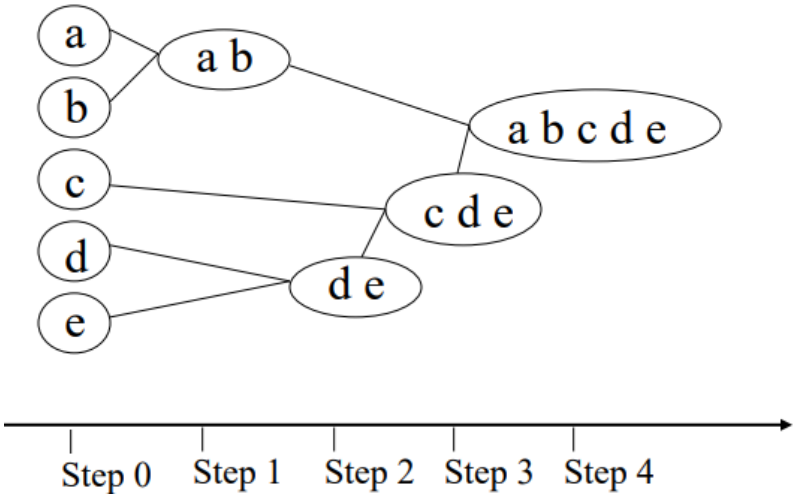
Hierarchical Clustering

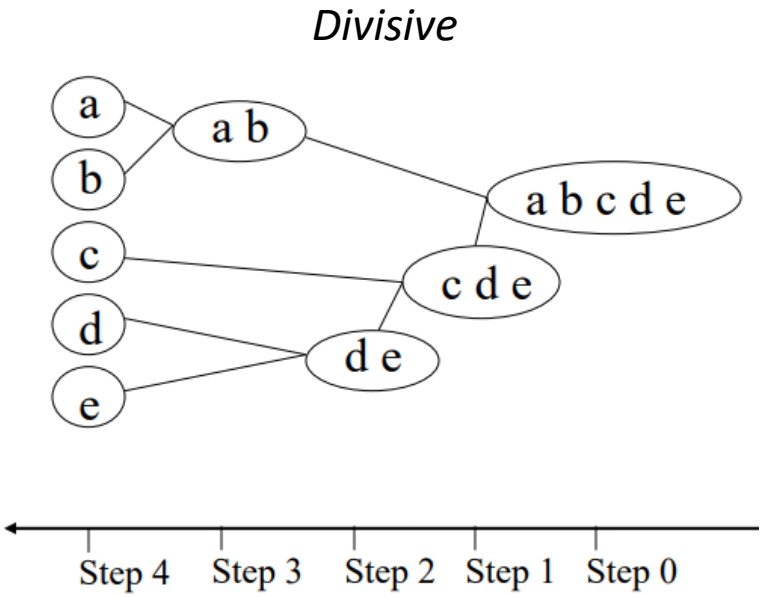
A **hierarchical clustering** is a set of nested clusters that are organized as a tree.

- Agglomerative vs Divisive
  - *Agglomerative*: each instance is its own cluster and the algorithm merges clusters
  - *Divisive*: begins with all instances in one cluster and divides it up

Agglomerative

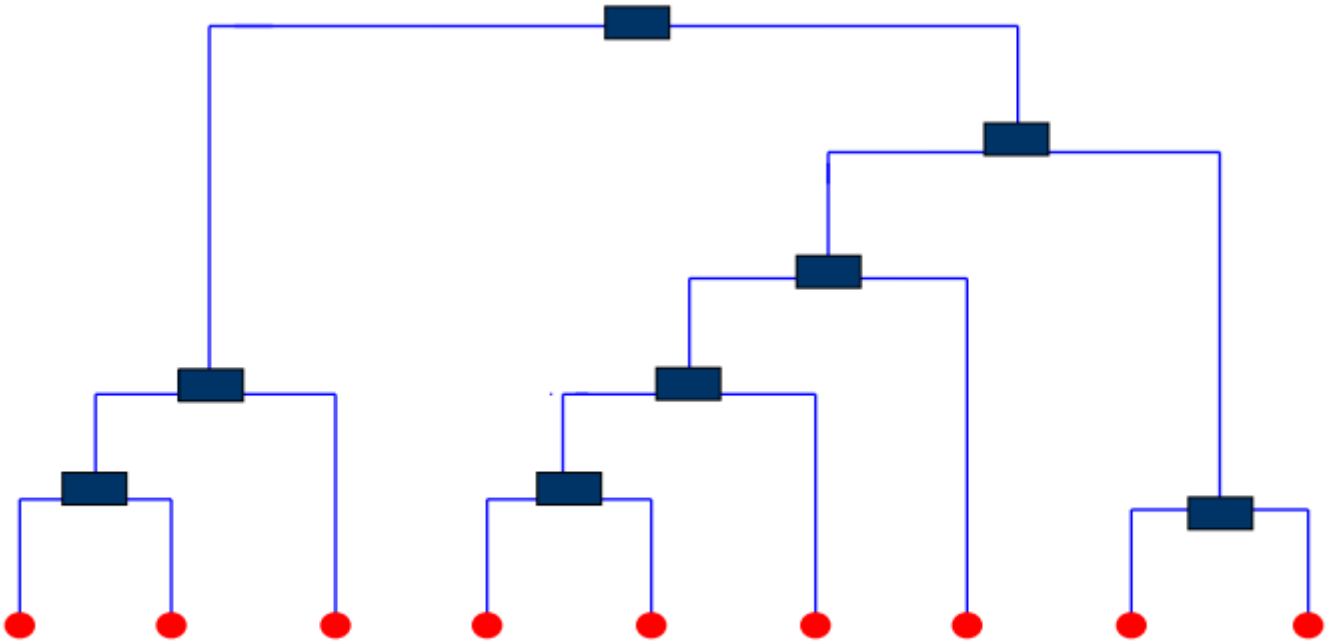


Divisive



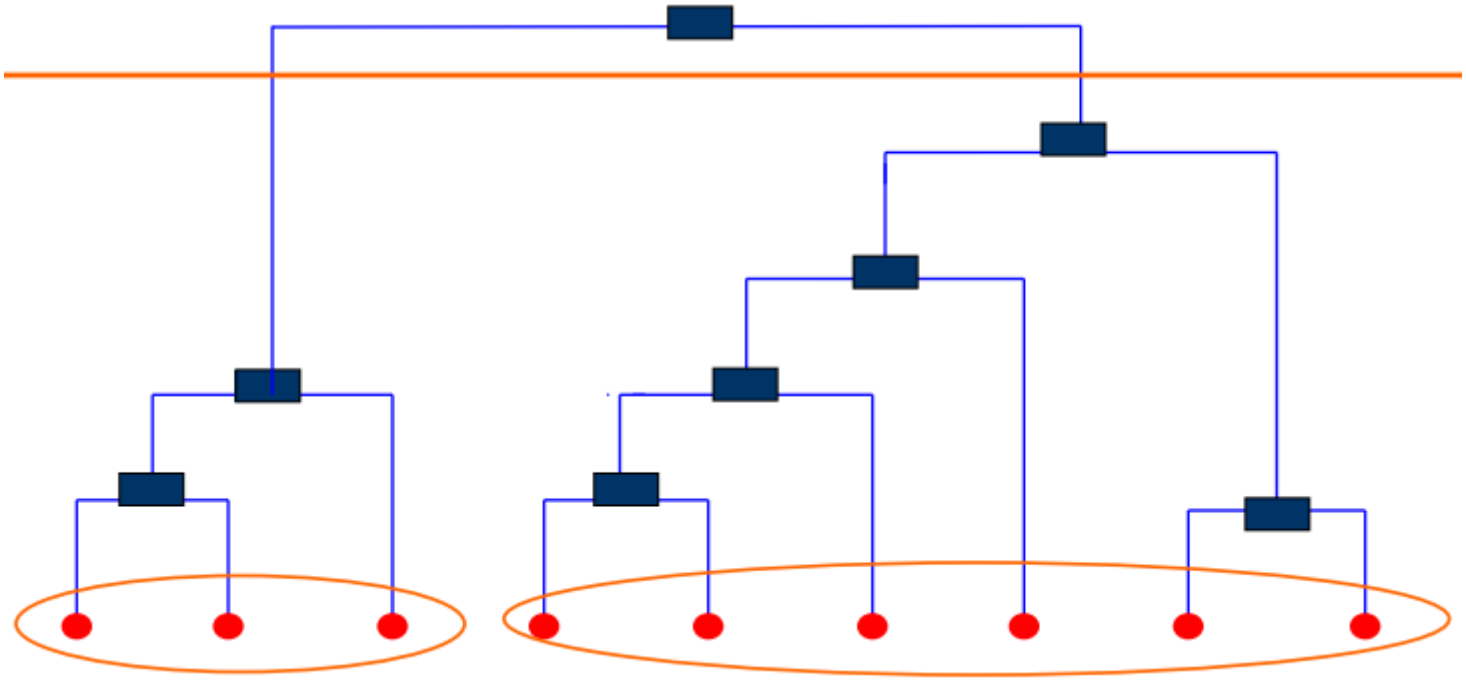
Dendrogram

- A tree that shows how clusters are merged/split hierarchically
- Each node on the tree is a cluster; each leaf node is a singleton cluster



Dendrogram

- A clustering of the data objects is obtained by cutting the dendrogram at the desired level, then each connected component forms a cluster



Agglomerative Clustering Algorithm

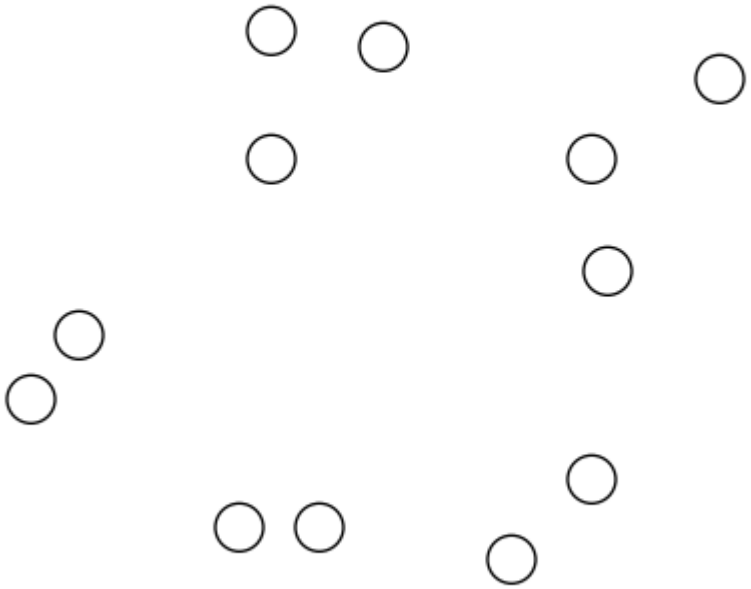
- More popular hierarchical clustering technique

- Basic algorithm is straightforward
  1. Compute the distance matrix
  2. Let each data point be a cluster
  3. Repeat
     4. Merge the two closest clusters
     5. Update the distance matrix
  6. Until only a single cluster remains

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

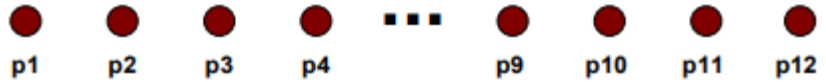
Starting Situation

Start with clusters of individual points and a distance matrix



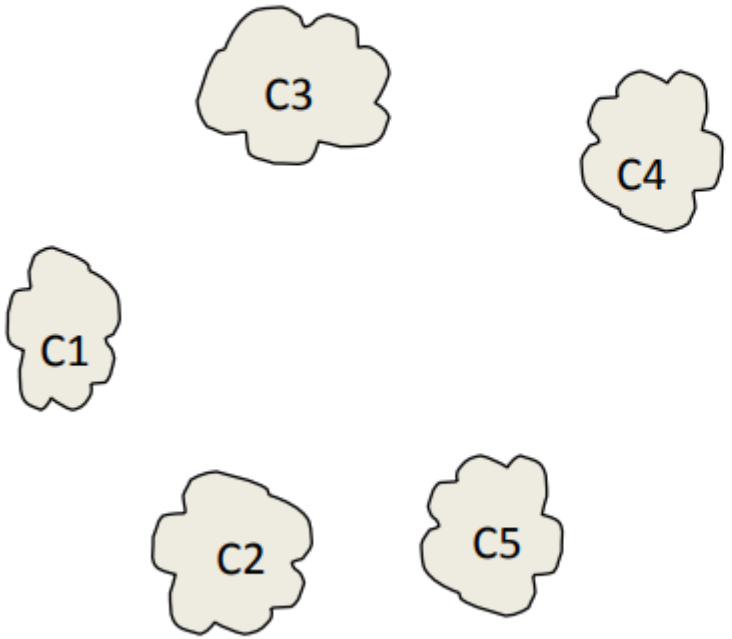
	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						

Distance Matrix



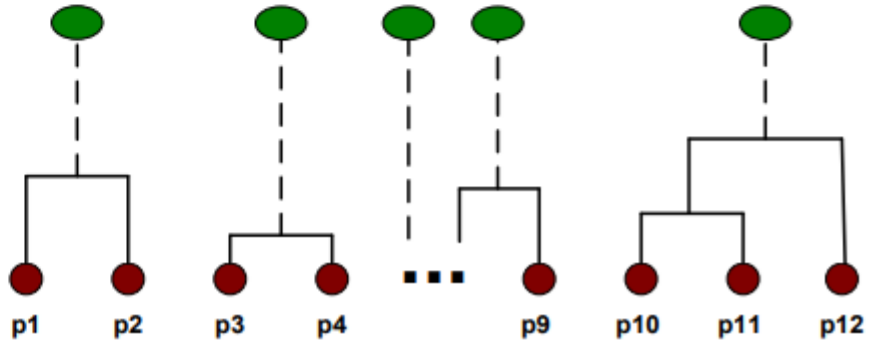
Intermediate Situation

- After some merging steps, we have some clusters
- Choose two clusters that has the smallest distance (largest similarity) to merge



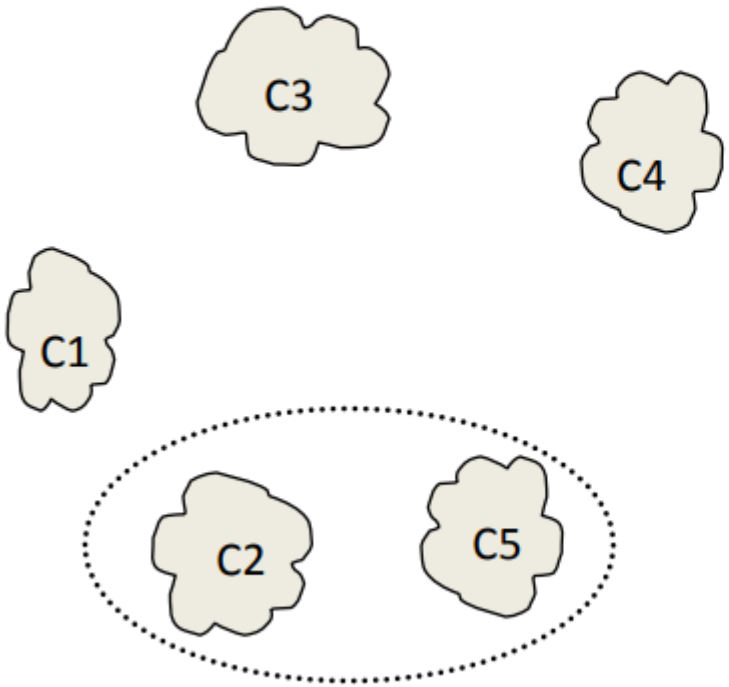
	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

Distance Matrix



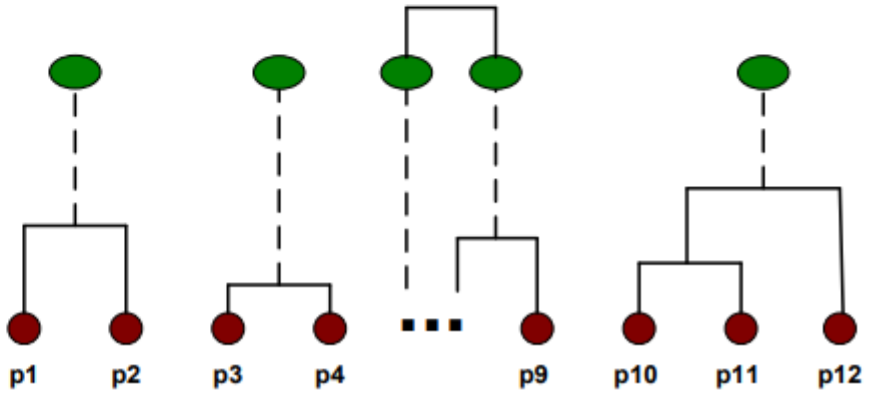
Intermediate Situation

- We want to merge the two closest clusters (C2 and C5) and update the distance matrix.



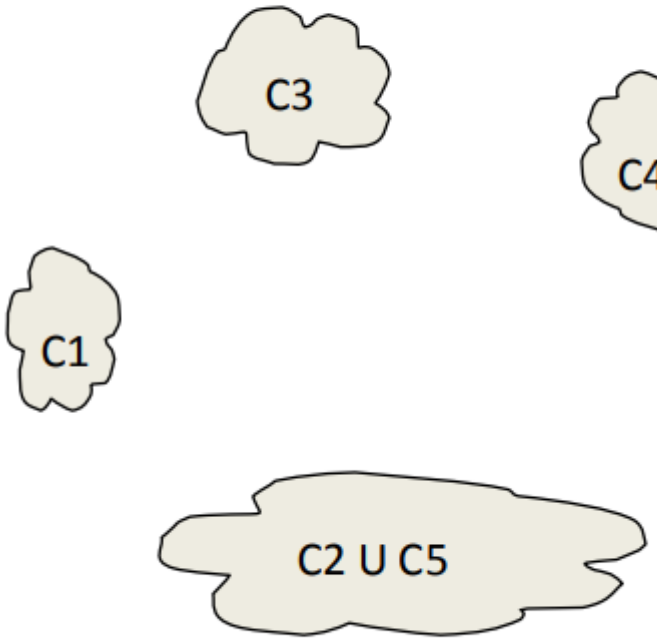
	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

Distance Matrix



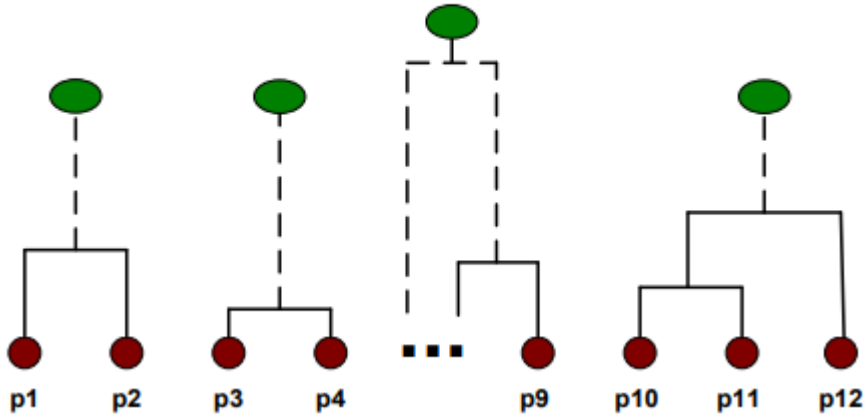
After Merging

The question is “How do we update the distance matrix?”

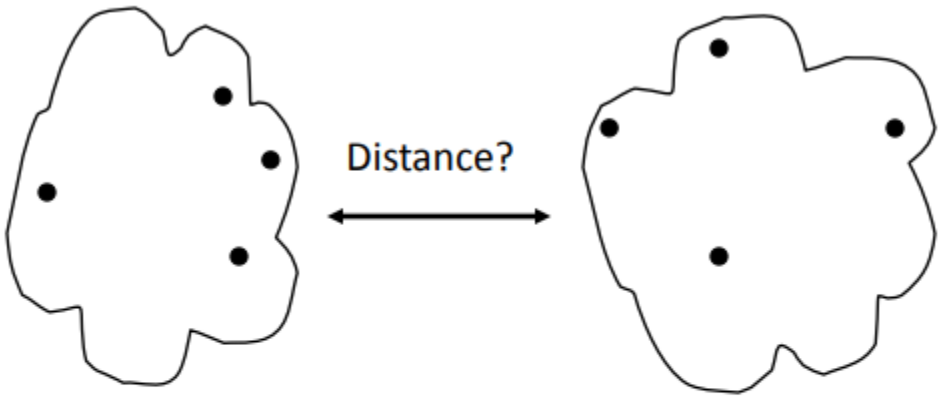


	C1	C2 U C5	C3	C4
C1		?		
C2 U C5	?	?	?	?
C3		?		
C4		?		

Distance Matrix



How to Define Inter-Cluster Distance



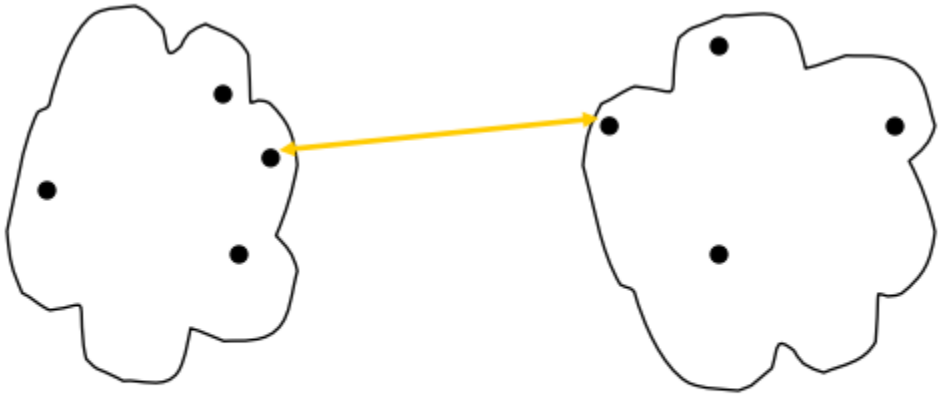
	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						

Distance Matrix

- Single link method (Min)
- Complete link method (Max)
- Average link (group Average)
- Centroid method (Distance between centriods)

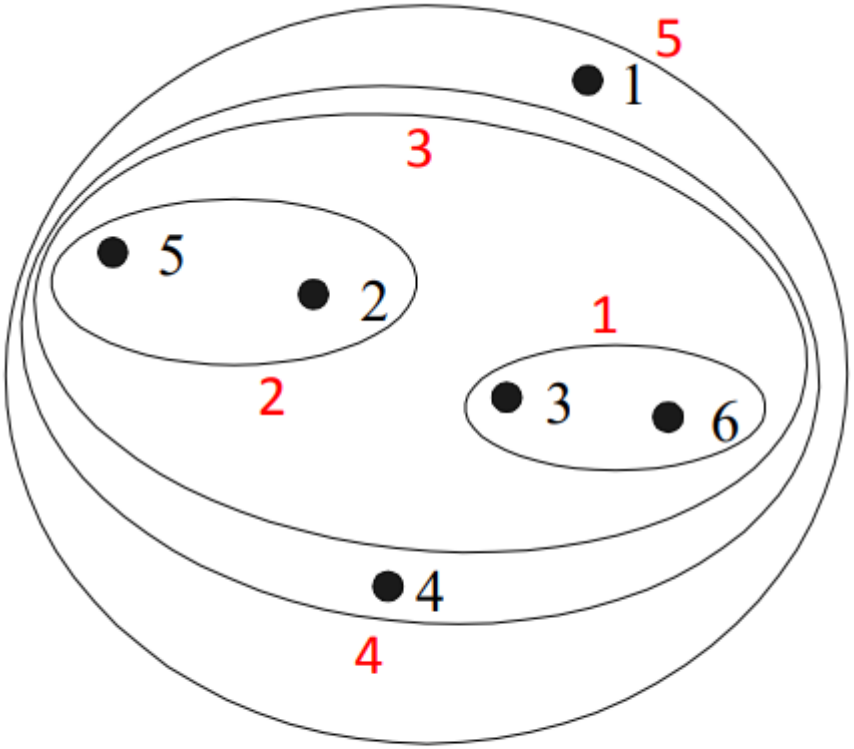
Single link method (Min)

- The distance between two clusters is represented by the distance of the closest pair of data objects belonging to different clusters.
- Determined by one pair of points, i.e., by one link in the proximity graph

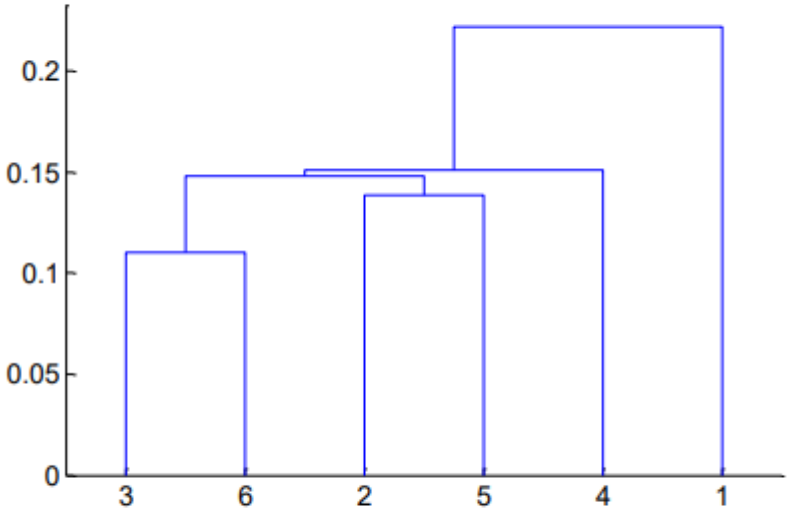


$$d_{\min}(C_i, C_j) = \min_{p \in C_i, q \in C_j} d(p, q)$$

Single link method (Min)



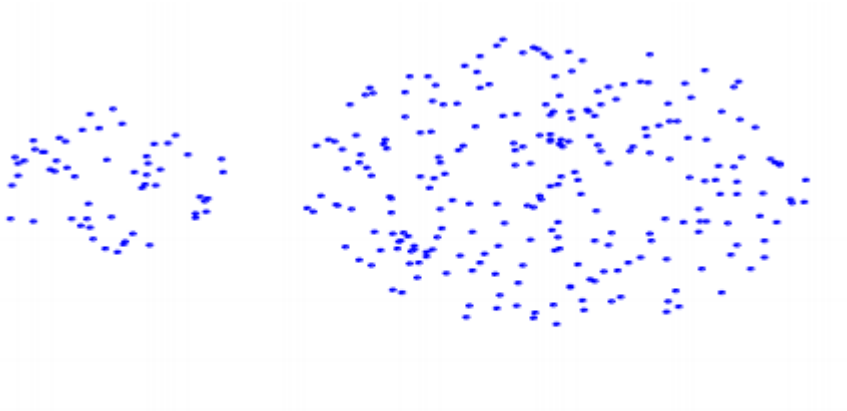
Nested Clusters



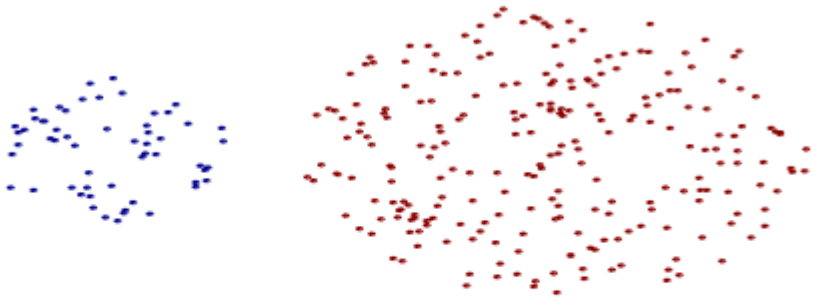
Dendrogram

Single link method (Min)

Can handle non-elliptical shapes



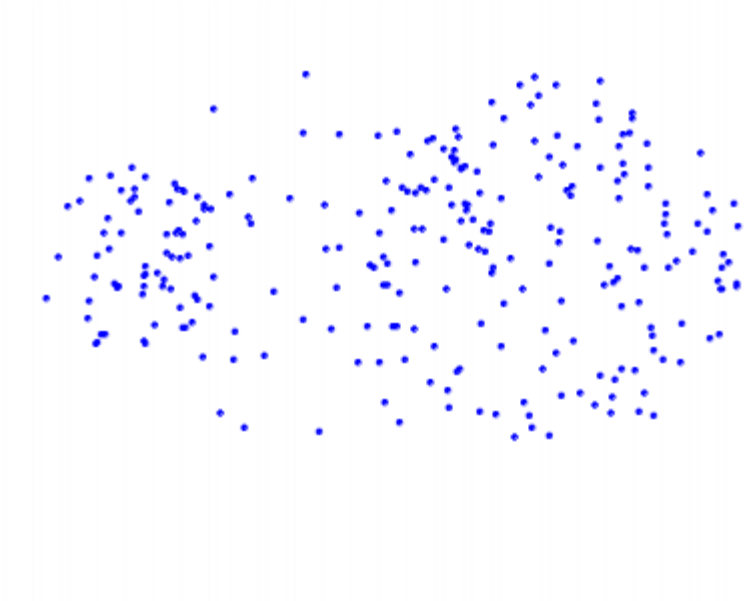
Original Points



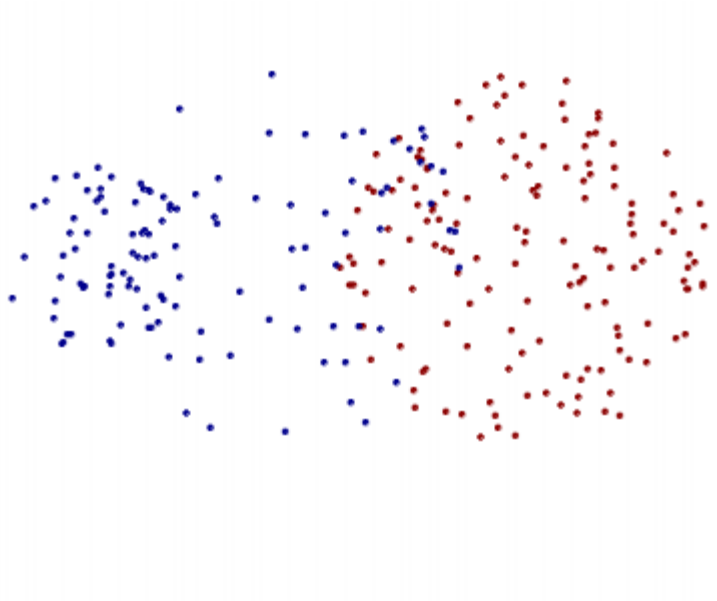
Two Clusters

Single link method (Min)

Sensitive to noise and outliers



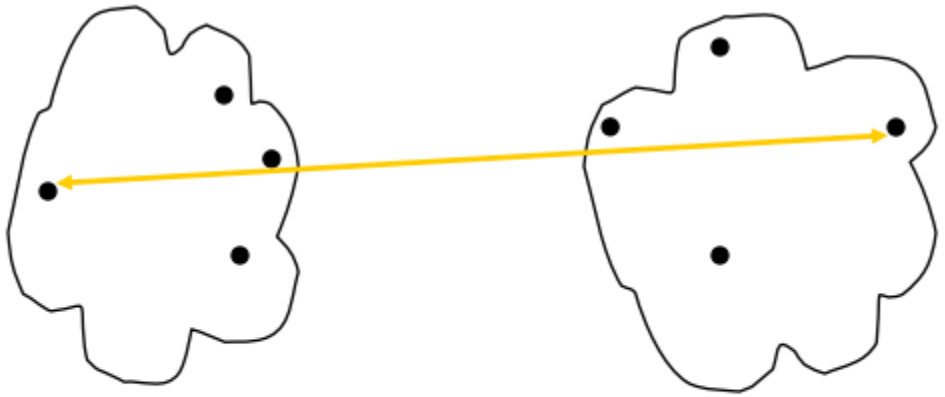
Original Points



Two Clusters

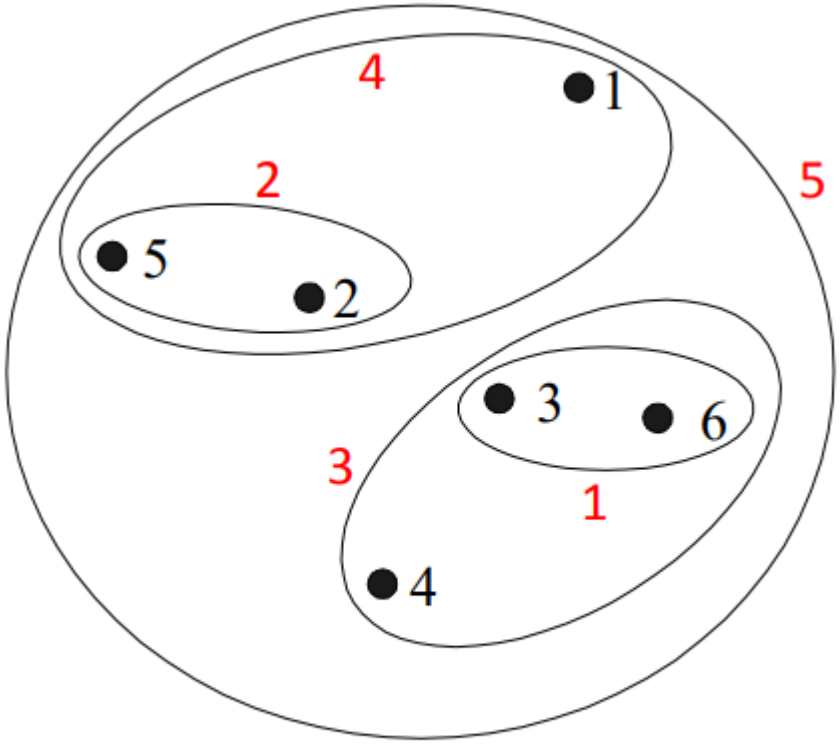
Complete link method (Max)

- The distance between two clusters is represented by the distance of the farthest pair of data objects belonging to different clusters

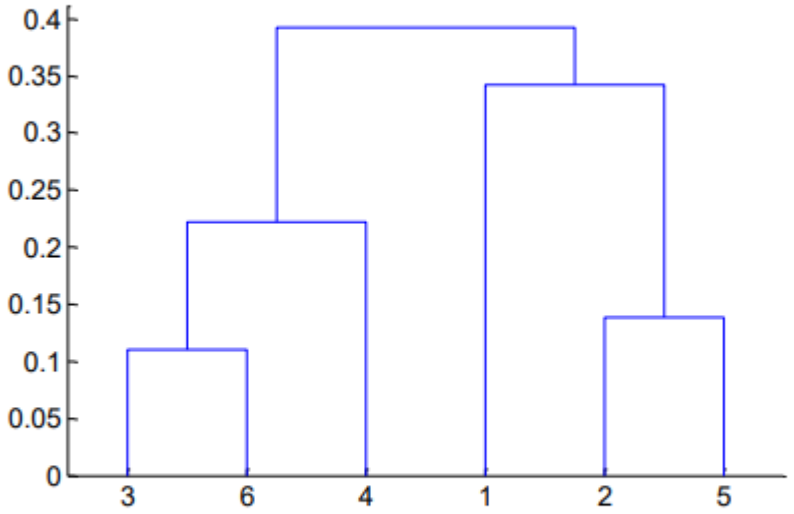


$$d_{\min}(C_i, C_j) = \max_{p \in C_i, q \in C_j} d(p, q)$$

Complete link method (Max)



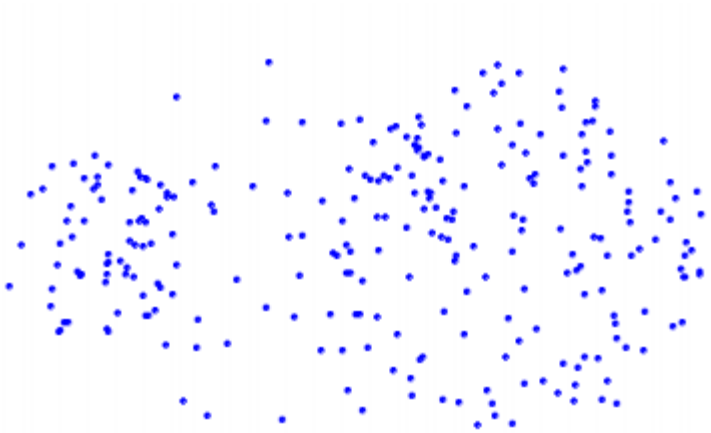
Nested Clusters



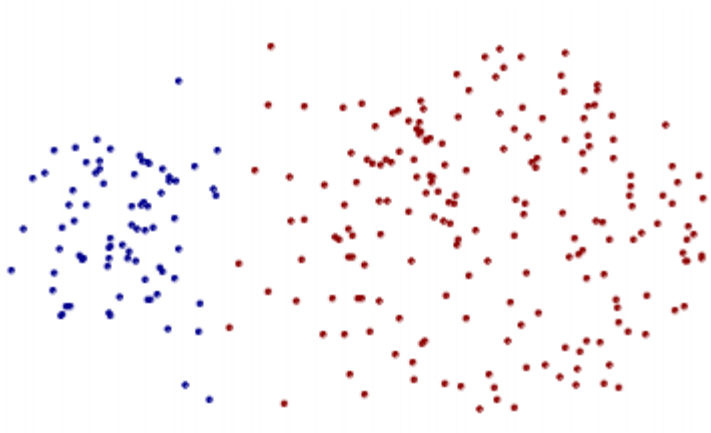
Dendrogram

Complete link method (Max)

Less susceptible to noise and outliers



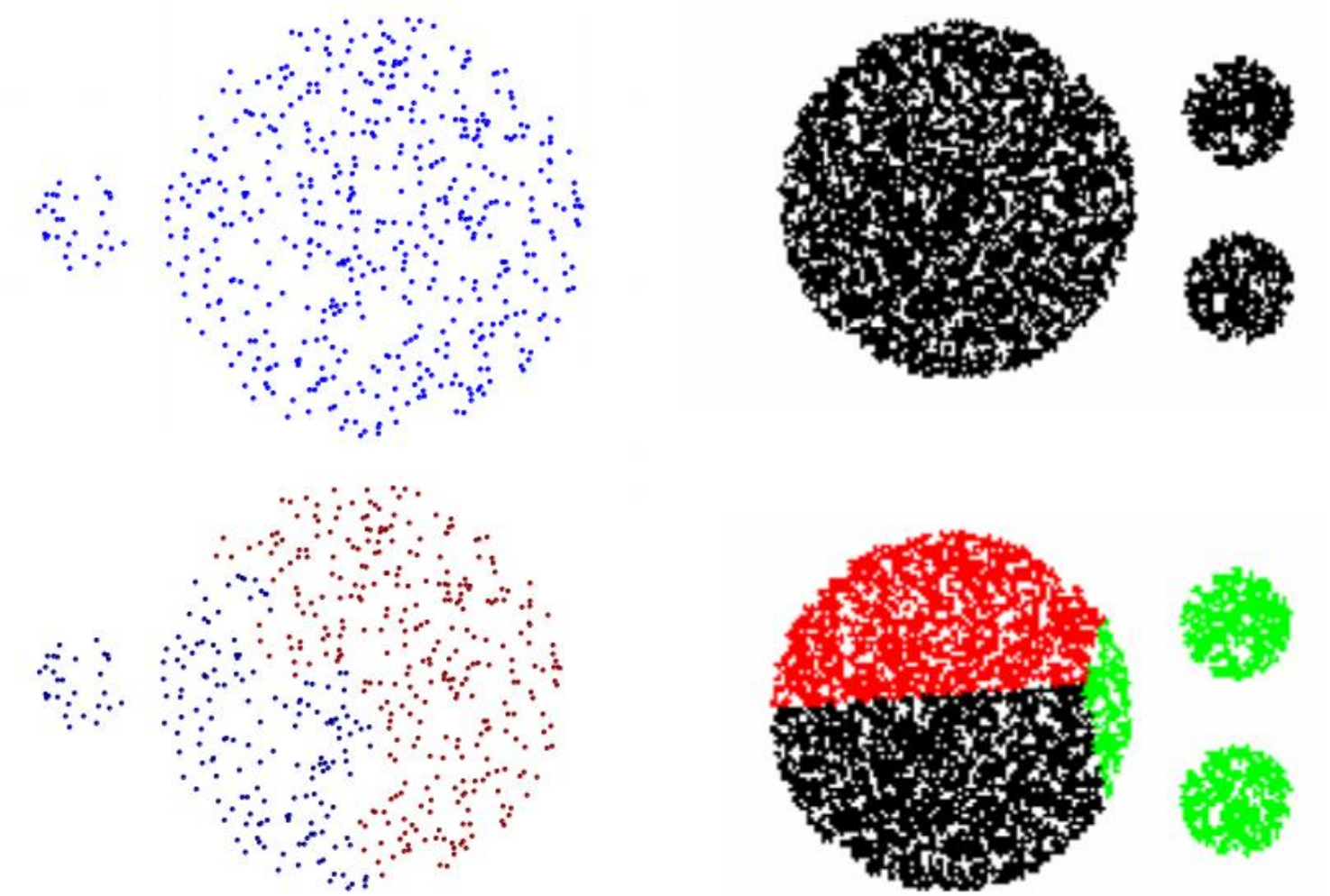
Original Points



Two Clusters

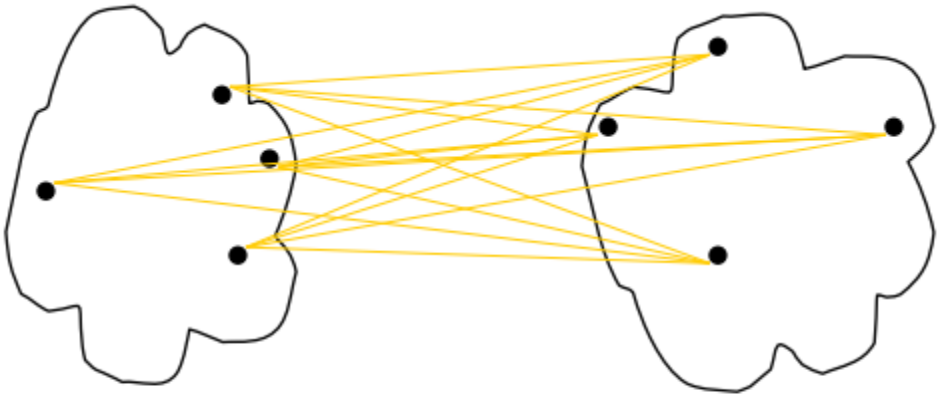
Complete link method (Max)

Tends to break large clusters



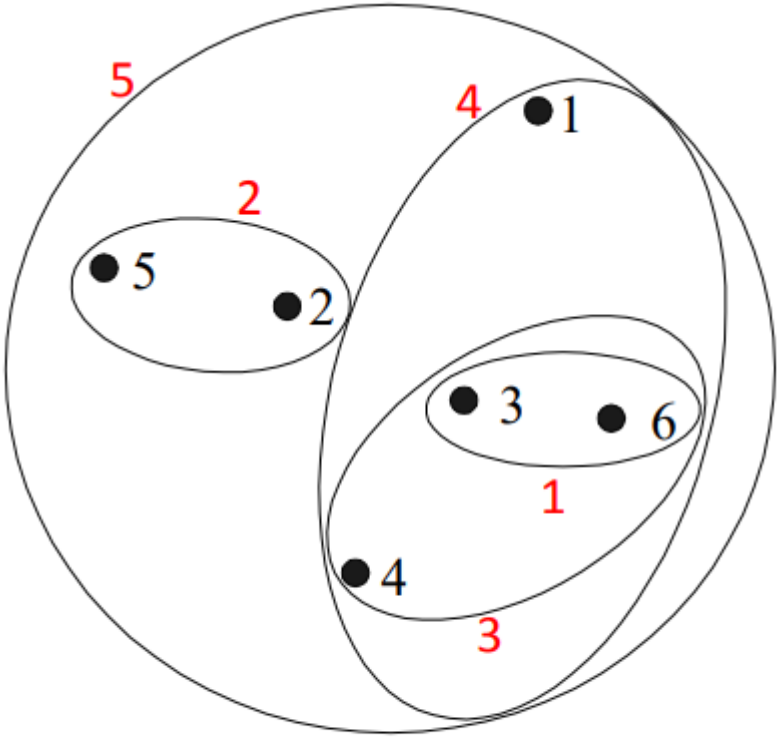
Average link (Group Average)

- The distance between two clusters is represented by the average distance of all pairs of data objects belonging to different clusters
- Determined by all pairs of points in the two clusters

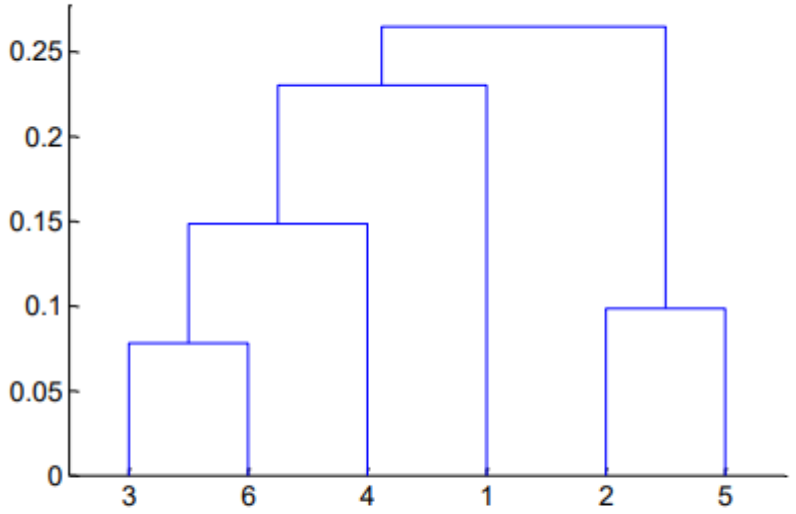


$$d_{\min}(C_i, C_j) = \underset{p \in C_i, q \in C_j}{avg} \; d(p,q)$$

Average link (Group Average)



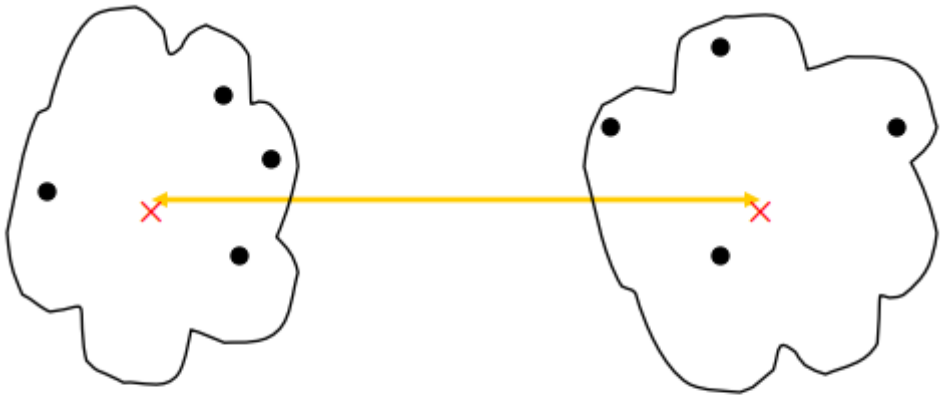
Nested Clusters



Dendrogram

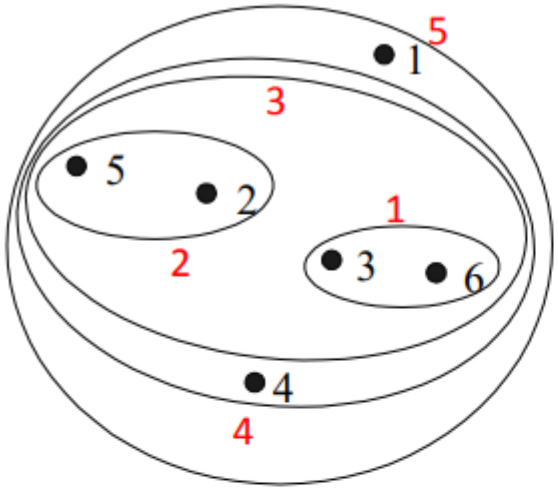
Centroid method (Distance between centroids)

- The distance between two clusters is represented by the distance between the centers of the clusters
- Determined by cluster centroids



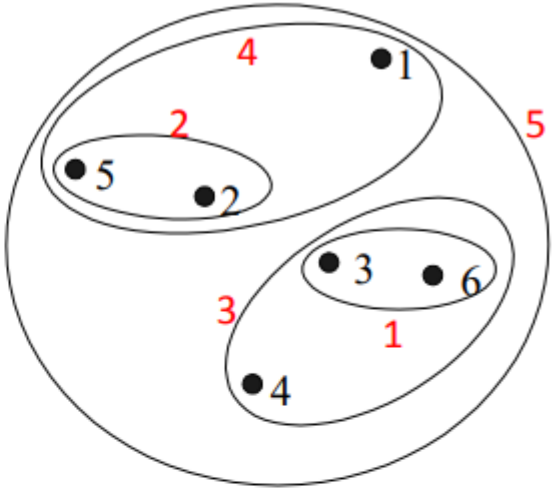
$$d_{mean}(C_i, C_j) = d(m_i, m_j)$$

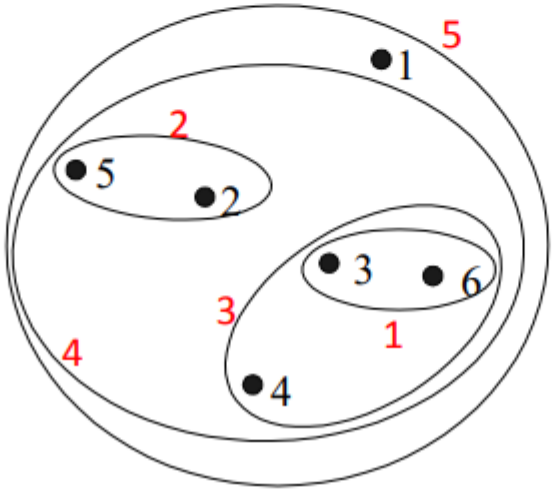
Comparison



MIN

MAX





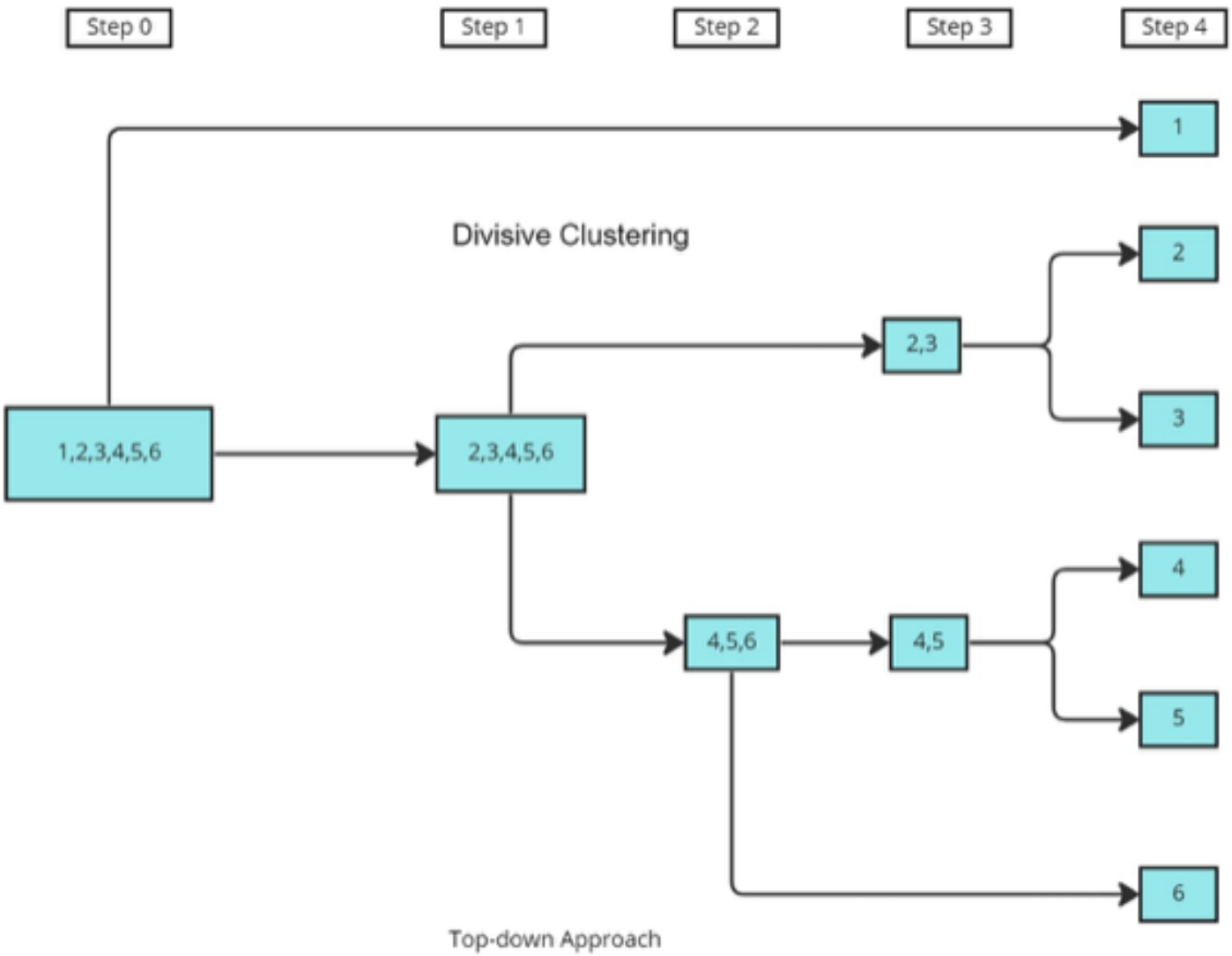
Group Average

Divisive clustering

- Divisive clustering works just the opposite of agglomerative clustering. It starts by considering all the data points into a big single cluster and later on splitting them into smaller heterogeneous clusters continuously until all data points are in their own cluster.
- Good at identifying large clusters. It follows a top-down approach and is more efficient than agglomerative clustering.
- Its complexity in implementation, it doesn’t have any predefined implementation in any of the major machine learning frameworks.

Steps in Divisive Clustering

- Consider all the data points as a single cluster.
- Split into clusters using any flat-clustering method, say K-Means.
- Choose the best cluster among the clusters to split further, choose the one that has the largest Sum of Squared Error (SSE).
- Repeat steps 2 and 3 until a single cluster is formed.


Step 0
Step 1
Step 2
Step 3
Step 4
Divisive Clustering
1,2,3,4,5,6
2,3,4,5,6
2,3
4,5,6
4,5
1
2
3
4
5
6
Top-down Approach

Density-based clustering

- This type of clustering algorithm groups together data points that are in high-density concentrations and separates points in low-concentrations regions.
- The basic idea is that it identifies regions in the data space that have a high density of data points and groups those points together into clusters. Example: DBSCAN(Density-Based Spatial Clustering of Applications with Noise)

Concepts: Preliminary

- **DBSCAN is a density-based algorithm**

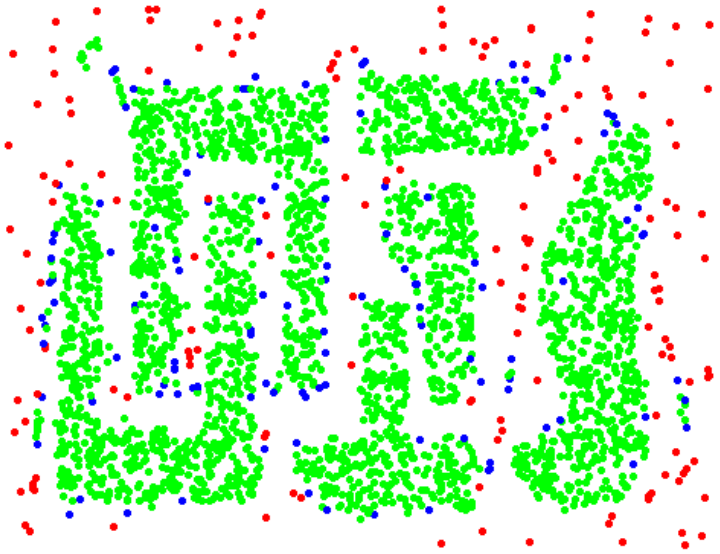
- DBScan stands for Density-Based Spatial Clustering of Applications with Noise

- Density-based Clustering locates regions of high density that are separated from one another by regions of low density

  Density = number of points within a specified radius (Eps)

Concepts: Preliminary

Original Points



Point types: core, border and noise

Eps = 10, MinPts = 4

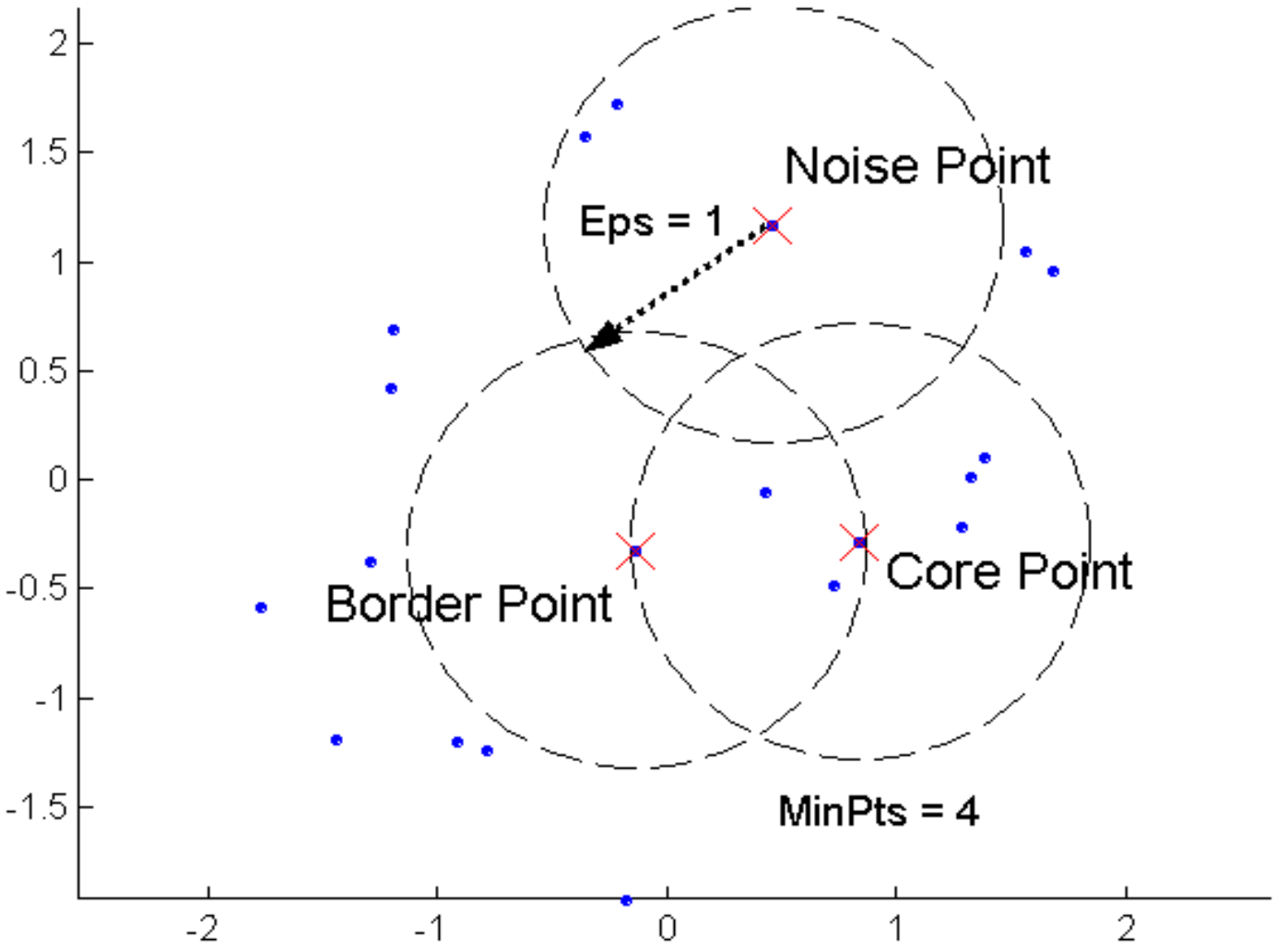
Concepts: Preliminary

- A point is a core point if it has more than a specified number of points (MinPts) within Eps
  - These are points that are at the interior of a cluster
- A border point has fewer than MinPts within Eps, but is in the neighborhood of a core point
- A noise point is any point that is not a core point or a border point

Concepts: Preliminary

- Any two core points are close enough– within a distance *Eps* of one another – are put in the same cluster
- Any border point that is close enough to a core point is put in the same cluster as the core point
- Noise points are discarded

Concepts: Core, Border, Noise



Parameter Estimation

parameters must be specified by the user.

ε = physical distance(radius),

minPts = desired minimum cluster size

minPts

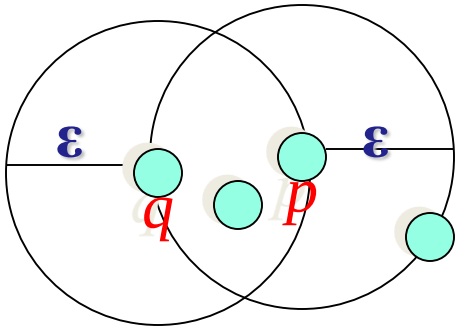
- derived from the number of dimensions *D* in the data set, as $minPts \geq D + 1$
- *minPts* = *1* does not make sense, as then every point on its own will already be a cluster
- *minPts* must be chosen at least 3. larger is better.
- larger the data set, the larger the value of *minPts* should be chosen.

ε

- value can be chosen by using a k-distance graph.
- if ε is chosen much too small, a large part of the data will not be clustered.
- if too high value ,majority of objects will be in the same cluster
- In general, small values of ε are preferable.

Concepts: ε-Neighborhood

- ε-Neighborhood - Objects within a radius of ε from an object. (epsilon-neighborhood)
- Core objects - ε-Neighborhood of an object contains at least MinPts of objects



ε-Neighborhood of *p*

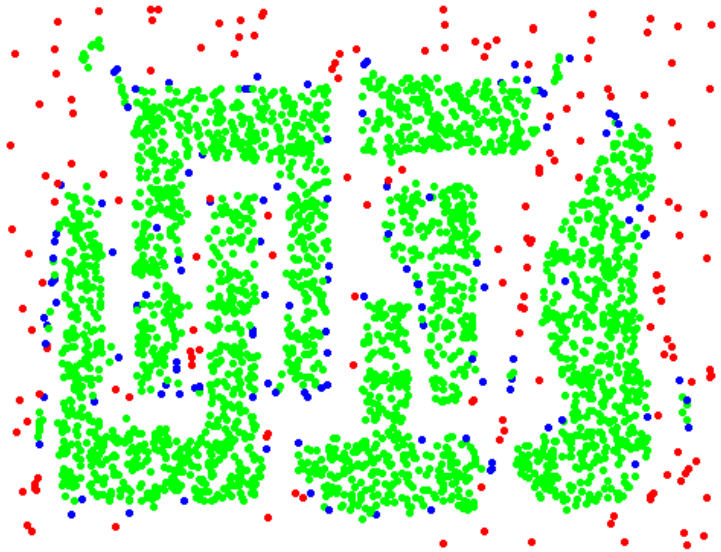
ε-Neighborhood of *q*

p is a core object (MinPts = 4)

q is not a core object

Core, Border, Noise points representation

Original Points

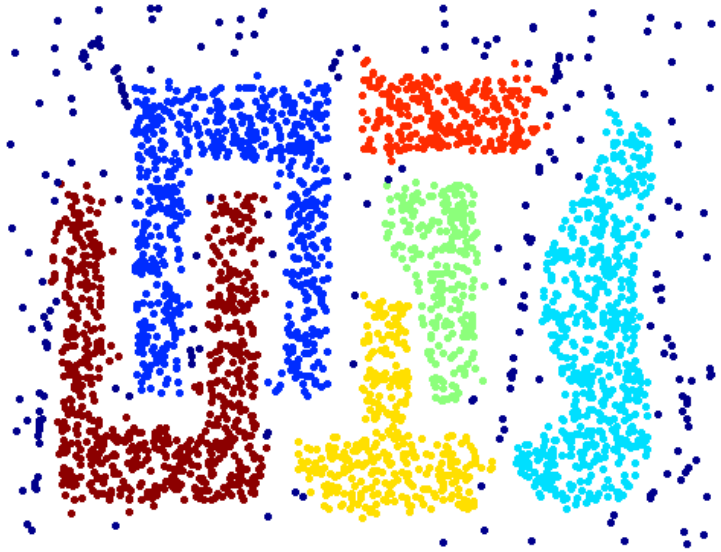


Point types: core, border and noise

Eps = 10, MinPts = 4

Clustering

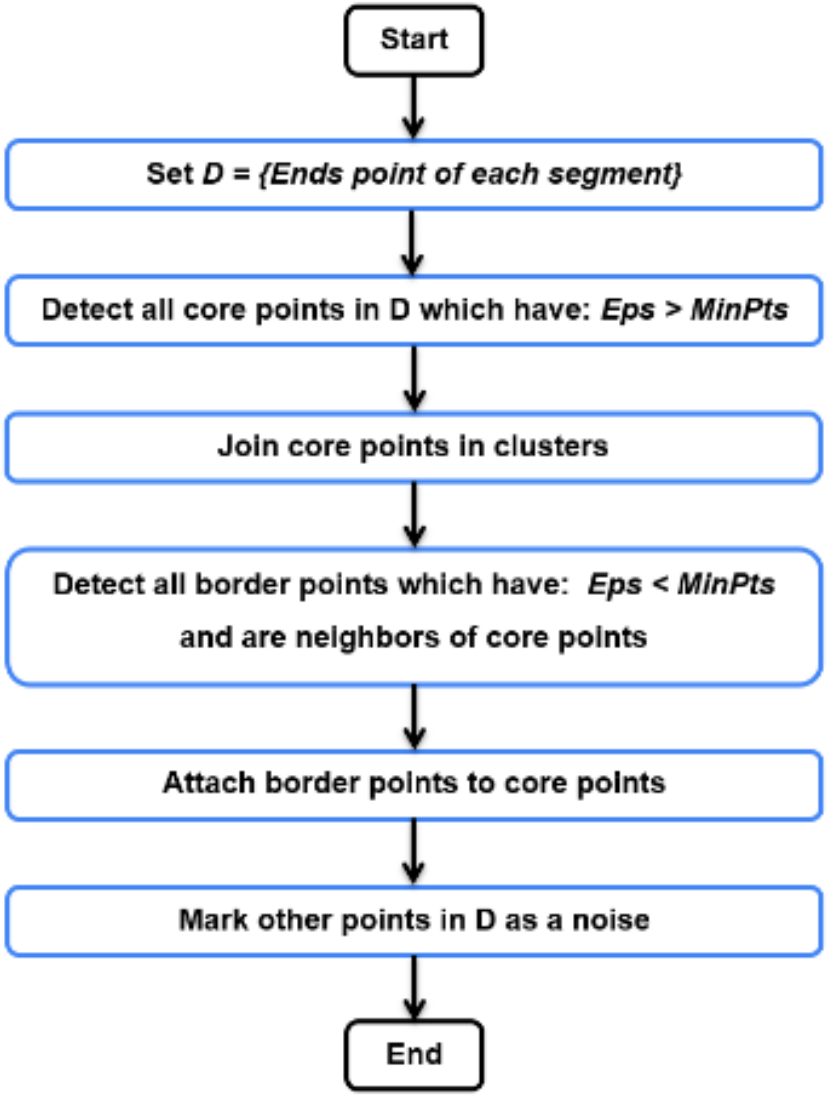
Original Points



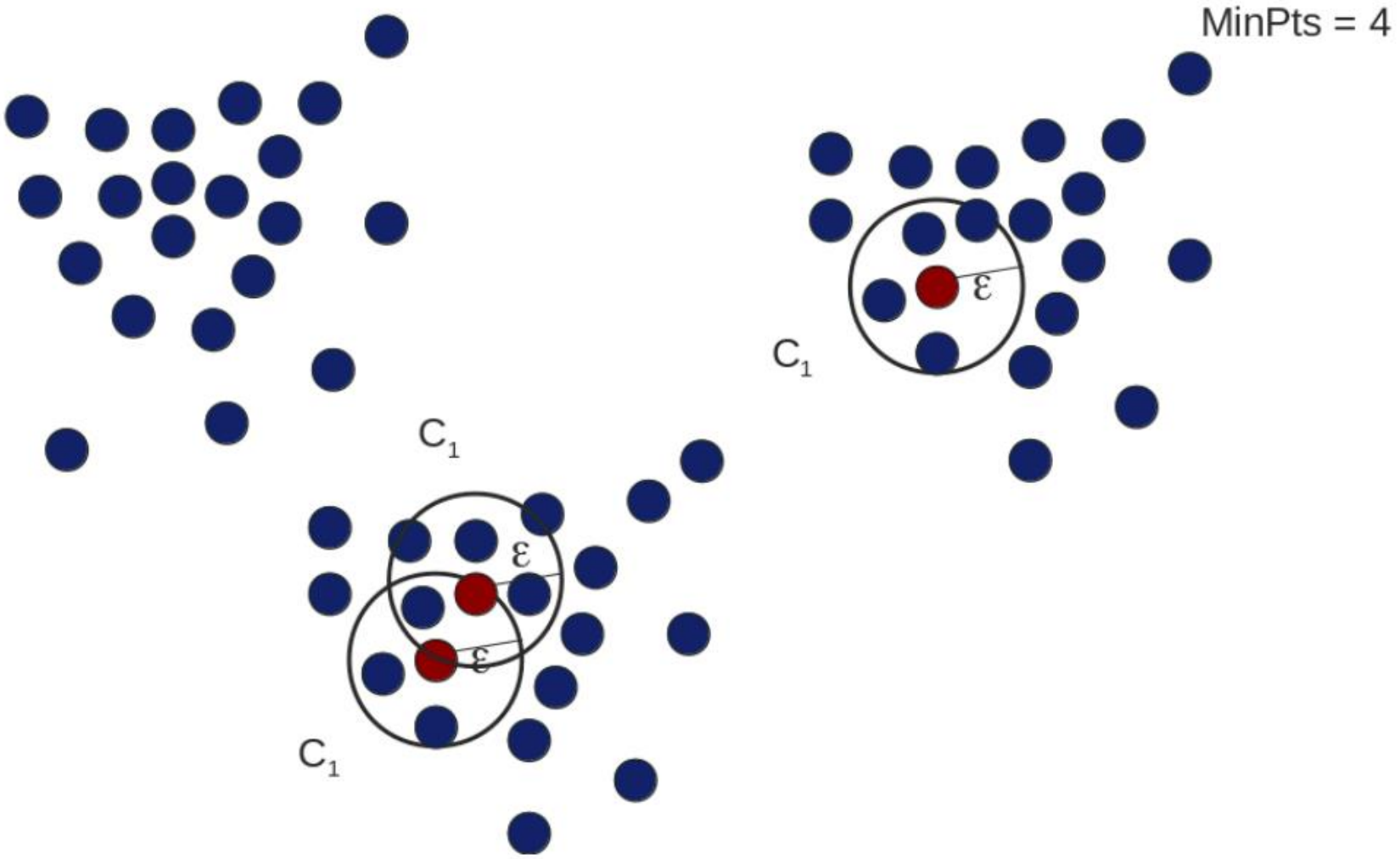
Clusters

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

DBScan :Flowchart



DBScan : Example



DBSCAN : Advantages

- Does not require one to specify the number of clusters in the data
- Can find arbitrarily shaped clusters. even find a cluster completely surrounded by a different cluster.
- Has a notion of noise, and is robust to outliers.
- Requires just two parameters and is mostly insensitive to the ordering of the points in the database.
- Designed for accelerate region queries.
- minPts and ε can be set by a domain expert

DBSCAN : Disadvantages

- DBSCAN is not entirely deterministic: Border points that are reachable from more than one cluster can be part of either cluster, depending on the order the data is processed.

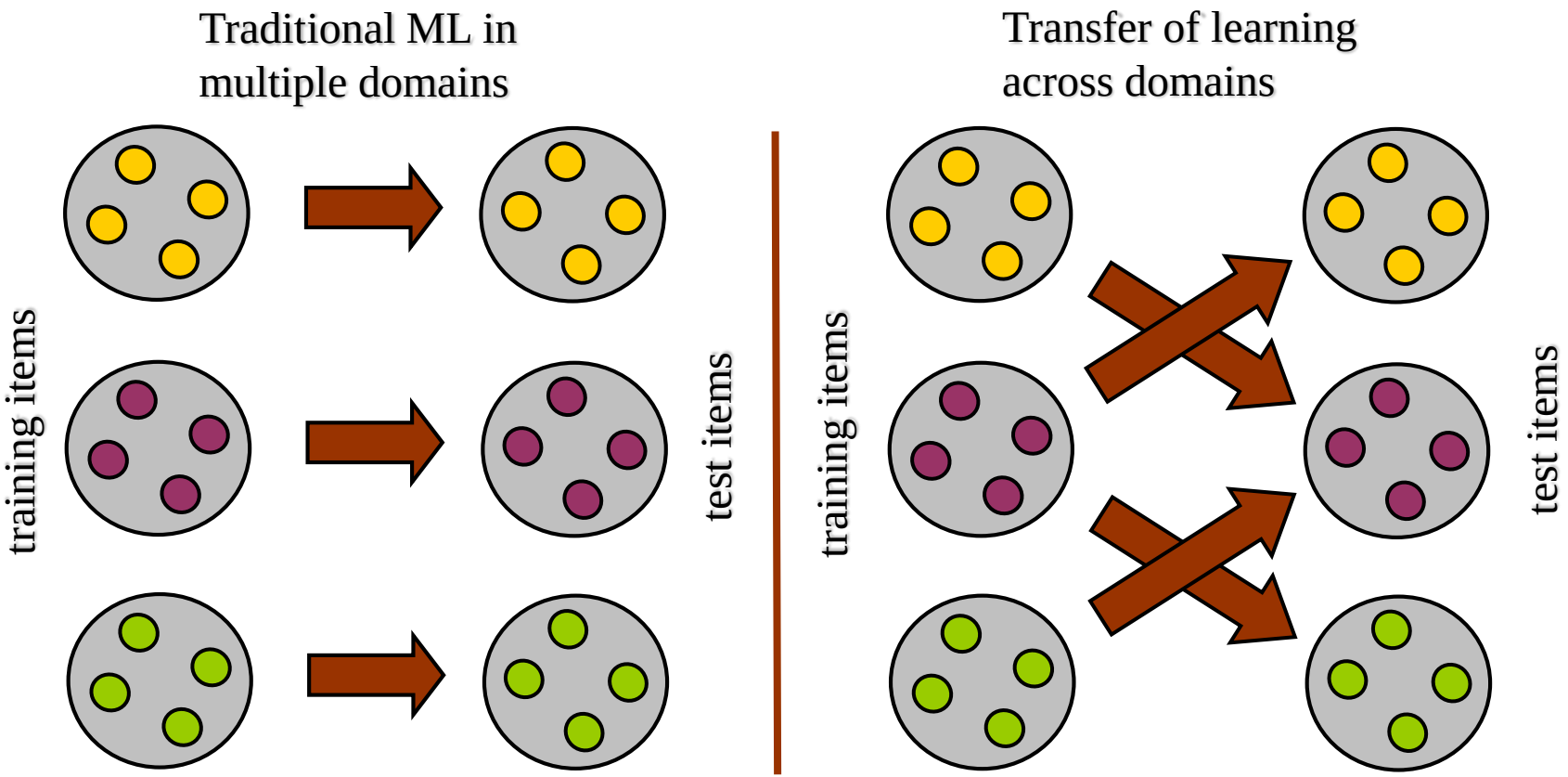
- The quality of DBSCAN depends on the distance measure used in the function regionQuery. (such as Euclidean distance)

- If the data and scale are not well understood, choosing a meaningful distance threshold ε can be difficult.

DBSCAN : Complexity

- **Time Complexity:** $O(n^2)$
  - for each point it has to be determined if it is a core point.
  - can be reduced to O(n*log(n)) in lower dimensional spaces by using efficient data structures (n is the number of objects to be clustered);
- **Space Complexity:** O(n).

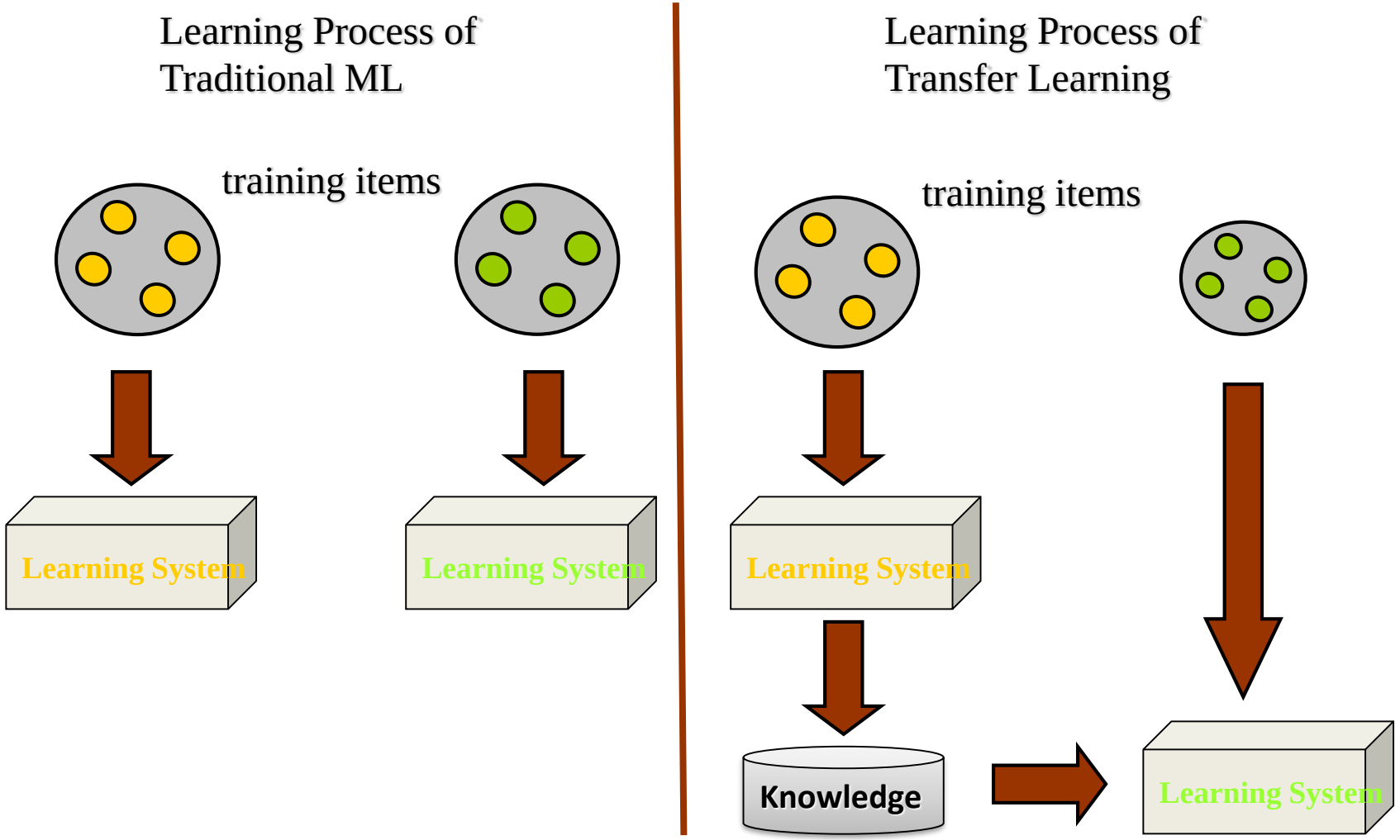
Traditional ML vs. TL



Humans can learn in many domains.

Humans can also transfer from one domain to other domains.

Traditional ML vs. TL



Why Transfer Learning?

- In some domains, labeled data are in short supply.
- In some domains, the calibration effort is very expensive.
- In some domains, the learning process is time consuming

  - *How to extract knowledge learnt from related domains to help learning in a target domain with a few labeled data?*
  - *How to extract knowledge learnt from related domains to speed up learning in a target domain?*

Transfer learning techniques may help!

Approaches to Transfer Learning

Transfer learning approaches	Description
Instance-transfer	To re-weight some labeled data in a source domain for use in the target domain
Feature-representation-transfer	Find a “good” feature representation that reduces difference between a source and a target domain or minimizes error of models
Model-transfer	Discover shared parameters or priors of models between a source domain and a target domain
Relational-knowledge-transfer	Build mapping of relational knowledge between a source domain and a target domain.

Text Mining Applications

1. Sentiment Analysis

Application: Analyzing customer reviews, social media posts, or survey responses to determine the sentiment (positive, negative, or neutral) of the text.

Use Case: Companies use sentiment analysis to monitor brand reputation, improve customer service, and make data-driven decisions.

2. Topic Modeling

Application: Automatically identifying topics or themes within a large corpus of text.

Use Case: News organizations use topic modeling to categorize articles, while businesses use it to analyze customer feedback and identify key issues.

3. Document Classification

Application: Classifying documents into predefined categories based on their content.

Use Case: Email filtering (spam vs. non-spam), categorizing legal documents, or sorting customer support tickets.

4. Named Entity Recognition (NER)

Application: Identifying and classifying proper nouns (e.g., names of people, organizations, locations) within a text.

Use Case: In information extraction tasks, such as automatically extracting names of key stakeholders from financial reports.

5. Information Retrieval

Application: Searching and retrieving relevant documents from a large corpus of text based on a user query.

Use Case: Search engines, legal document retrieval, and digital libraries use information retrieval techniques to provide relevant results.

6. Text Summarization

Application: Generating a concise summary of a longer text document while preserving the main ideas.

Use Case: Automated news summarization, summarizing legal documents, or generating abstracts for scientific papers.

7. Fraud Detection

Application: Detecting fraudulent activities by analyzing patterns in text data, such as transaction descriptions or insurance claims.

Use Case: Banks and insurance companies use text mining to identify suspicious activities and prevent fraud.

8. Market Intelligence

Application: Analyzing news, social media, and other online content to gain insights into market trends, competitor activities, and customer preferences.

Use Case: Companies use market intelligence to develop strategies, identify new opportunities, and stay ahead of competitors.

9. Healthcare and Biomedicine

Application: Analyzing clinical notes, research articles, and patient feedback to extract valuable medical information.

Use Case: Text mining helps in drug discovery, patient diagnosis, and personalized medicine by analyzing large volumes of medical literature and patient records.