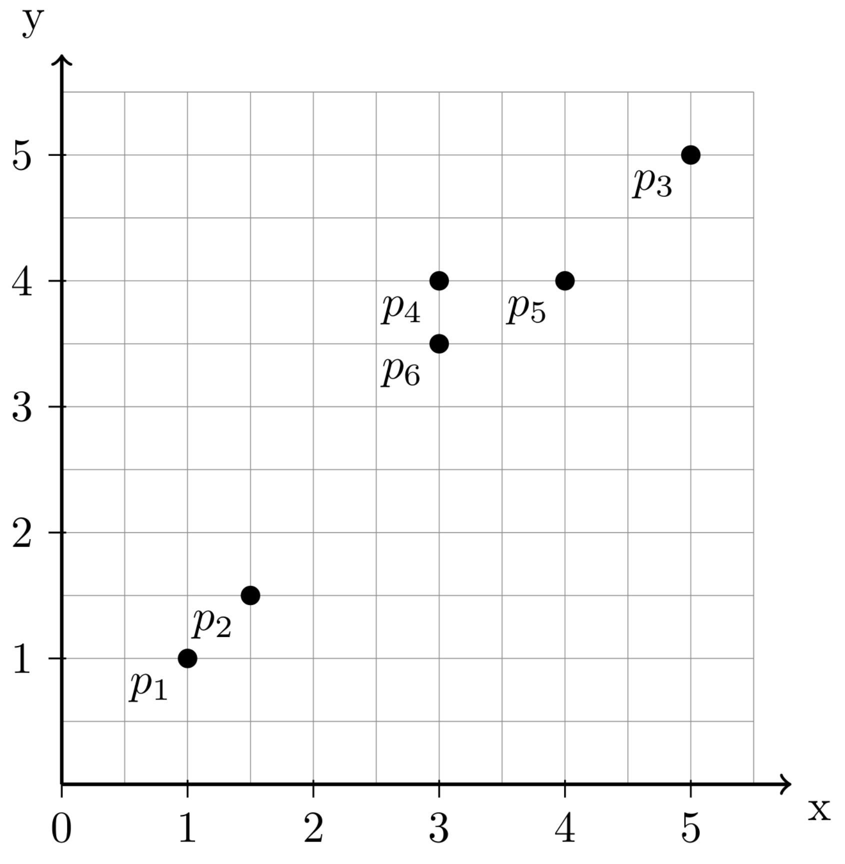
# AGNES: **AG**glomerative **NES**ting algorithm

- **Steps:**
  1. Initialization: Start with $n$ points, each treated as its own cluster (i.e., $n$ clusters)
  2. Find the nearest pair of clusters based on a predefined measure
  3. Merge these two clusters
     - Combine them into one cluster
     - Update the distance matrix to reflect the new cluster distances
  4. Repeat step 2 & 3 until all data points belong to a single cluster

# AGNES: An Example

- Initialize with 6 clusters
  - $\{p_1\}, \{p_2\}, \{p_3\}, \{p_4\}, \{p_5\}, \{p_6\}$
- Calculate distances

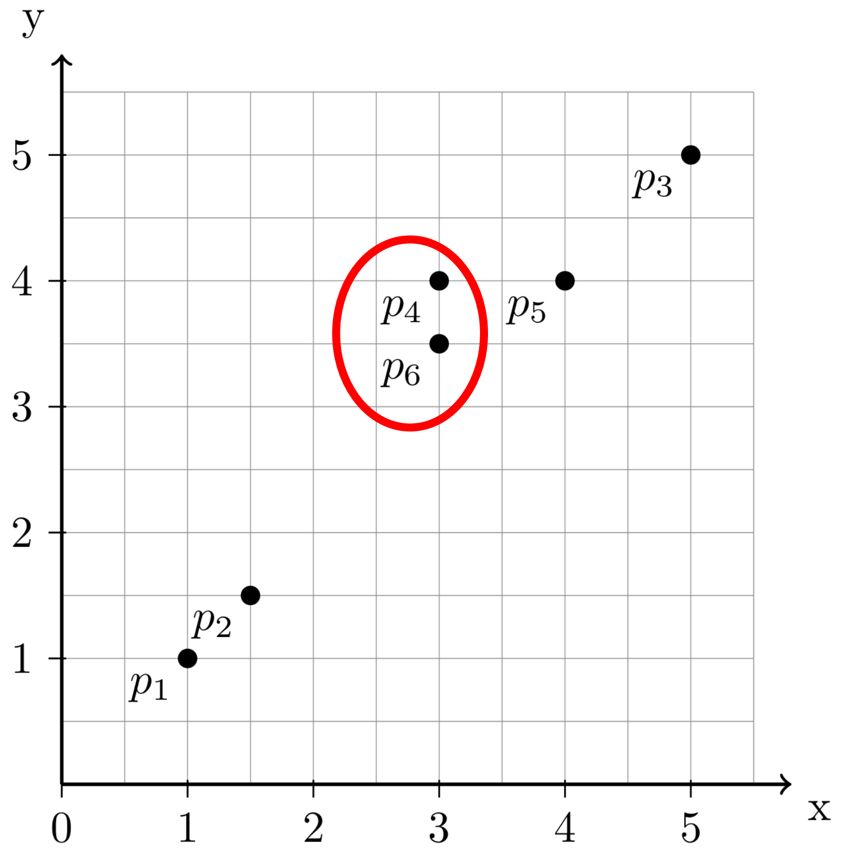
| | $\{p_1\}$ | $\{p_2\}$ | $\{p_3\}$ | $\{p_4\}$ | $\{p_5\}$ | $\{p_6\}$ |
|---|---|---|---|---|---|---|
| $\{p_1\}$ | 0.0 | | | | | |
| $\{p_2\}$ | 0.7 | 0.0 | | | | |
| $\{p_3\}$ | 5.7 | 4.9 | 0.0 | | | |
| $\{p_4\}$ | 3.6 | 2.9 | 2.2 | 0.0 | | |
| $\{p_5\}$ | 4.2 | 3.5 | 1.4 | 1.0 | 0.0 | |
| $\{p_6\}$ | 3.2 | 2.5 | 2.5 | 0.5 | 1.1 | 0.0 |



# AGNES: An Example

- Merge $\{p_4\}$ and $\{p_6\}$
  - $\{p_1\}, \{p_2\}, \{p_3\}, \{p_4, p_6\}, \{p_5\}$
- Calculate distances

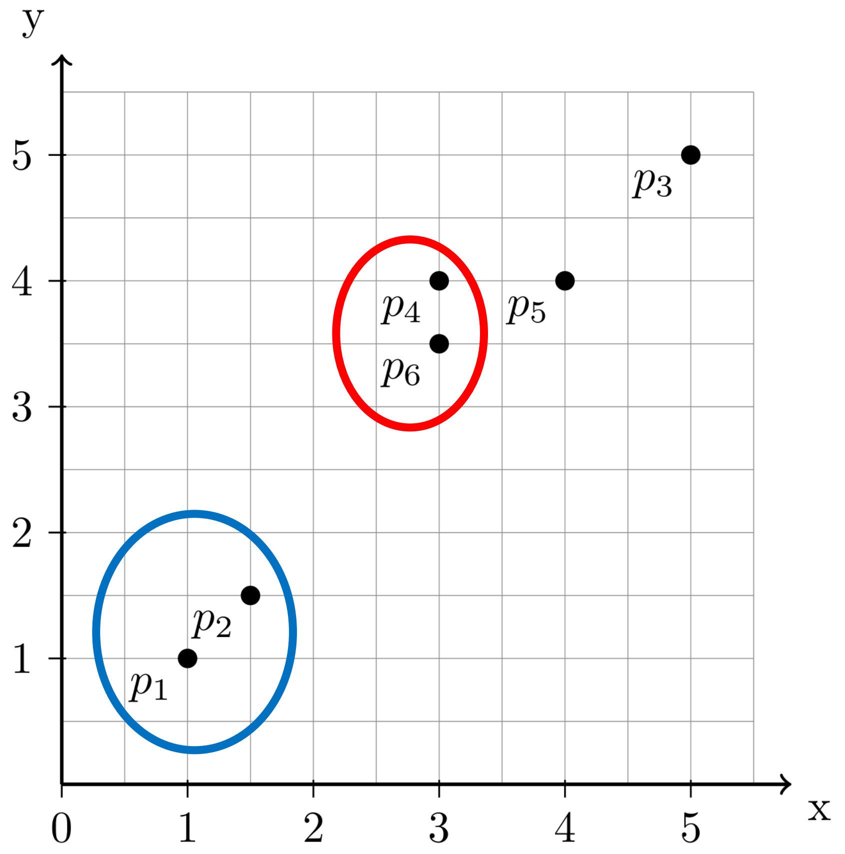
| | $\{p_1\}$ | $\{p_2\}$ | $\{p_3\}$ | $\{p_4, p_6\}$ | $\{p_5\}$ |
|---|---|---|---|---|---|
| $\{p_1\}$ | 0.0 | | | | |
| $\{p_2\}$ | 0.7 | 0.0 | | | |
| $\{p_3\}$ | 5.7 | 4.9 | 0.0 | | |
| $\{p_4, p_6\}$ | 3.2 | 2.5 | 2.2 | 0.0 | |
| $\{p_5\}$ | 4.2 | 3.5 | 1.4 | 1.0 | 0.0 |



# AGNES: An Example

- Merge $\{p_1\}$ and $\{p_2\}$
  - $\{p_1, p_2\}, \{p_3\}, \{p_4, p_6\}, \{p_5\}$
- Calculate distances

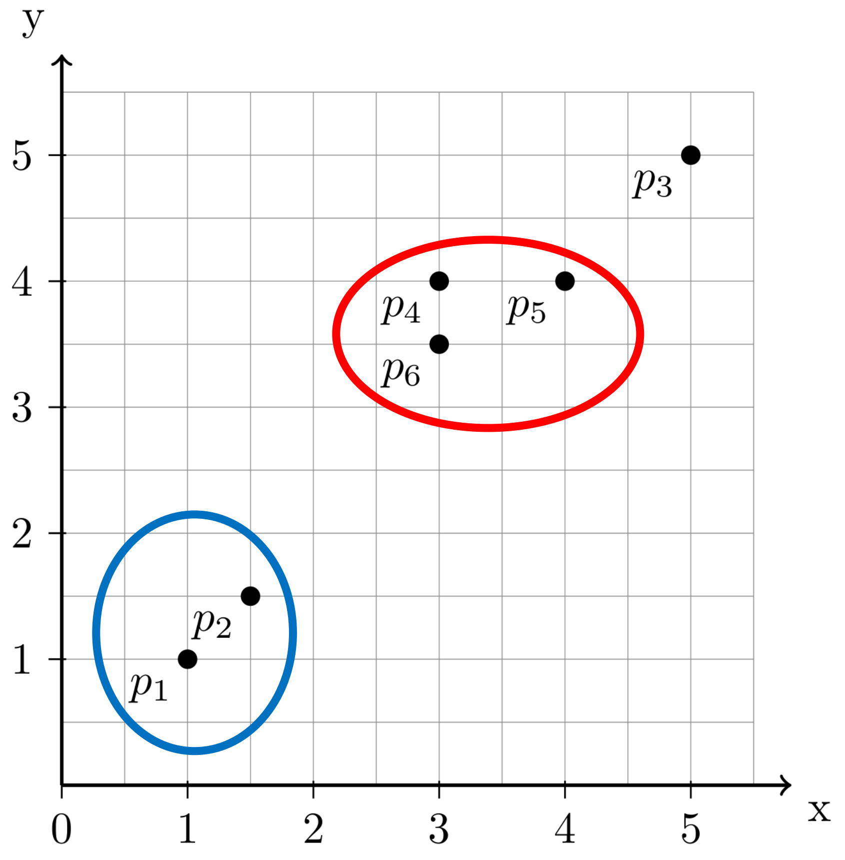
| | $\{p_1, p_2\}$ | $\{p_3\}$ | $\{p_4, p_6\}$ | $\{p_5\}$ |
|---|---|---|---|---|
| $\{p_1, p_2\}$ | 0.0 | | | |
| $\{p_3\}$ | 4.9 | 0.0 | | |
| $\{p_4, p_6\}$ | 2.5 | 2.2 | 0.0 | |
| $\{p_5\}$ | 3.5 | 1.4 | 1.0 | 0.0 |



# AGNES: An Example

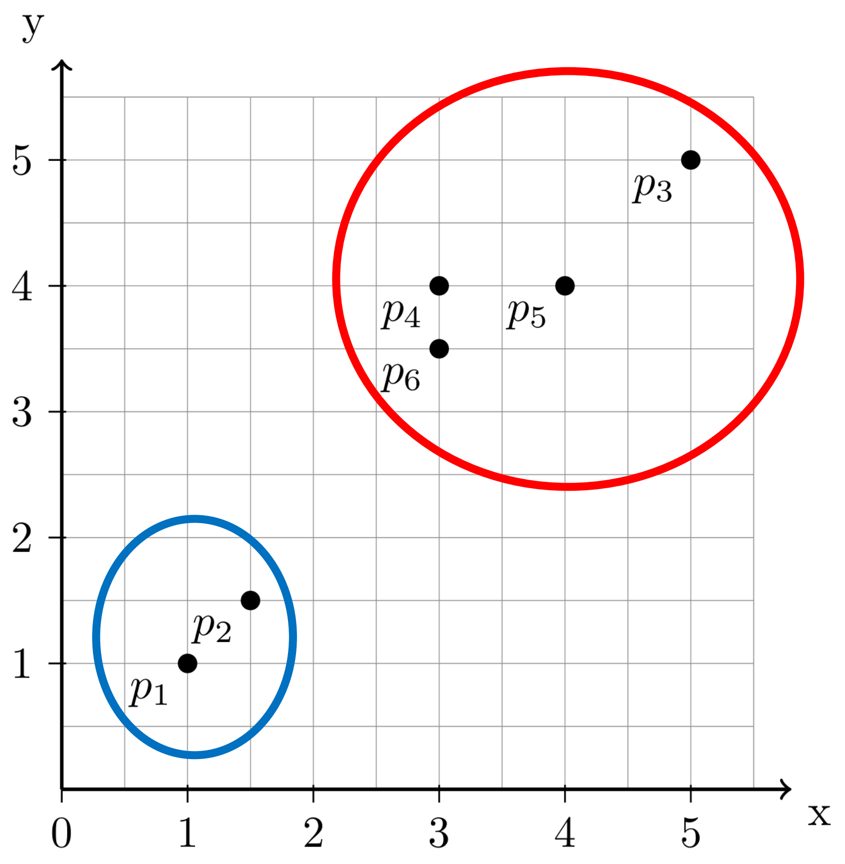
- Merge $\{p_4, p_6\}$ and $\{p_5\}$
  - $\{p_1, p_2\}, \{p_3\}, \{p_4, p_5, p_6\}$
- Calculate distances

| | $\{p_1, p_2\}$ | $\{p_3\}$ | $\{p_4, p_5, p_6\}$ |
|---|---|---|---|
| $\{p_1, p_2\}$ | 0.0 | | |
| $\{p_3\}$ | 4.9 | 0.0 | |
| $\{p_4, p_5, p_6\}$ | 2.5 | 1.4 | 0.0 |



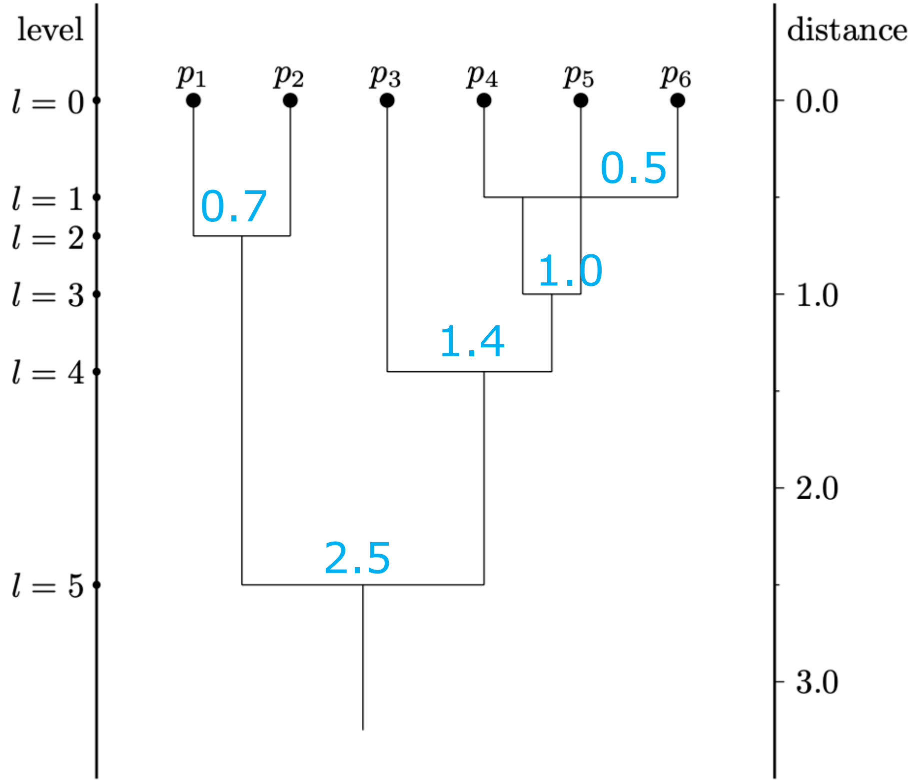
# AGNES: An Example

- Merge $\{p_4, p_5, p_6\}$ and $\{p_3\}$
  - $\{p_1, p_2\}, \{p_3, p_4, p_5, p_6\}$
- Calculate distances
  - $dist(\{p_1, p_2\}, \{p_3, p_4, p_5, p_6\}) = 2.5$
- Merge into one cluster



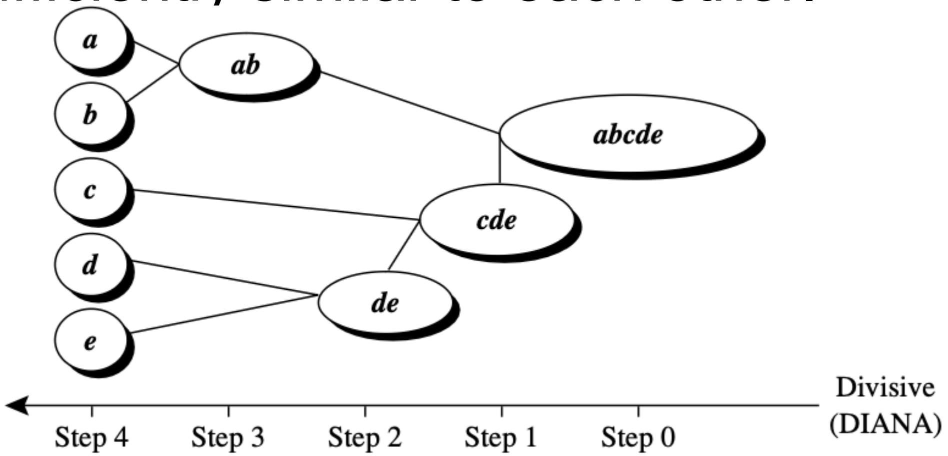
# Visualization Using **Dendrogram**

- A tree structure to represent the process of hierarchical clustering
  - $\{p_4\}$ and $\{p_6\}$ merges at distance 0.5
  - $\{p_1\}$ and $\{p_2\}$ merges at distance 0.7
  - $\{p_4, p_6\}$ and $\{p_5\}$ merges at distance 1.0
  - $\{p_4, p_5, p_6\}$ and $\{p_3\}$ merges at distance 1.4
  - $\{p_3, p_4, p_5, p_6\}$ and $\{p_1, p_2\}$ merges at distance 2.5



# Divisive Hierarchical Clustering

- Top-down strategy
  - Start by placing all objects in one cluster (the hierarchy's root)
  - Divide the root cluster into several smaller subclusters, and recursively partitions those clusters into smaller ones.
  - **Stop**: the partitioning process continues until each cluster at the lowest level is coherent enough –
    - *either* containing only one object
    - *or* the objects within a cluster are sufficiently similar to each other.



DBSCAN

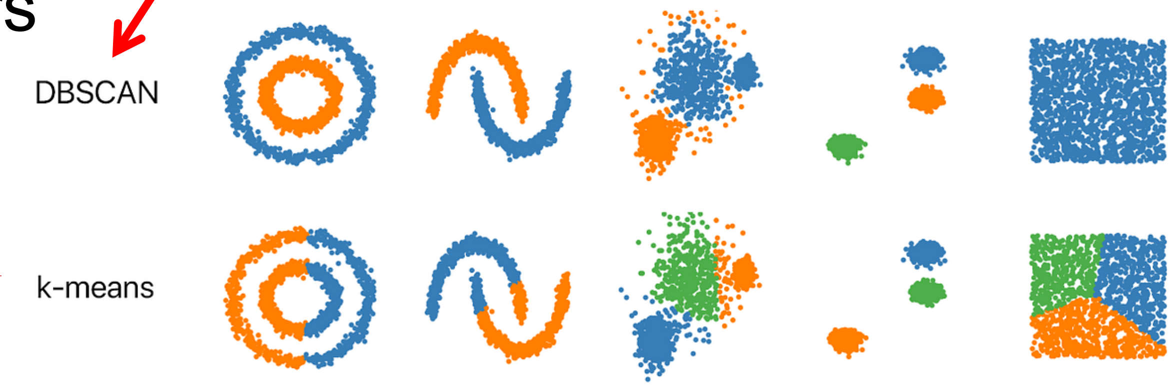
# DENSITY-BASED CLUSTERING

# Density-based Clustering

- **Idea**: Clusters are identified as **dense regions** separated by areas of lower density (sparse).
  - Group data points into clusters based on the density in a region.
  - Regions with high densities of points form clusters, while regions with low densities are treated as noise or outliers.

- **Advantages**:
  - No predefined number of clusters
  - Deal with arbitrary shapes
  - Robust to noises

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

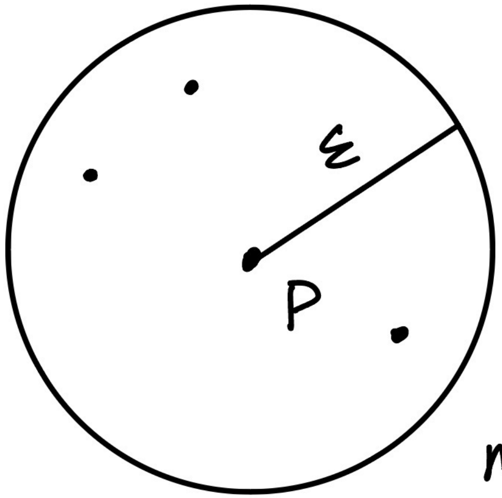


# DBSCAN – Basic Concepts

- **Input**: radius of neighborhood $\epsilon$, minimum points $minPts$
  - An object's $\epsilon$**-neighborhood**: $N_\epsilon(p) = \{o | dist(p, o) \leq \epsilon\}$
    - The maximum distance for points to be considered 'neighbors'
  - $p$ is a core object $\Leftrightarrow |N_\epsilon(p)| \geq minPts$
    - The minimum number of points required to form a dense region

**Three types of objects**:

- Core objects: points with at least *minPts* neighbors within $\epsilon$
- Border objects: points within $\epsilon$-neighborhood of a core object but with fewer than *minPts* neighbors
- Noise: points that are not within the $\epsilon$-neighborhood of any core object

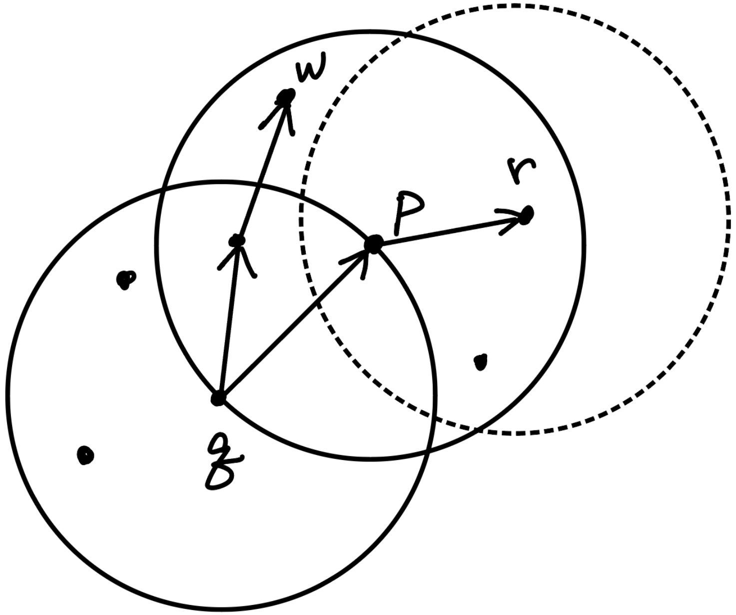


# DBSCAN – Density-Reachability

- $p$ is directly density-reachable from $q$ if
  1. $q$ is a core object
  2. $p$ is within $q$'s $\epsilon$-neighbourhood, i.e., $p \in N_\epsilon(q)$

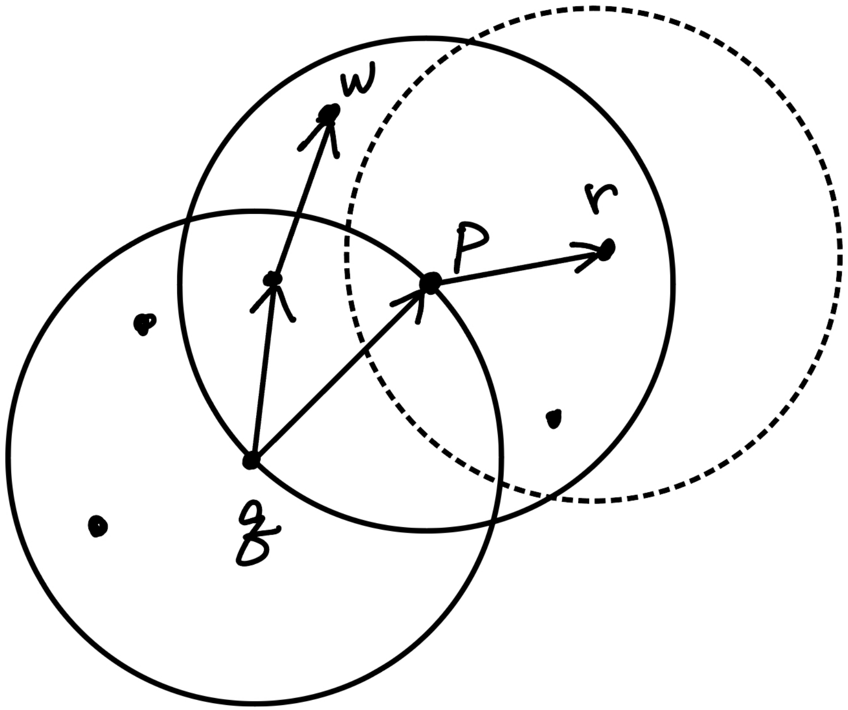
- $p$ is density-reachable from $q$ if there exists a chain of objects $p_1, p_2, \ldots, p_n$, s.t.,
  1. $p_1 = q, p_n = p$
  2. $p_{i+1}$ is directly density-reachable from $p_i$, for $1 \leq i < n$

***minPts* = 4**
- q has 4 neighbors
- p has 5 neighbors
- r has 2 neighbors



# DBSCAN – Density-Reachability (Properties)

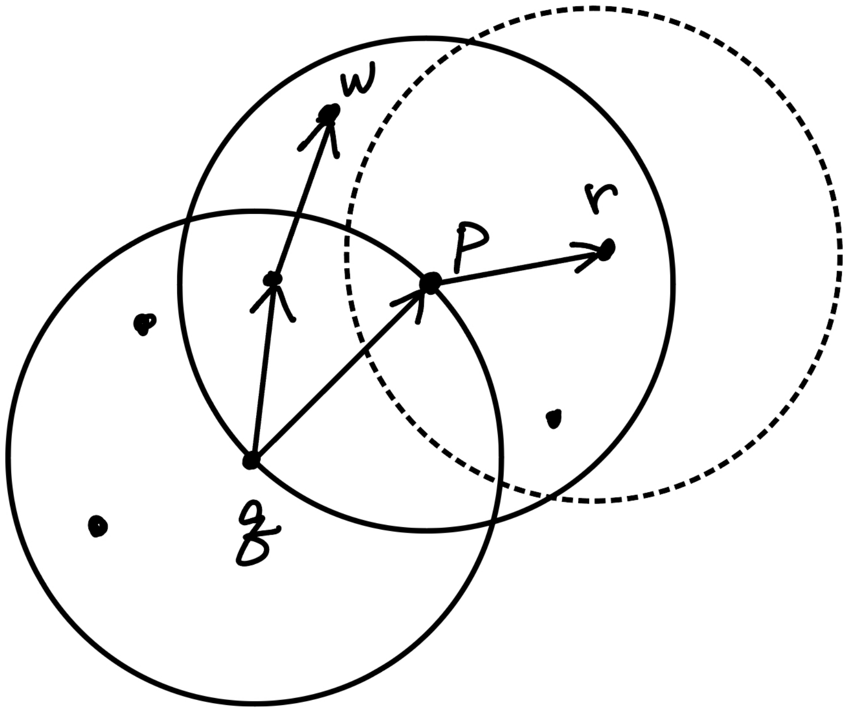
- Density-reachability is transitive
  - $q \rightarrow p$ and $p \rightarrow r$, then $q \rightarrow r$

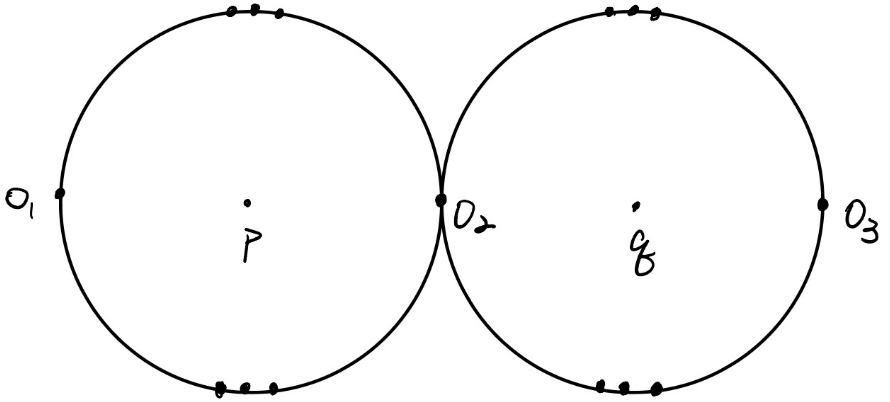



- Density-reachability is NOT symmetric
  - $p \rightarrow w$ does not mean $w \rightarrow p$
  - Density-reachability is symmetric for core objects

# DBSCAN – Density-Connectivity

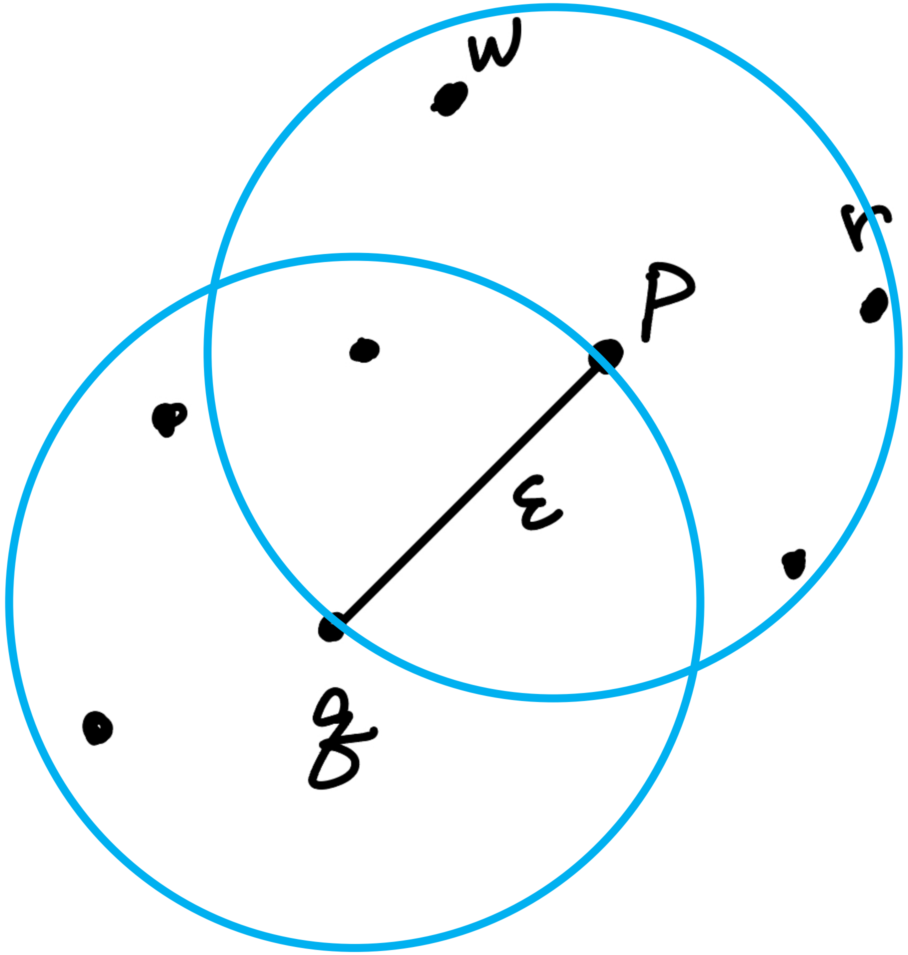
- $w$ and $r$ are density-connected if there exists a core object $p$, s.t.,
  - $p \rightarrow w$ and $p \rightarrow r$
- **Properties**
  - Density-connectivity is symmetric
    - If $w$ is density-connected to $r$, then $r$ is also density-connected to $w$.
  - Density-connectivity is NOT transitive
    - If $o_1$ and $o_2$ are density-connected, $o_2$ and $o_3$ are density-connected, it does not mean $o_1$ and $o_3$ are density-connected.



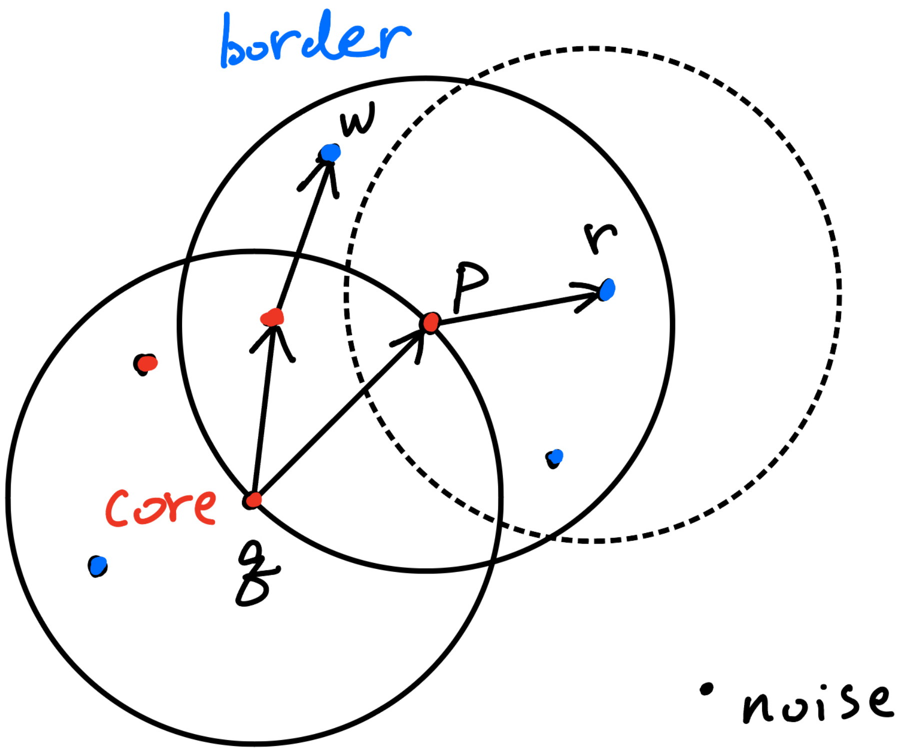
# Density-based Clusters

- A density-based cluster $C$ satisfies:
  - Connectivity: For any $p, q \in C$, $p$ and $q$ are density-connected
  - Maximality: For any $p, q$, if $q \in C$ and $p$ is density-reachable from $q$, then $p \in C$
    - If a core object $p \in C$, then all objects density-reachable from $p$ belong to $C$



# Density-based Clusters

- **Core objects**
- **Border objects**: $p$ belongs to a cluster $C$, but $p$ is not a core object
  - A border object can belong to multiple clusters (hub or bridge)
- **Noises**: objects belong to no clusters

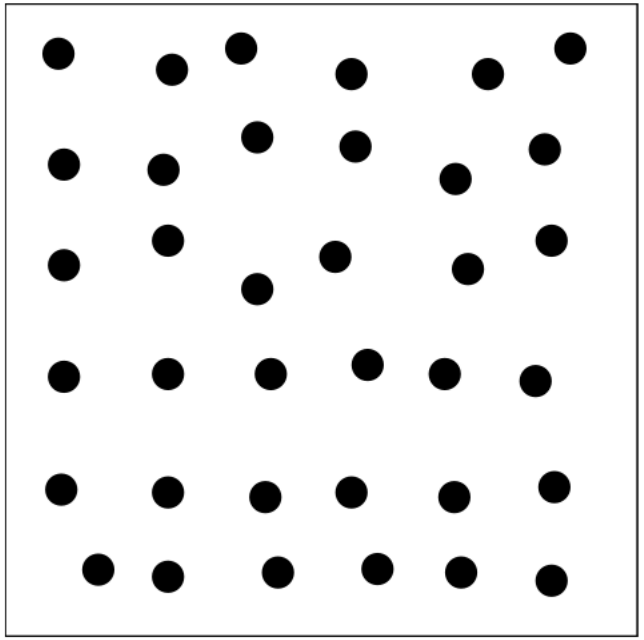


quality, stability, tendency, …

# EVALUATION OF CLUSTERING

# Major Tasks in Clustering Evaluation

- **Clustering Quality**
  - To evaluate the goodness of the clustering, e.g., how well the clusters fit the data set, or how well the clusters match the "*ground truth*" (if available)
- **Clustering Stability**
  - To understand the sensitivity of the clustering result to parameters, e.g., # clusters $K$
- **Clustering Tendency**
  - To assess the suitability of clustering, i.e., whether the data has any inherent *non-random* grouping structure – meaningful clusters



A data set that is uniformly distributed in the data space.

# Measuring Clustering Quality

- **Evaluation**: Evaluating the goodness of clustering results
  - No universally recognized 'best' measure in practice!
- **Three categorization of measures**
  - **Internal**: Unsupervised, criteria derived from data itself
    - How well the clusters are separated and how compact the clusters are
  - **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 measures
  - **Relative**: Directly compare different clustering, usually those obtained by varying parameters for the same algorithm

# Measuring Clustering Quality: External Methods

- Given the ground truth $T$, compare the clustering result ($C$) with the ground truth using a quality measure $Q(C, T)$. It is considered good if it satisfies the following essential criteria:
  - **Cluster homogeneity**: the purer clusters, the better clustering
  - **Cluster completeness**: objects belonging to the same category in the ground truth should be assigned to the same cluster
  - **Small cluster preservation**: Splitting a small category into pieces is more harmful than splitting a large category into pieces
  - **Rag bag**: putting a heterogeneous object into a pure cluster is worse than putting it into a rag bag (i.e., “other” class)

# Summary

- **Cluster**: a collection of data objects that are similar to one another within the same cluster and are dissimilar to the objects in others.
- **Clustering**: the unsupervised process of grouping objects
  - As a standalone data mining tool to gain insight into data distribution
  - As a preprocessing step for other data mining algorithms
- **Partitioning-based**: k-means, k-medoids, k-medians, k-modes, ...
- **Hierarchical-based**: either agglomerative (bottom-up) or divisive (top-down), based on how the hierarchical decomposition is formed
- **Density-based**: DBSCAN, arbitrary shapes, robust to noise
- **Clustering evaluation**: quality, stability, tendency – subjective!

Email: fengmei.jin@polyu.edu.hk

Office: PQ747

# THANK YOU!