Nile University
جامعة النيل

School of
Information Technology
and Computer Science

Hidden Structures,
Unseen Insights

Unsupervised Machine Learning: a Foundation.

By: Noha Gamal El-Din,
Assistant Prof. School of ITCS, NU.
ngamal@nu.edu.eg

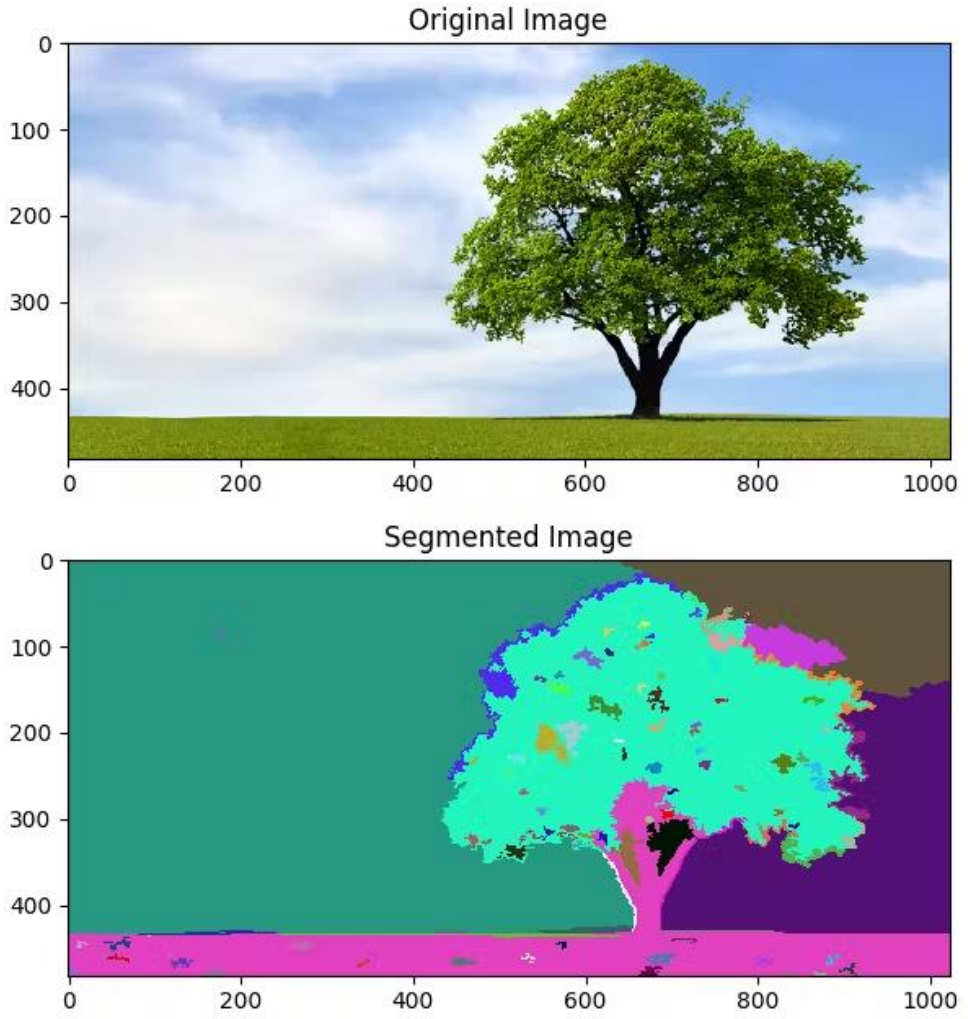
My De-code

- PhD in Computer Systems, Ain Shams University.
- Master’s in Information Security, Nile University.
- Bachelor’s in Communications Engineering, Alexandria University.
- Specialized in AI, Information Systems, Networks, Security systems.

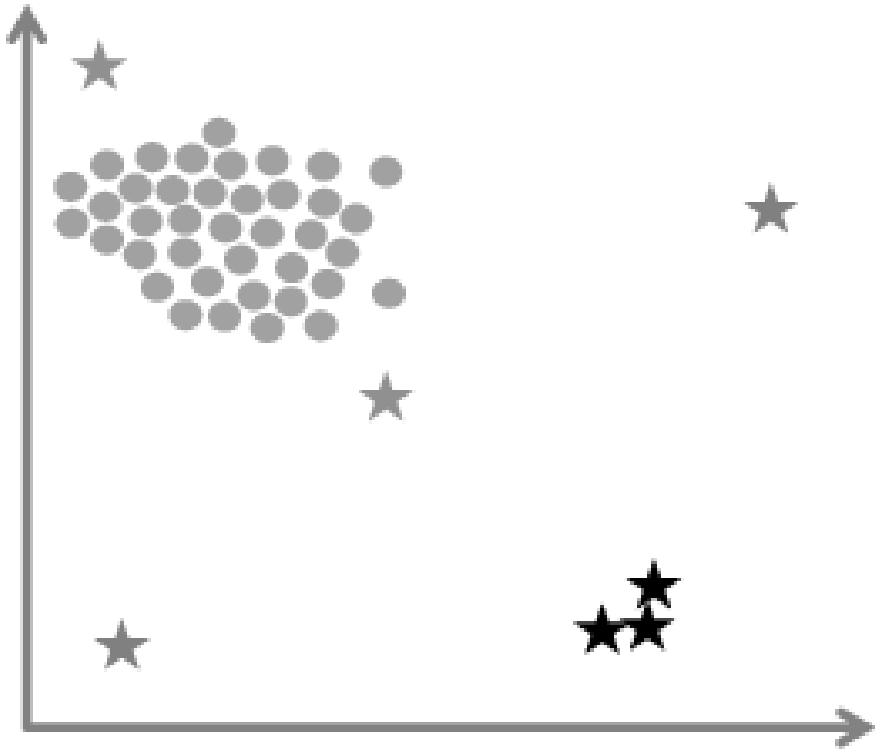
Nile University

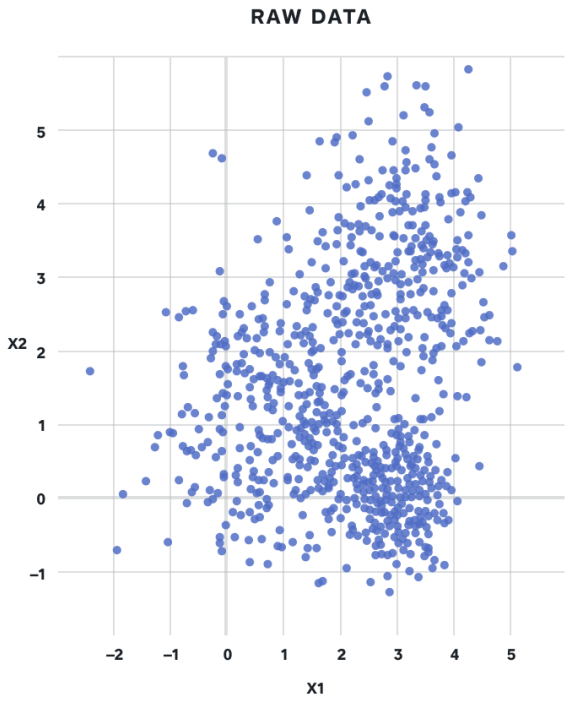
What is going on here?

“As a human, If you had a dataset with no labels, could you still make sense of it?”

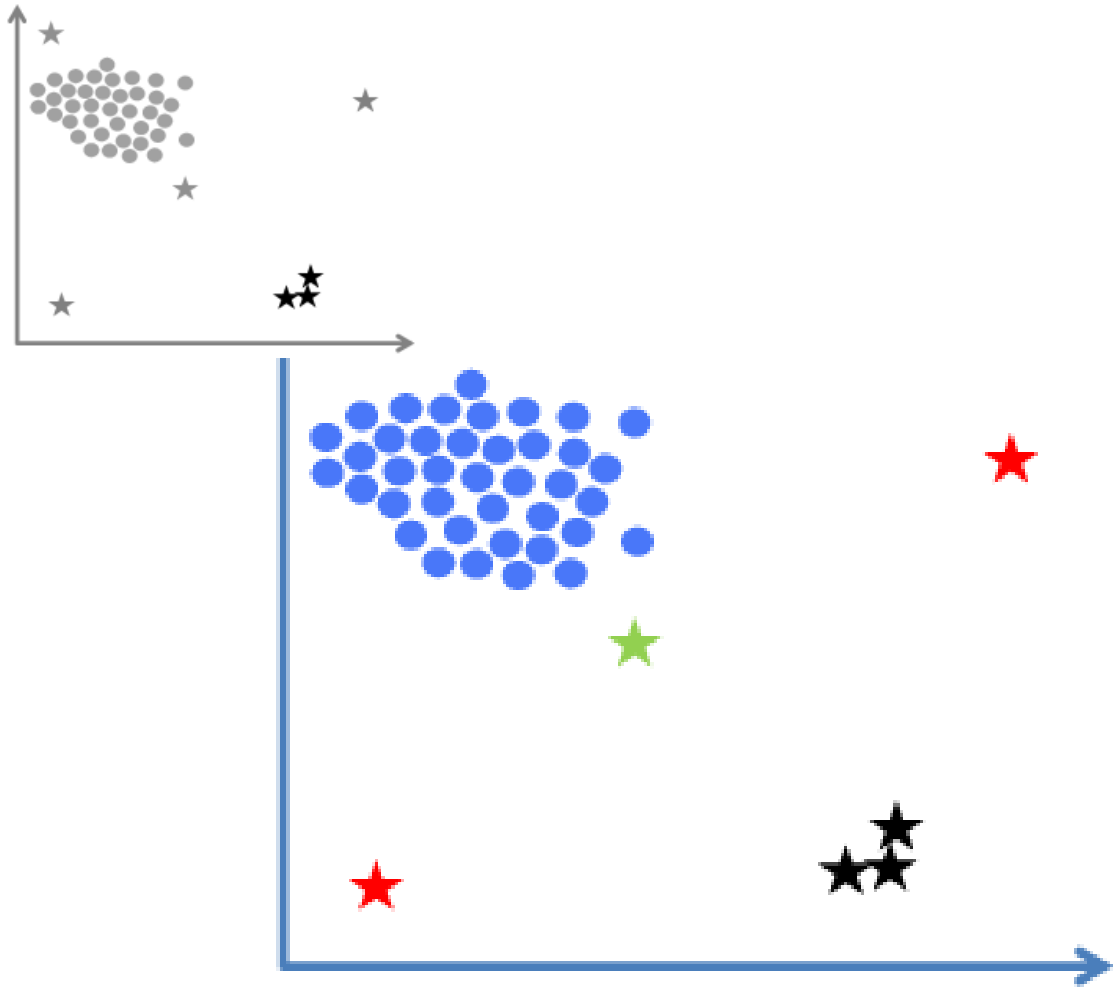


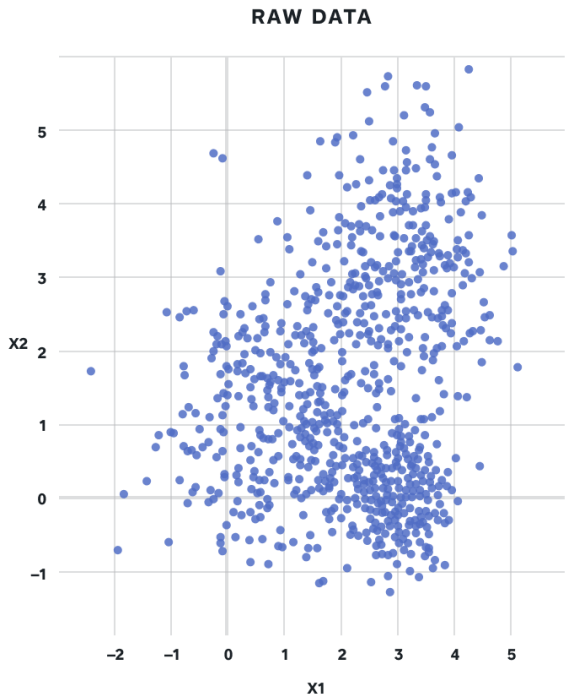
“As a human If you had a dataset with no labels, could you still make sense of it?”

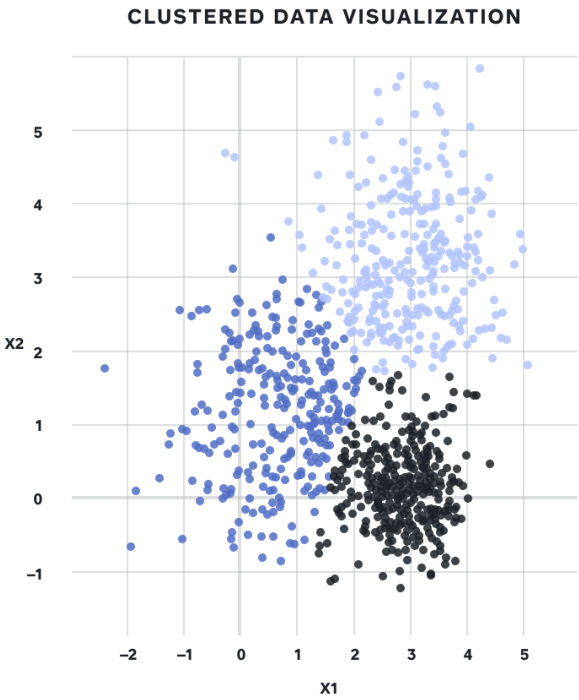




“As a human If you had a dataset with no labels, could you still make sense of it?”

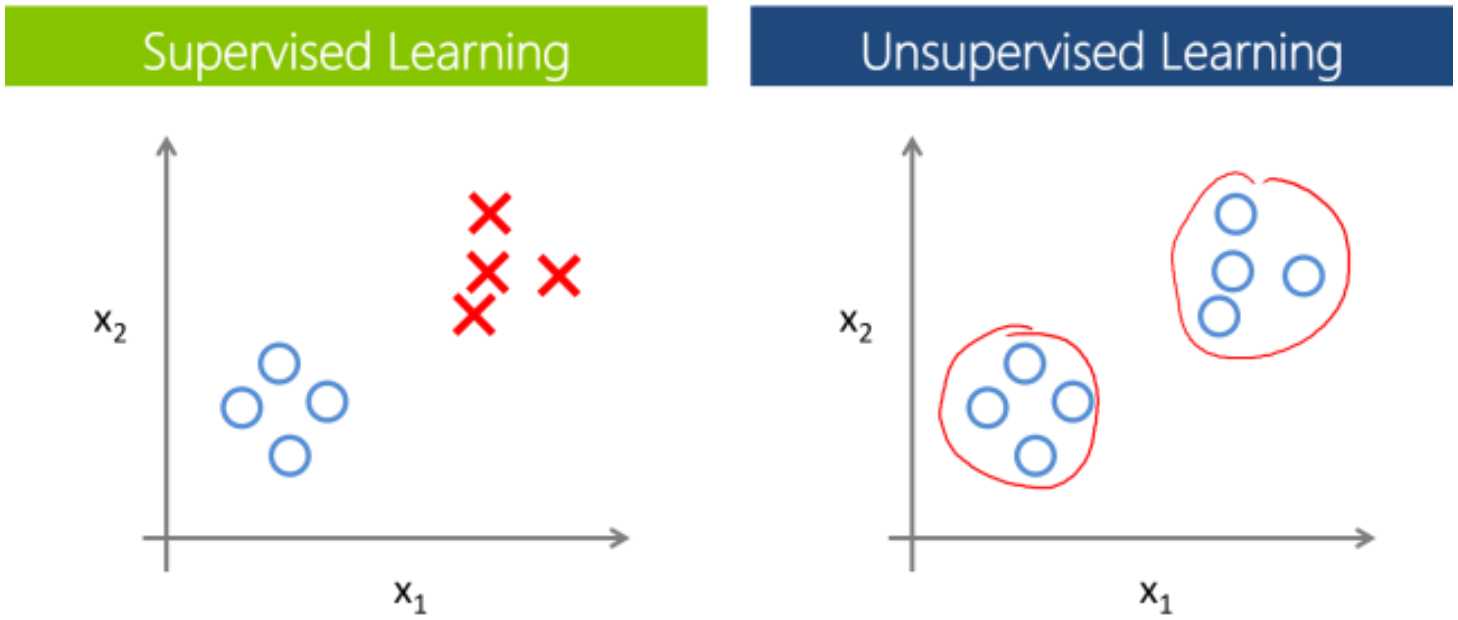






So, our objective here, is to make the machine learn from patterns not labels like humans can do, which is unsupervised ML.

Unsupervised Learning



Definition: Learning from unlabeled datasets — the model **finds patterns and structure without explicit instructions**.

Goal: Discover hidden patterns, groupings, or structure in data.

Outline

Unsupervised Vs. Supervised Learning

Why Unsupervised Learning

Types of Unsupervised Learning

Clustering

Association

Anomaly Detection

Data Representation

Dimensionality reduction

Cluster Analysis

What is cluster analysis

How to find patterns in data

Types of Cluster Analysis

Partitional (K-Means)

Hierarchical (Agglomerative)

Density-based (DB-Scan)

Why Unsupervised Learning?

- Can you tell the market size of data annotation world wide?

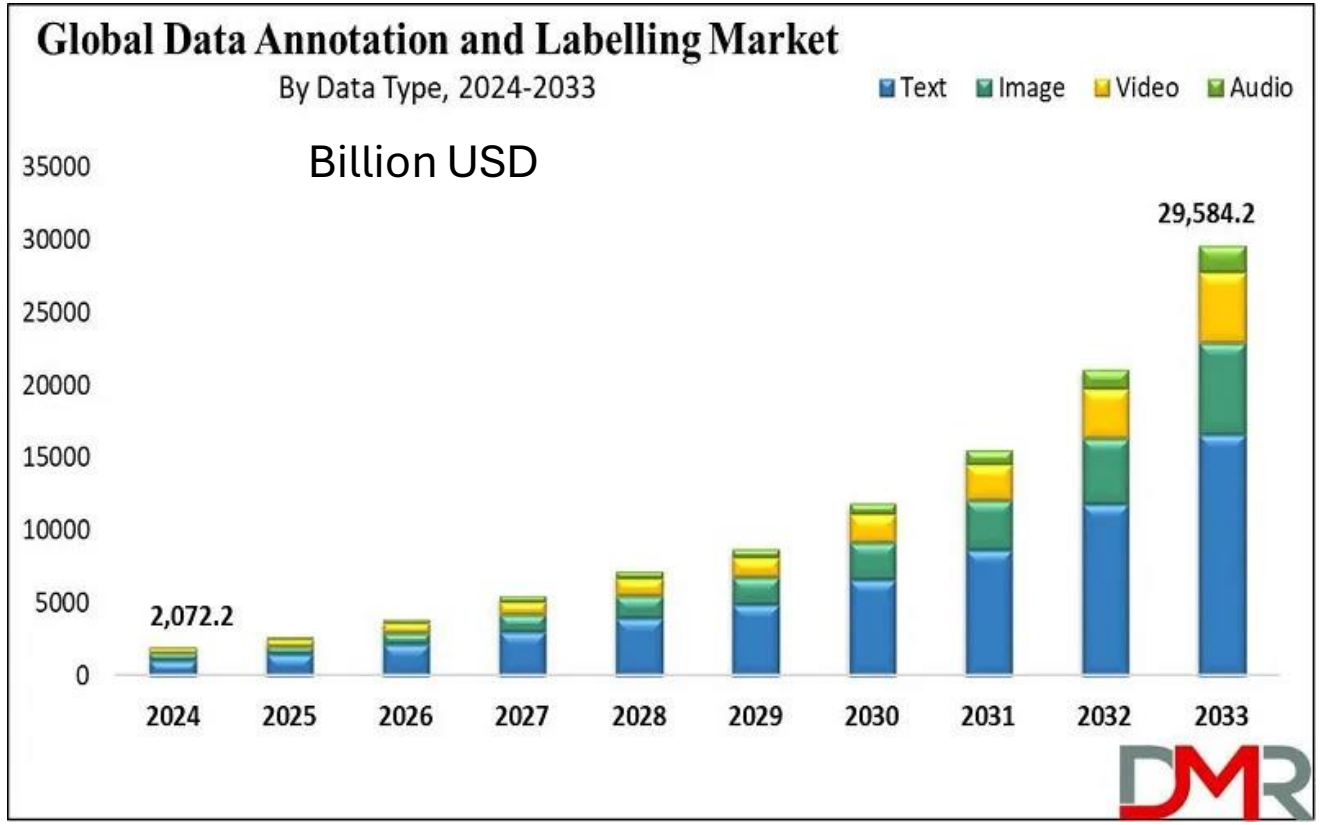
Why Unsupervised Learning?

Data Reality

- **Vast majority of data is unlabeled** → images, logs, sensor readings, text, clicks.
- **Labeling is expensive** → requires time, human experts, or domain specialists.

Unsupervised Learning Adaptability to the reality of data

Works when labels are **unavailable** or **impractical** to obtain.



Why Unsupervised Learning?

2. Discovery of Hidden Structure

- Reveals natural groupings

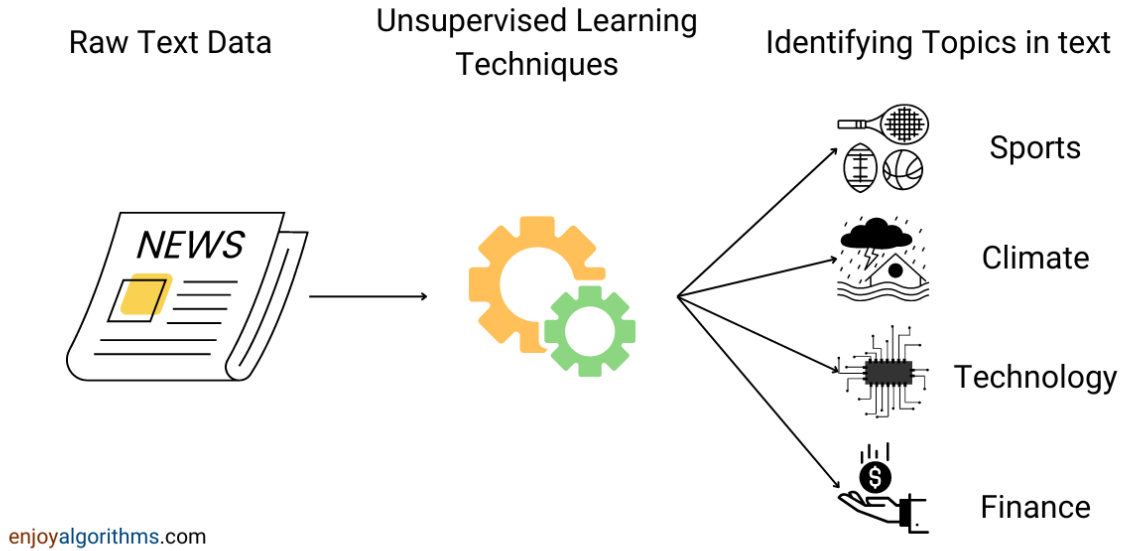
(e.g., customer segments, species types, behavior clusters).

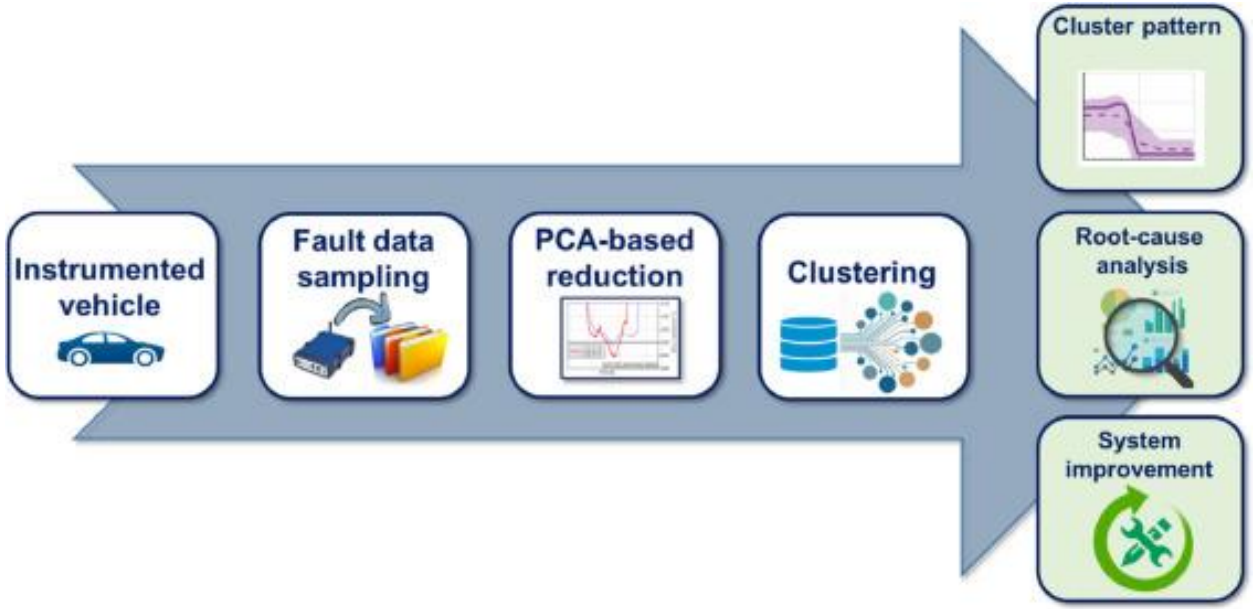
- Finds correlations and patterns not visible to the human eye.
- Supports exploratory analysis before committing resources to labeling or hypothesis testing.

Why Unsupervised Learning?

3. Preprocessing for Supervised Learning

- **Latent Dirichlet Allocation (LDA)** – *Unsupervised topic modeling*
- Feature extraction and Dimensionality reduction (e.g., via PCA, embeddings).
- **→ faster training & less overfitting.**





Why Unsupervised Learning?

4. Real-Time & Dynamic Applications

- ***Incremental clustering algorithms***

    - Unlike traditional batch clustering (like classic K-Means), *incremental* algorithms update clusters as new data arrives without retraining from scratch.

    - These models Can form *new* clusters on the fly if patterns change

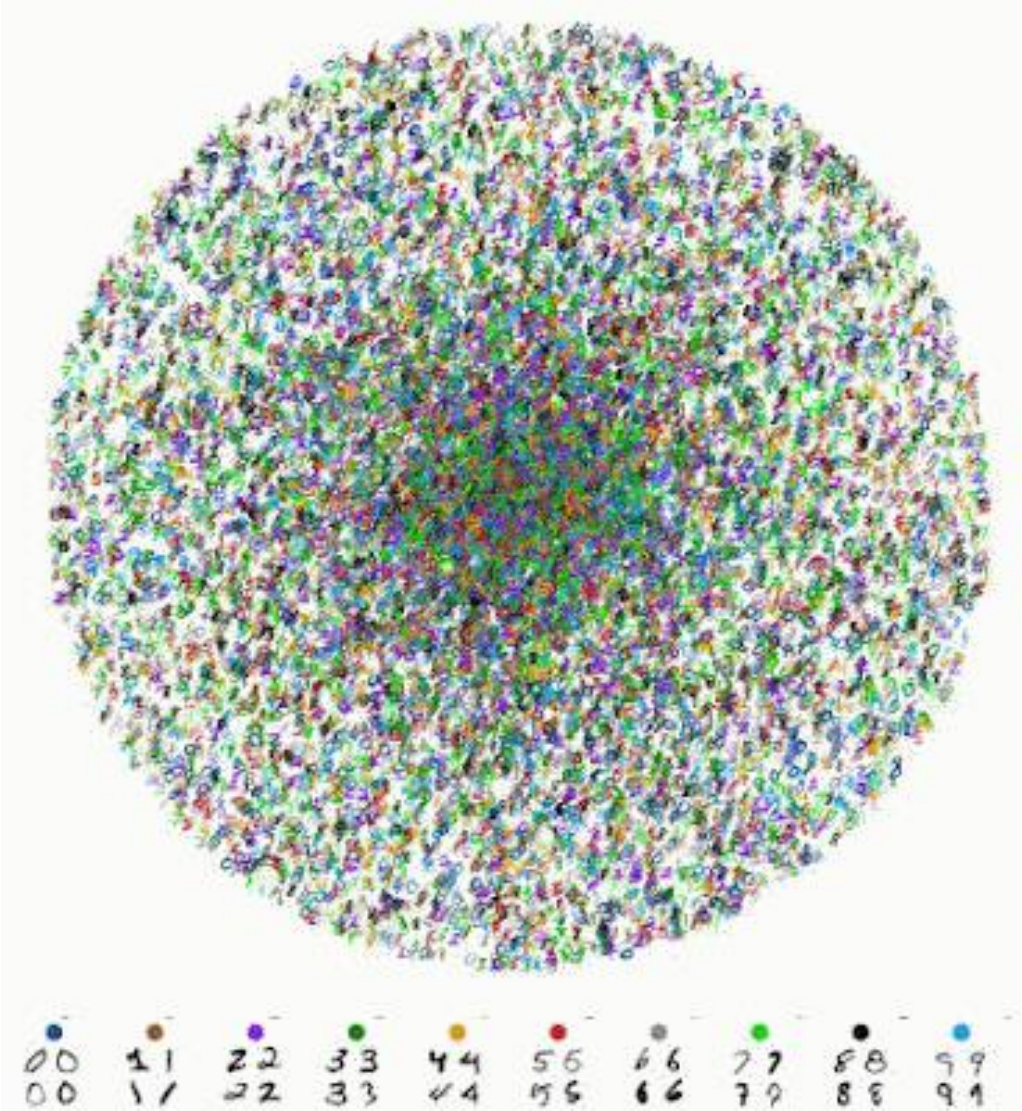
Ex. **Streaming K-Means** – keeps centroids updated as the stream flows.

- ***Low-latency anomaly detection***
    - Clustering models can flag points far from all clusters as anomalies in real time (used in fraud detection, intrusion detection).

Incremental clustering => *"Where does this new point fit among evolving, unlabeled groups?"*

2

Incremental clustering = not only answers the question but also update clusters based on the new data.



So, to cut story short,

Advantages of Unsupervised Learning

1.No labels required

Advantages of Unsupervised Learning

1. No labels required
2. Data Exploration

Advantages of Unsupervised Learning

1. No labels required
2. Data Exploration
3. Feature Discovery

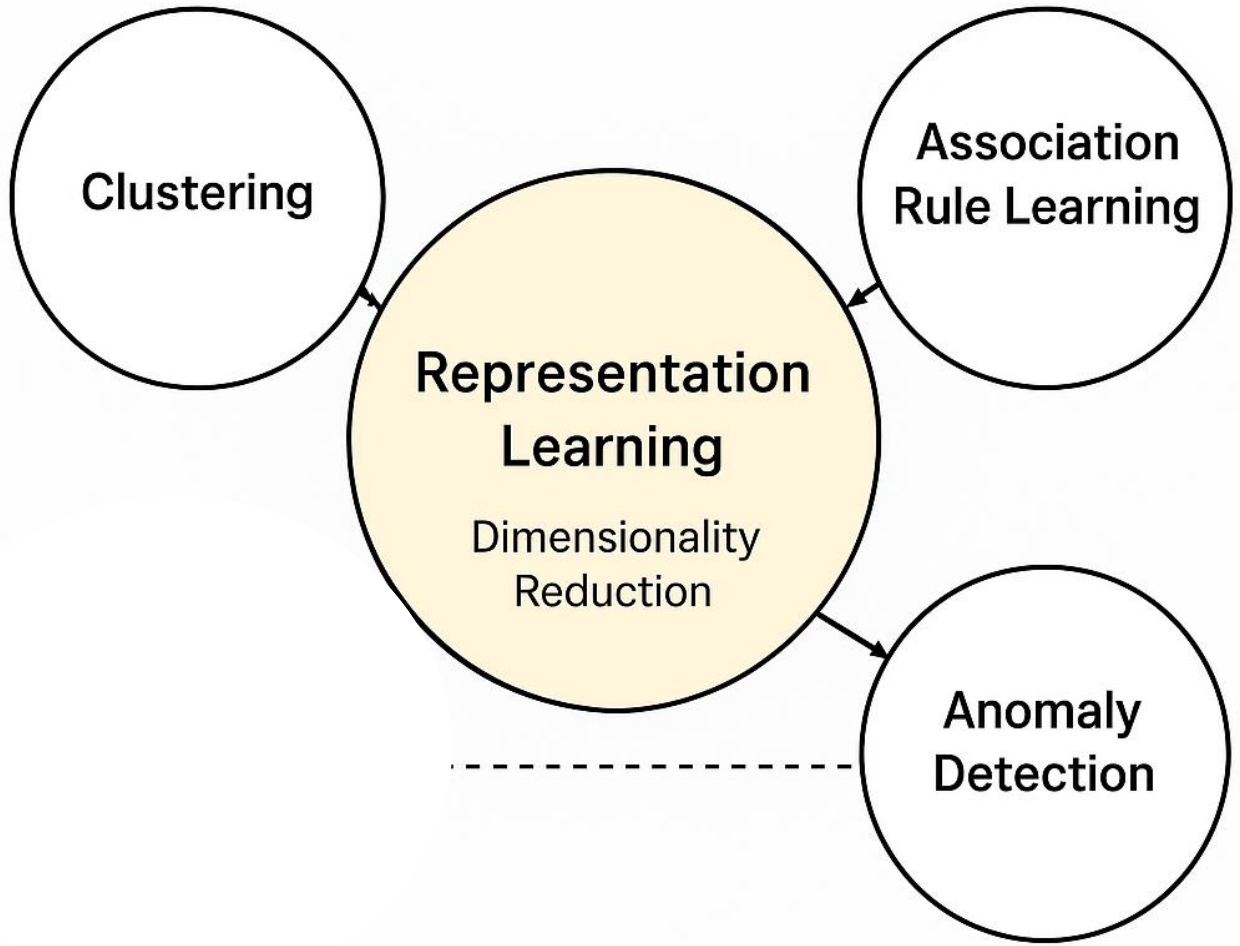
Advantages of Unsupervised Learning

1. No labels required
2. Data Exploration
3. Feature Discovery
4. Anomaly Detection

Advantages of Unsupervised Learning

1. No labels required
2. Data Exploration
3. Feature Discovery
4. Anomaly Detection
5. Reduced Bias

Types of Unsupervised Learning



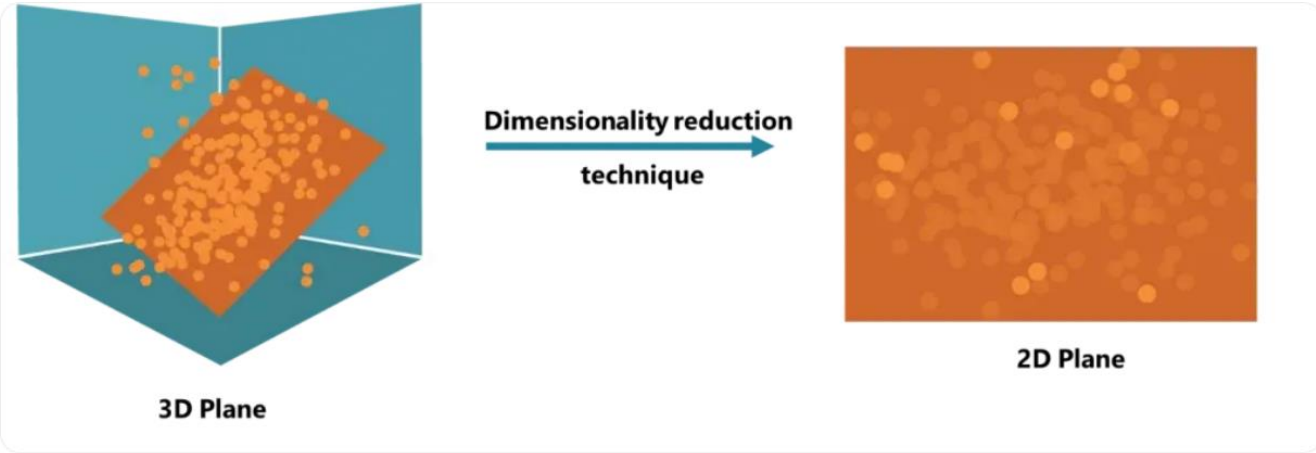
Representation Learning

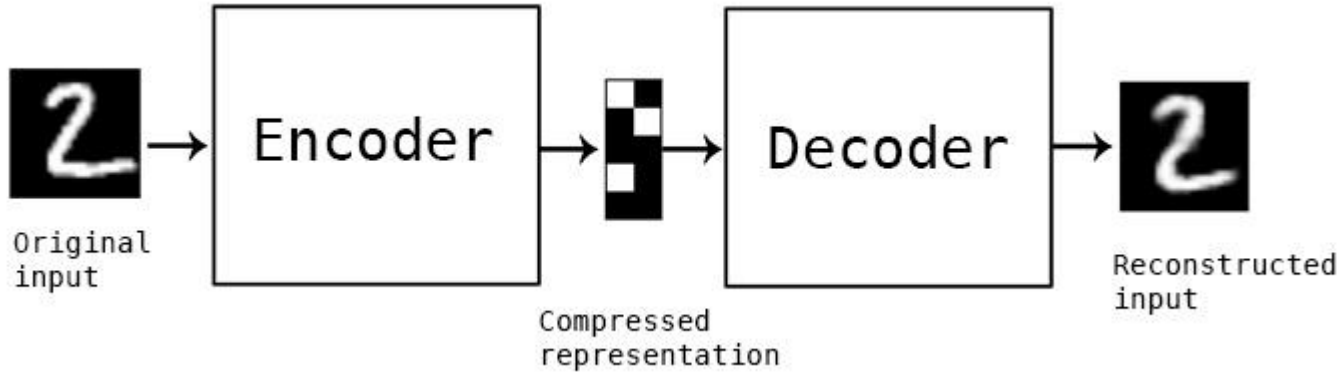
Learns meaningful feature representations from raw data.

Dimensionality Reduction: ex. PCA, t-SNE, UMAP.

Feature Learning: ex. Autoencoders

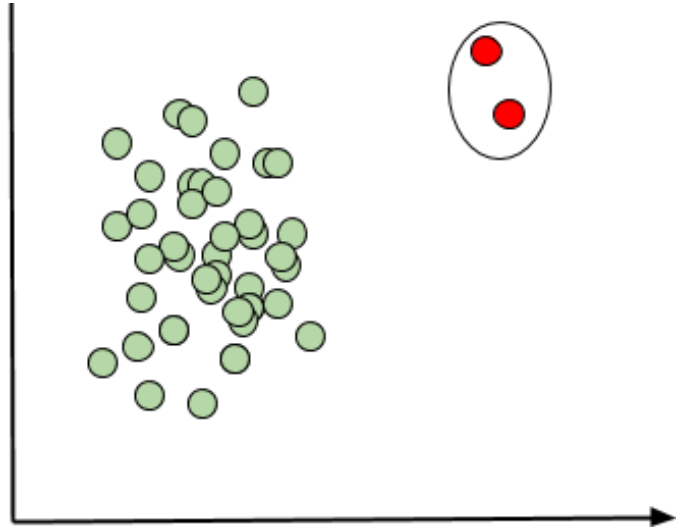
Example Applications: Data visualization, compression, preprocessing.

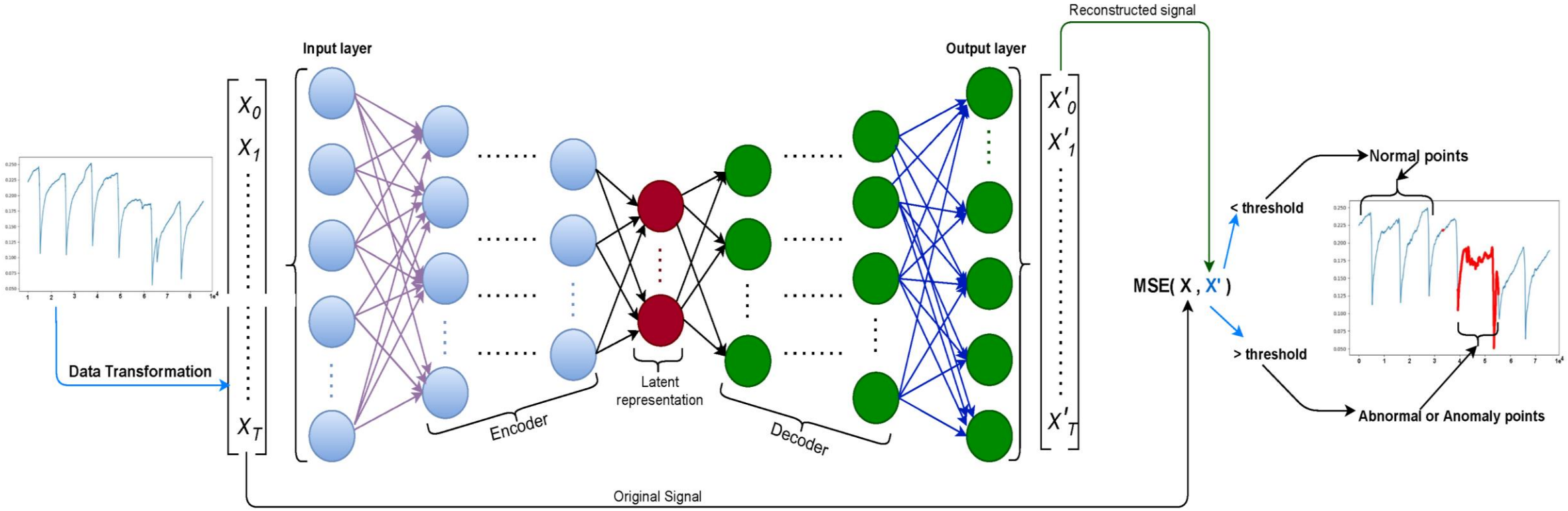




Anomaly Detection

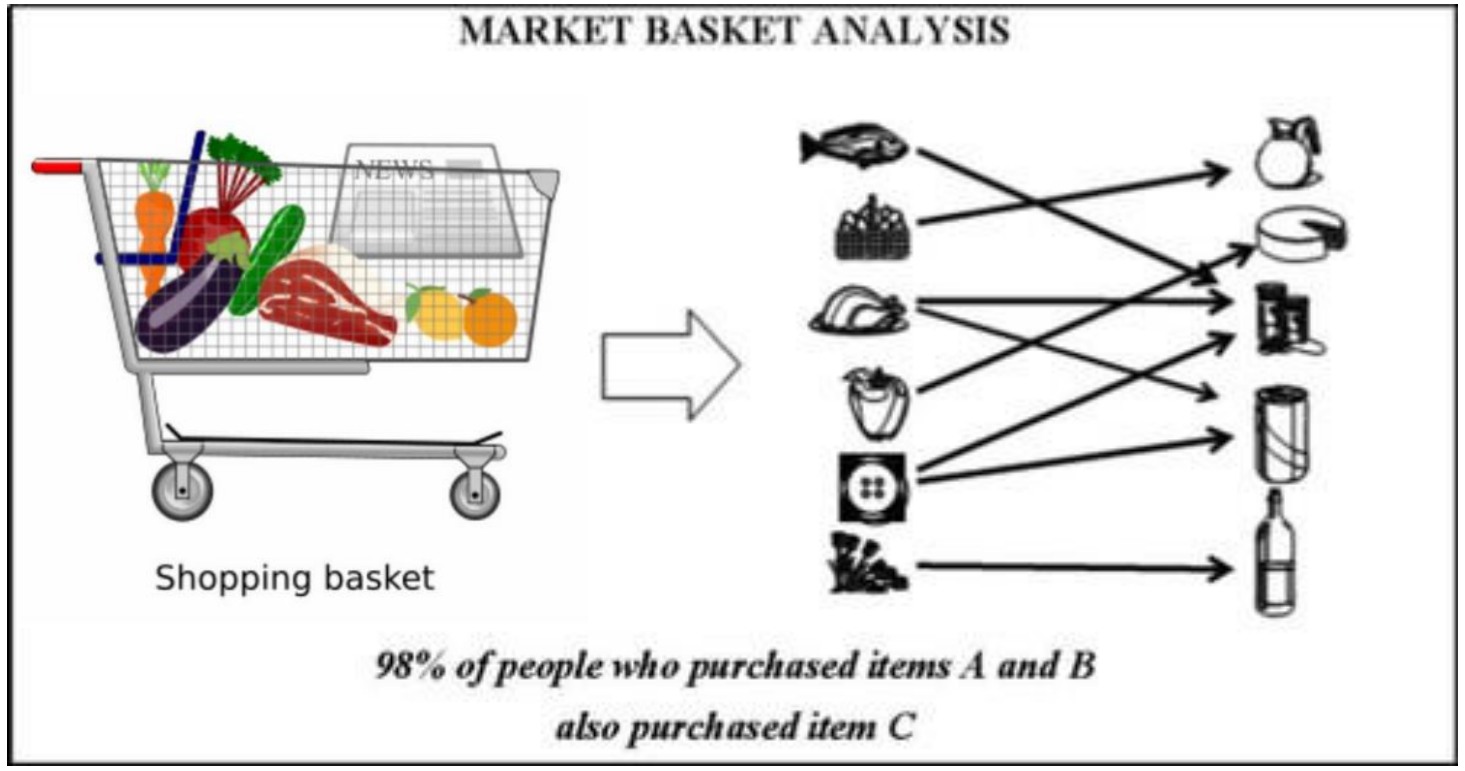
- Identifies data points that deviate from the norm.
- **Example Applications:** Fraud detection, network intrusion detection.





Rule-based Association Learning

- Finds frequent co-occurrence patterns or rules.
- **Methods:** Apriori, FP-Growth
- **Support and Confidence** are two important metrics used with association rule learning
- **Applications:** Market basket analysis, recommendation systems.



$$Rule\ X \Longrightarrow Y$$

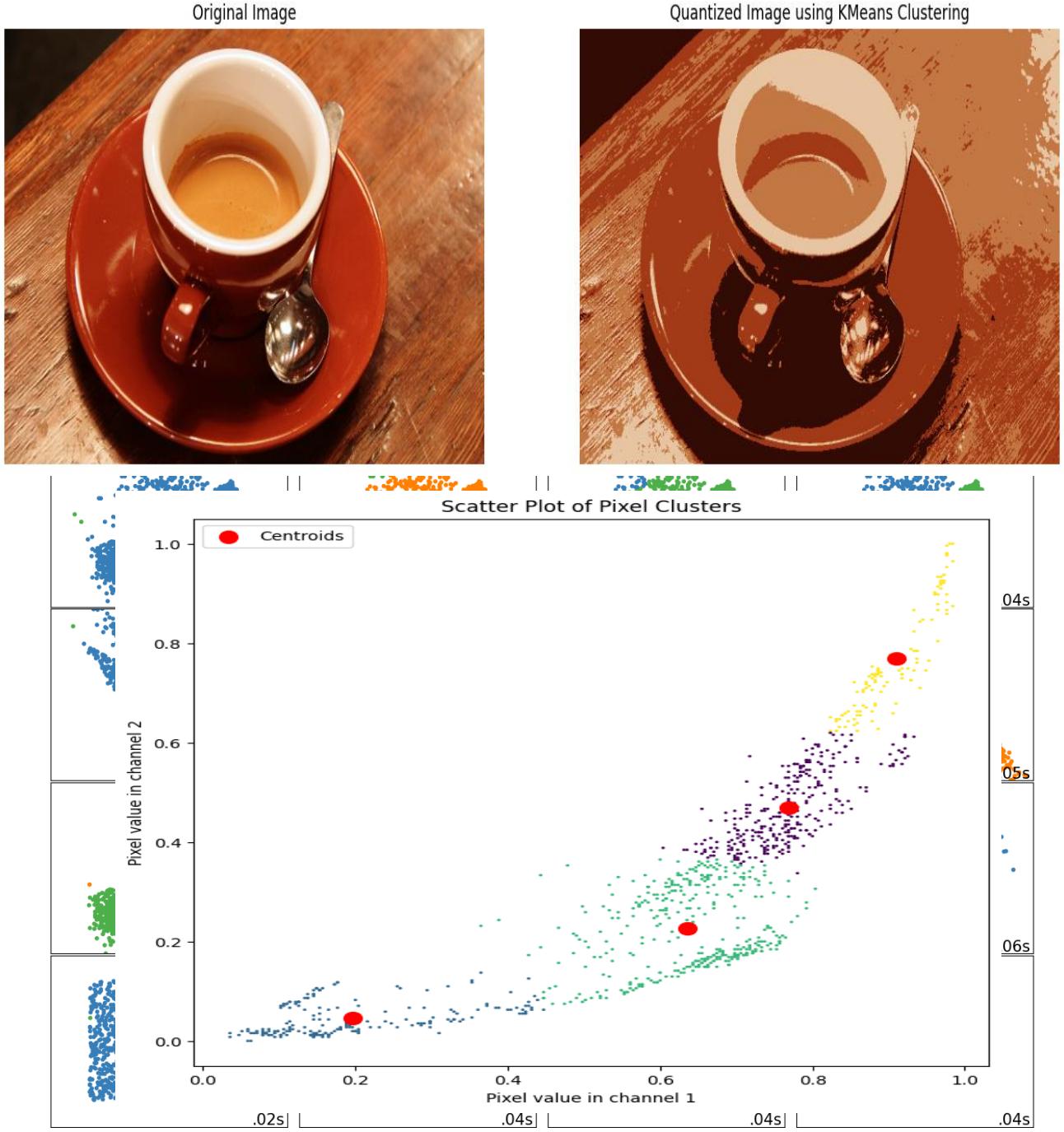
$$Support = \frac{Frequency\ (X,Y)}{N}$$

$$Confidence = \frac{Frequency\ (X,Y)}{Frequency(X)}$$

$$Lift = \frac{Support}{Support(X) * Support(Y)}$$

Clustering/ Cluster Analysis

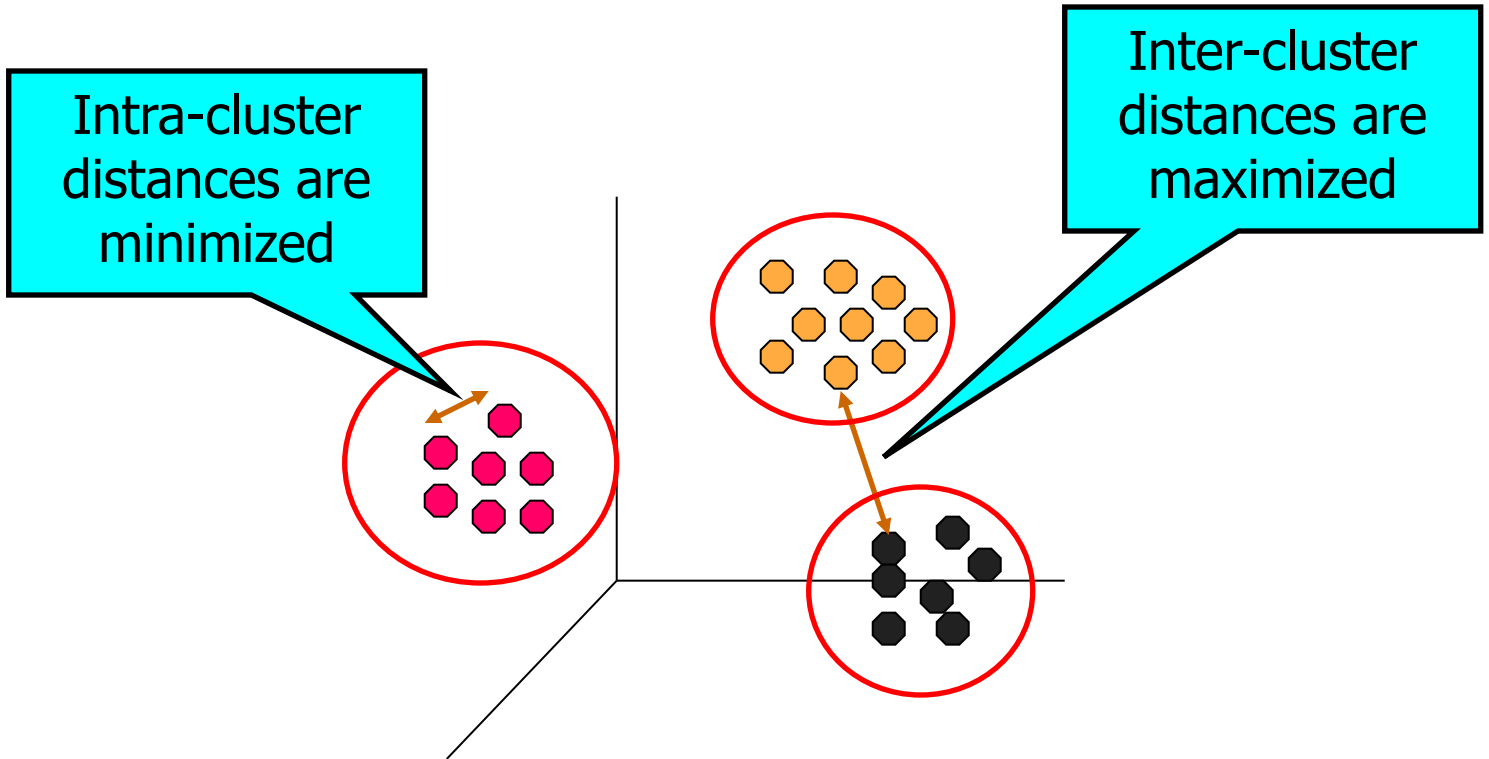
- Clustering: Groups similar data points without labels.
- Methods:
  - Partitional → K-Means, Mini-Batch K-Means
  - Hierarchical → Agglomerative, Divisive
  - Density-based → DBSCAN, OPTICS
- Applications: Customer grouping, image segmentation.



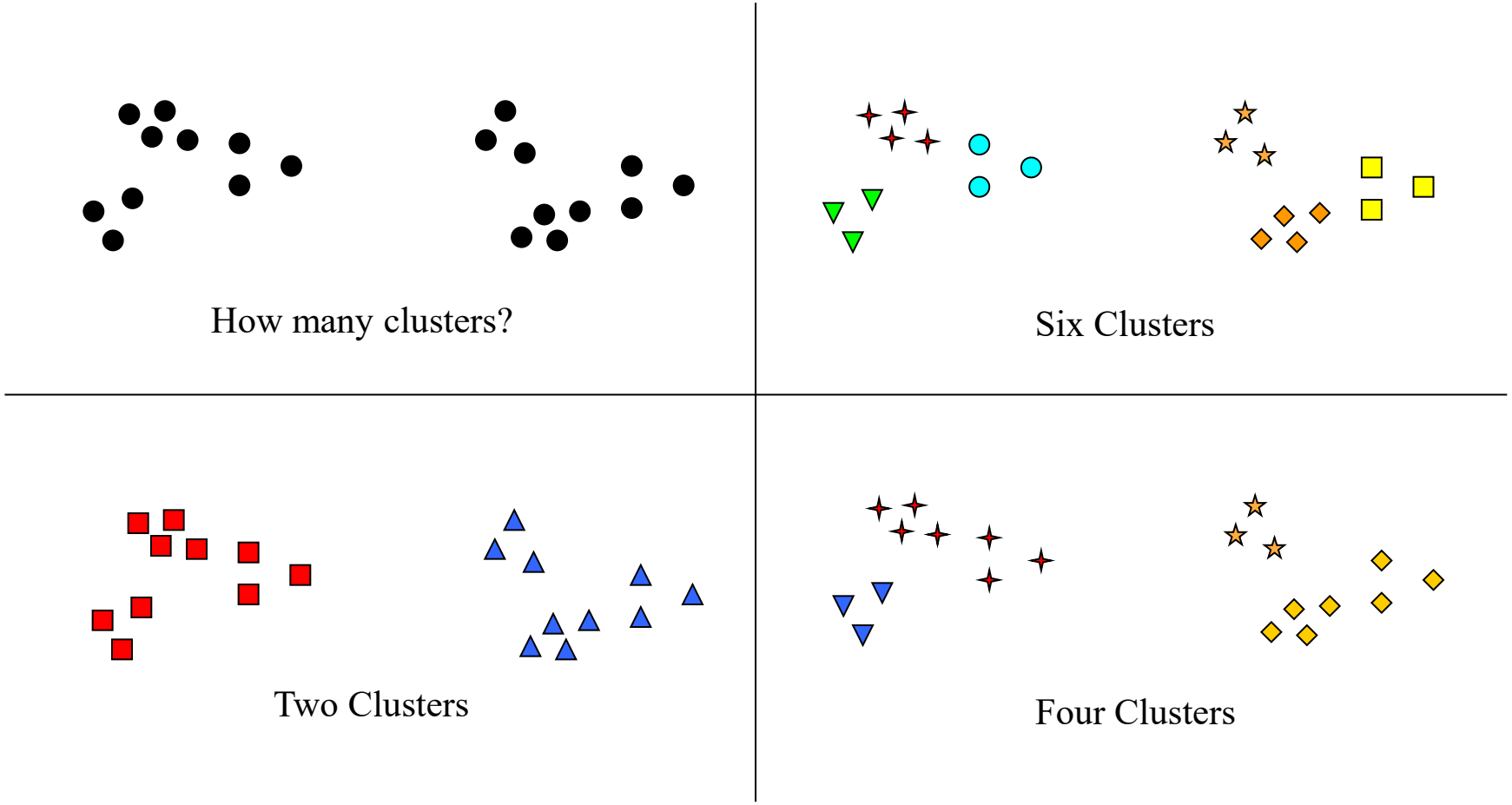
What is Cluster Analysis?

- **Finding groups of objects** such that the **objects in a group will be similar** (or related) to one another and **different from (or unrelated to) the objects in other groups without prior training**.
- **Clustering is a method used in unsupervised learning to:**

find these natural groupings in data.

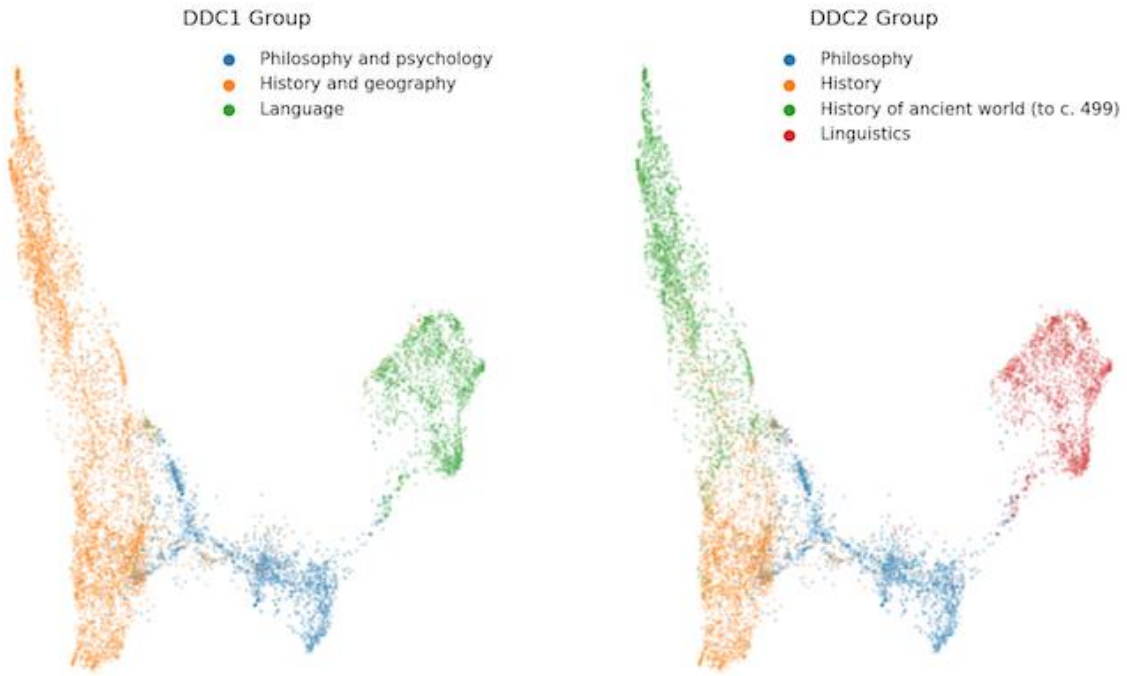


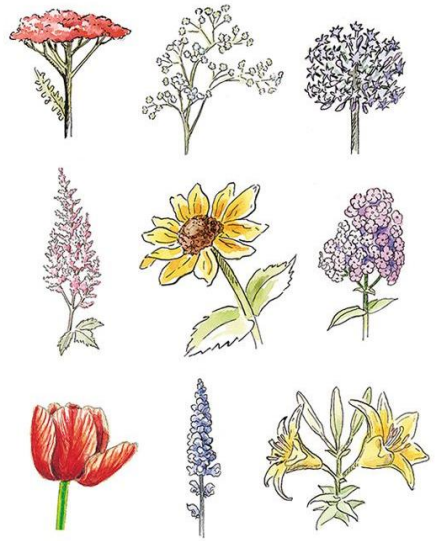
Notion of a Cluster can be Ambiguous



Notion of a Cluster can be Ambiguous

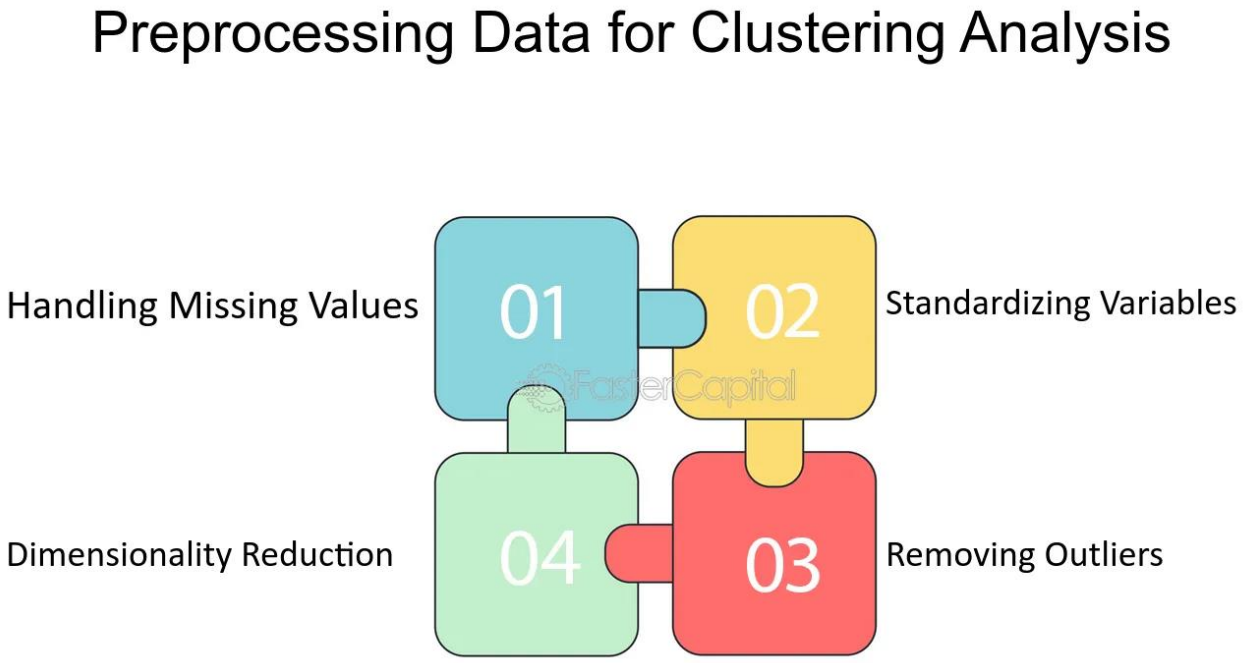
- **No Single "Correct" Answer:** No ground truth labels. Different algorithms or parameter settings can produce different groupings on the same dataset.
- **Sensitive to Distance Measures:** Choice of similarity metric (Euclidean, cosine, Manhattan) can change cluster shapes and boundaries.
- **Sensitive to Feature Selection & Scaling:** Including or excluding certain features can radically change the grouping. Features with larger numeric ranges can dominate unless scaled.
- **Overlap Between Clusters:** Real-world data often has fuzzy boundaries — a point could plausibly belong to more than one cluster
- **Different Algorithm Biases**: K-Means favors spherical clusters. DBSCAN favors dense regions and can mark points as noise. Hierarchical methods may merge or split clusters differently based on linkage criteria.





How to Find Patterns in Data?

- **Step 1 – Represent the Data**
- Choose relevant features (attributes) that capture important characteristics.
- Preprocess:
    - **Scaling** (e.g., StandardScaler, MinMaxScaler) to avoid feature dominance.
    - **Encoding** categorical variables.



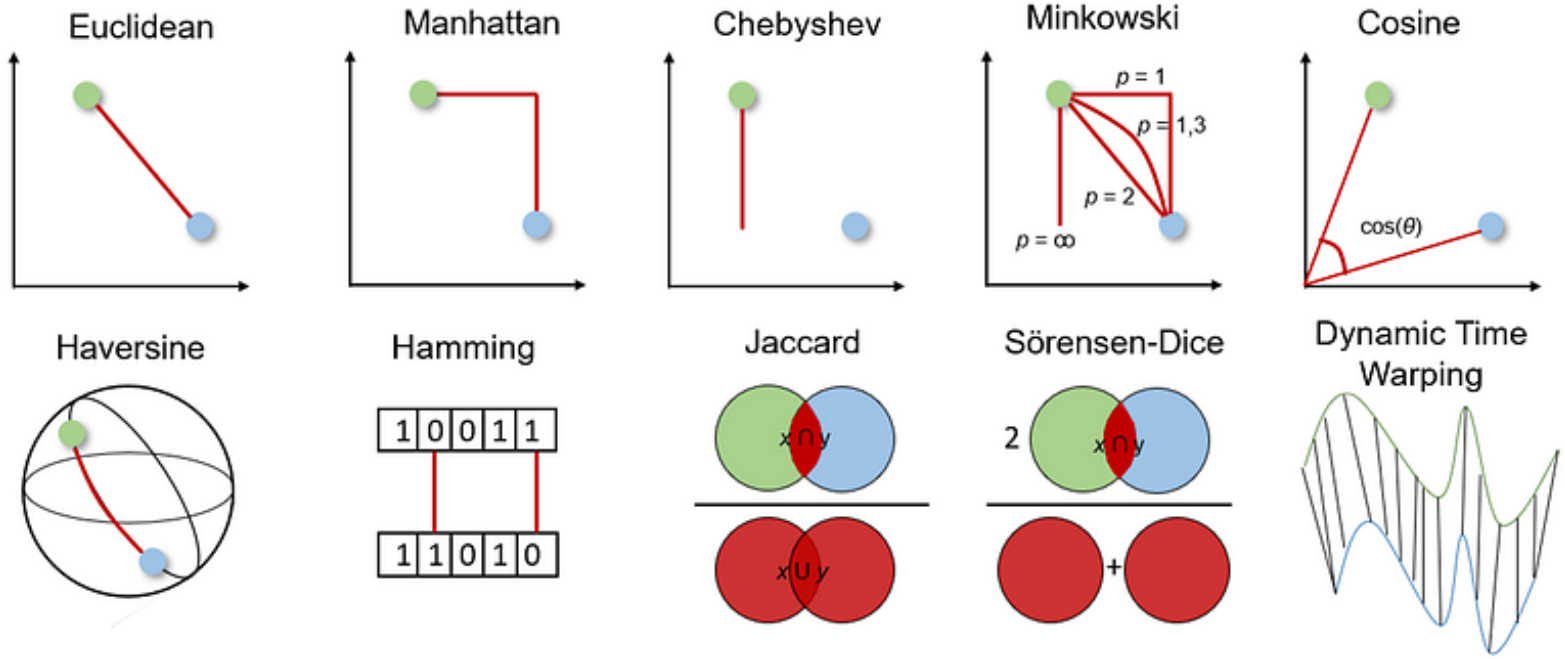
How to Find Patterns in Data?

Step 2 – Choose a Similarity or Distance Measure
The “pattern” in clustering is defined by how we measure closeness.

Measure	Best For	Example Use
Euclidean Distance	Continuous numeric data, spherical clusters	K-Means on physical measurements
Manhattan Distance	Grid-like movement, high-dimensional data	Pathfinding, urban traffic data
Cosine Similarity	Text, directional similarity (magnitude-less)	Document clustering, recommender systems
Jaccard Index	Binary attributes or sets	Market basket analysis
Hamming Distance	Categorical strings of equal length	DNA sequence clustering

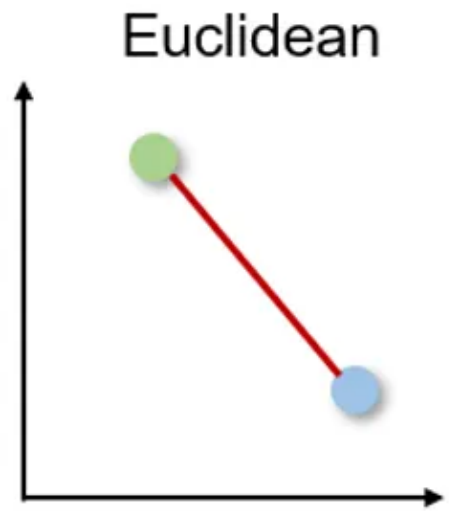
How to Find Patterns in Data?

Euclidean	$d(x,y) = \sqrt{\sum (x_i - y_i)^2}$
Squared Euclidean	$d(x,y) = \sum (x_i - y_i)^2$
Manhattan	$d(x,y) = \sum \lvert (x_i - y_i) \rvert$
Canberra	$d(x,y) = \sum \frac{\lvert x_i - y_i \rvert}{\lvert x_i + y_i \rvert}$
Chebychev	$d(x,y) = \max(\lvert x_i - y_i \rvert)$
Bray Curtis	$d(x,y) = \frac{\sum \lvert x_i - y_i \rvert}{\sum x_i + y_i}$
Cosine Correlation	$d(x,y) = \frac{\sum (x_i y_i)}{\sqrt{\sum (x_i)^2 \sum (y_i)^2}}$
Pearson Correlation	$d(x,y) = \frac{\sum (x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum (y_i - \overline{y})^2} \sqrt{\sum (y_i - \overline{y})^2}}$
Uncentered Peason Correlation	$d(x,y) = \frac{\sum x_i y_i}{\sqrt{\sum (y_i - \overline{y})^2} \sqrt{\sum (y_i - \overline{y})^2}}$
Euclidean Nullweighted	Same as Euclidean, but only the indexes where both x and y have a value (not NULL) are used, and the result is weighted by the number of values calculated. Nulls must be replaced by the missing value calculator (in dataloader).



Euclidean distance

The Euclidean distance measures the shortest distance between two real-valued vectors. Because of its intuitive use, simple implementation, and good results for many use cases, it is the most common distance measure and the default distance measure of many applications.

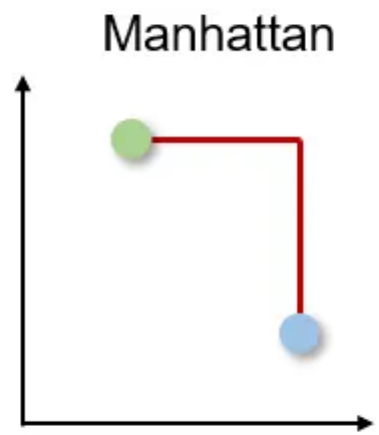


The Euclidean distance can also be refered to as a L2-norm and is calculated as:

$$d = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

Manhattan distance

The Manhattan distance is also called the Taxicab or City-Block distance as the distance between two real-valued vectors is calculated as if one could only move at right angles. This distance measure is often used for discrete and binary attributes to get a realistic path.

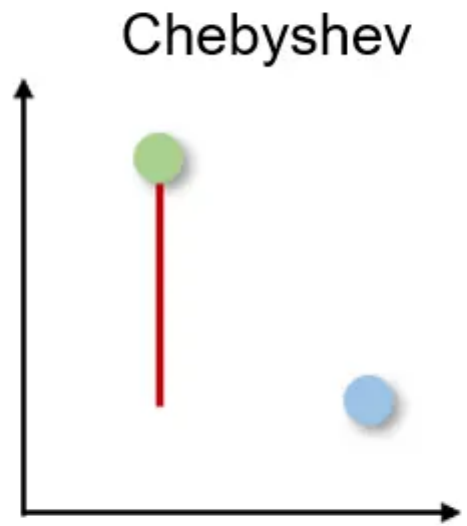


The Manhattan distance is based on a L1-norm and is calculated by:

$$d = \sum_{i=1}^{n} (x_i - y_i)$$

Chebyshev distance

The Chebyshev distance is also referred to as the chessboard distance as it is the greatest distance on any dimension between two real-valued vectors. The distance measure is often used in warehouse logistics in which the longest path determines the time it takes to get from one point to the next.

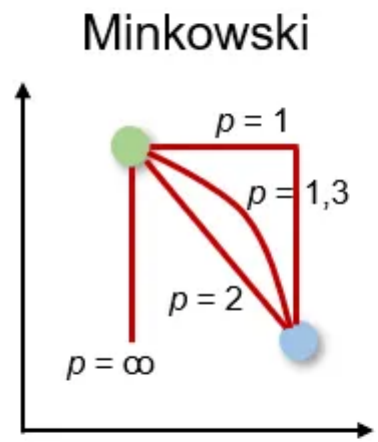


The Chebyshev distance is calculated by the L-infinity-norm:

$$d = \max_i \left(\left| x_i - y_i \right| \right)$$

Minkowski distance

The Minkowski distance is a generalized form of the above-mentioned distance measures. Hence, it can be used for the same use cases while providing a high flexiblity. We can choose the p-value to find the most suitable distance measure.

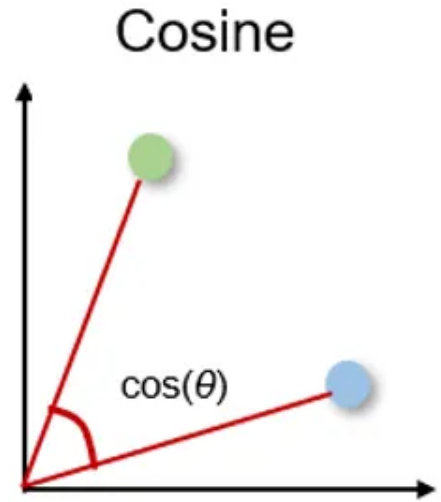


The Minkowski distance is calulated by:

$$d = \sqrt[p]{\sum_{i=1}^{n} (x_i - y_i)^p}$$

Cosine similarity and distance

The Cosine Similarity is a measure of orientation, determined by the cosine between two vectors, which neglects the magnitude of the vectors. The Cosine Similarity is often used in higher dimensionality where the magnitude of the data does not matter much, e.g., for recommendation systems or text analyses in which the data is represented by the word count.

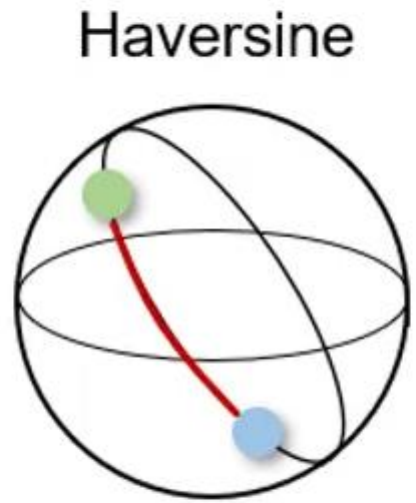


The Cosine Similarity can lie between -1 (opposite orientation) and 1 (same orientation) and is calculated by:

$$d = \cos(\alpha) = \frac{x\,y}{\|x\|\,\|y\|}$$

Haversine distance

The Haversine distance measures the shortest distance between two points on a sphere. Hence, the distance is used for navigation where points have longitudes and latitueds and in which the curvature has an effect.

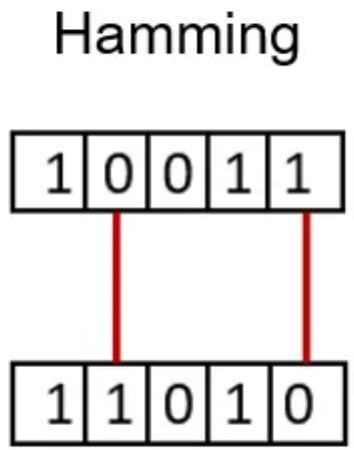


The Haversine distance can be determined by:

$$d = 2r \arcsin\left(\sqrt{\sin^2\left(\frac{\varphi_2 - \varphi_1}{2}\right) + \cos\varphi_1 \cos\varphi_2 \sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right)$$

Hamming distance

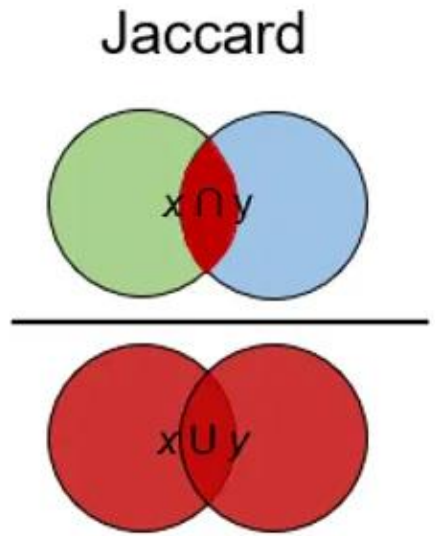
The Hamming distance measures the dissimilarity between two binary vectors or strings.



For this, the vectors are compared element-wise and the number of differences is averaged. The resulting distance lies between 0 if both vectors are identical and 1 if both vectors are completly different.

Jaccard Index and distance

The Jaccard Index is used to determine the similarity between two sample sets. It reflects how many one-to-one matches exist compared to the overall data set. The Jaccard Index is often used for binary data to compare the prediction of a deep learning model for image recognition with labeled data or to compare text patterns in documents based on the overlap of words.

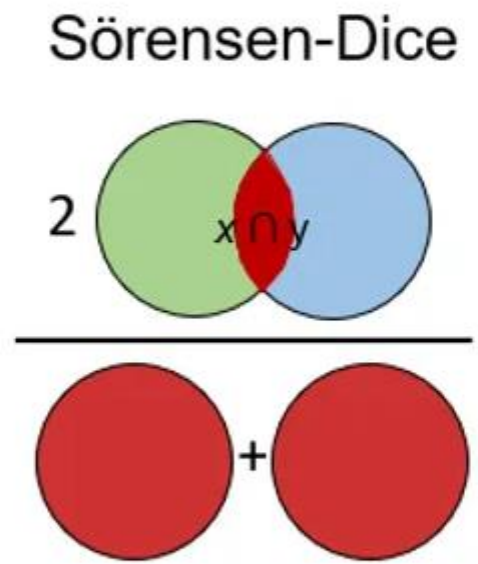


The Jaccard distance is calculated by:

$$d = 1 - \frac{|x \cap y|}{|x \cup y|}$$

Sörensen-Dice Index

The Sörensen-Dice Index is similar to the Jaccard Index as it measures the similarity and diversity of sample sets. However, the index is more intuitive as it calculates the percentage of the overlap. The Sörensen-Dice Index is often used for image segmentation and text similarity analysis.



The Sörensen-Dice distance is determined by:

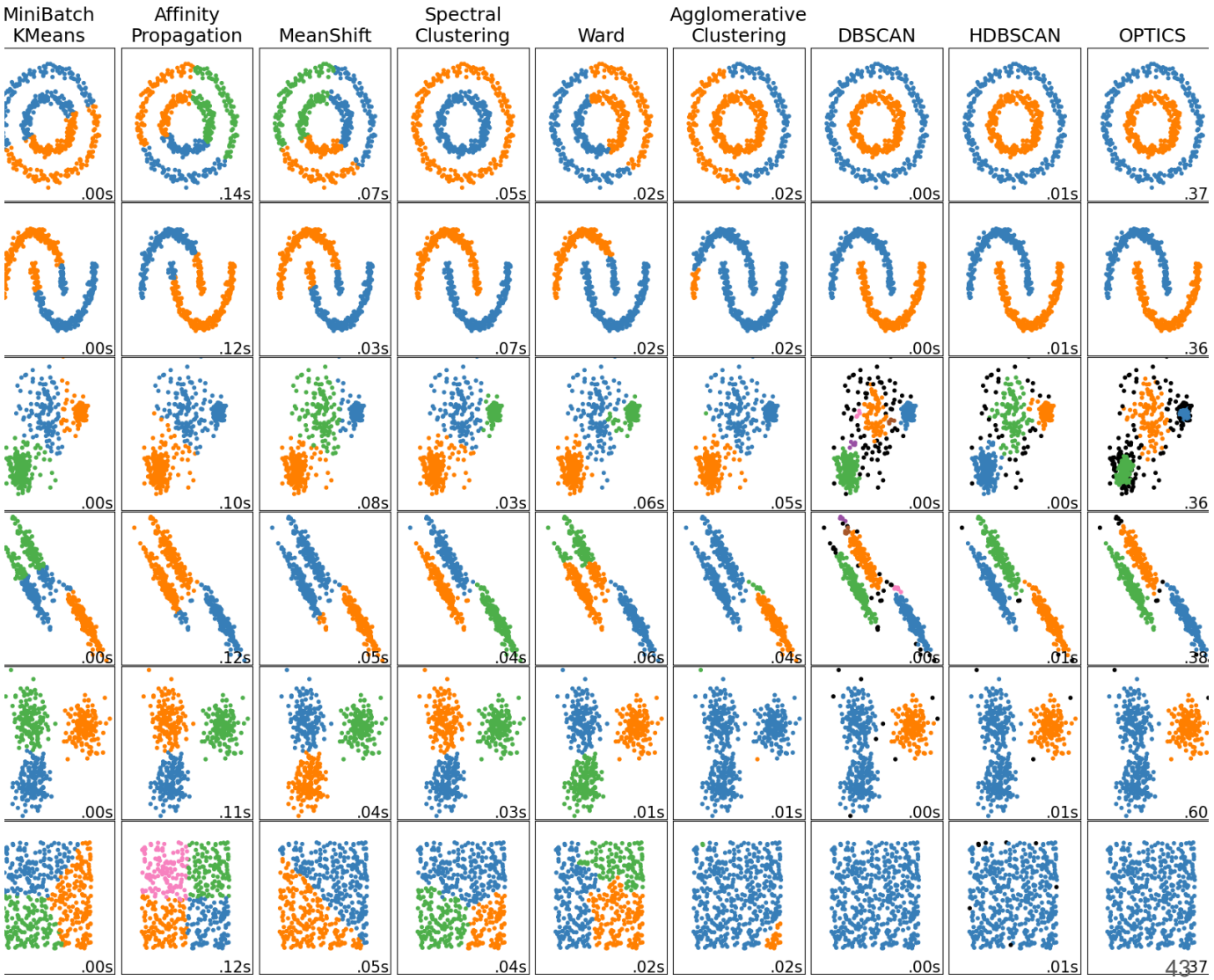
$$d = \frac{2|x \cap y|}{|x| + |y|}$$

How to Find Patterns in Data?

Step 3 – Pick the Clustering Approach Based on Patterns

- **Centroid-based (e.g., K-Means)** → Best for compact, spherical clusters.
- **Density-based (e.g., DBSCAN)** → Best for arbitrary-shaped clusters & noise handling.
- **Hierarchical** → Best when cluster relationships/hierarchy are important.

- **So, let's check next the Types of Clustering Algorithms**



Types of Clustering **Algorithms**

- This categorization refers to the **algorithmic approach** taken to create clusters from data, or **the process of how clusters are found**

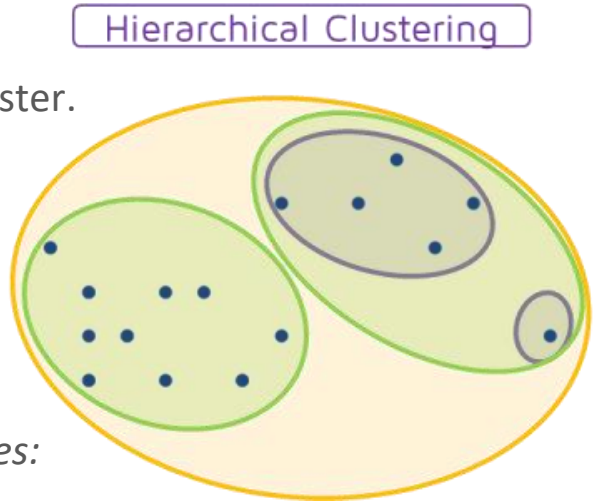
1. Partitional Clustering
Divides data into non-overlapping clusters; each point belongs to one cluster. *Examples:* K-Means, K-Medoids, K-Modes

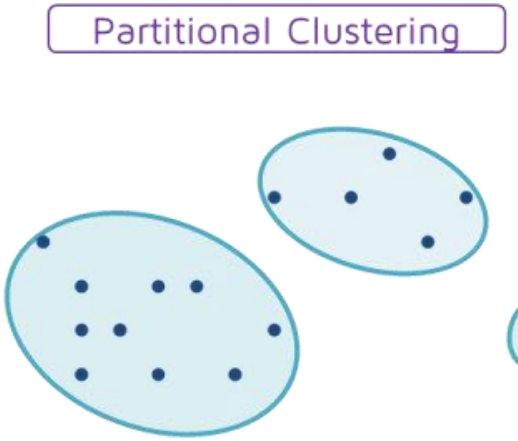
2. Hierarchical Clustering
Nested clusters arranged in a tree structure. *Examples:* Agglomerative, Divisive.

3. Density-Based Clustering
Forms clusters from dense regions; labels sparse points as noise. *Examples:* DBSCAN, OPTICS.

4. Model-Based Clustering
Assumes data comes from a mix of probability distributions. *Examples:* GMM, EM.

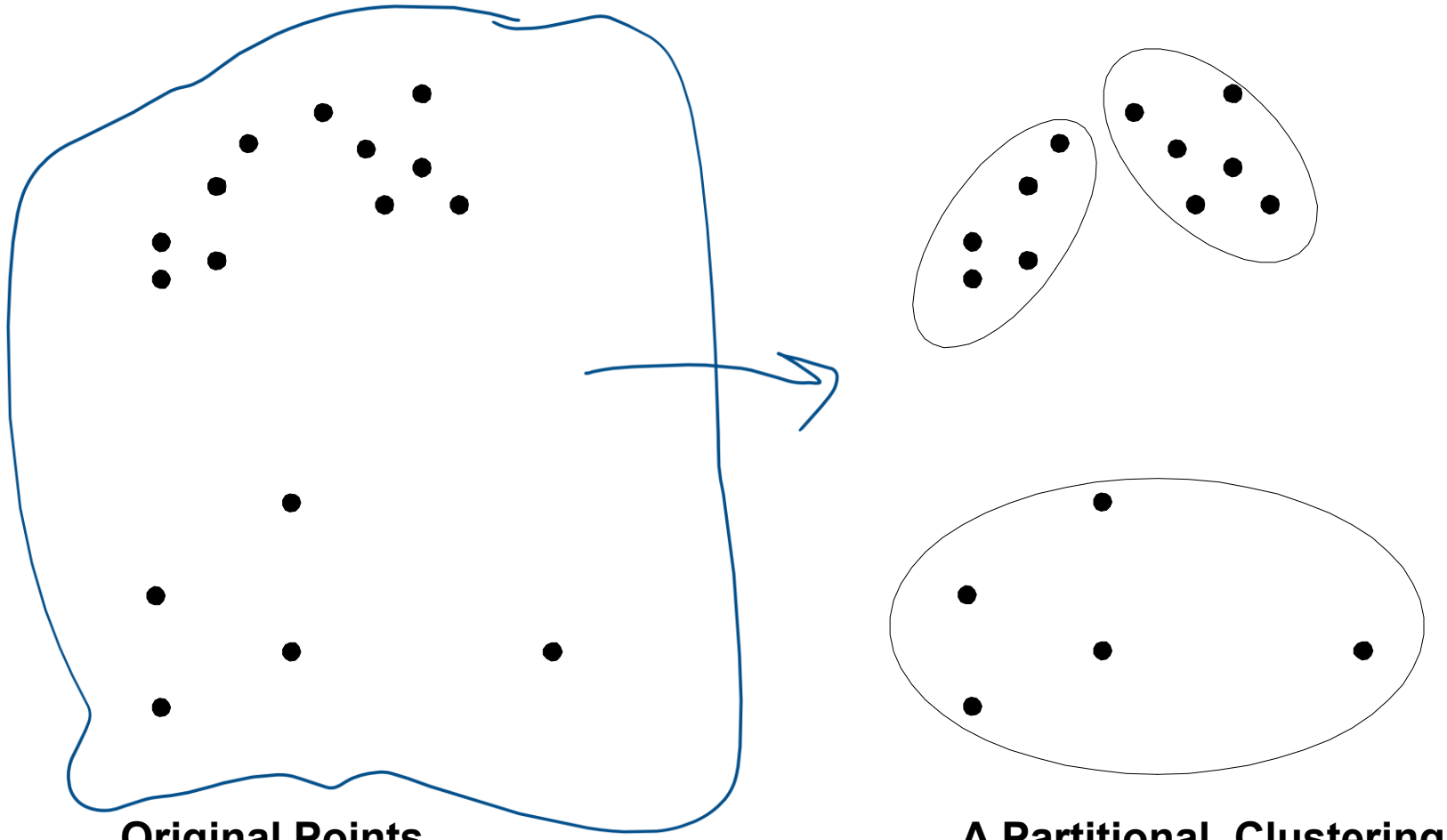
5. Grid-Based Clustering
Divides data space into grid cells; clusters dense cells. *Examples:* STING, CLIQUE.





Partitional Clustering

Divides data into non-overlapping clusters; each point belongs to one cluster. *Examples:* K-Means, K-Medoids, K-Modes

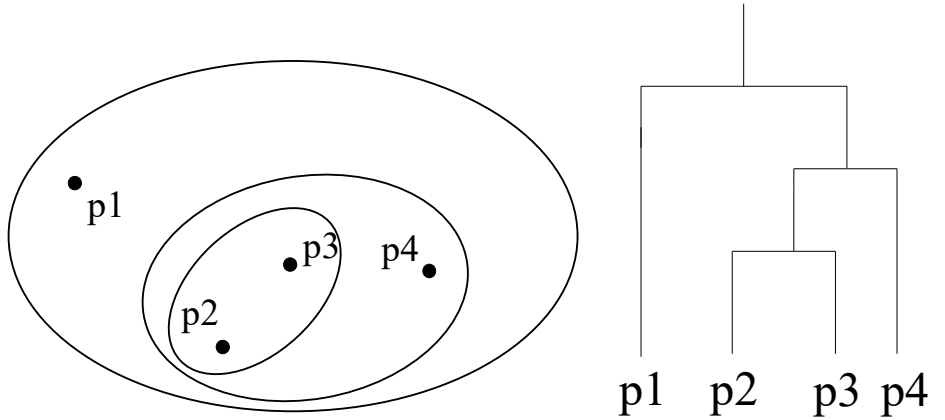


Original Points

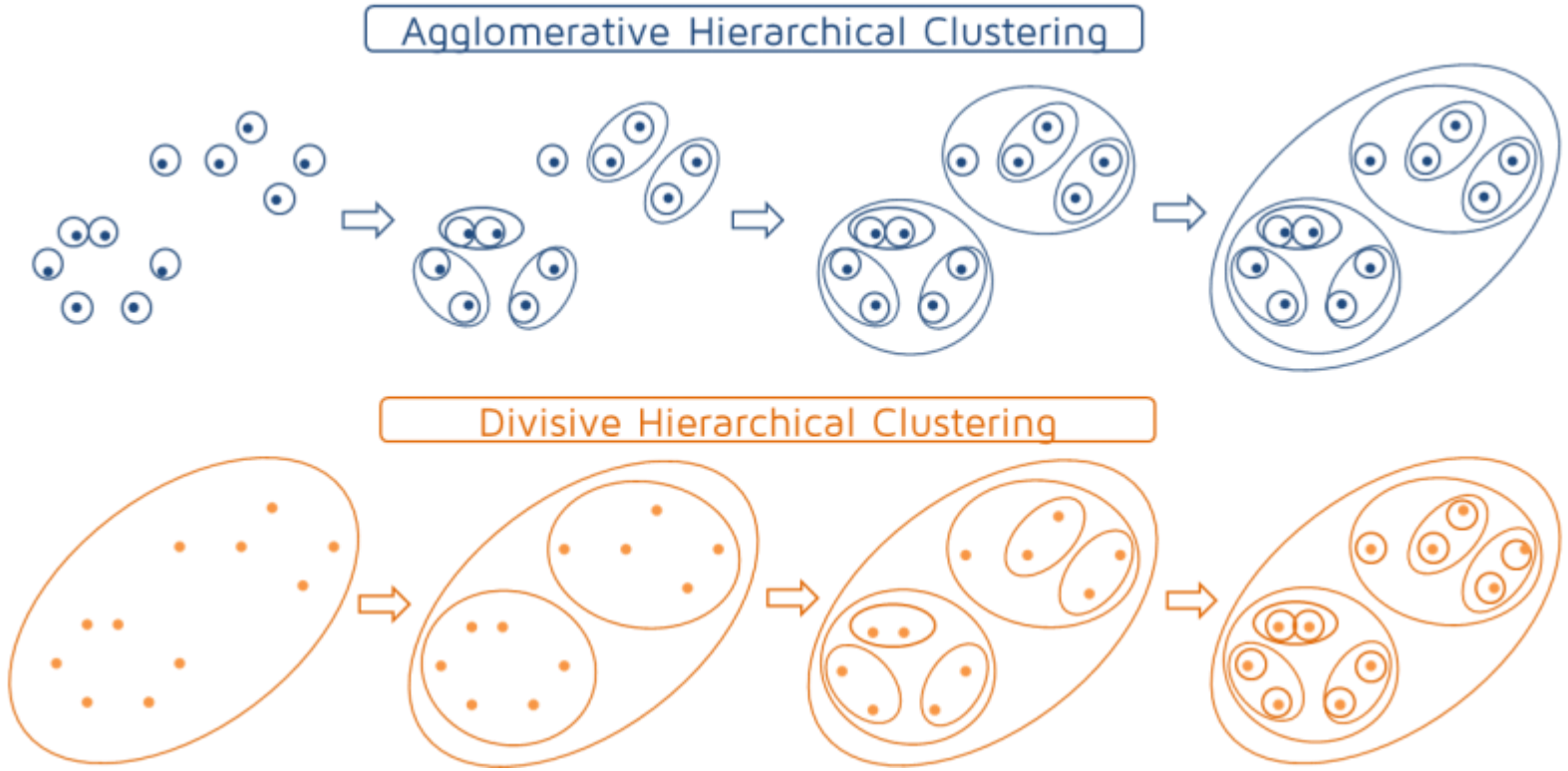
A Partitional Clustering

Hierarchical Clustering

- Nested clusters arranged in a tree structure.
- *Examples:* Agglomerative, Divisive.

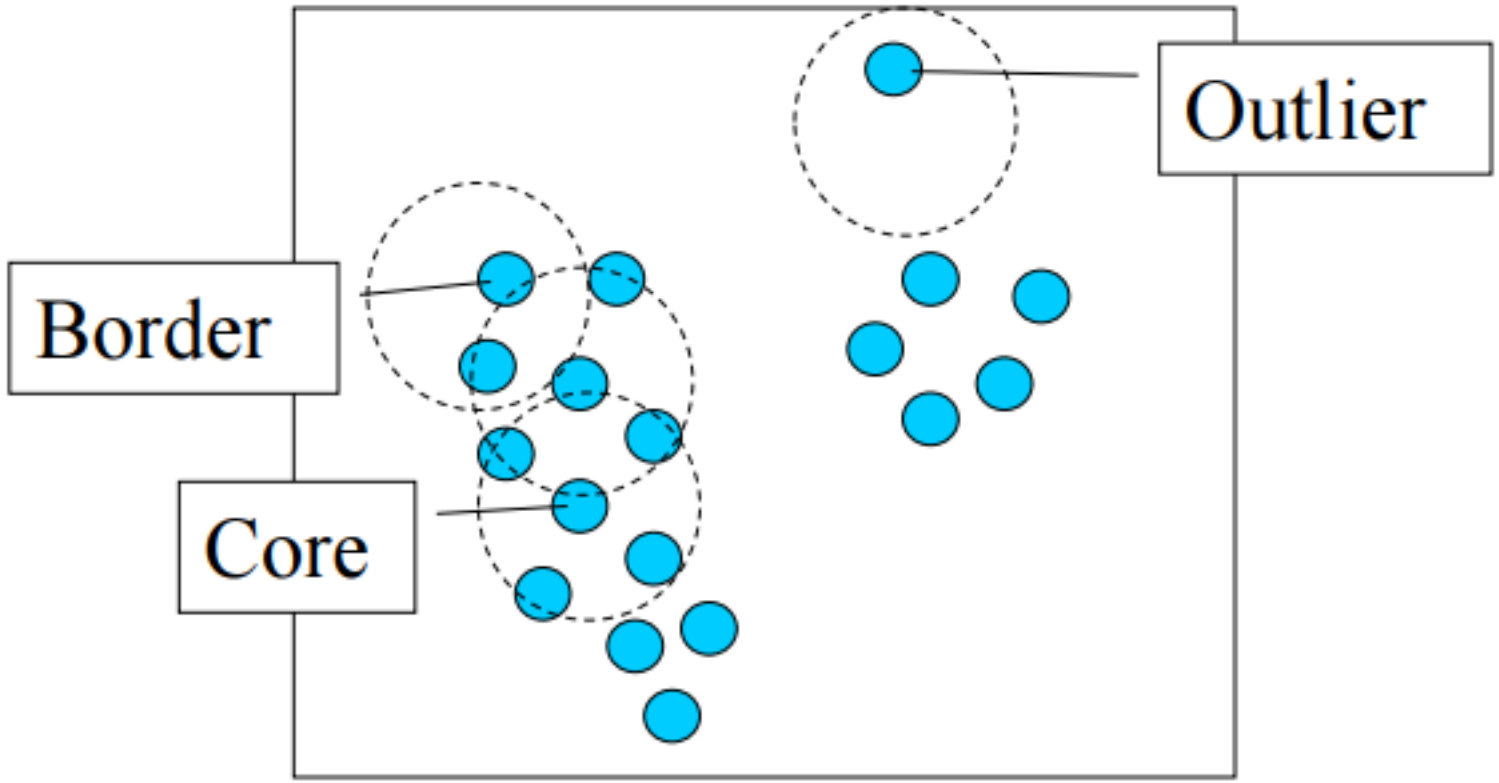


Traditional Hierarchical Clustering Traditional Dendrogram



Density-based Clustering

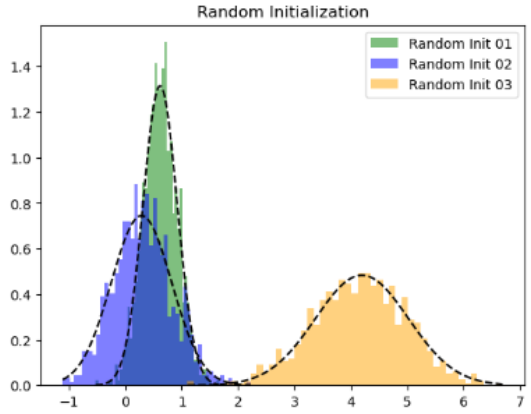
Forms clusters from dense regions; labels sparse points as noise. *Examples:* DBSCAN, OPTICS.

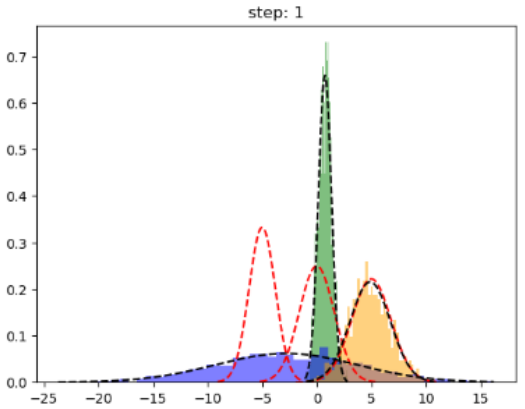


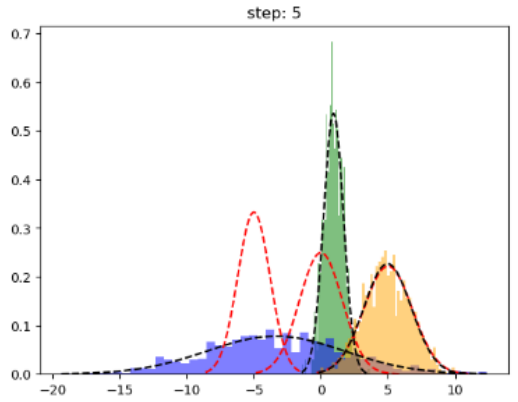
Model-based Clustering

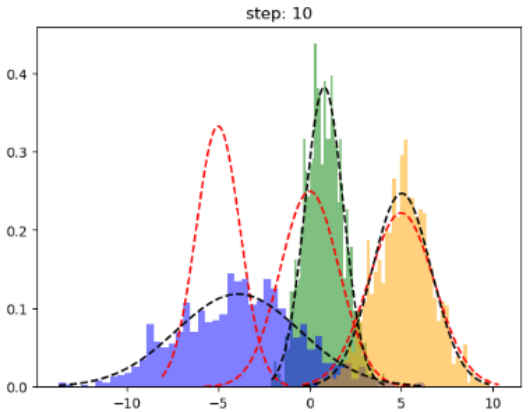
- Assumes data comes from a finite mix of probability distributions, each representing a cluster.

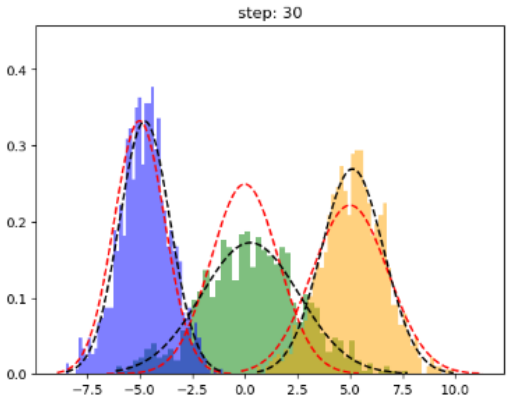
Examples: GMM, EM.







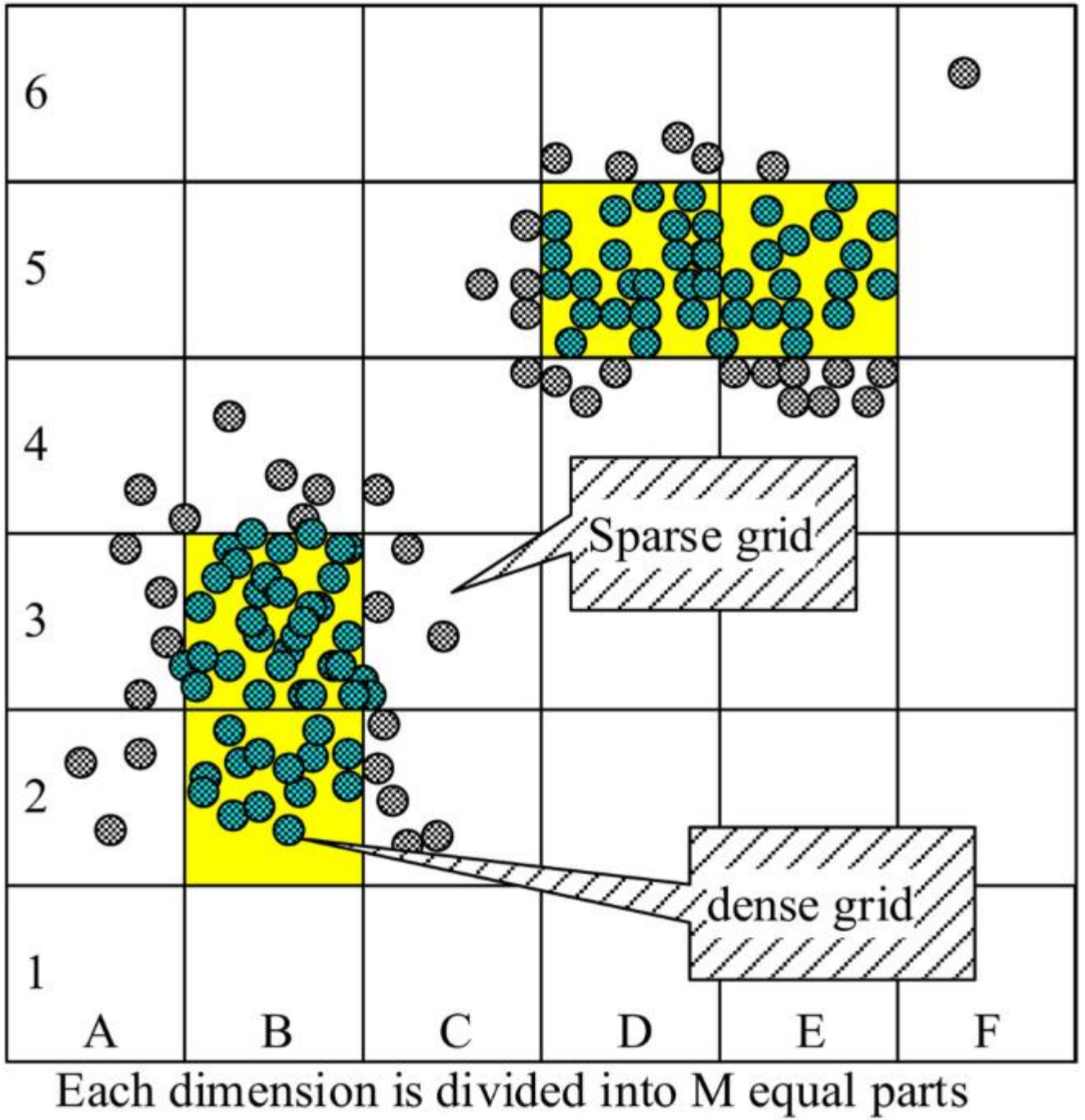




Grid-based Clustering

- Divides data space into grid cells;

clusters dense cells. *Examples:* STING, CLIQUE.



Types of Cluster **Models (another categorization view!)**

- This categorization refers to the **conceptual characterization** of **what defines a cluster within the data or define how data points are grouped.**

.

- Distance Based
  - Well-separated clusters
  - Center-based clusters
  - Contiguous clusters
- Density Based
  - Density-based clusters

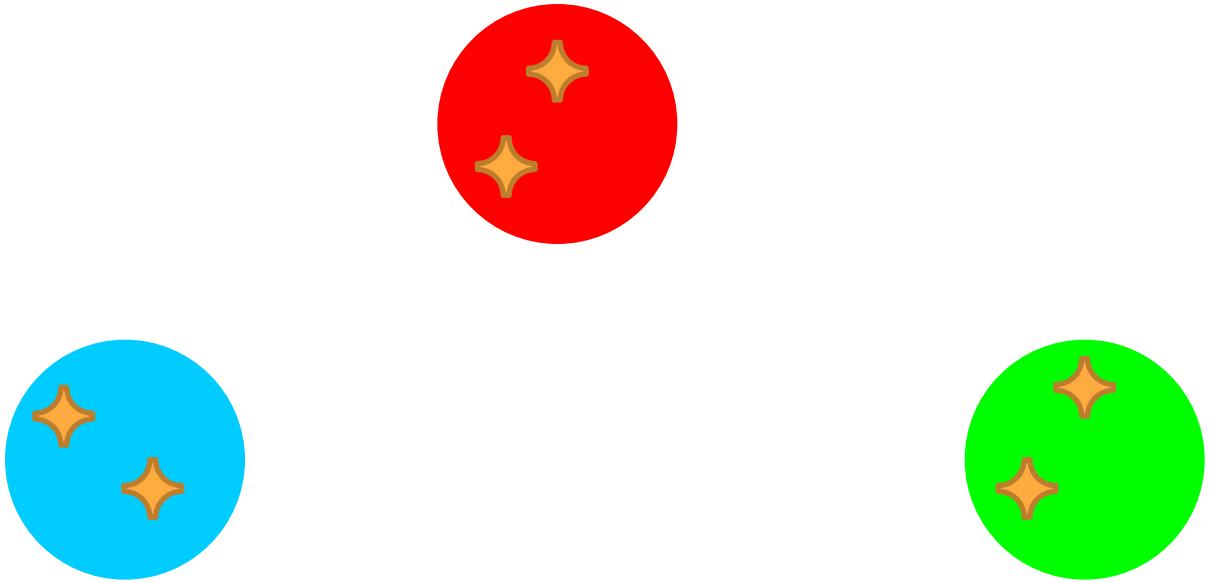
Well-Separated Clusters:

- **Description**: In well-separated clusters, **each data point in a cluster is closer (more similar) to every other point in the cluster than to any point not in the cluster.**
- **Mathematical Notation**:
  - Given a distance metric *d*, and two points *x* and *y* in cluster *C*, and any point *z* not in *C*, the following condition holds:

$\forall x, y \in C, \forall z \notin C\colon\ d(x, y) < d(x, z)$ and $d(y, x) < d(y, z)$

Types of Clusters: **Well-Separated**

- Well-Separated Clusters:
  - A cluster is a set of points such that any point in a cluster is closer (or more similar) to every other point in the cluster than to any point not in the cluster.



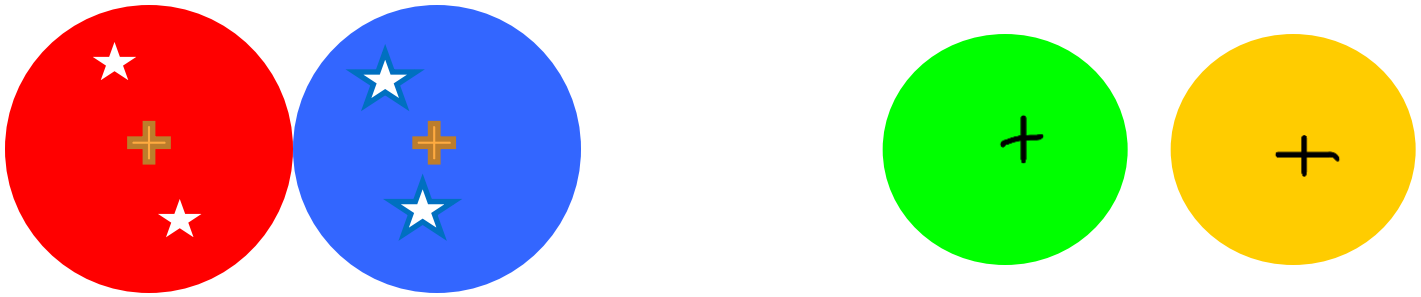
3 well-separated clusters

Center-Based Clusters:

- **Description**: Center-based clusters are defined by a centroid, which is either a center point or the mean position of all the points in the cluster. **Every point in the cluster is closer to its cluster centroid than to other cluster centroids.**
- **Mathematical Notation**:
  - $For\ a\ set\ of\ clusters$
  - $\{C_1, C_2, \ldots, C_k\}\ with\ centroids\ \{\mu_1, \mu_2, \ldots, \mu_k\},$
  - $a\ point\ x\ belongs\ to\ cluster\ C_i\ if\colon\ \forall j \neq i \colon d(x, \mu i) < d(x, \mu j)$

Types of Clusters: **Center-Based**

- Center-based
  - A cluster is a set of objects such that an object in a cluster is closer (more similar) to the “center” of a cluster, than to the center of any other cluster
  - The center of a cluster is often a centroid, the average of all the points in the cluster, or a medoid, the most “representative” point of a cluster



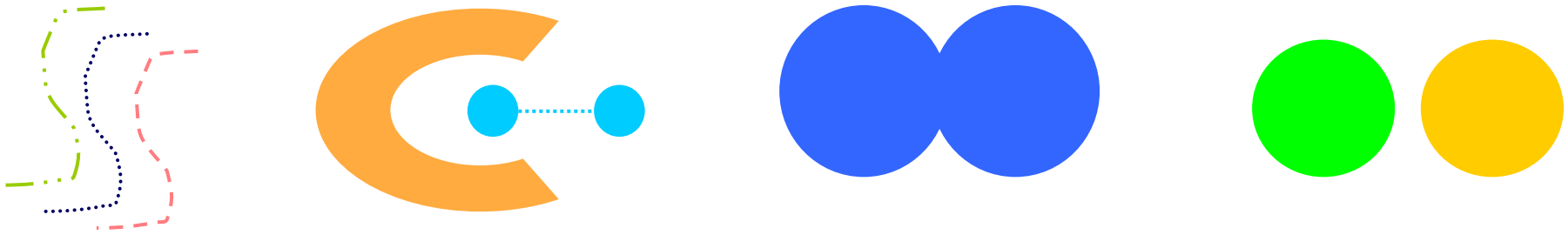
4 center-based clusters

Contiguous Clusters (also known as Connectivity-based Clusters):

- **Description**: Clusters are formed **by points that are contiguous or connected to one another. A point is in the same cluster as its nearest points.**
- **Mathematical Notation**:
  - Given a set of points X and a threshold distance ϵ, a point $x \in X$ is connected to point $y \in X$ forming a cluster C if $d(x,y) < \epsilon$.
  - Clusters can then be defined as maximal sets of connected points.

Types of Clusters: **Contiguity-Based**

- Contiguous Cluster (Nearest neighbor or Transitive)
  - A cluster is a set of points such that a point in a cluster is closer (or more similar) to one or more other points in the cluster than to any point not in the cluster. A point is in the same cluster as its nearest points..



8 contiguous clusters

Density-Based Clusters:

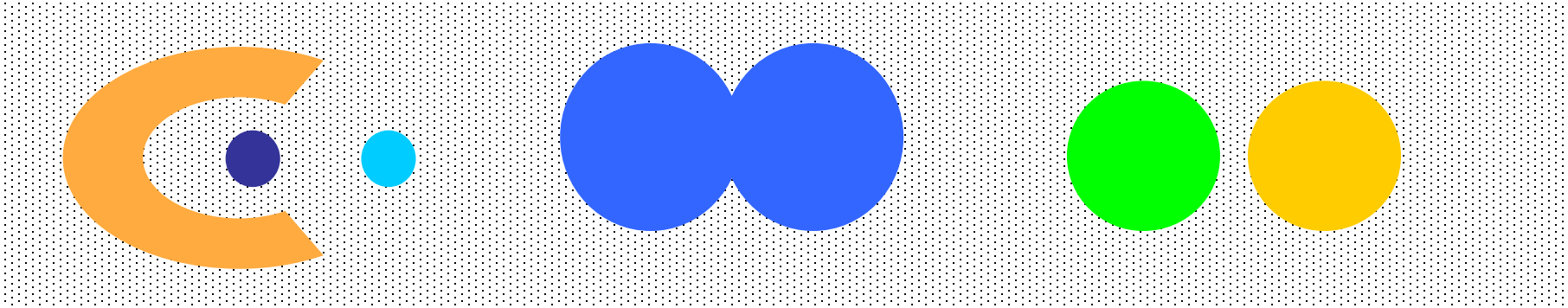
- Description: **Clusters are regions of higher data point density compared to the rest of the dataset. Sparse areas, which are needed to separate clusters, are typically considered noise or border points.**
- Mathematical Notation:
- Density Function (ρ): Measures point density in a data region by counting points within a radius (ϵ) of a target point (x).
- Cluster (C): Defined as a region where the point density exceeds a threshold (ρ0). Formally, for a set of points X, cluster C is defined as:

$$C=\{x \in X \mid \rho(x) > \rho_0\}$$

- meaning points in C have density greater than ρ0.

Types of Clusters: **Density-Based**

- Density-based
  - A cluster is a dense region of points, which is separated by low-density regions, from other regions of high density.
  - Used when the clusters are irregular or intertwined, and when noise and outliers are present.



6 density-based clusters

Clustering Algorithms

- K-means and its variants
- Hierarchical clustering
- Density-based clustering

Distance Based

- Well-separated clusters
- Center-based clusters
- Contiguous clusters

Density Based

- Density-based clusters

Help me add connection lines between these two sets!

K-means Clustering

- Partitional clustering approach
- Each cluster is associated with a centroid (center point)
- Each point is assigned to the cluster with the closest centroid
- Number of clusters, K, must be specified
- The basic algorithm is very simple

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

Example

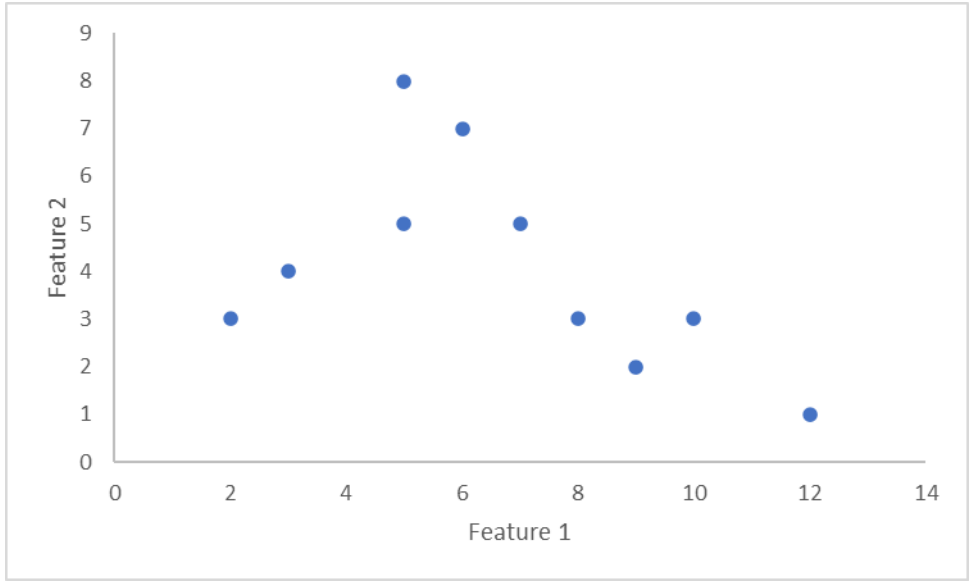
Considering the following data points, apply k-means clustering algorithm where k = 2.

1- Considering initial centroids are P3 and P7

2- Calculate the distance from each node to each centroid and assign nodes to the nearest centroid.

The distance between a node *P*(*xp*,*yp*) and a centroid *C*(*xc*,*yc*) is calculated using the Euclidean distance (E.D) formula:

$$d(P,C) = ((x_p - xc)2 + (y_p - y_c)^2)^{0.5}$$



	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
X1	2	3	5	6	5	7	8	9	10	12
X2	3	4	8	7	5	5	3	2	3	1

- 3- Assign each node to the centroid with the shortest E.D.
- 4- Once all nodes have been assigned to centroids, calculate the new centroid of each cluster by averaging the coordinates of the nodes in that cluster.

- 5- Repeat steps 2, 3 and 4 until the centroids do not change or changes are below a certain threshold.

Step 2: Assignment

We calculate the Euclidean distance from each node to each centroid:

For example, the distance from node A to centroid C_1 is:

$$d(A, C_1) = \sqrt{(2-5)^2 + (3-8)^2} = \sqrt{9+25} = \sqrt{34}$$

And the distance from node A to centroid C_2 is:

$$d(A, C_2) = \sqrt{(2-8)^2 + (3-3)^2} = \sqrt{36} = 6$$

Since $\sqrt{34} < 6$, node A would be assigned to centroid C_1.

We would do this for each node.

Step 3: Update

After all nodes have been assigned to a centroid, we calculate the new centroid for each cluster by averaging the coordinates of all the nodes in the cluster.

For Cluster 1 (with C_1), if we assume nodes A, B, C, D, and E were assigned to it, the new centroid would be:

$$C_1' = \left(\frac{2+3+5+6+5}{5}, \frac{3+4+8+7+5}{5}\right) = \left(\frac{21}{5}, \frac{27}{5}\right) = (4.2, 5.4)$$

For Cluster 2 (with C_2), if we assume nodes F, G, H, I, and J were assigned to it, the new centroid would be:

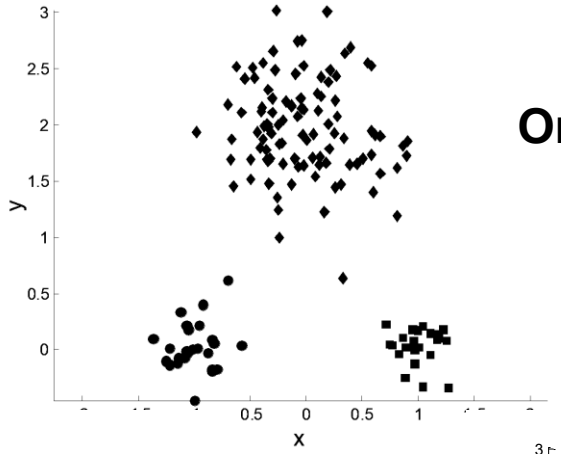
$$C_2' = \left(\frac{7+8+9+10+12}{5}, \frac{5+3+2+3+1}{5}\right) = \left(\frac{46}{5}, \frac{14}{5}\right) = (9.2, 2.8)$$

P	X1	X2	P,C11	P,C21	Cluster Ass	P, C12	P, C22	Cluster Ass
P1	2	3	5.830952	6	C1	3.255764	7.202777	C1
P2	3	4	4.472136	5.09902	C1	1.843909	6.315061	C1
P3 (C11)	5	8	0	5.830952	C1	2.720294	6.68431	C1
P4	6	7	1.414214	4.472136	C1	2.408319	5.280152	C1
P5	5	5	3	3.605551	C1	0.894427	4.741308	C1
P6	7	5	3.605551	2.236068	C2	2.828427	3.11127	C1
P7 (C21)	8	3	5.830952	0	C2	4.494441	1.216553	C2
P8	9	2	7.211103	1.414214	C2	5.882176	0.824621	C2
P9	10	3	7.071068	2	C2	6.276942	0.824621	C2
P10	12	1	9.899495	4.472136	C2	8.955445	3.328663	C2
C12	4.2	5.4			C13	2.3	5.5	
C22	9.2	2.8			C23	6.4	1.5	

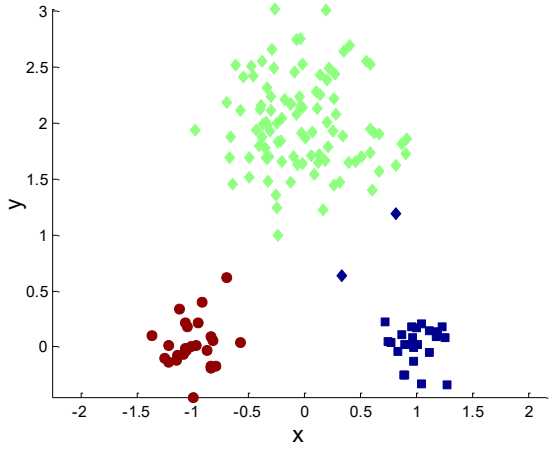
K-means Clustering – Details

- Initial centroids are often chosen randomly.
  - Clusters centroids vary from one run to another.
- The centroid is (typically) the mean of the points in the cluster.
- ‘Closeness’ is measured by Euclidean distance, cosine similarity, correlation, etc.
- K-means will converge for common similarity measures mentioned above. meaning that it will reach a point where the centroids no longer change significantly, using these common measures of similarity.
- Most of the convergence happens in the first few iterations.
  - Often the stopping condition is changed to ‘Until relatively few points change clusters’. A common approach is to continue iterating until only a small number of points switch clusters between iterations, indicating that a stable clustering has been reached.
- Complexity is O(n * K * I)
  - n = number of points, K = number of clusters,
    I = number of iterations

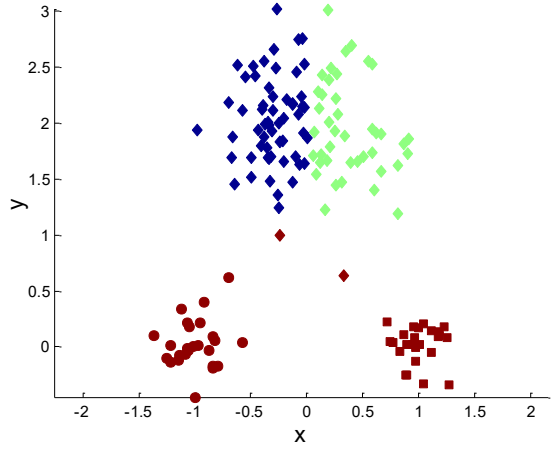
Two different K-means Clustering: number of assumed clusters? Initial centroids?



Original Points

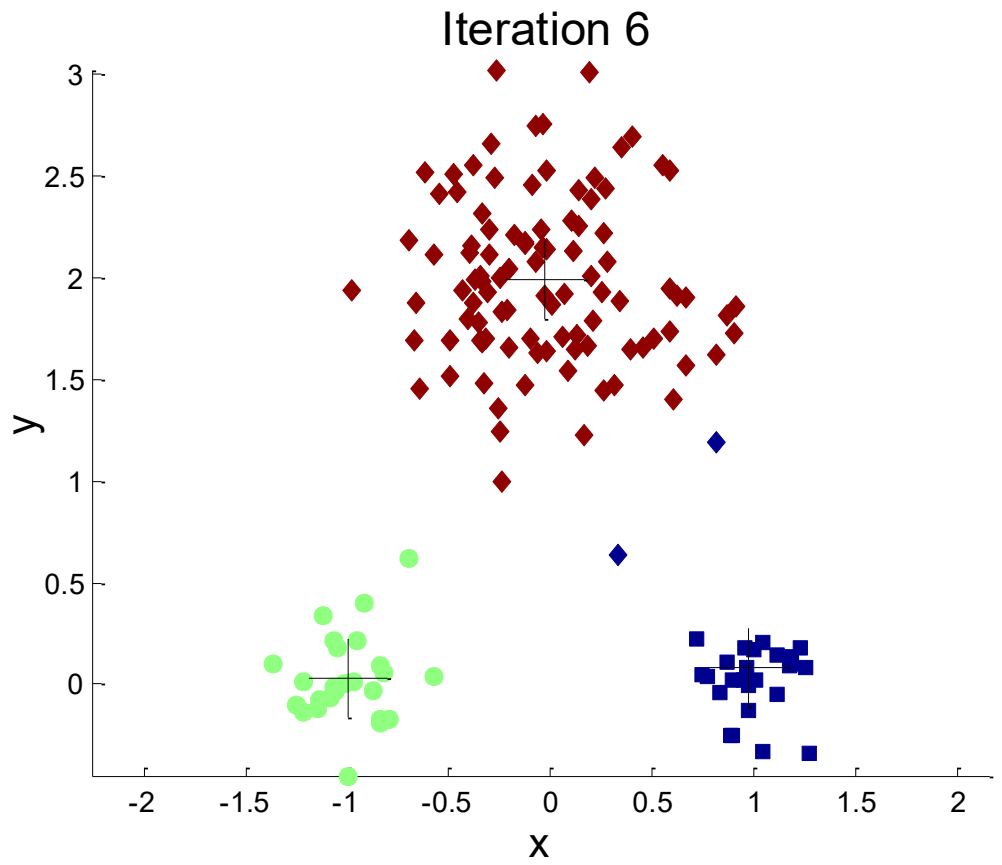


Optimal Clustering

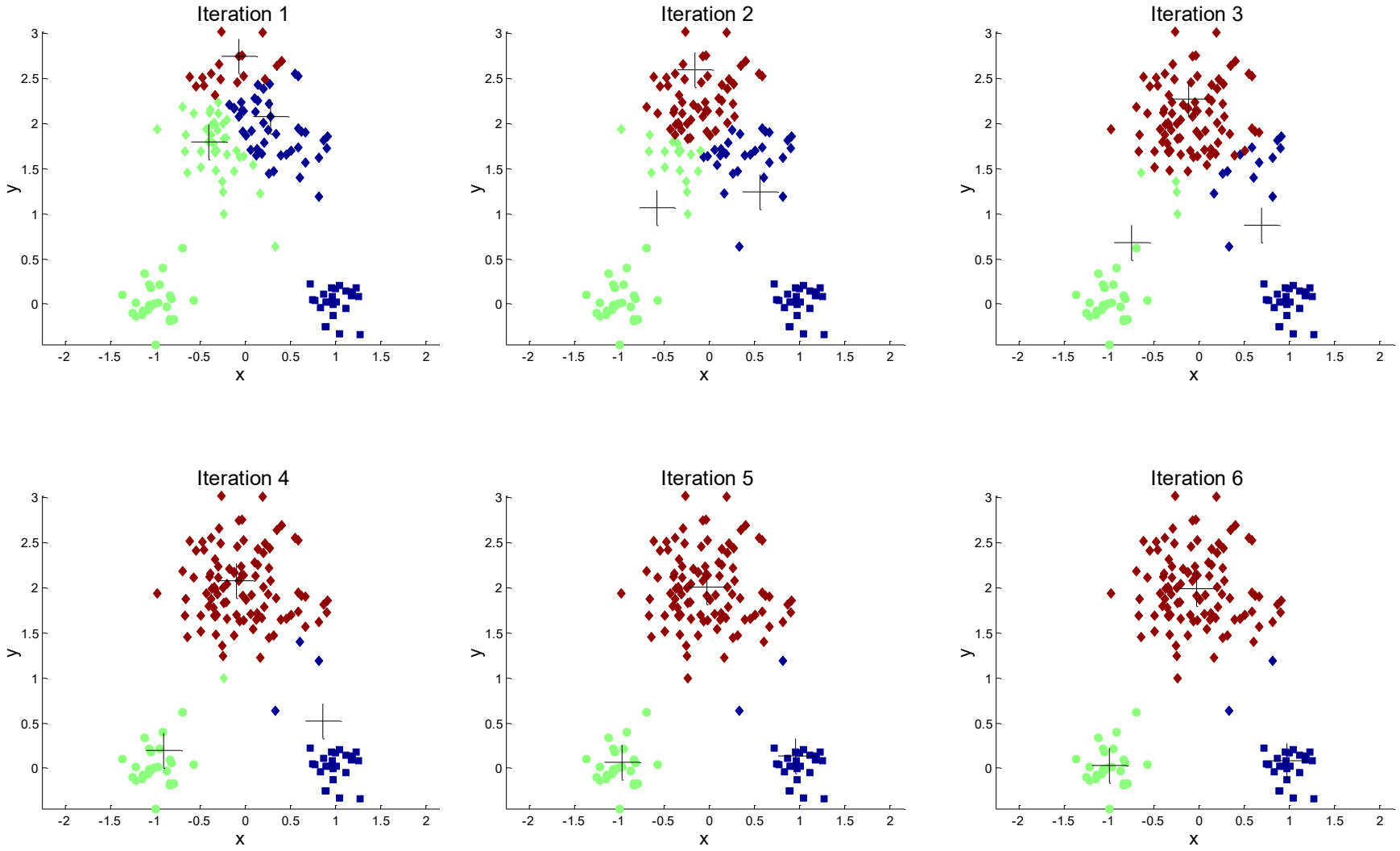


Sub-optimal Clustering

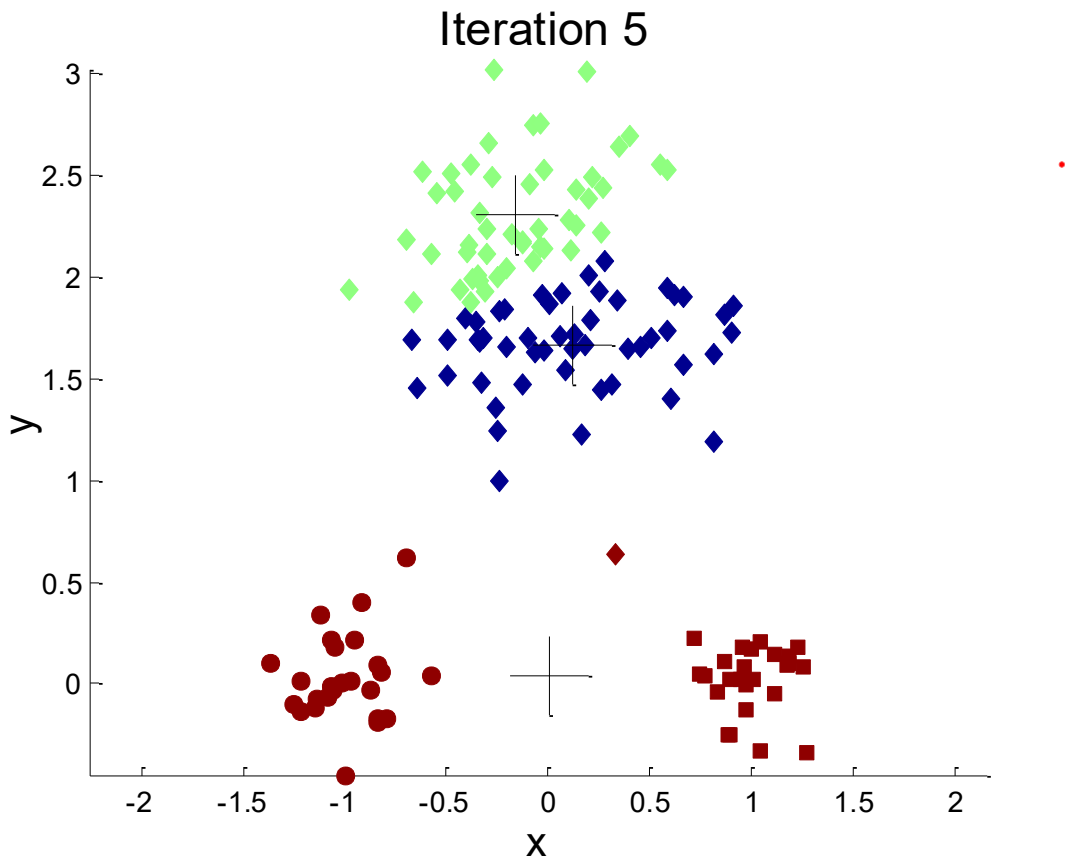
Importance of Choosing Initial Centroids, **the case when choosing initial centroids that lead to optimal clustering**



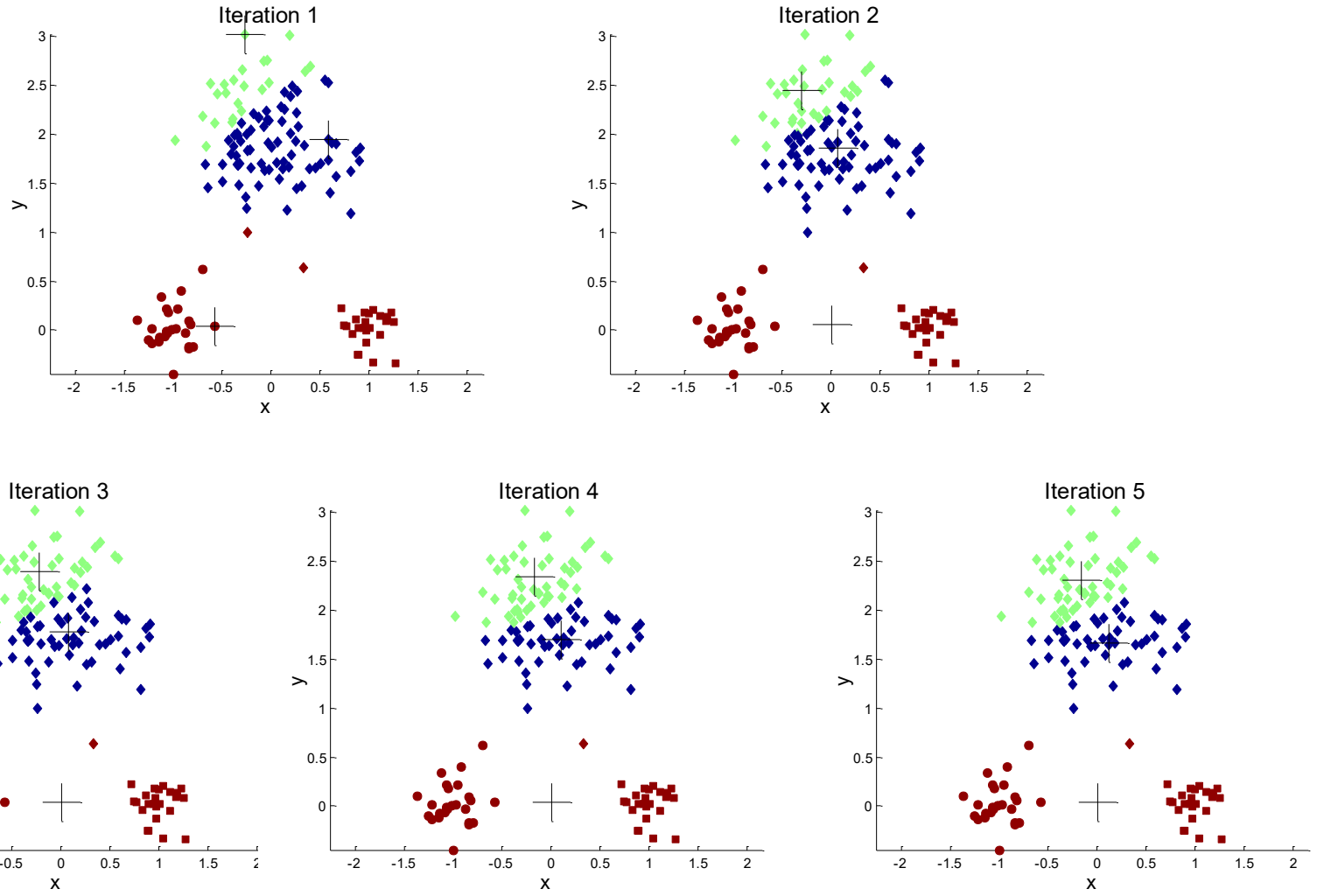
Importance of Choosing Initial Centroids



Importance of Choosing Initial Centroids, **the case when choosing initial centroids that lead to sub-optimal clustering …**



Importance of Choosing Initial Centroids …



Evaluating K-means Clusters

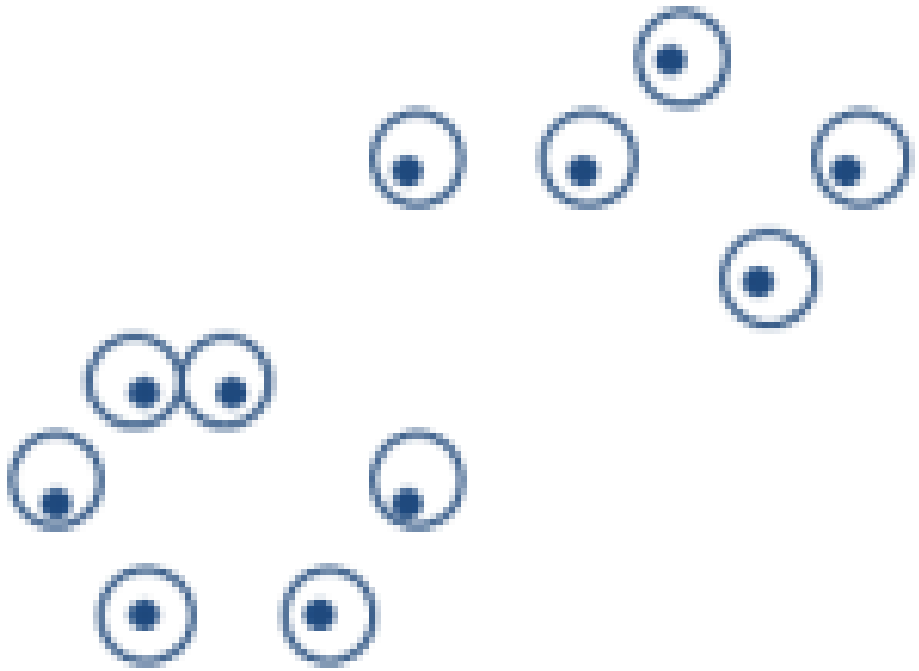
Most common measure is Sum of Squared Error (SSE)

- For each point, the error is the distance to the nearest cluster (the cluster the point was finally assigned)
- To get SSE, we square these errors and sum them. The min intra-cluster distance, the better

$$SSE = \sum_{i=1}^{K} \sum_{x \in C_i} dist^2(m_i, x)$$

This measures the intra-cluster distance between each point x in the cluster c and the cluster's center or mean m, for all clusters K

- Interpretation: The distance from each point in the cluster to its centroid is squared and summed up to compute the SSE.
- For the same data, Given two clustering, one with K1 clusters and the other with K2 clusters, we can choose the one with the smallest error.

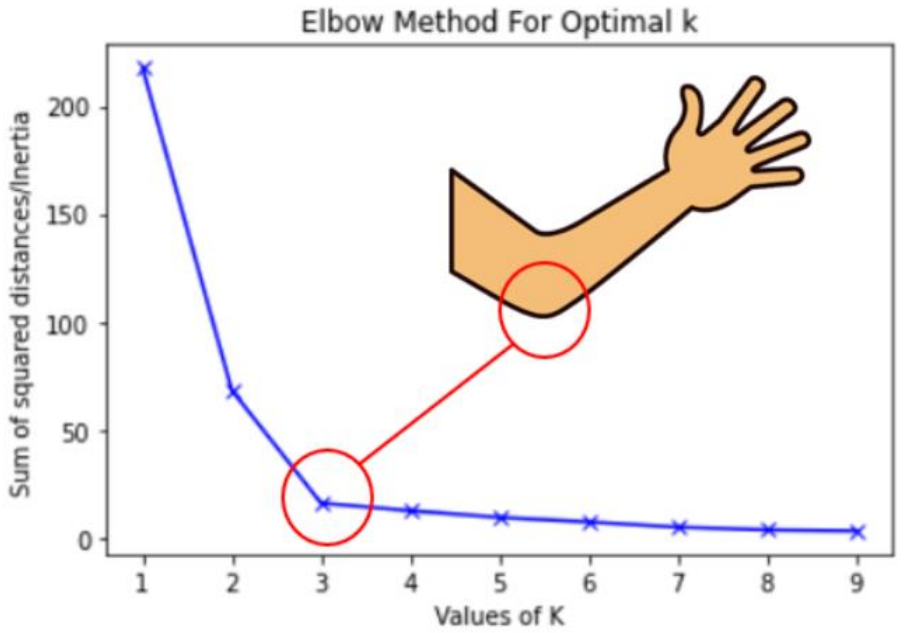


- What is the value of SSE in this case?

Evaluating K-means Clusters

Relationship between SSE and Number of Clusters (K)

- General Trend: As K increases, SSE tends to decrease. Increasing the number of clusters typically leads to a smaller distance between points and their respective centroids, thus reducing SSE.
- Choosing Optimal K: A lower SSE or conversely, higher K is not always better as it can lead to overfitting (too many clusters).
- Techniques like the Elbow Method can be used to find an optimal balance, where an increase in K does not significantly decrease SSE.



Line plot between K and inertia

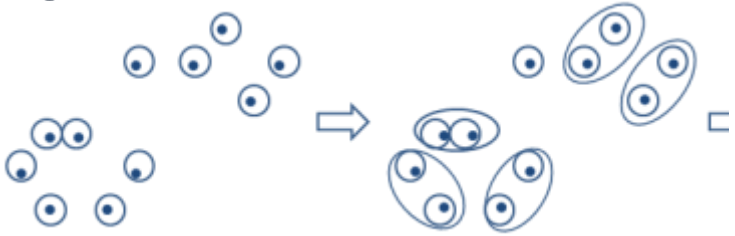
Solutions to Initial Centroids Problem

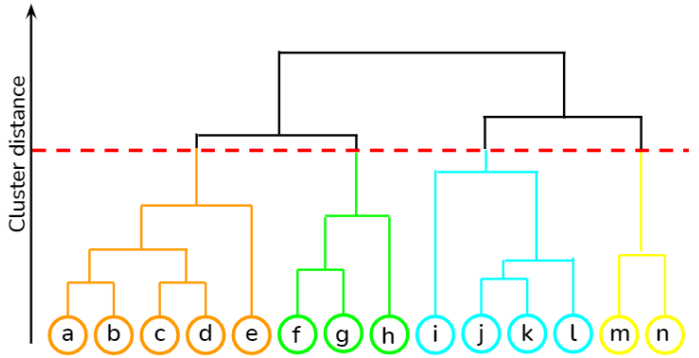
Multiple Runs

- **What it is**: Running the K-means algorithm several times with different random initial centroids.
- **Benefit**: Increases the chance of finding a better clustering solution by avoiding poor initial placements.

Hierarchical Clustering for Initial Centroids

- **Approach**: Use hierarchical clustering to first create a dendrogram and then pick initial centroids based on this.
- **Advantage**: Provides a more informed starting point, potentially leading to better clustering.





Solutions to Initial Centroids Problem

Selecting More Than K Initial Centroids

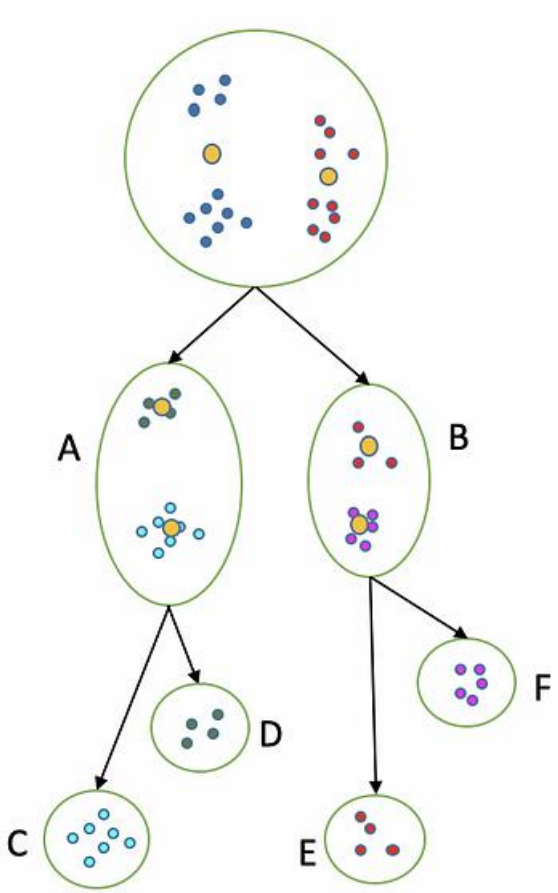
- **Strategy**: Start with a larger number of initial centroids than the final desired number of clusters.
- **Process**: Gradually reduce the number of centroids to K by combining them, based on proximity or similarity.

Postprocessing

- **Idea**: Adjust the centroids after the initial run of K-means.
- **Goal**: Refine the centroids for better cluster quality by considering additional criteria or corrections.

Bisecting K-means

- **Concept**: A variant of K-means where the algorithm starts with all points in a single cluster and then successively splits clusters.
- **Advantage**: Less sensitive to initial centroid selection, as it progressively refines the clusters.



Handling Empty Clusters

- **Issue of Empty Clusters**
- **What Happens**: Sometimes, during the iterative process of K-means clustering, one or more clusters may end up with no points assigned to them (only the centroid is there!!)
- **Why It's a Problem**: Empty clusters can cause computational issues and may lead to inefficient clustering results.
- **Strategies for Dealing with Empty Clusters**
    - **Reassigning Centroids**
    - **Approach**: If a cluster becomes empty, reposition its centroid to a random data point or a point that contributes most to SSE (Sum of Squared Errors) of another cluster.
    - Repositioning the empty cluster's centroid to this "most contributing point" of another cluster, thereby redistributing points and potentially improving overall clustering performance. maintain the initial number of clusters.
    - **Merging Clusters**
    - **Approach**: Combine the empty cluster with another nearby or similar cluster to maintain the desired number of clusters, which is less than the initial.

Limitations of K-means

Differing Cluster Sizes

Issue: K-means tends to struggle with clusters of varying sizes**. It often biases towards clusters of similar sizes**, potentially overlooking smaller clusters.

Differing Densities

Challenge: Clusters with different densities can lead to misclassified points, as **K-means might favor denser clusters.**

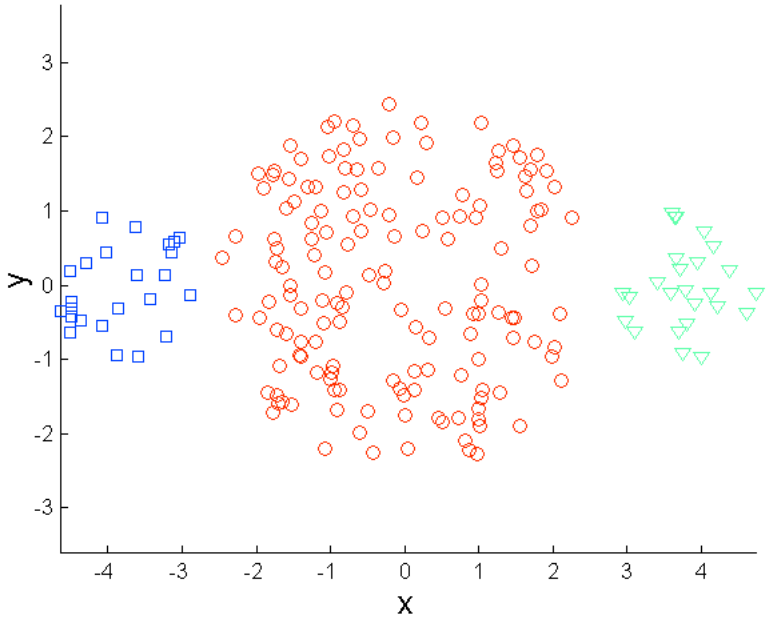
Non-Globular Shapes

Problem: K-means assumes clusters are spherical (globular). **It performs poorly with clusters having non-globular shapes such as elongated or circular shapes.**

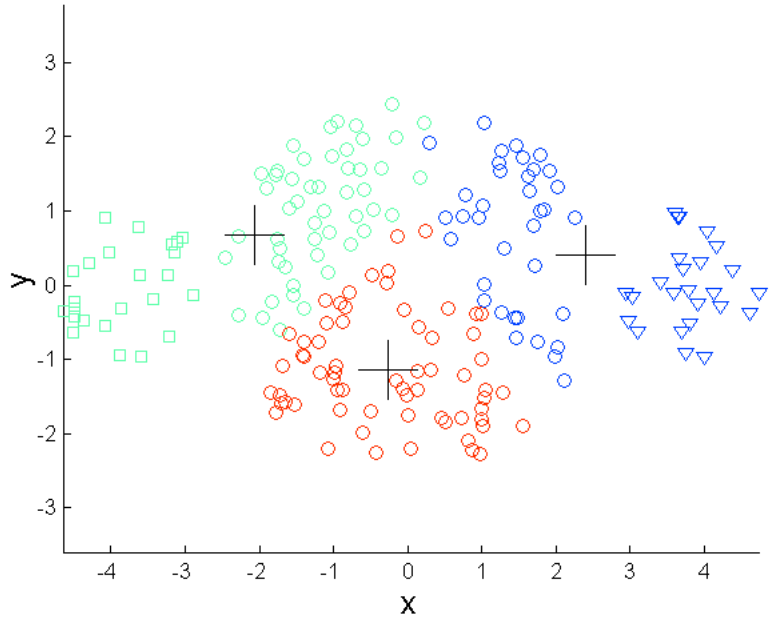
Data with Outliers

Concern: Outliers can significantly skew the centroids of clusters in K-means, leading to inaccurate clustering.

Limitations of K-means: Differing Sizes (tends to balancing the cluster sizes, because of its dependency on mean-based centroids)

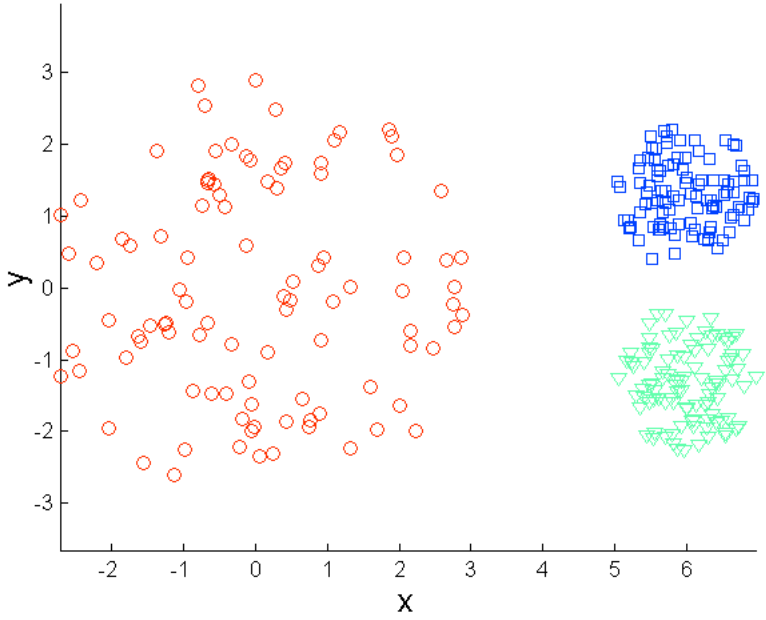


Original Points

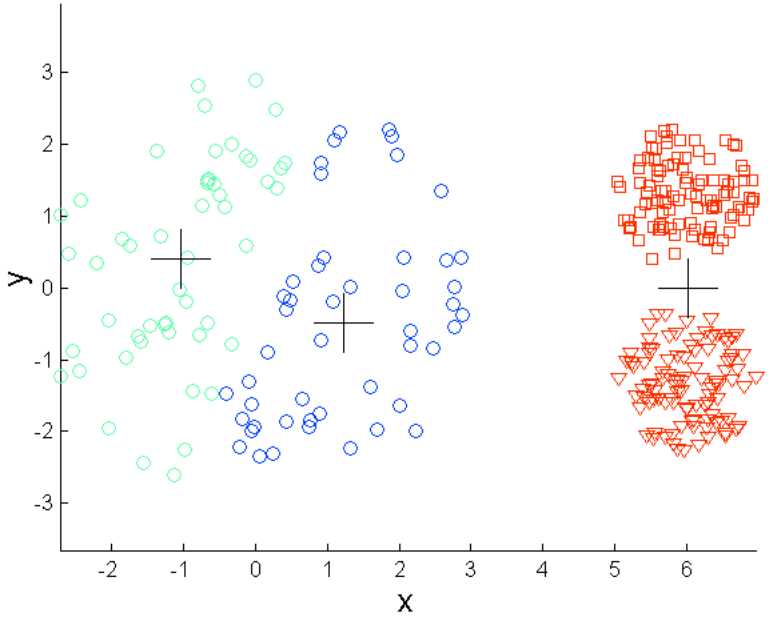


K-means (3 Clusters)

Limitations of K-means: Differing Density

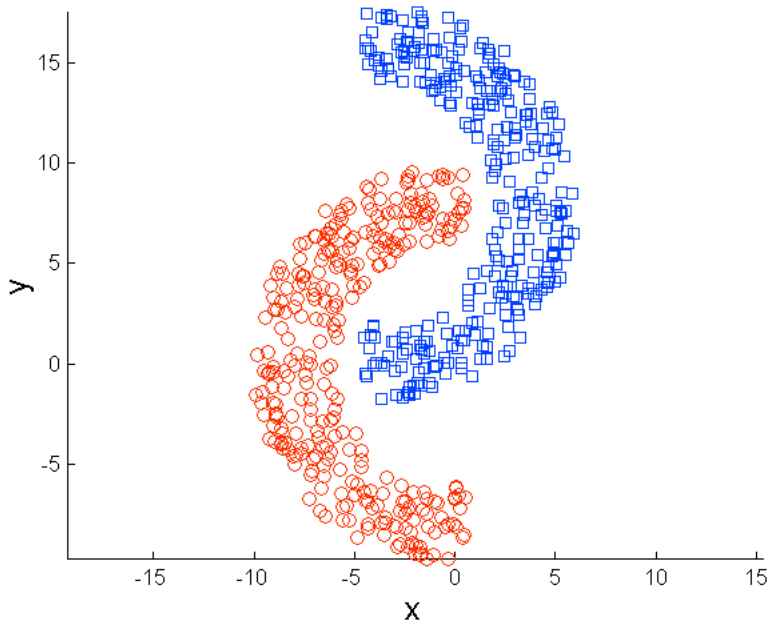


Original Points

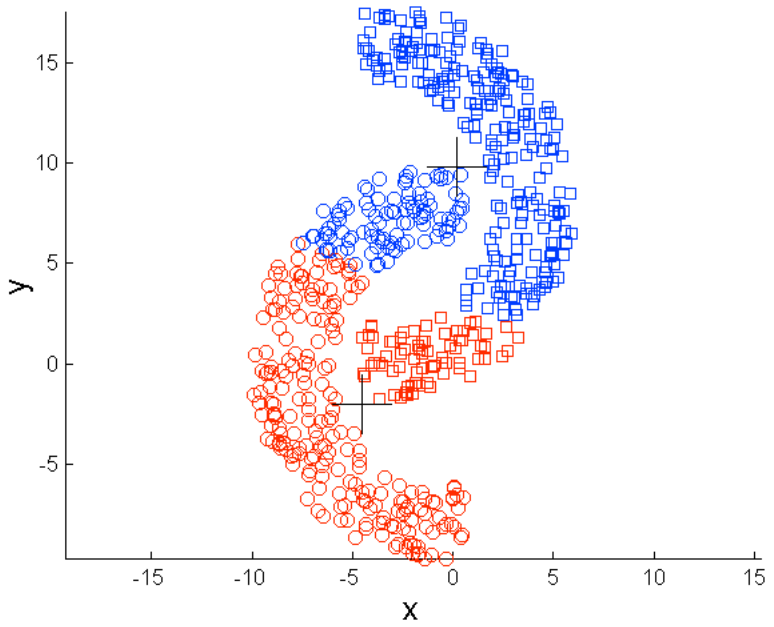


K-means (3 Clusters)

Limitations of K-means: Non-globular Shapes



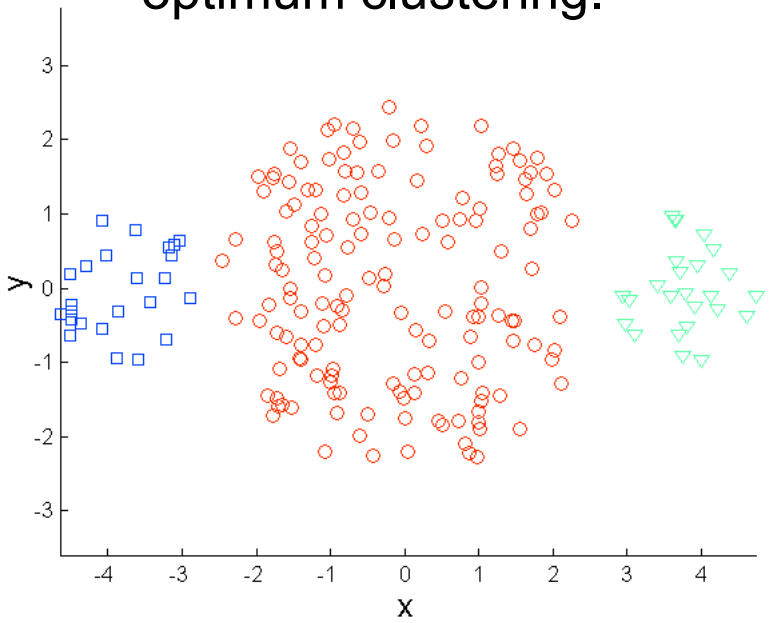
Original Points



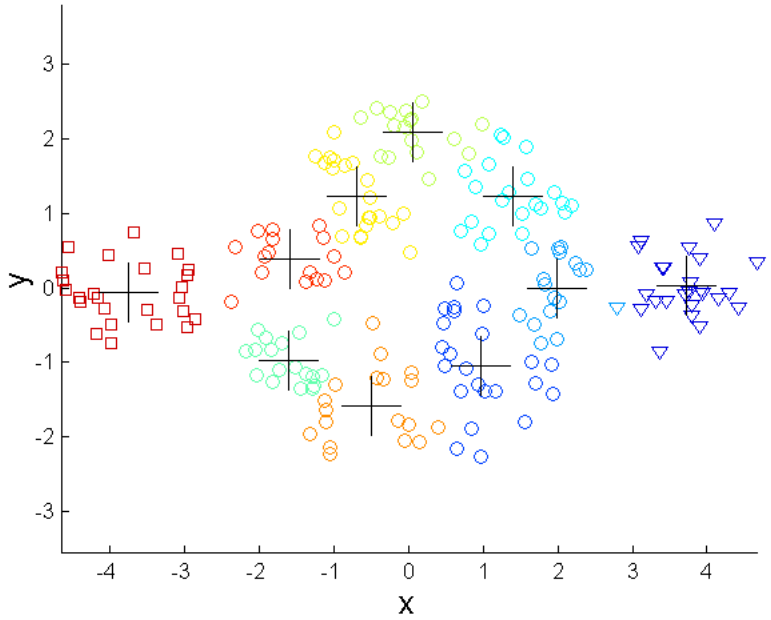
K-means (2 Clusters)

Overcoming K-means Limitations

One solution is to use many clusters.
Or work for more clusters than the expected number. These small parts of clusters need to put together after that (it is like using both hierarchical and partitional algorithms in a **hybrid** format to reach optimum clustering.

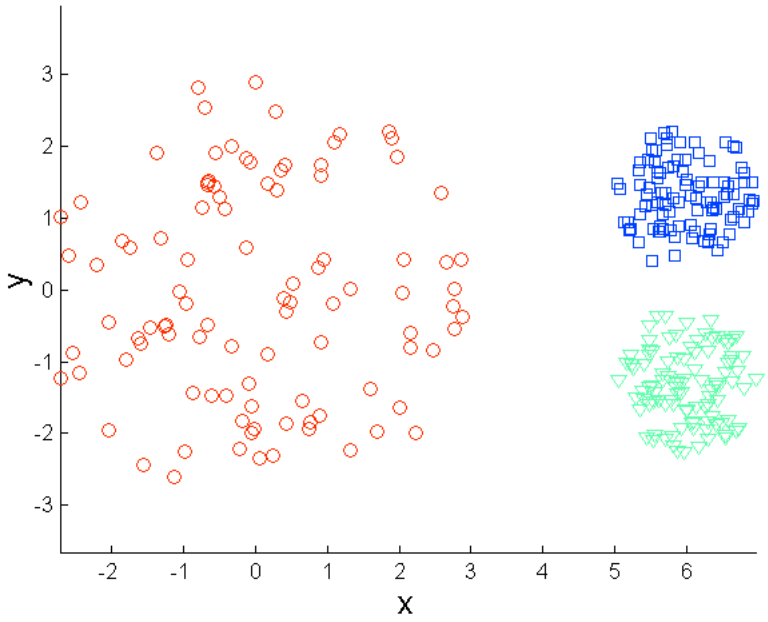


Original Points

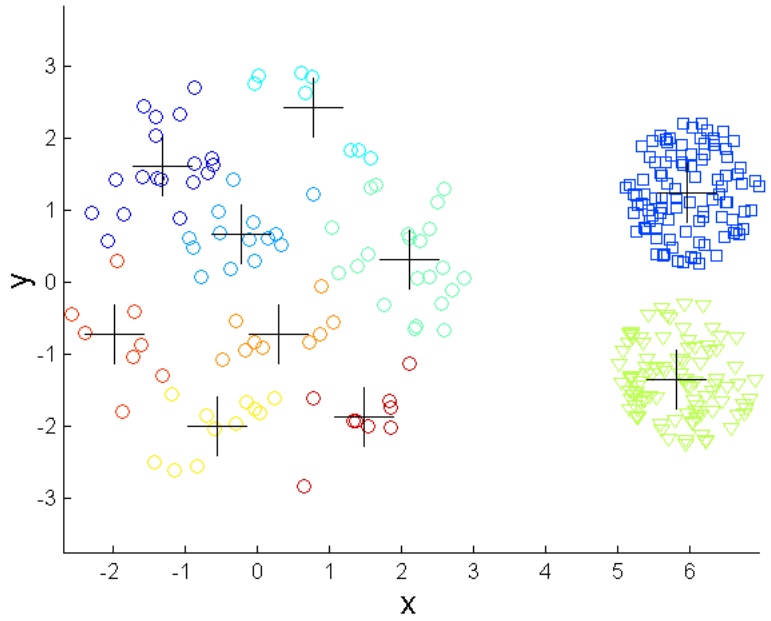


K-means Clusters

Overcoming K-means Limitations

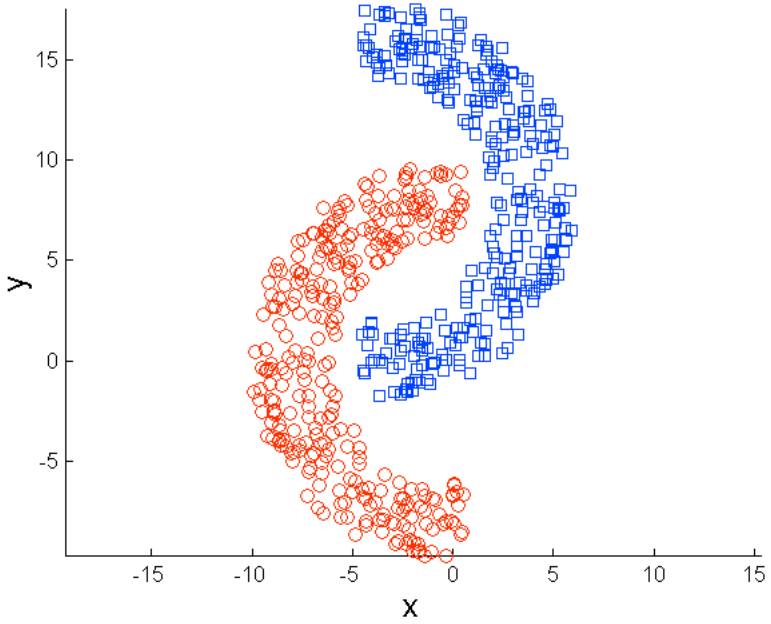


Original Points

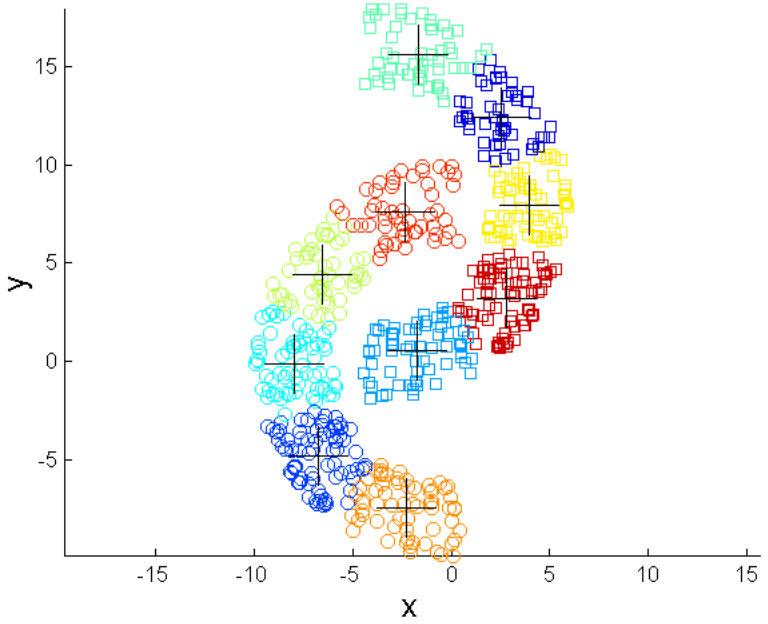


K-means Clusters

Overcoming K-means Limitations



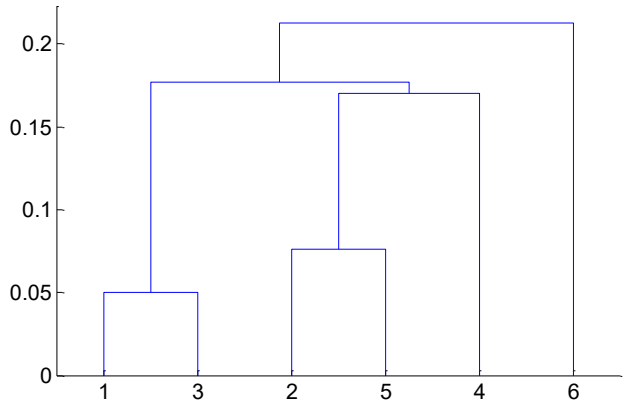
Original Points

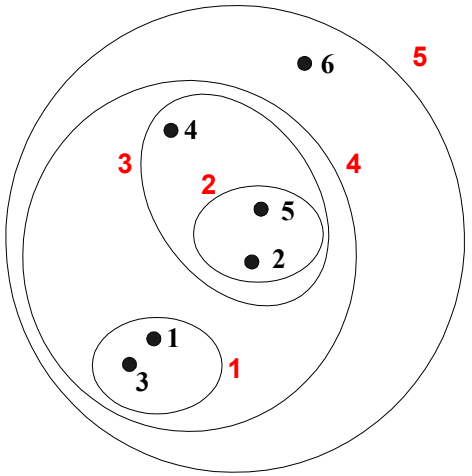


K-means Clusters

Hierarchical Clustering

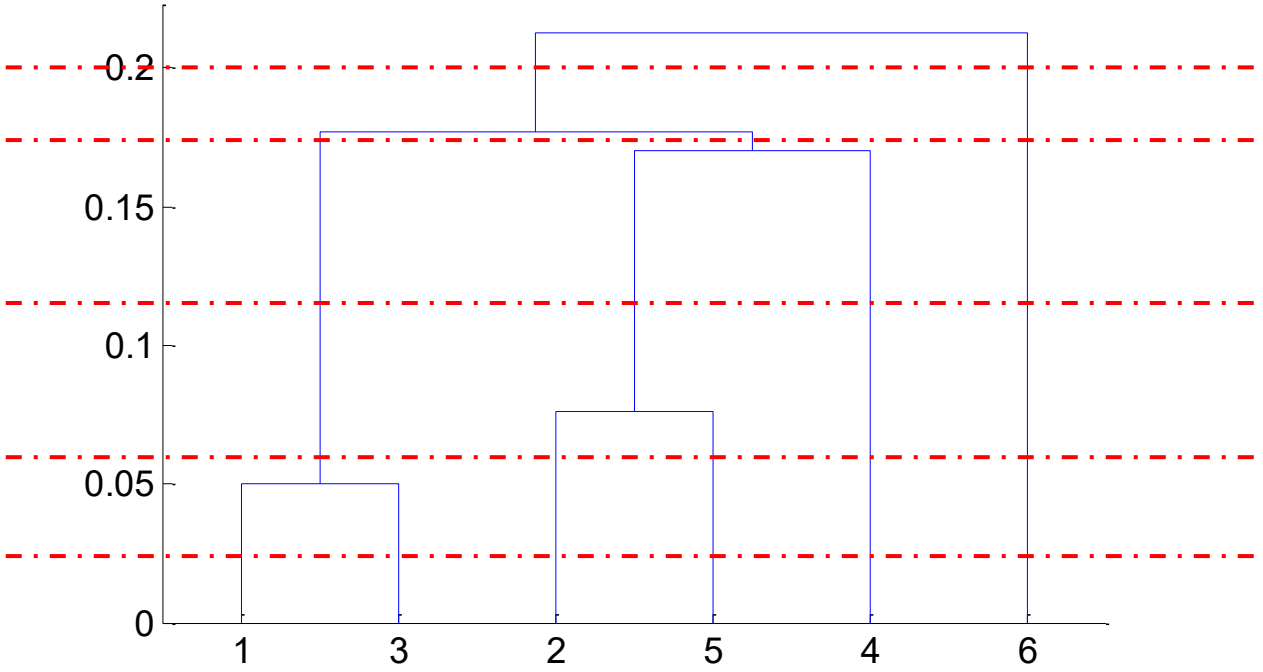
- Produces a set of nested clusters organized as a hierarchical tree
- Can be visualized as a dendrogram
  - A tree like diagram that records the sequences of merges or splits





Strengths of Hierarchical Clustering

- Do not have to assume any particular number of clusters or centroids.
  - Any desired number of clusters can be obtained by ‘cutting’ the dendrogram at the proper level



Hierarchical Clustering

- **Two main types of hierarchical clustering**
  - Agglomerative:
    - Start with the points as individual clusters
    - At each step, merge the closest pair of clusters until only one cluster (or k clusters) left
  - Divisive:
    - Start with one, all-inclusive cluster
    - At each step, split a cluster until each cluster contains a point (or there are k clusters)
- **Traditional hierarchical algorithms use a similarity or distance matrix**
  - Merge or split one cluster at a time

Agglomerative Clustering Algorithm

	P1	P2	P3
P1	0		
P2		0	
P3			0

Distance Matrix measured by ED for example

- Most popular hierarchical clustering technique
- Basic algorithm is straightforward as follows
  - 1. Compute the proximity matrix (Similarity)
  - 2. Let each data point be a cluster
  - 3. Repeat
    - 3.1 Merge the two closest clusters
    - 3.2 Update the proximity matrix
  - 4. Continue Until only a single cluster remains

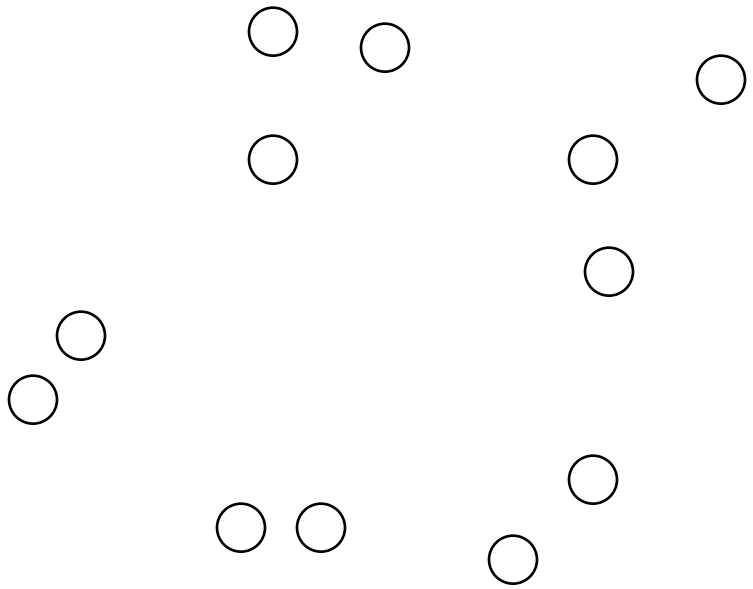
	P1	P2	P3
P1	1		
P2		1	
P3			1

Similarity Matrix measured by Corr for example

- **Key operation is the computation of the proximity of two clusters**
  - **Different approaches to defining the distance between clusters distinguish the different algorithms**

Starting Situation

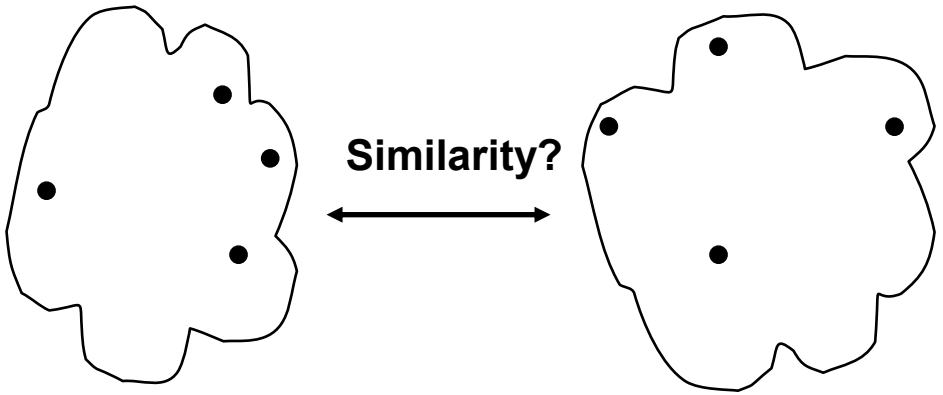
- Start with clusters of individual points and a proximity matrix



	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

How to Define Inter-Cluster Similarity

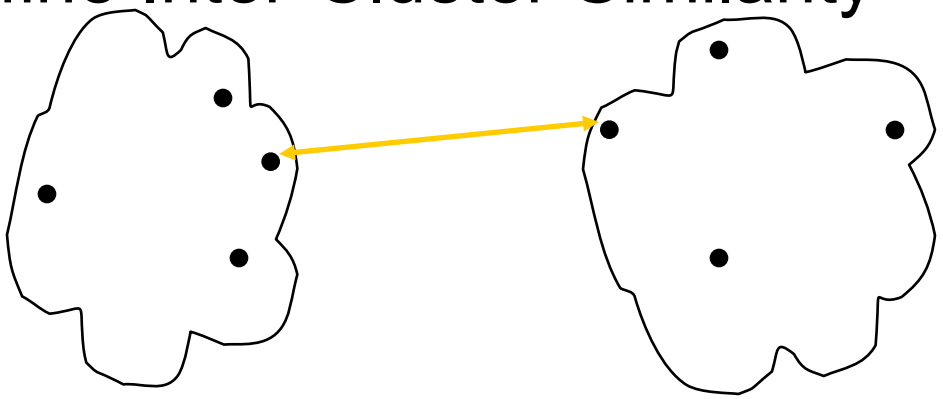


- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
  - Ward's Method uses squared error

	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						

Proximity Matrix

How to Define Inter-Cluster Similarity

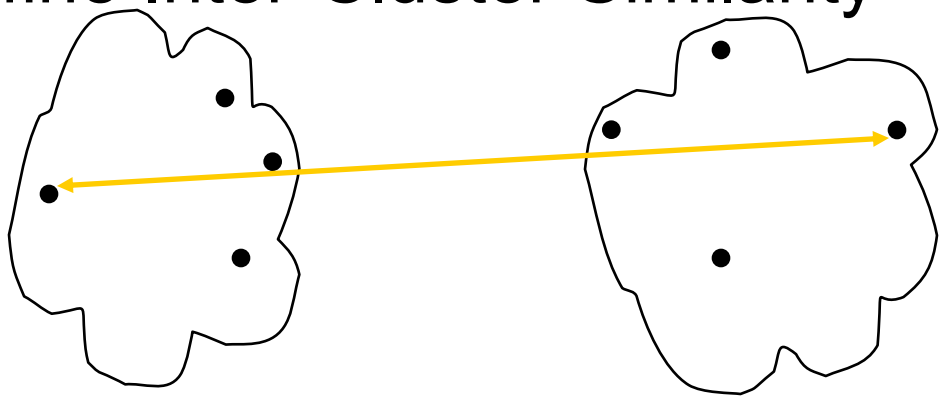


- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
  - Ward's Method uses squared error

	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

How to Define Inter-Cluster Similarity

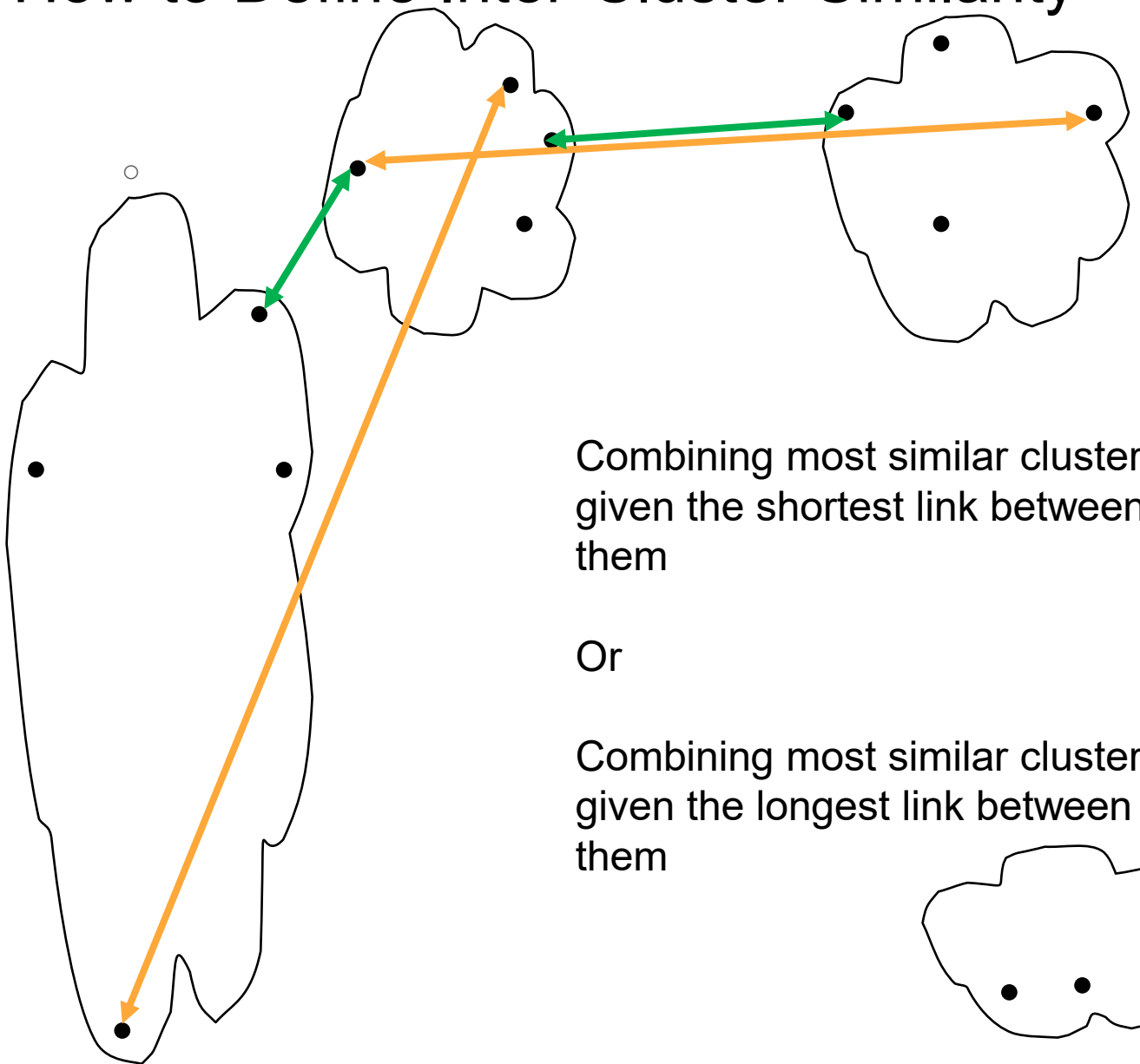


- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
  - Ward's Method uses squared error

	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

How to Define Inter-Cluster Similarity



Combining most similar clusters given the shortest link between them

Or

Combining most similar clusters given the longest link between them

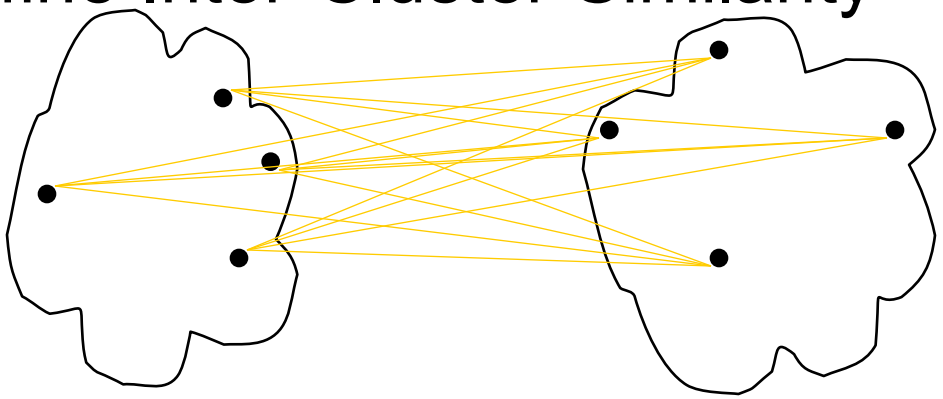
	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						

.

.

Proximity Matrix

How to Define Inter-Cluster Similarity

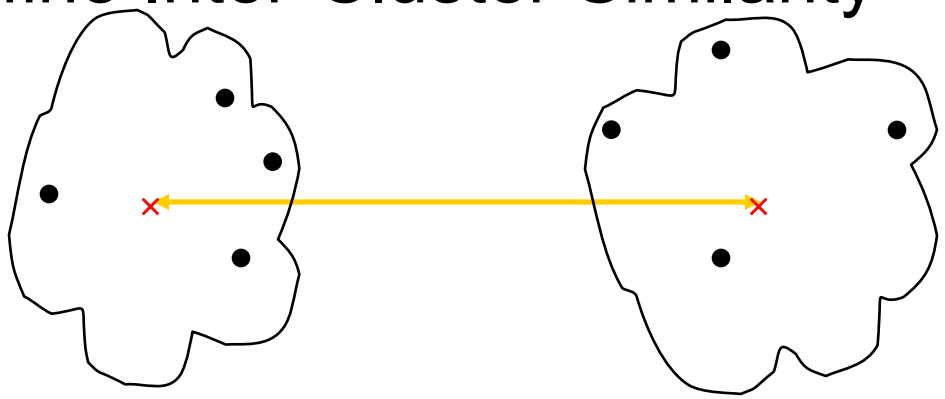


- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
  - Ward's Method uses squared error

	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

How to Define Inter-Cluster Similarity



- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
  - Ward's Method uses squared error

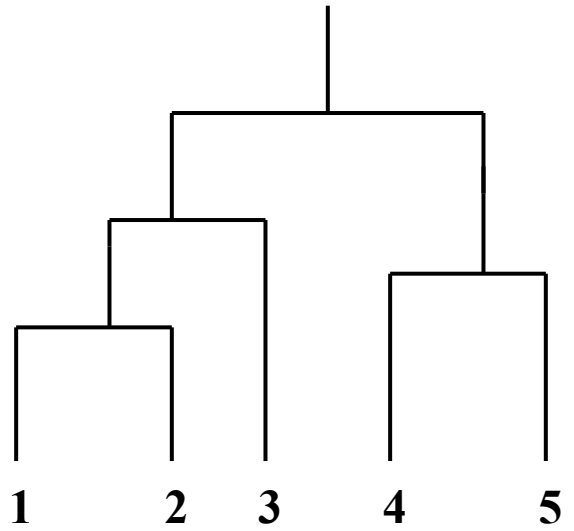
	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

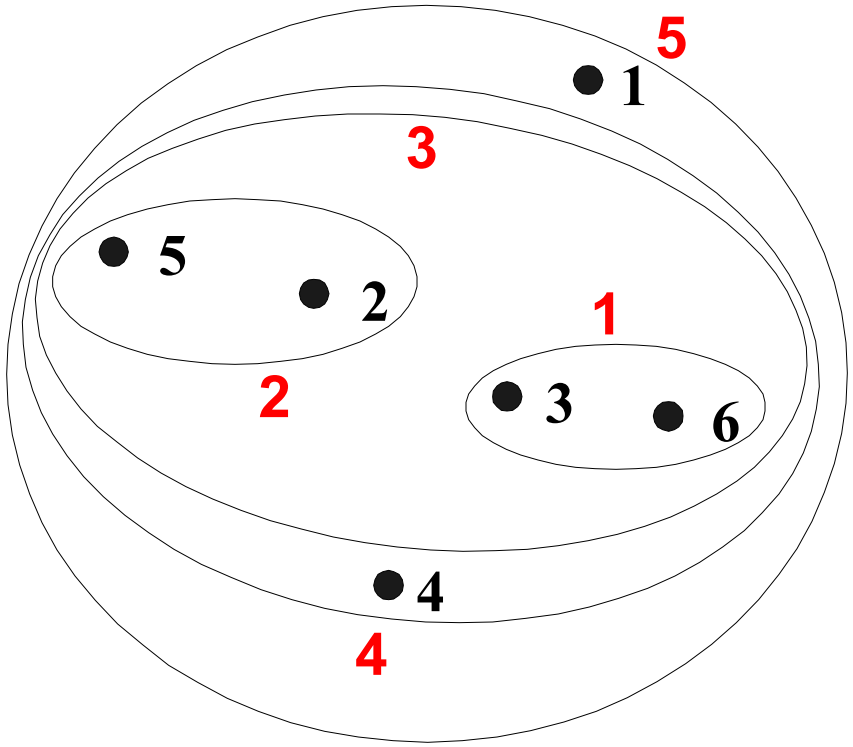
Cluster Similarity: MIN or Single Link

- Similarity of two clusters is based on the two most similar (closest) points in the different clusters
  - Determined by one pair of points, i.e., by one link in the proximity graph.

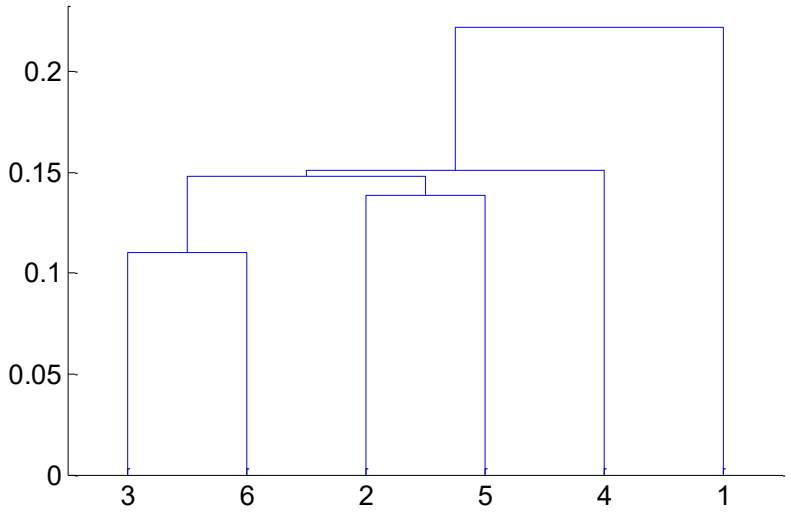
	I1	I2	I3	I4	I5
I1	1.00	0.90	0.10	0.65	0.20
I2	0.90	1.00	0.70	0.60	0.50
I3	0.10	0.70	1.00	0.40	0.30
I4	0.65	0.60	0.40	1.00	0.80
I5	0.20	0.50	0.30	0.80	1.00



Hierarchical Clustering: MIN



Nested Clusters

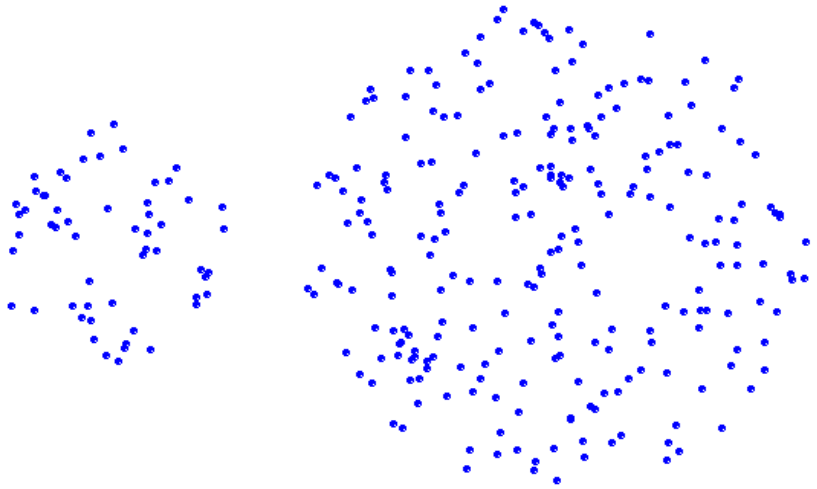


Dendrogram

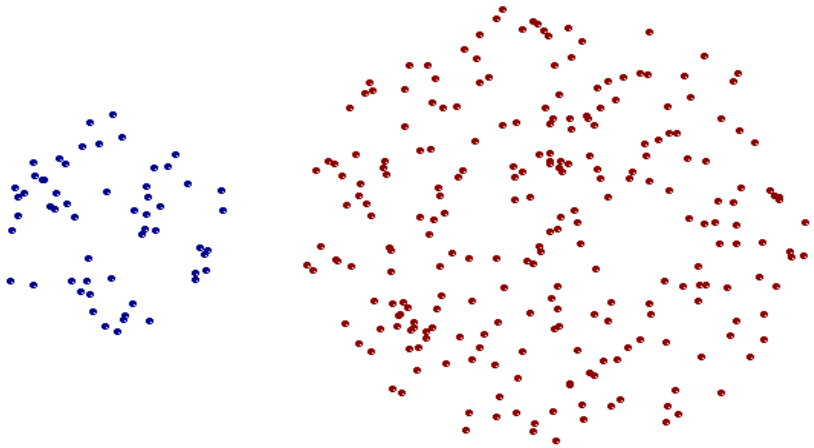
Single Linkage (Min Link)

- **Pros**:
- Can handle clusters with non-globular shapes.
- Can find elongated and irregularly shaped clusters.
- **Cons**:
- Tends to suffer from the "chaining effect," where clusters may be merged due to a series of points forming a bridge between two clusters, leading to long and loose clusters.
- Sensitive to noise and outliers.
- **Suitable For**: Datasets where the goal is to find clusters with non-globular shapes and where elongated clusters are expected.

Strength of MIN



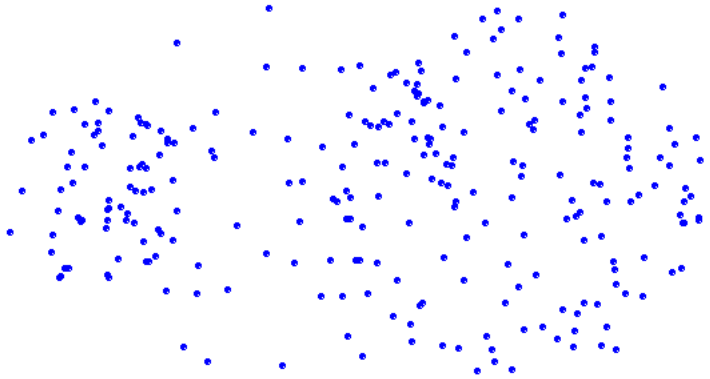
Original Points



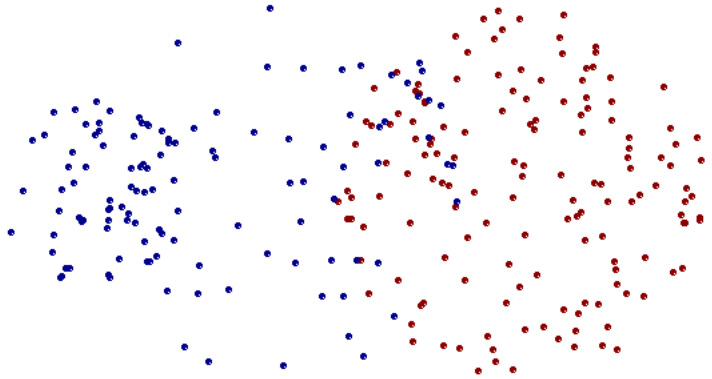
Two Clusters

• **Can handle non-elliptical shapes:** non-elliptic (stretched) shapes are often used informally to describe shapes that are elongated or stretched but lack the precise mathematical definition of an ellipse.

Limitations of MIN



Original Points



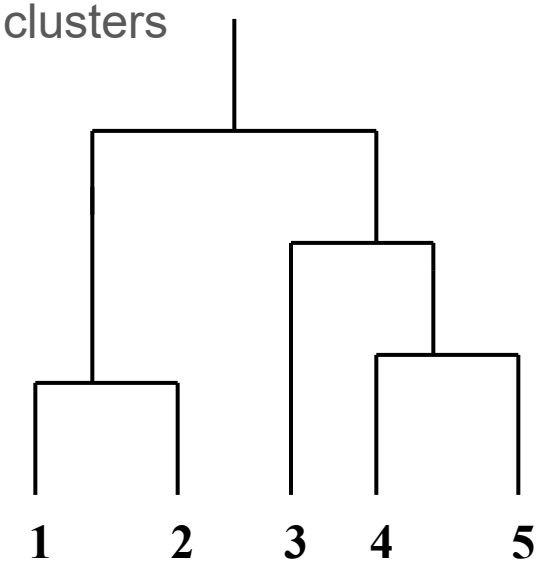
Two Clusters

- **Sensitive to noise and outliers (**chaining effect)

Cluster Similarity: MAX or Complete Linkage

- In complete linkage clustering, the distance between two clusters is defined as the maximum distance between any single point in the first cluster and any single point in the second cluster
- Meaning that, Similarity of two clusters is based on the two least similar (most distant) points in the different clusters
    - Determined by all pairs of points in the two clusters

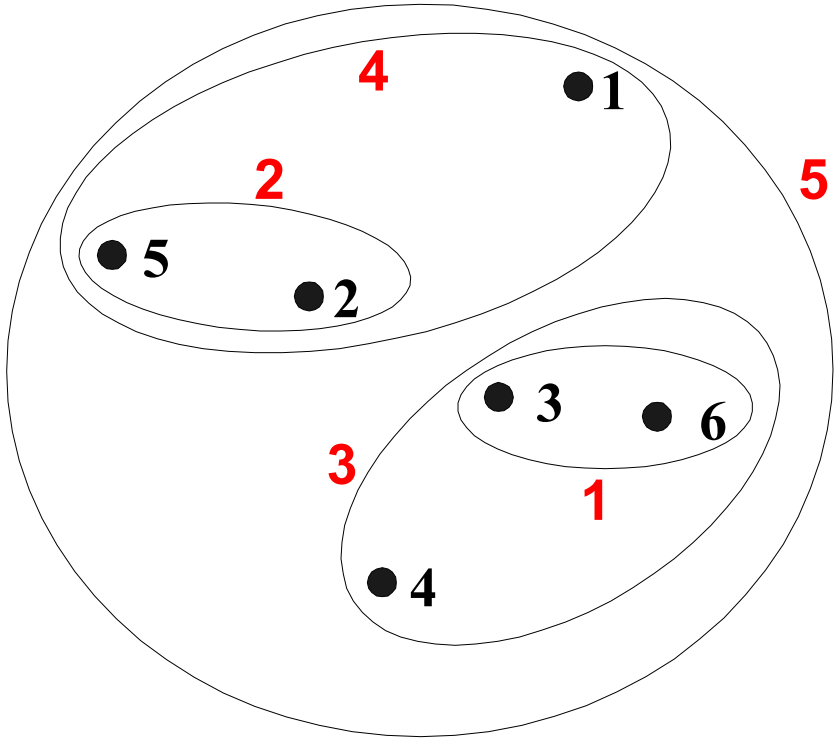
	I1	I2	I3	I4	I5
I1	1.00	0.90	0.10	0.65	0.20
I2	0.90	1.00	0.70	0.60	0.50
I3	0.10	0.70	1.00	0.40	0.30
I4	0.65	0.60	0.40	1.00	0.80
I5	0.20	0.50	0.30	0.80	1.00



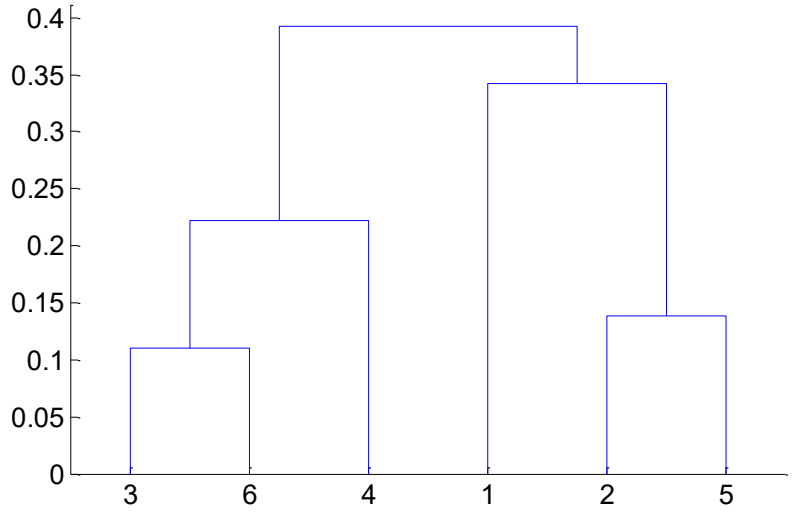
Complete Linkage (Max Link)

- **Pros**:
- Produces more compact clusters compared to single linkage.
- Less sensitive to noise and outliers compared to single linkage.
- **Cons**:
- Tends to force clusters to be more spherical or globular.
- Can sometimes break larger clusters into smaller ones if they are not compact (dense enough).
- **Suitable For**: Datasets where the goal is to find compact, globular clusters and where robustness to noise is desired.

Hierarchical Clustering: MAX

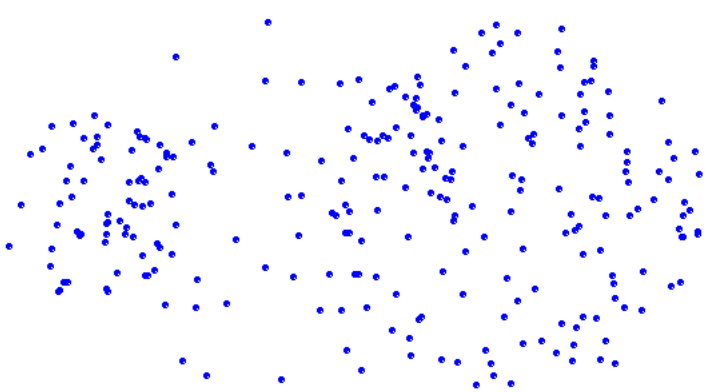


Nested Clusters

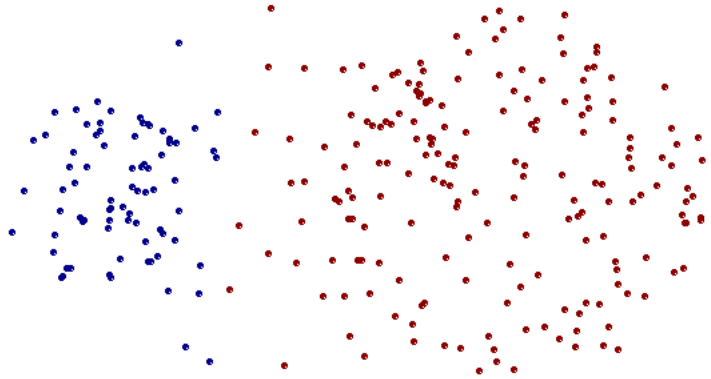


Dendrogram

Strength of MAX



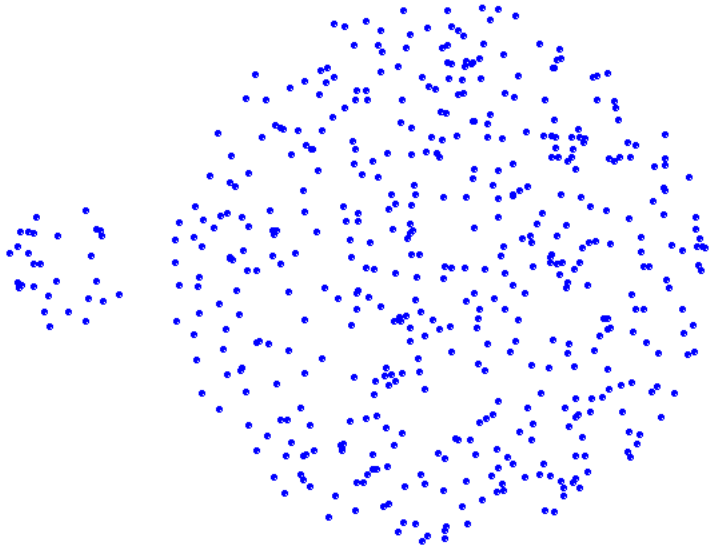
Original Points

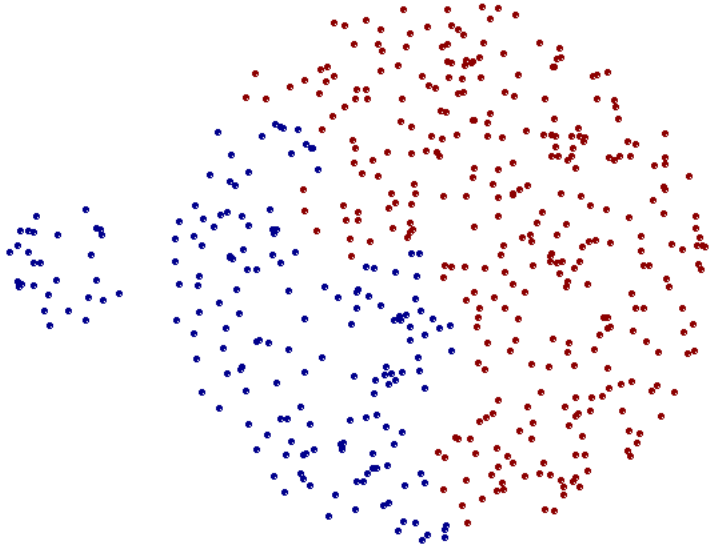


Two Clusters

- **Less susceptible to noise and outliers**

Limitations of MAX





Original Points

Two Clusters

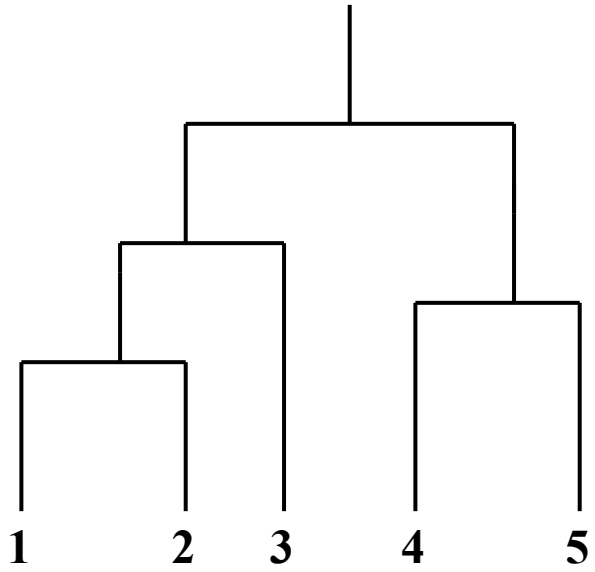
- **Tends to break large clusters**
- **Biased towards globular clusters**

Cluster Similarity: Group Average

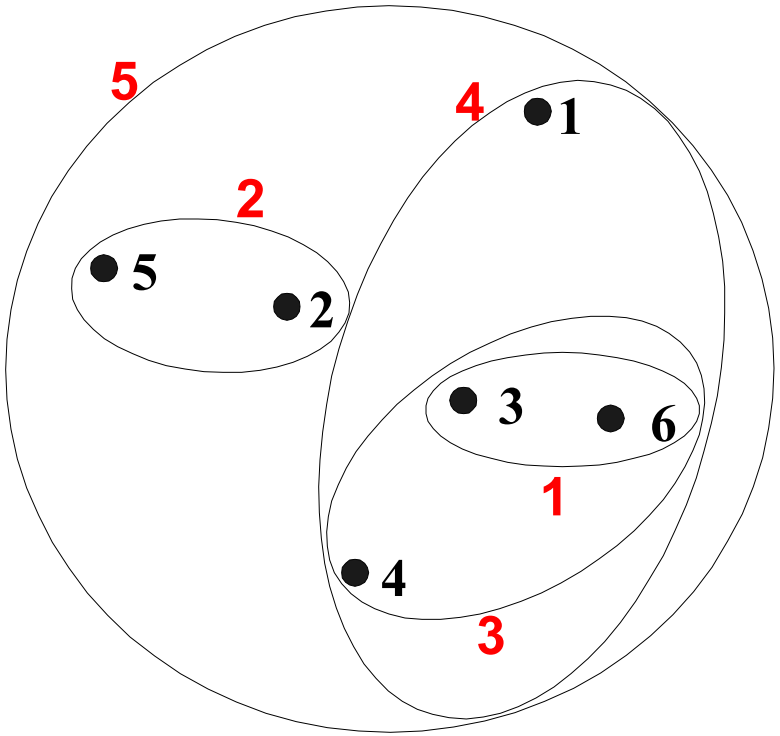
- Proximity of two clusters is the average of pairwise proximity between points in the two clusters.

- Need to use average connectivity for scalability since total proximity favors large clusters

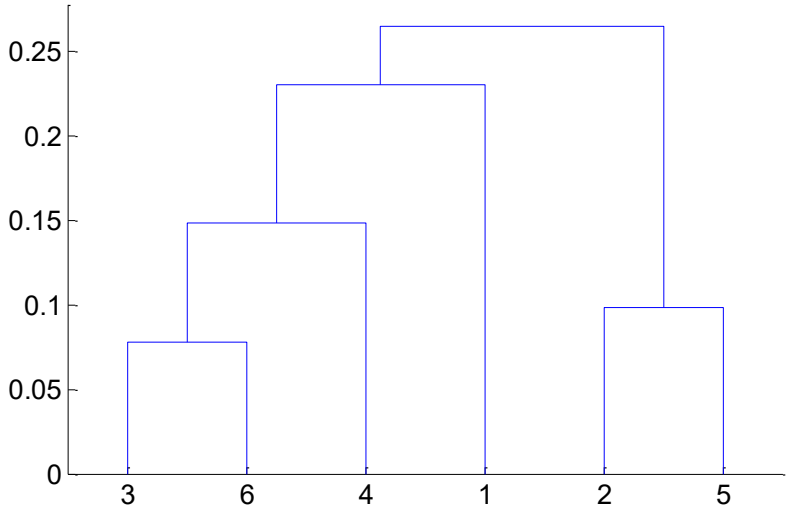
	I1	I2	I3	I4	I5
I1	1.00	0.90	0.10	0.65	0.20
I2	0.90	1.00	0.70	0.60	0.50
I3	0.10	0.70	1.00	0.40	0.30
I4	0.65	0.60	0.40	1.00	0.80
I5	0.20	0.50	0.30	0.80	1.00



Hierarchical Clustering: Group Average



Nested Clusters



Dendrogram

Average Linkage (Avg Link)

- **Pros**:
- Balances the properties of single and complete linkage.
- Less susceptible to chaining effect compared to single linkage.
- Produces relatively compact clusters compared to single linkage.
- **Cons**:
- May not perform well with clusters of varying sizes and densities.
- Still can be influenced by outliers, though less so than single linkage.
- **Suitable For**: Datasets where a balance between compactness and shape flexibility is needed, and where clusters of similar sizes and densities are expected.

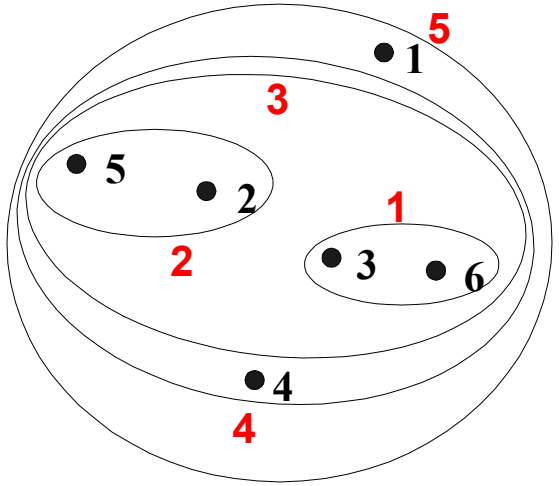
Cluster Similarity: Ward’s Method

- Similarity of two clusters is based on the increase in squared error when two clusters are merged
  - Similar to group average if distance between points is distance squared
- Less susceptible to noise and outliers
- Biased towards globular clusters
- Hierarchical analogue of K-means
  - Can be used to initialize K-means

Summary Table

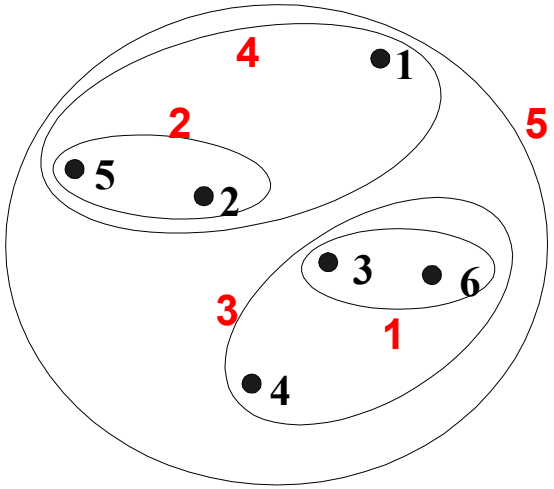
Method	Description	Suitable For	Pros	Cons
Single Linkage	Min distance between any point in the two clusters	Non-globular, elongated clusters	Handles irregular shapes	Chaining effect, sensitive to noise
Complete Linkage	Max distance between any point in the two clusters	Compact, globular clusters	Produces compact clusters, less sensitive to noise	Forces globular-ity may split large clusters if not dense enough
Average Linkage	Average distance between all pairs of points	Balanced cluster shapes	Balances compactness and flexibility	Struggles with varying sizes and densities
Ward's Method	Minimizes total within-cluster variance	Compact, spherical clusters	Very compact clusters, robust to outliers	Assumes globular-ity, splits non-compact clusters

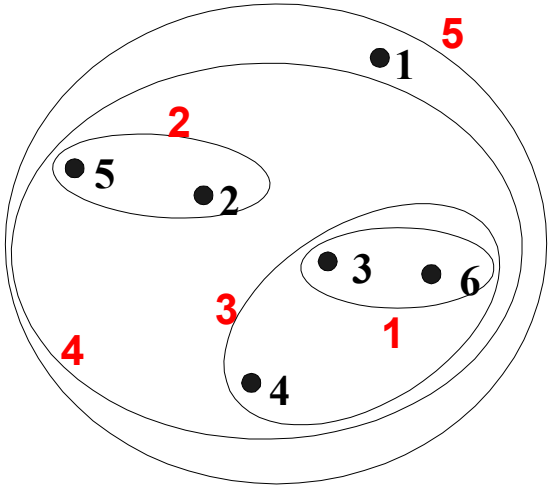
Hierarchical Clustering: Comparison



MIN

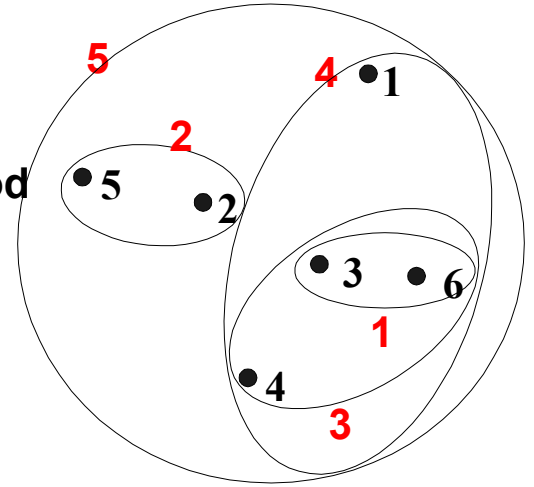
MAX

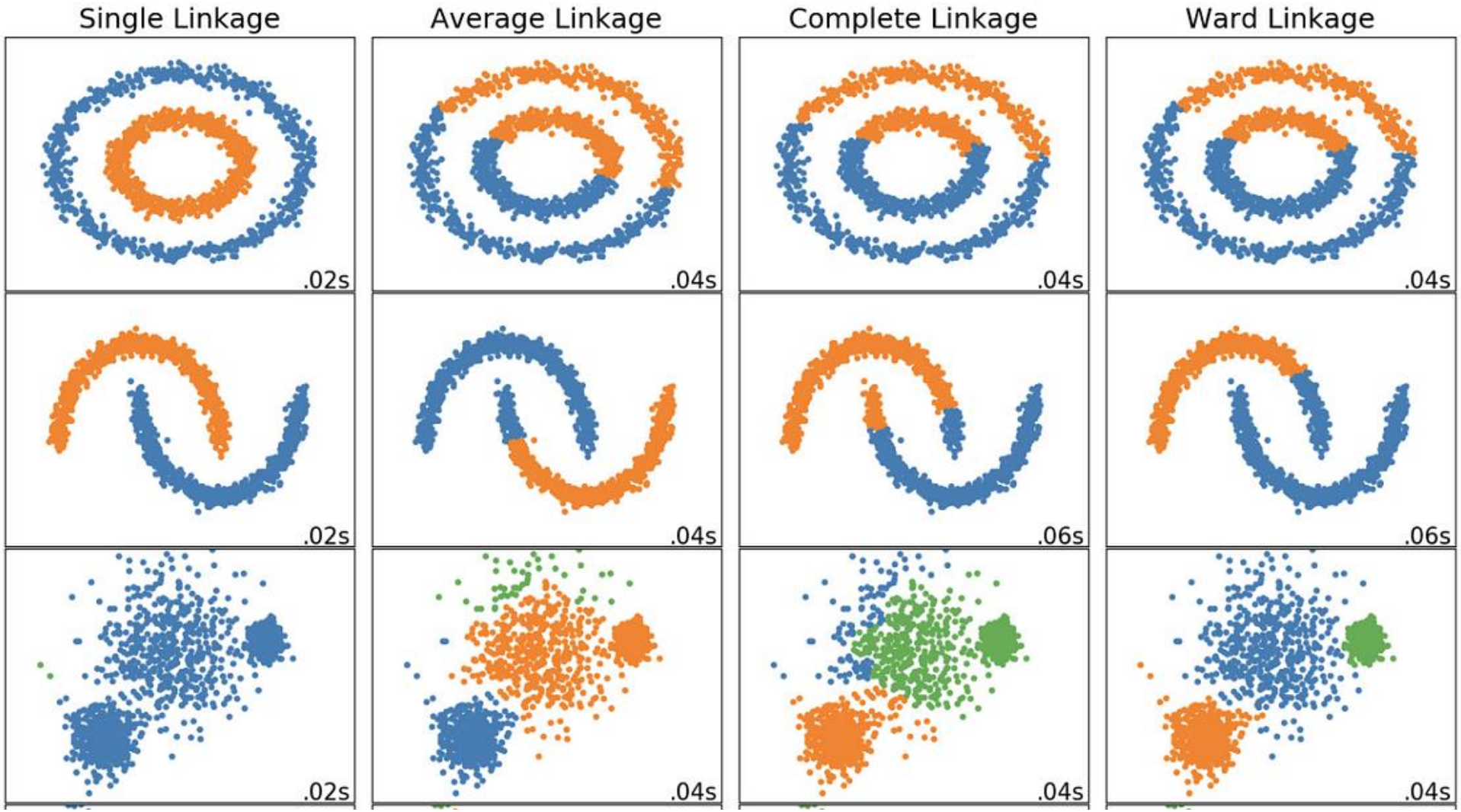




Group Average

Ward's Method




Single Linkage
Average Linkage
Complete Linkage
Ward Linkage
.02s
.04s
.04s
.04s
.02s
.04s
.06s
.06s
.02s
.04s
.04s
.04s

Hierarchical Clustering: Time and Space requirements

- $O(N^2)$ space requirements since it uses the proximity matrix.
  - N is the number of points.
- $O(N^3)$ time requirements in many cases
  - There are N steps and at each step the size, N^2, proximity matrix must be updated and searched
  - Complexity can be reduced to $O(N^2 \log(N))$ for some approaches by using more sophisticated data structures like **priority queues**, which can **help in more efficiently finding the closest pair of clusters** .

Hierarchical Clustering: Problems and Limitations

- Once a decision is made to combine two clusters, it cannot be undone
- No objective function is directly minimized, unlike to what we learnt in k-means with Sum of Squared Errors.
- Different schemes have problems with one or more of the following:
  - Sensitivity to noise and outliers (ex. Agglomerative model with single link)
  - Difficulty handling different sized clusters and convex shapes (ex. Agglomerative model with Average/complete link)
  - Breaking large clusters (ex. Agglomerative model with complete link)

DBSCAN (Density-based Clustering)

- DBSCAN is a density-based algorithm.

It can identify clusters in large datasets by **grouping together points that are closely packed together, while marking as outliers the points that lie alone in low-density regions.**

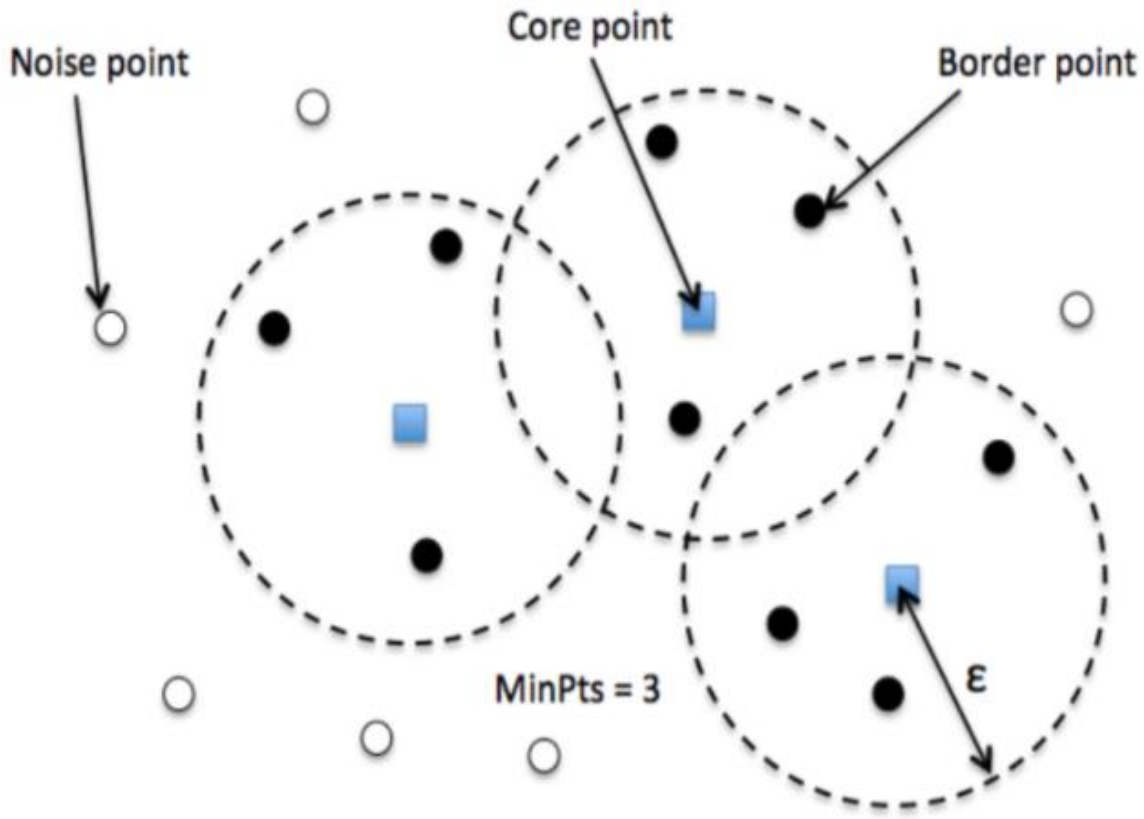
- To proceed with a DBSCAN algorithm, you need to define two parameters:

1. **eps (epsilon),** the maximum distance between two points for them to be considered as in the same neighborhood,

2. **minPts**, the minimum number of points required to form a dense region (i.e., for a region to be considered a cluster).

   - Both parameters are required In order to be able to compute the **Density = count of points within a specified radius (Eps). This count can be greater than, less than or equal to MinPts**.

DBSCAN: Core, Border, and Noise Points

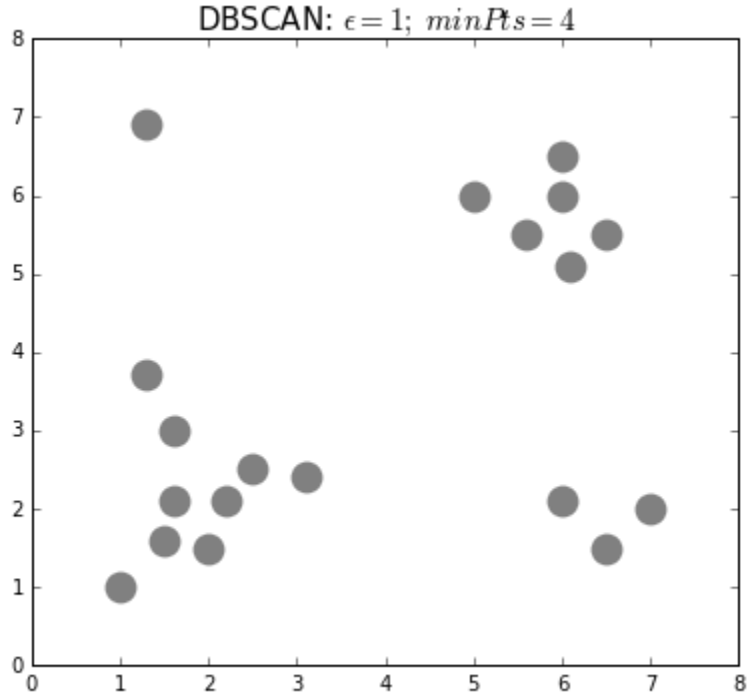


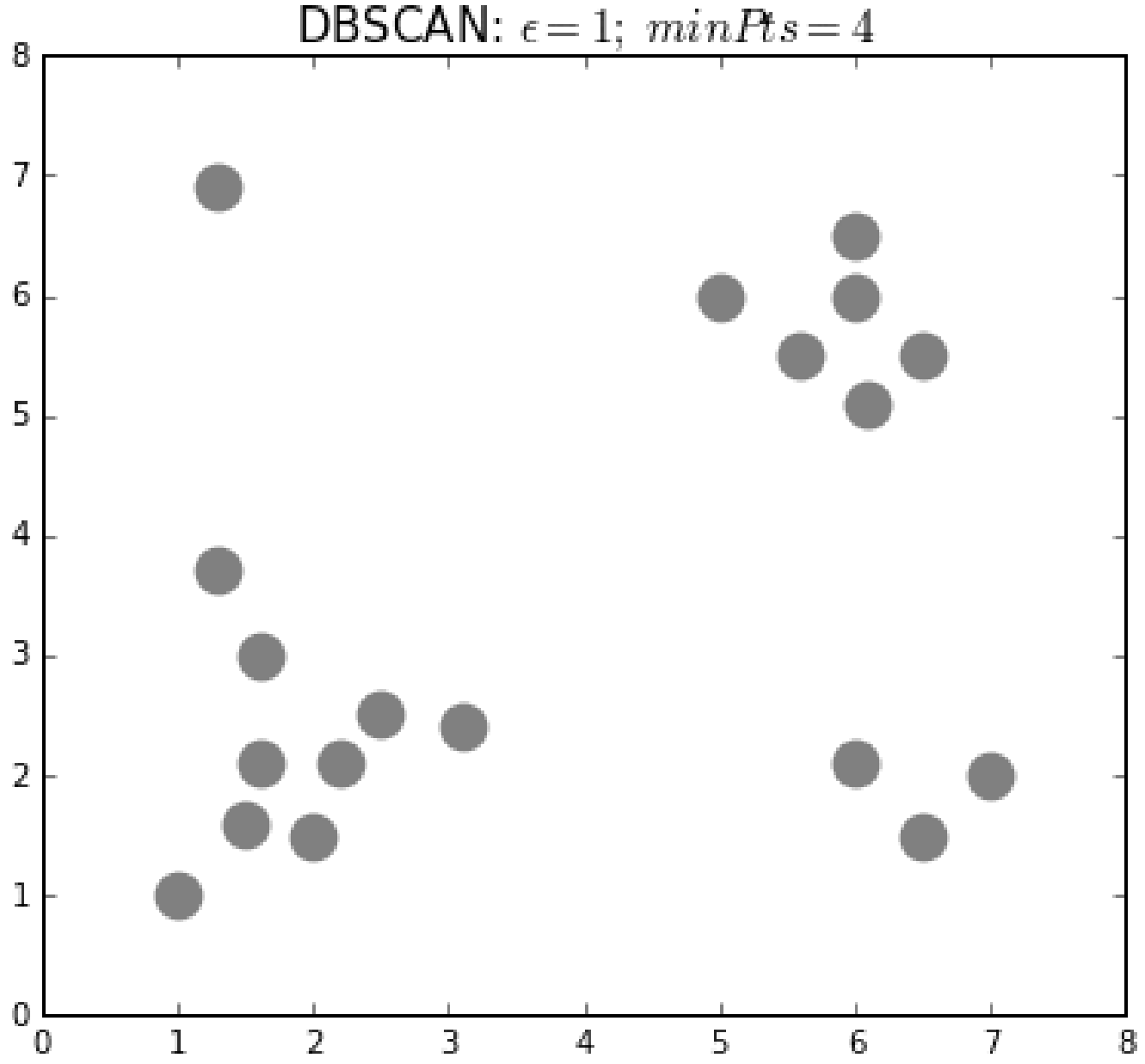
A point is considered **a core point** if at least minPts points are within distance eps of it (including the point itself). These are points that are located inside the clusters.

A **border point** is not a core point itself, but lies within distance eps of a core point. Border points are near to or on the edges of clusters.

All other points are considered **noise points** or outliers.

- The algorithm proceeds by arbitrarily picking up a point in the dataset (until all points have been visited).
- If there are at least ‘minPoint’ points within a radius of ‘ε’ to the point then we consider all these points to be part of the same cluster.
- The clusters are then expanded by recursively repeating the neighborhood calculation for each neighboring point




DBSCAN: $\epsilon = 1;\ minPts = 4$
0
1
2
3
4
5
6
7
8

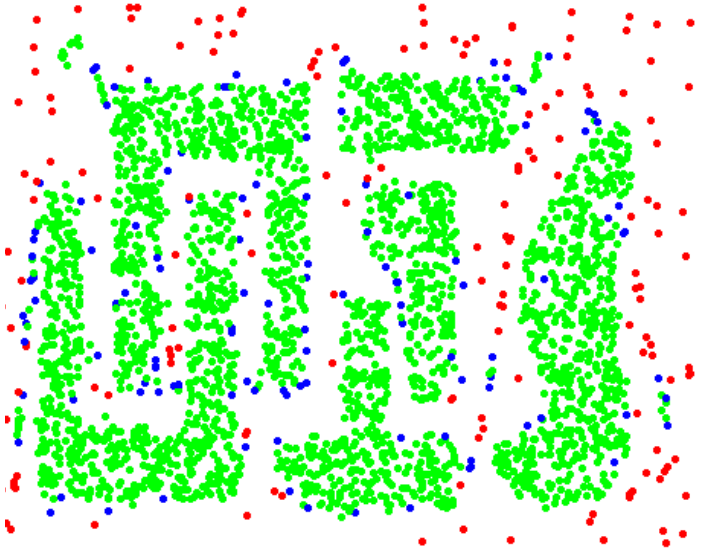
DBSCAN Algorithm

- Eliminate noise points
- Perform clustering on the remaining points

$current\_cluster\_label \leftarrow 1$
for all core points **do**
  if the core point has no cluster label **then**
    $current\_cluster\_label \leftarrow current\_cluster\_label + 1$
    Label the current core point with cluster label $current\_cluster\_label$
  end if
  for all points in the Eps-neighborhood, except i^{th} the point itself **do**
    if the point does not have a cluster label **then**
      Label the point with cluster label $current\_cluster\_label$
    end if
  end for
end for

DBSCAN: Core, Border and Noise Points

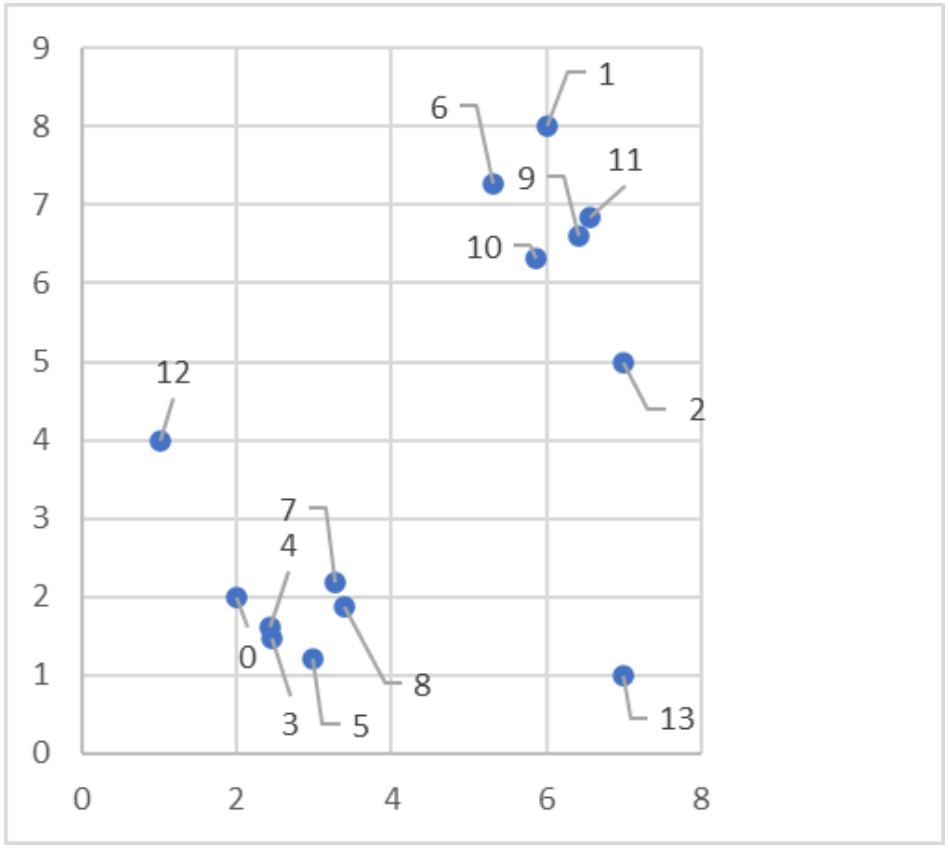
Original Points



Eps = 10, MinPts = 4

Point types: core, border and noise

Point	x-coordinate	y-coordinate
0	2	2
1	6	8
2	7	5
3	2.446	1.477
4	2.431	1.622
5	2.991	1.208
6	5.31	7.272
7	3.264	2.187
8	3.387	1.881
9	6.414	6.612
10	5.854	6.329
11	6.558	6.851
12	1	4
13	7	1



Exercise: Considering the following data points, apply DBSCAN clustering algorithm where EPS = 2 and MinPts = 6.

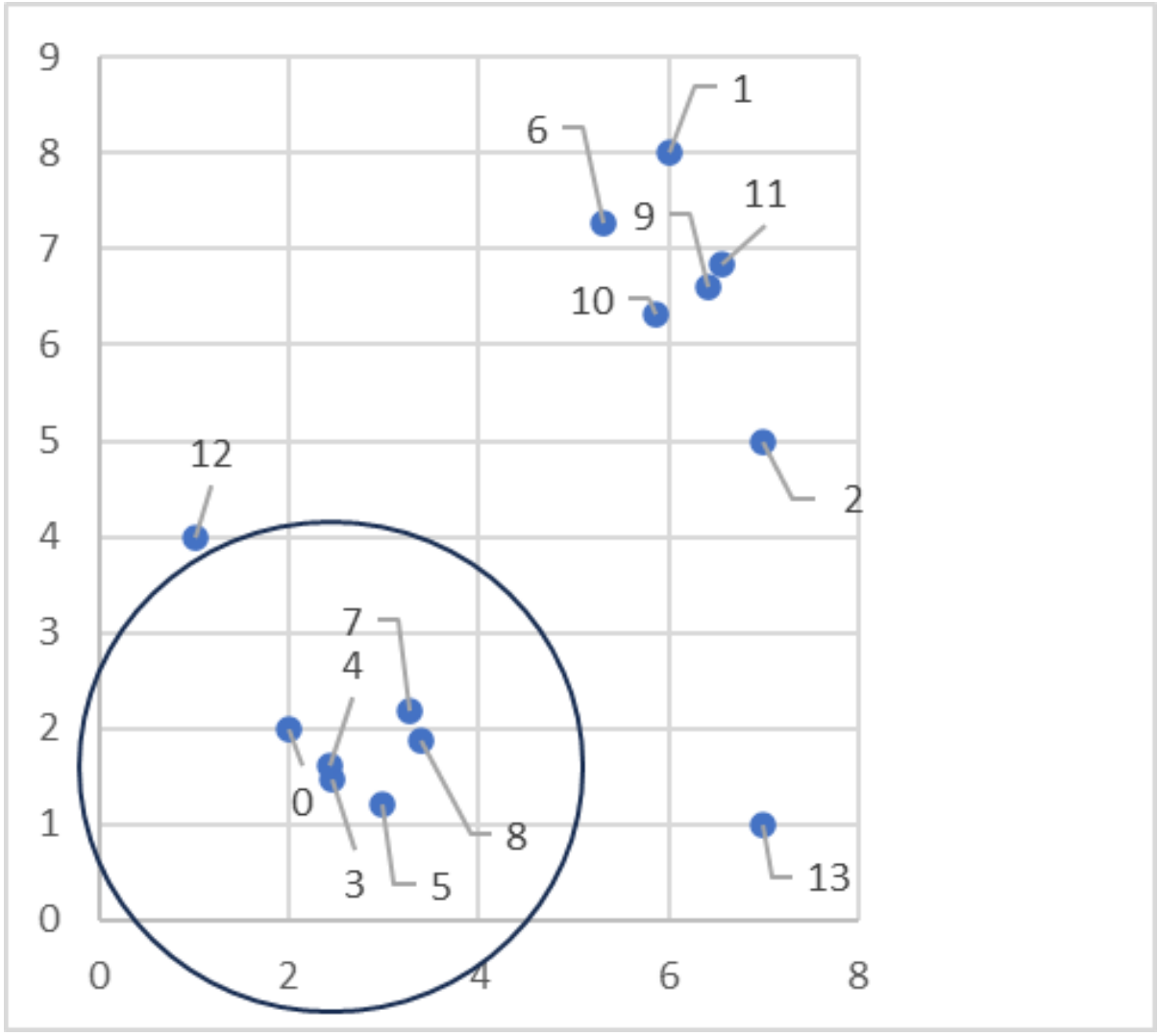
Q5.1) How many clusters are resulting?

Q5.2) Which points will belong to each cluster, show you answer on the scatter graph?

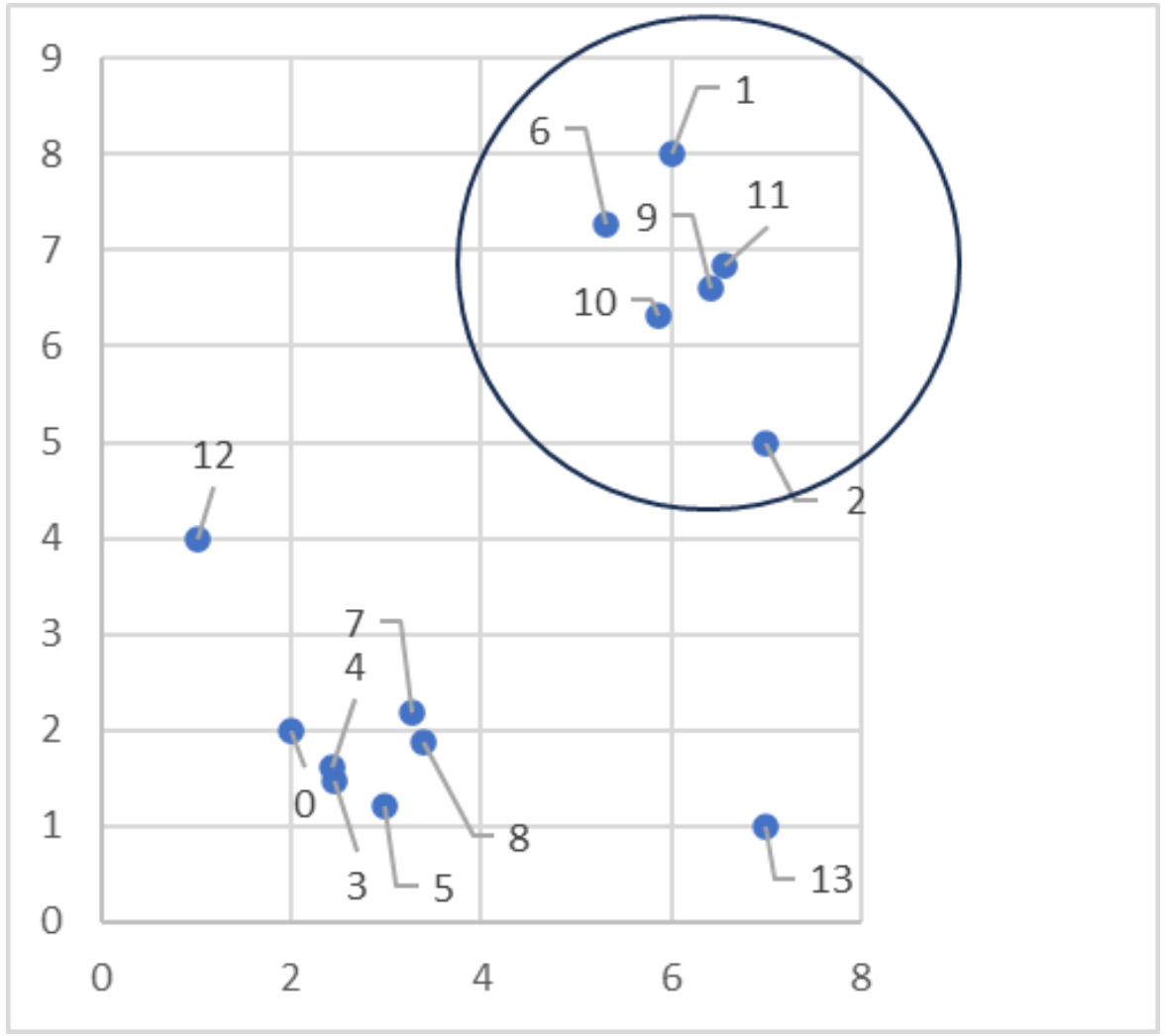
Q5.3) Show on the graph which points will be categorized as Core, Border, and Noise point?

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	count (eps<2)
0	**0**	7.21	5.83	0.69	0.57	1.27	6.22	1.28	1.39	6.38	5.8	6.66	2.24	5.1	6
1	7.21	**0**	3.16	7.43	7.31	7.43	1	6.42	6.65	1.45	1.68	1.28	6.4	7.07	5
2	5.83	3.16	**0**	5.76	5.68	5.52	2.83	4.68	4.77	1.72	1.75	1.9	6.08	4	4
3	0.69	7.43	5.76	**0**	0.15	0.61	6.46	1.08	1.02	6.49	5.93	6.77	2.91	4.58	6
4	0.57	7.31	5.68	0.15	**0**	0.7	6.34	1.01	0.99	6.38	5.82	6.66	2.78	4.61	6
5	1.27	7.43	5.52	0.61	0.7	**0**	6.49	1.02	0.78	6.4	5.87	6.68	3.43	4.01	6
6	6.22	1	2.83	6.46	6.34	6.49	**0**	5.48	5.72	1.29	1.09	1.32	5.41	6.5	5
7	1.28	6.42	4.68	1.08	1.01	1.02	5.48	**0**	0.33	5.43	4.89	5.71	2.9	3.92	6
8	1.39	6.65	4.77	1.02	0.99	0.78	5.72	0.33	**0**	5.62	5.09	5.9	3.19	3.72	6
9	6.38	1.45	1.72	6.49	6.38	6.4	1.29	5.43	5.62	**0**	0.63	0.28	6.01	5.64	6
10	5.8	1.68	1.75	5.93	5.82	5.87	1.09	4.89	5.09	0.63	**0**	0.88	5.38	5.45	6
11	6.66	1.28	1.9	6.77	6.66	6.68	1.32	5.71	5.9	0.28	0.88	**0**	6.25	5.87	6
12	2.24	6.4	6.08	2.91	2.78	3.43	5.41	2.9	3.19	6.01	5.38	6.25	**0**	6.71	1
13	5.1	7.07	4	4.58	4.61	4.01	6.5	3.92	3.72	5.64	5.45	5.87	6.71	**0**	1

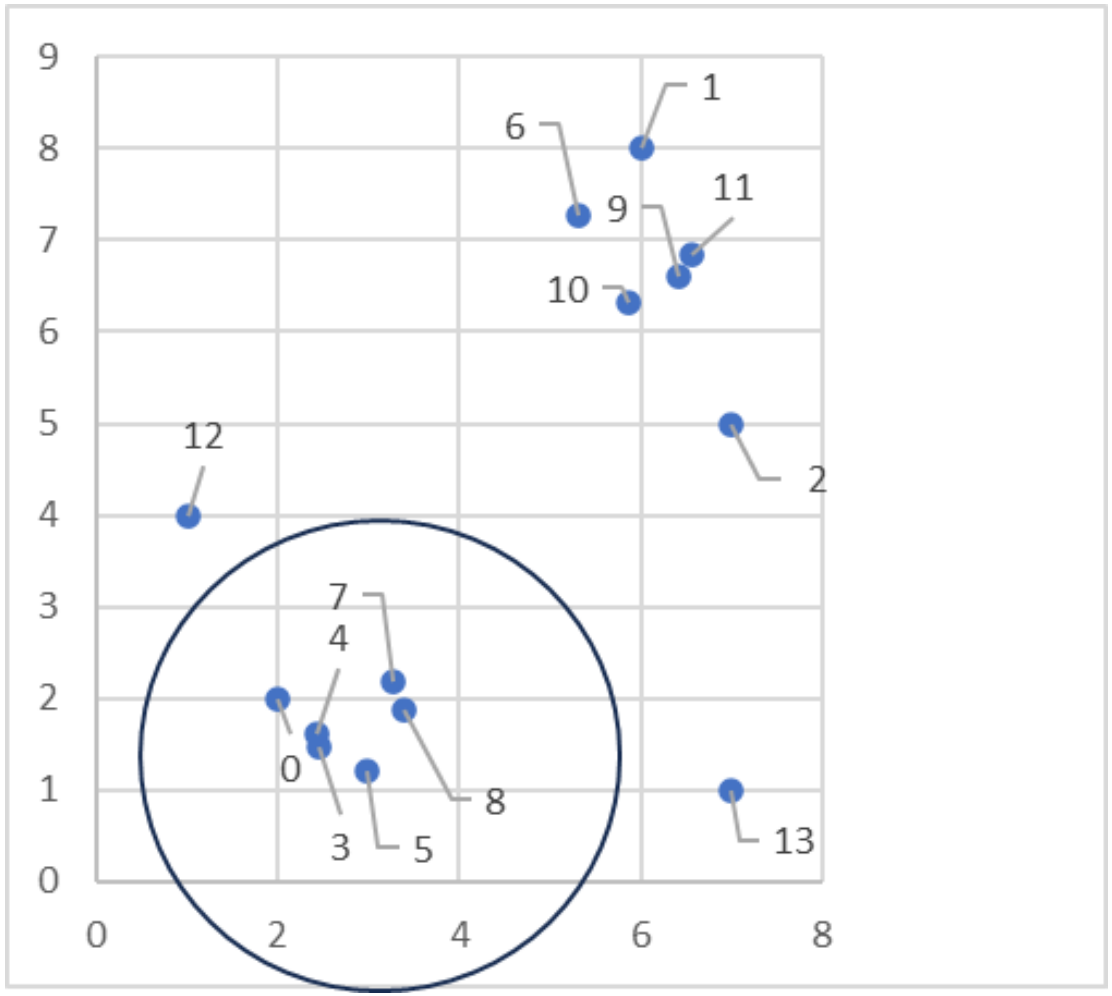
Point 0 is core

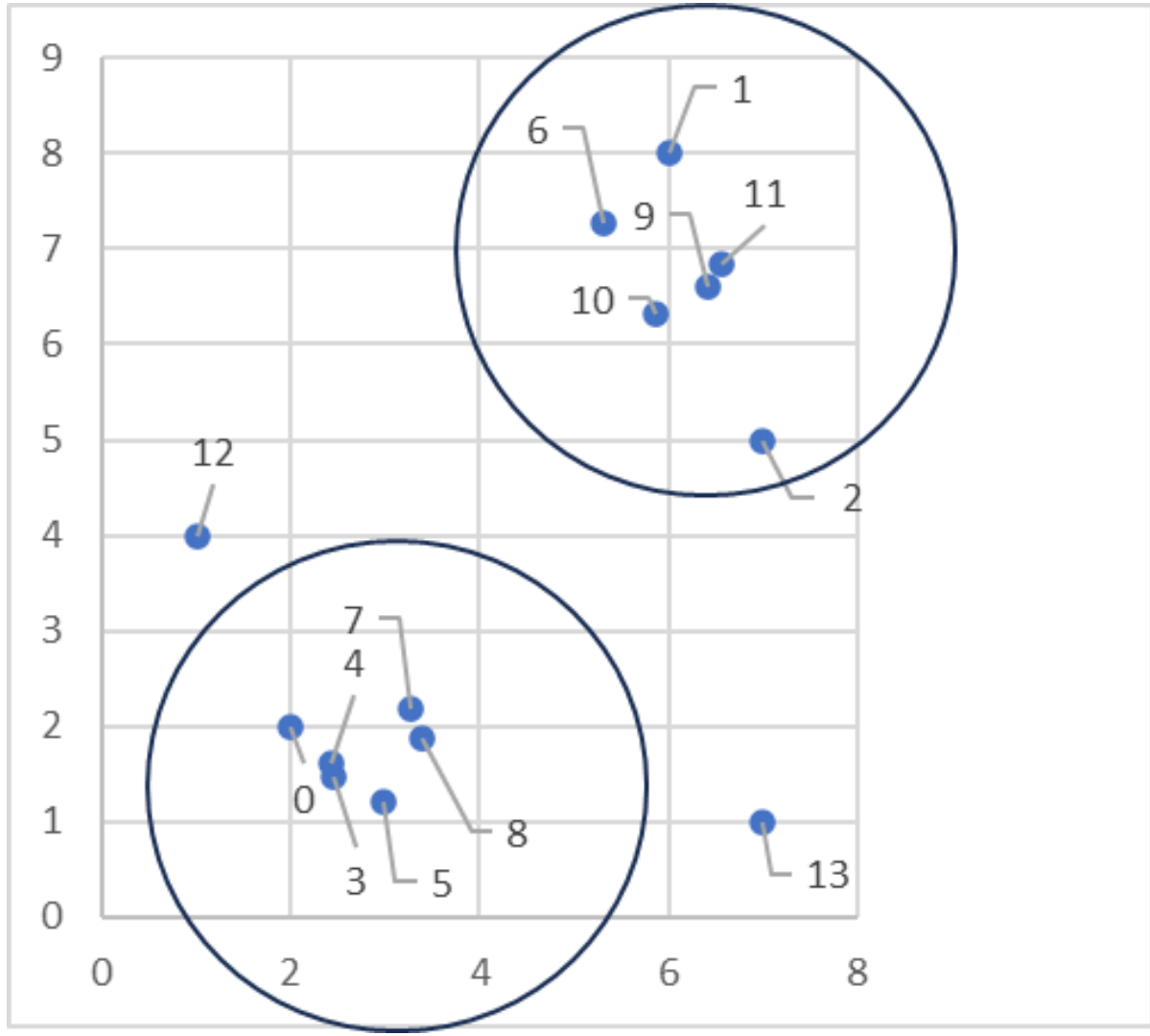


Point 9 is core



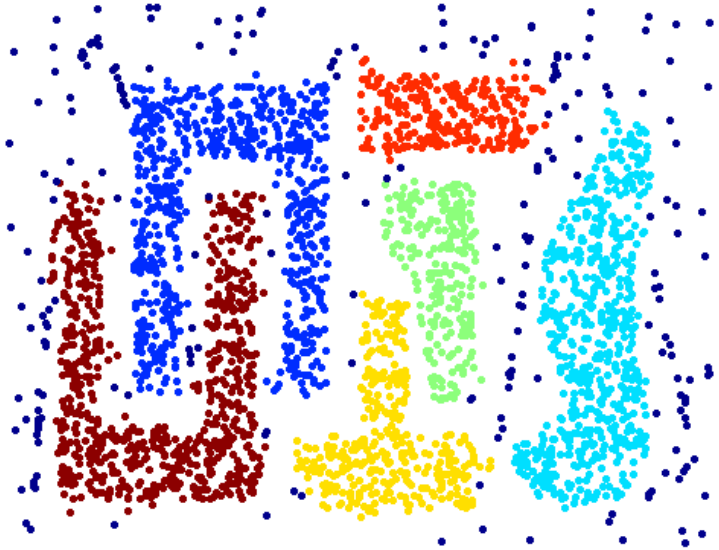
Point 5 is core




9
8
7
6
5
4
3
2
1
0
0
2
4
6
8
1
6
9
11
10
2
12
7
4
0
3
5
8
13

When DBSCAN Works Well

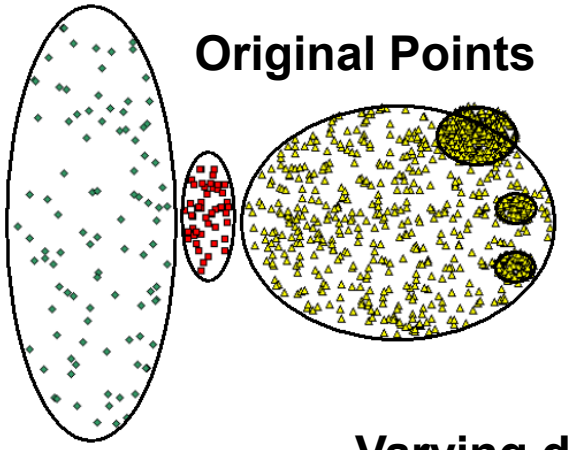
Original Points



Clusters

- **Resistant to Noise**
- **Can handle clusters of different shapes and sizes**

When DBSCAN Does NOT Work Well

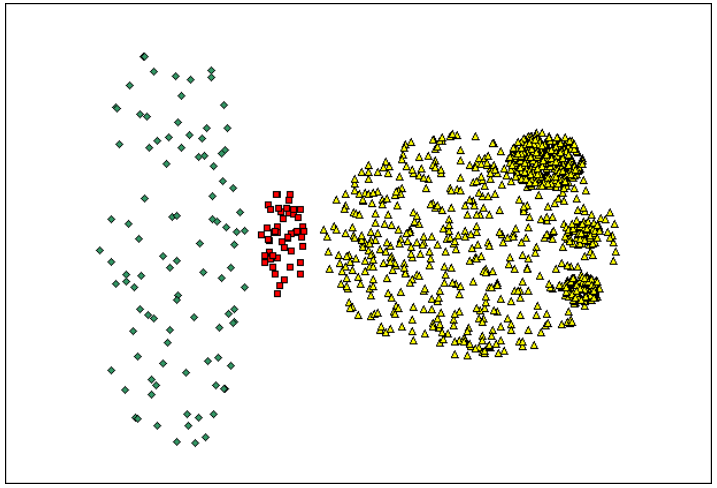


Original Points

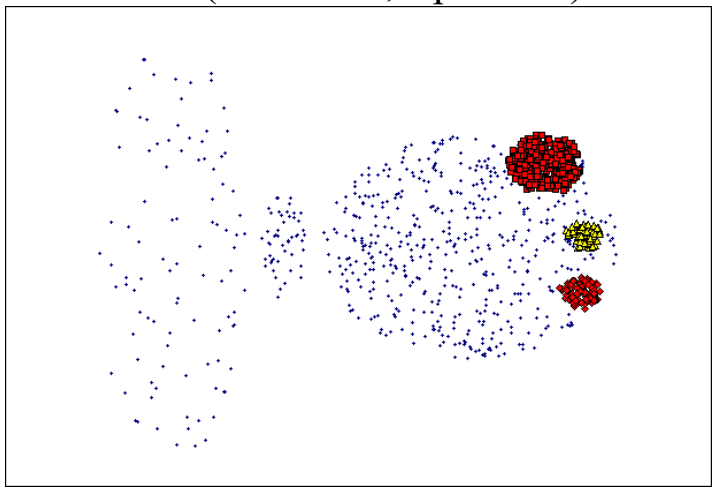
Varying densities since DBSCAN assumes a uniform density of clusters, which means it has a single set of parameters (ε and MinPts) that applies to the entire dataset.

High-dimensional data: If the dataset is high-dimensional, the "curse of dimensionality" can make it difficult for DBSCAN to effectively measure densities and distances.

In very high dimensions, the contrast between different distances diminishes. 'nearest neighbor' becomes less meaningful, as all points tend to be approximately equidistant from each other.



(MinPts=4, Eps=9.75).

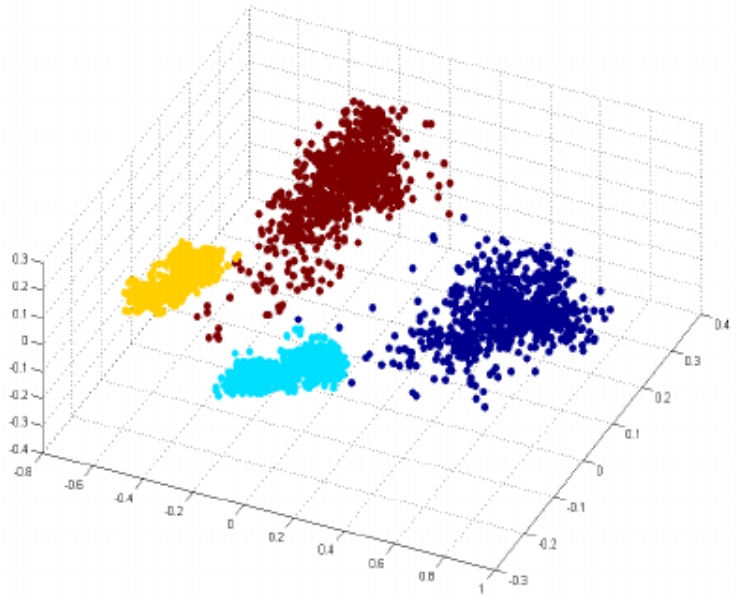


(MinPts=4, Eps=9.92)

The outcome of DBSCAN is highly sensitive to the choice of parameters

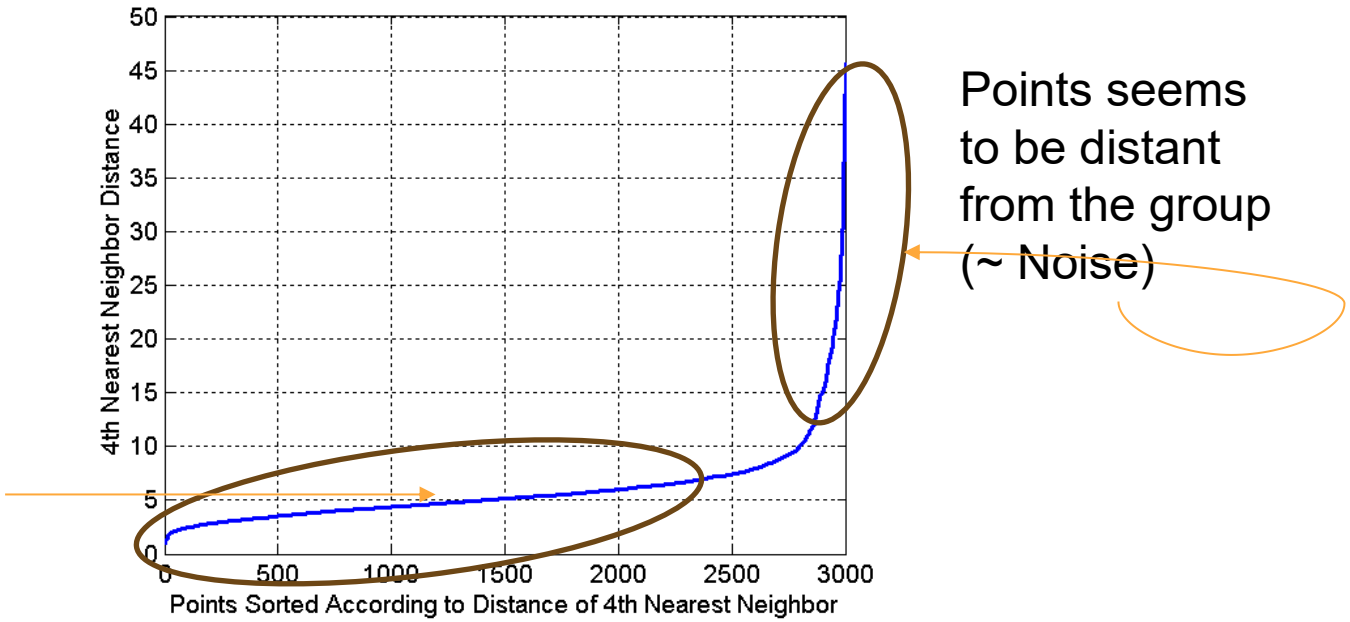
DBSCAN: Determining EPS and MinPts

- **Determining MinPts**
- MinPts: A good starting point for MinPts is based on the dimensionality of your dataset.
- A common approach is to set MinPts at least one higher than the dimensionality of the data.
- For 2D data, MinPts could be 3 or more, and for higher dimensions, you may start with MinPts equal to the dimensionality of the data plus one, although this should be determined based on domain knowledge and the density distribution of your data. In the example we have 3 dimensional data, so we might start with minPts = 4



DBSCAN: Determining EPS and MinPts

- **Determining EPS**
- Idea is that for points in a cluster, their kth nearest neighbors are at roughly the same distance.
- Noise points have the kth nearest neighbor at farther distance
- So, plot sorted distance of every point to its kth nearest neighbor

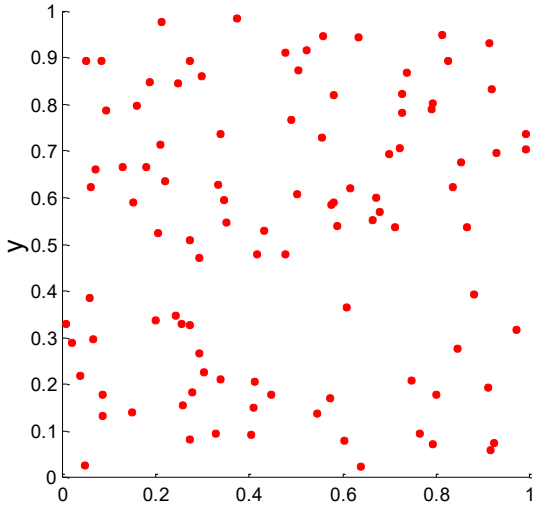


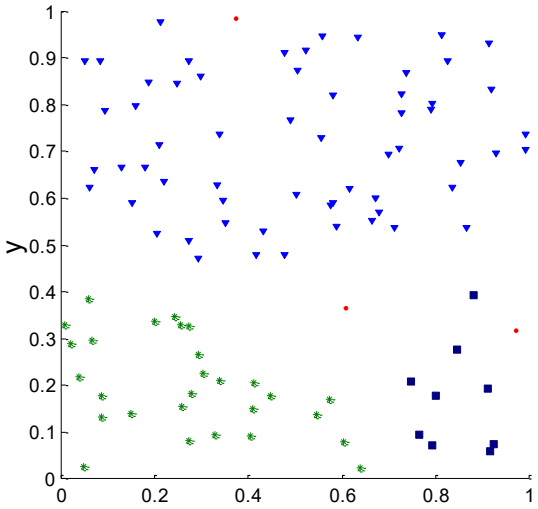
Cluster Validity

- For **supervised classification** we have a variety of measures to evaluate how good our model is
  - **Accuracy, precision, recall**
- **For cluster analysis, the analogous question is how to evaluate the "goodness" of the resulting clusters?**

Clusters found in Random Data: we may misinterpret random noise as meaningful clusters

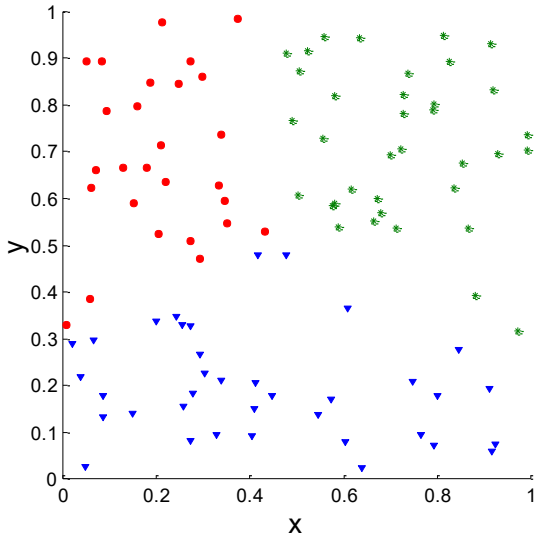
Random Points

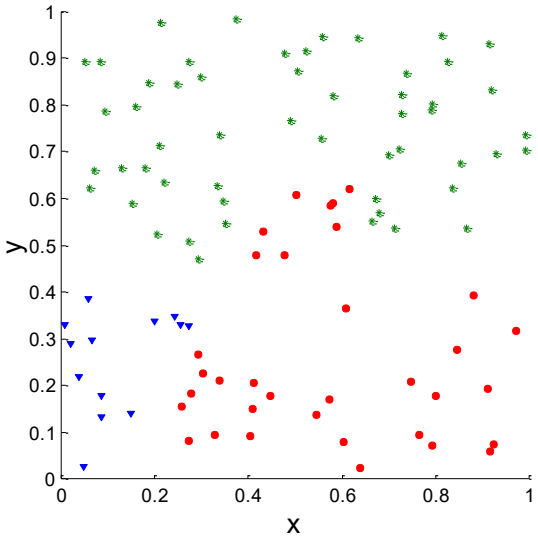




DBSCAN

K-means





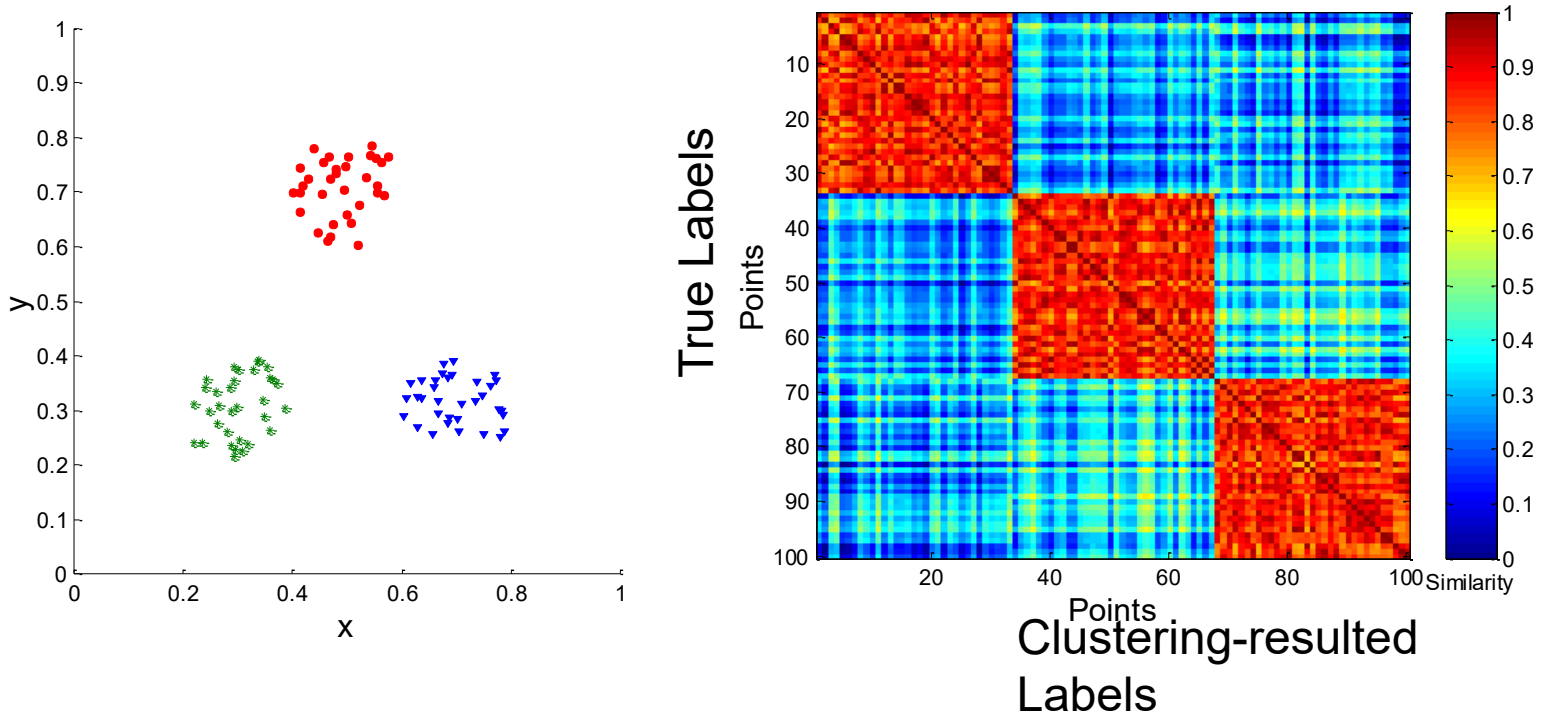
Complete Link

Measures of Cluster Validity

- Numerical measures that are applied to judge various aspects of cluster validity, are classified into the following three types.
  - **External Index:** Used to measure **the extent to which cluster labels match externally supplied class labels**.
  - **Internal Index:** Used to **measure the goodness of a clustering structure** without respect to external information, ex. based on the distances between data points and cluster centroids
  - **Relative Index:** Used to **compare two different clusterings** or clusters.
    - **To compare clustering algorithms** Different clustering algorithms can produce very different sets of clusters on the same data. For instance, some algorithms may be better at detecting spherical clusters, while others may excel at identifying clusters of arbitrary shapes.
    - **To compare two sets of clusters** compare clustering results across different runs of the same algorithm (with different parameters)
    - **To compare two clusters** we might want to compare two individual clusters within the same run to understand their characteristics

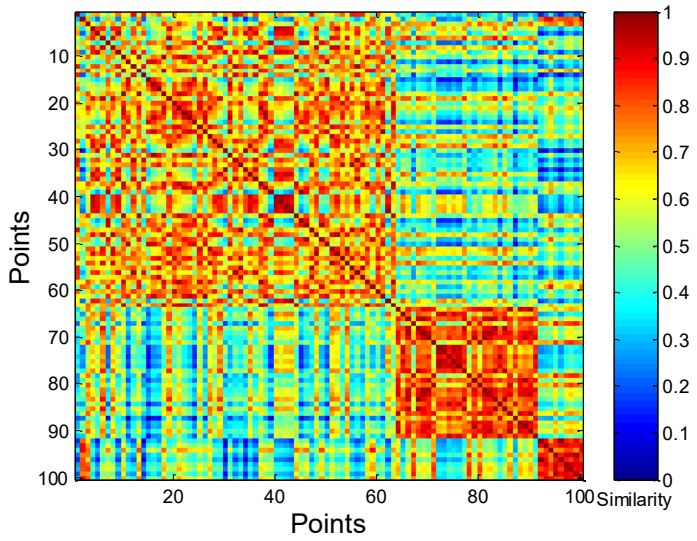
Using Similarity Matrix for Cluster Validation

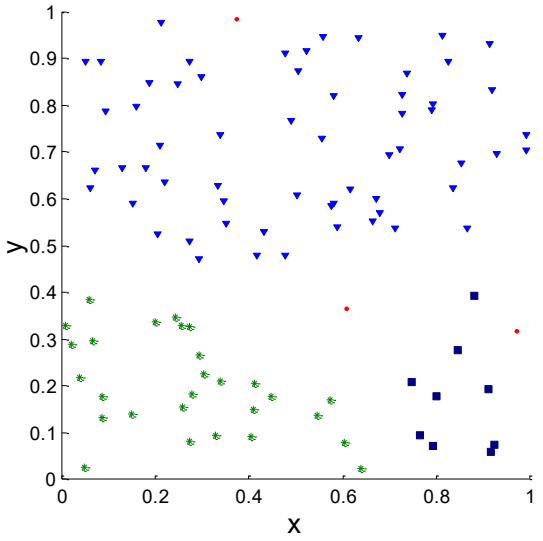
- **Order the similarity matrix with respect to cluster labels and inspect visually.**



Using Similarity Matrix for Cluster Validation

- Clusters in random data are not so crisp

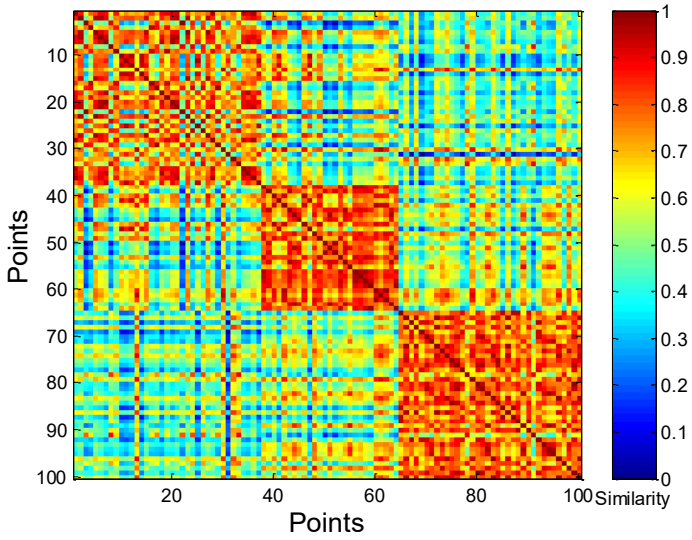


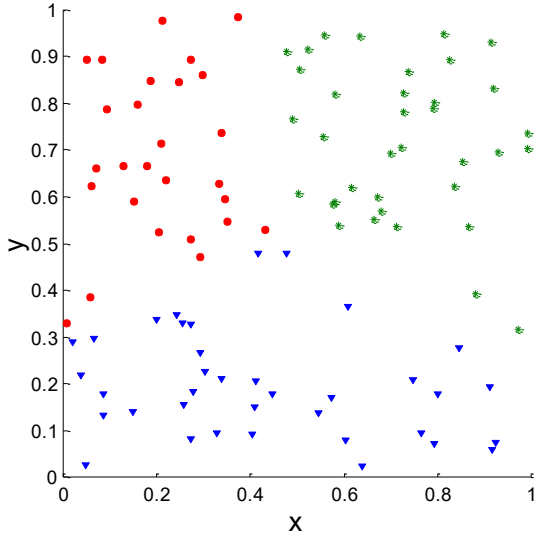


DBSCAN

Using Similarity Matrix for Cluster Validation

- Clusters in random data are not so crisp

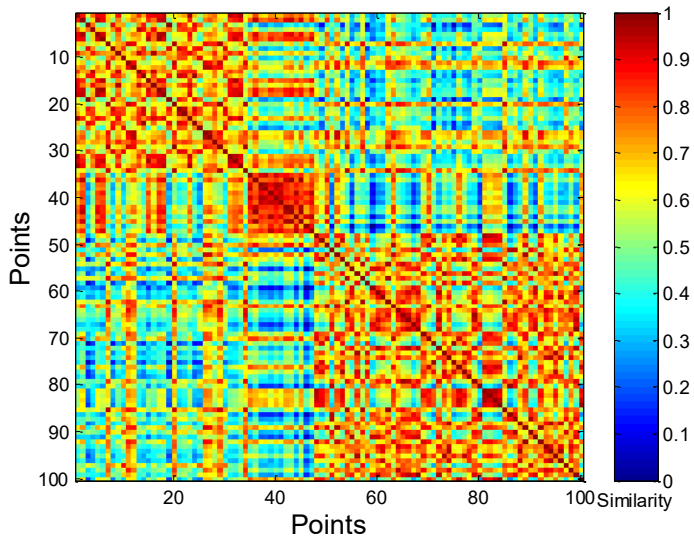


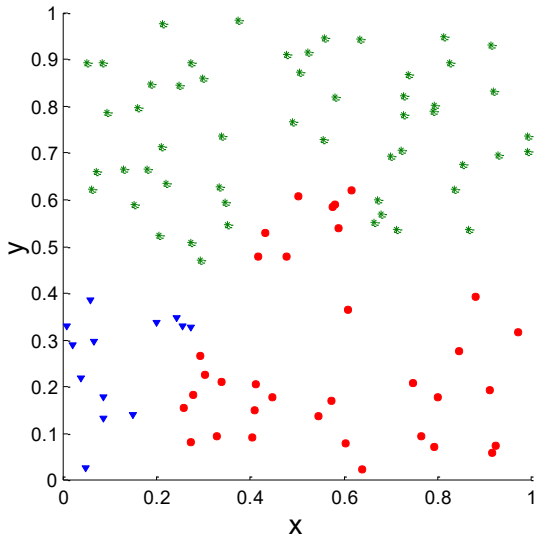


K-means

Using Similarity Matrix for Cluster Validation

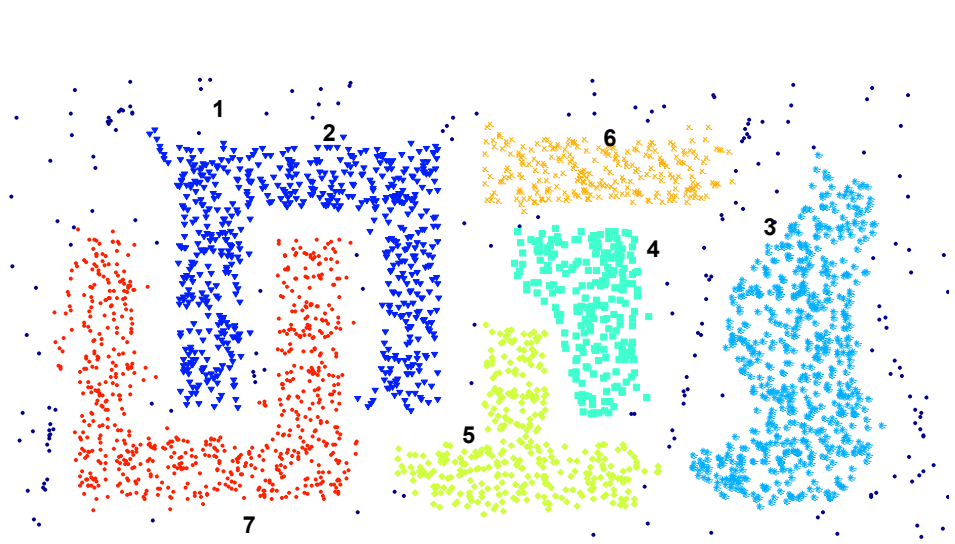
- Clusters in random data are not so crisp

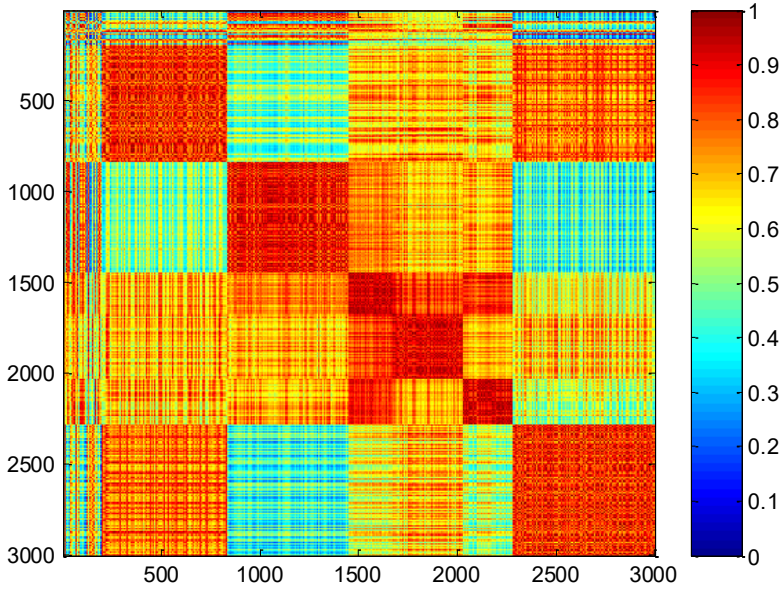




Complete Link

Using Similarity Matrix for Cluster Validation

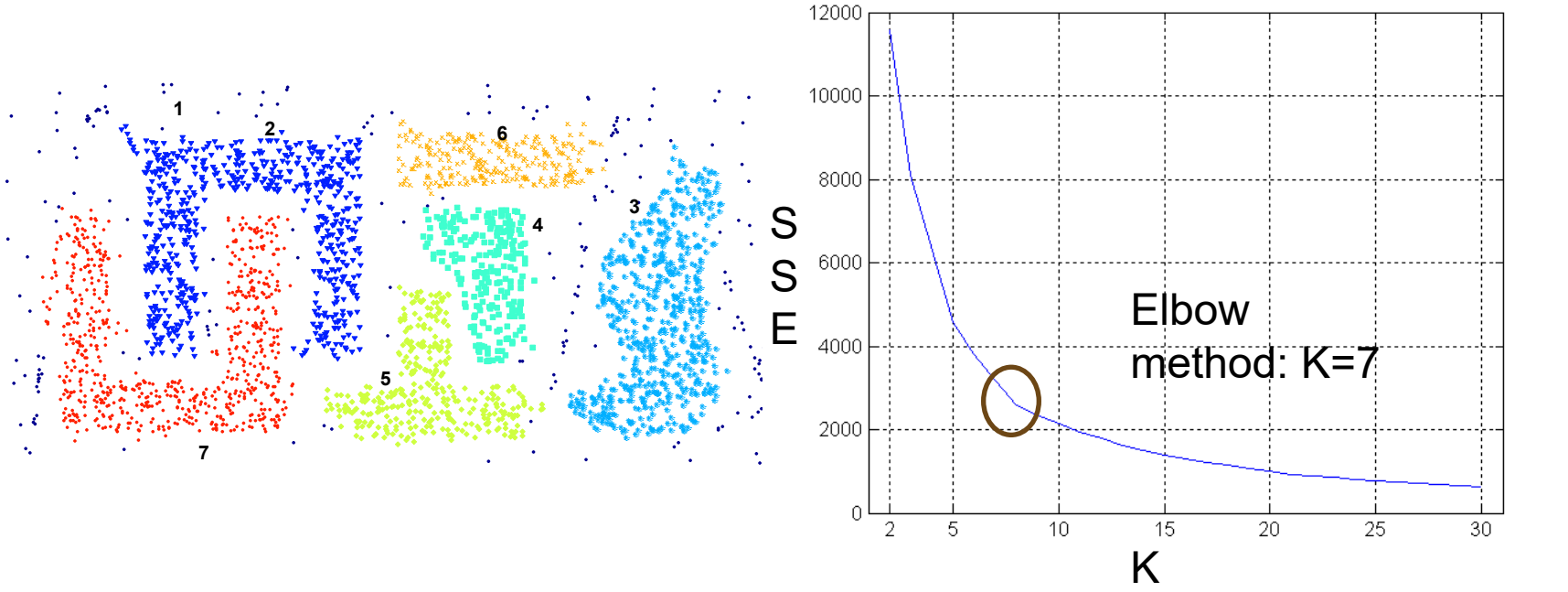




DBSCAN

Internal Measures: SSE

- SSE curve for a more complicated data set



SSE of clusters found using K-means

		1	2	6	9	10	11	0	3	4	5	7	8	12	13	count
		C1	C1	C1	C1	C1	C1	C2	C2	C2	C2	C2	C2	C3	C3	
1	C1	0	3.16	1	1.45	1.68	1.28	7.21	7.43	7.31	7.43	6.42	6.65	6.4	7.07	5
2	C1	3.16	0	2.83	1.72	1.75	1.9	5.83	5.76	5.68	5.52	4.68	4.77	6.08	4	4
6	C1	1	2.83	0	1.29	1.09	1.32	6.22	6.46	6.34	6.49	5.48	5.72	5.41	6.5	5
9	C1	1.45	1.72	1.29	0	0.63	0.28	6.38	6.49	6.38	6.4	5.43	5.62	6.01	5.64	6
10	C1	1.68	1.75	1.09	0.63	0	0.88	5.8	5.93	5.82	5.87	4.89	5.09	5.38	5.45	6
11	C1	1.28	1.9	1.32	0.28	0.88	0	6.66	6.77	6.66	6.68	5.71	5.9	6.25	5.87	6
0	C2	7.21	5.83	6.22	6.38	5.8	6.66	0	0.69	0.57	1.27	1.28	1.39	2.24	5.1	6
3	C2	7.43	5.76	6.46	6.49	5.93	6.77	0.69	0	0.15	0.61	1.08	1.02	2.91	4.58	6
4	C2	7.31	5.68	6.34	6.38	5.82	6.66	0.57	0.15	0	0.7	1.01	0.99	2.78	4.61	6
5	C2	7.43	5.52	6.49	6.4	5.87	6.68	1.27	0.61	0.7	0	1.02	0.78	3.43	4.01	6
7	C2	6.42	4.68	5.48	5.43	4.89	5.71	1.28	1.08	1.01	1.02	0	0.33	2.9	3.92	6
8	C2	6.65	4.77	5.72	5.62	5.09	5.9	1.39	1.02	0.99	0.78	0.33	0	3.19	3.72	6
12	C3	6.4	6.08	5.41	6.01	5.38	6.25	2.24	2.91	2.78	3.43	2.9	3.19	0	6.71	1
13	C3	7.07	4	6.5	5.64	5.45	5.87	5.1	4.58	4.61	4.01	3.92	3.72	6.71	0	1

Thanks