Data Mining

Cluster Analysis
Basic Concepts and Algorithms

Slides by Tan, Steinbach, Kumar adapted by Pimprapai Thainiam

Topics

- **Introduction**
- Types of Clustering
- Type of Clusters
- Basic Clustering Algorithms
  - K-Means Clustering
    - Basic K-means Algorithm
    - Bisecting K-means Algorithm
  - Hierarchical Clustering
  - DBSCAN Algorithm

What is Cluster Analysis?

- **Cluster analysis** groups data objects based only on information found in the data that describes the objects and their relationships.
- There are two major purposes of the clustering:

  1. **Clustering for Understanding**

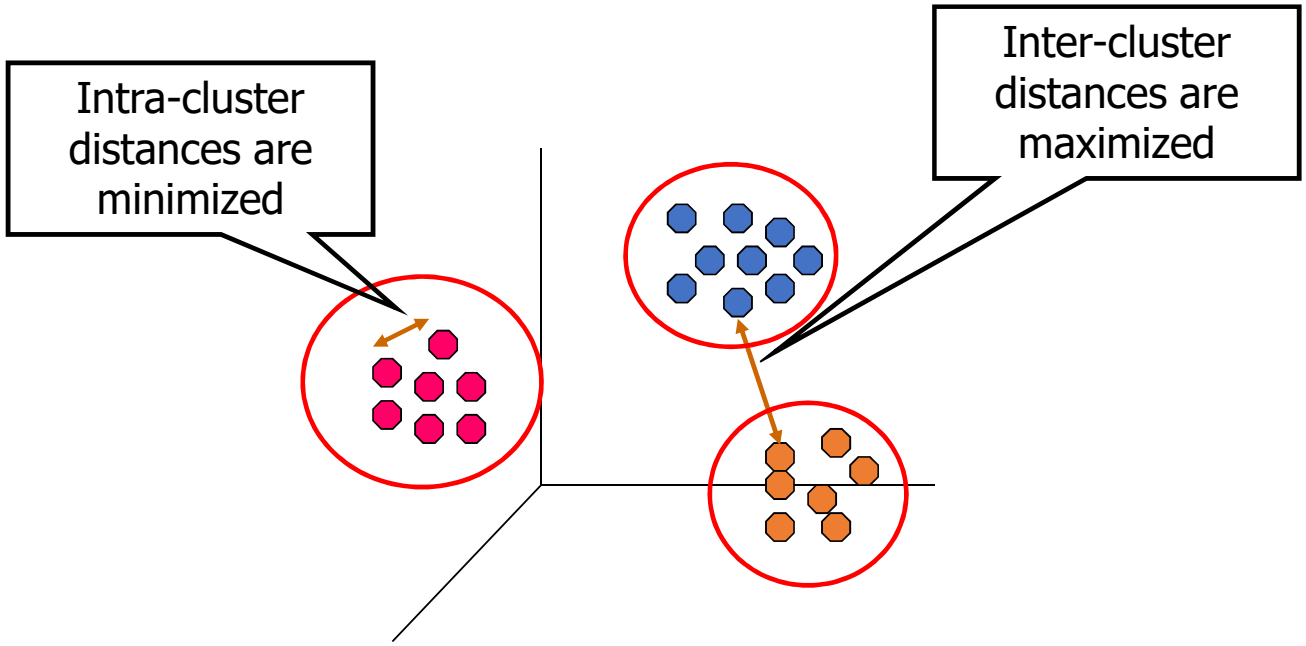
     Clusters are potential classes and cluster analysis is the study of techniques for automatically finding classes.

  2. **Clustering for Utility**

     Cluster analysis is the study of techniques for finding the most representative data object (cluster prototype).

What is Cluster Analysis?

- The **goal of clustering** is that the objects within a groups be similar (or related) to one another and different from (or unrelated to) the objects in other groups.
- The greater the similarity (or homogeneity) within a group and the greater the difference between groups the better or more distinct the clustering.



Clustering the same set of points

Three different ways of dividing points into clusters.

Original points

Two Clusters

Four Clusters

Six Clusters

Note: This figure illustrates that the definition of a cluster is imprecise and that the best definition depends on the nature of data and the desired results (e.g., number of clusters).

Supervised and Unsupervised Classification

- **Classification** is **supervised classification** where a new unlabeled object is assigned a class label using a model developed from objects with known class labels.
- **Clustering** can be regarded as a form of classification to create a labeling of objects with class (cluster) labels which are derived only from the data. Thus, cluster analysis is referred to as **unsupervised classification**.

Topics

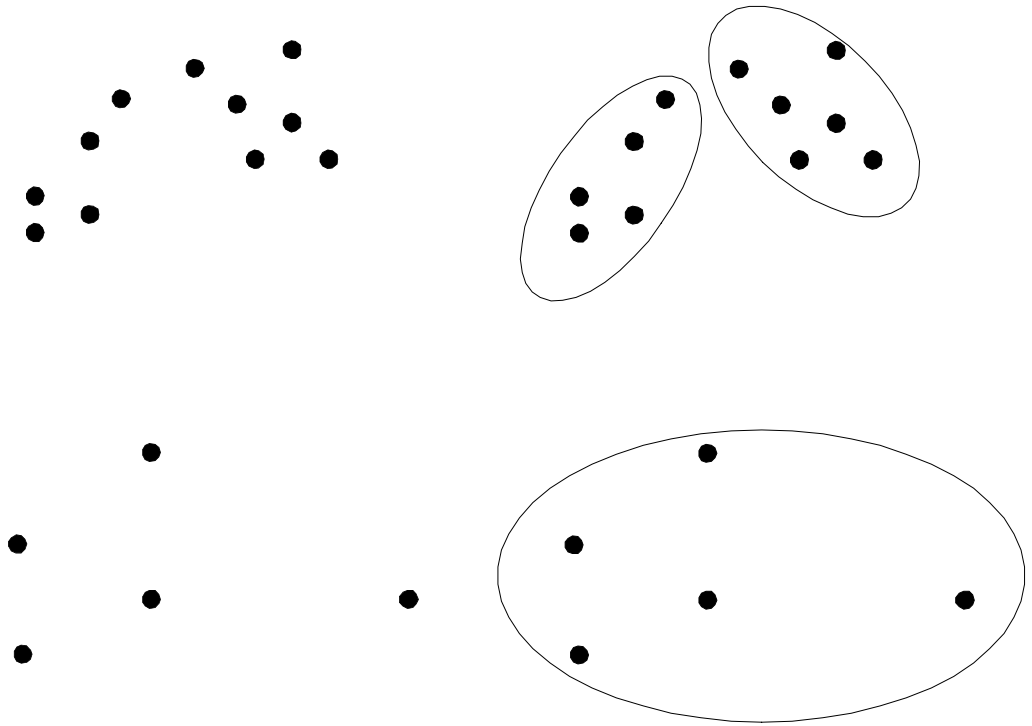
- Introduction
- **Types of Clustering**
- Type of Clusters
- Basic Clustering Algorithms
  - K-Means Clustering
    - Basic K-means Algorithm
    - Bisecting K-means Algorithm
  - Hierarchical Clustering
  - DBSCAN Algorithm

Types of Clustering

1. Hierarchical versus Partitional Clustering

- It classifies clustering techniques based on whether the set of clusters is nested (hierarchical) or unnested (partitional).
- **Partitional clustering** is a division of the set of data objects into non-overlapping subsets (clusters) such that each data object is in exactly one subset.
- **Hierarchical clustering** is a set of nested clusters that are organized as a tree where each node (cluster) in the tree (except for the leaf nodes) is the union of its children (subclusters), and the root of the tree is the cluster containing all the objects.

Partitional Clustering

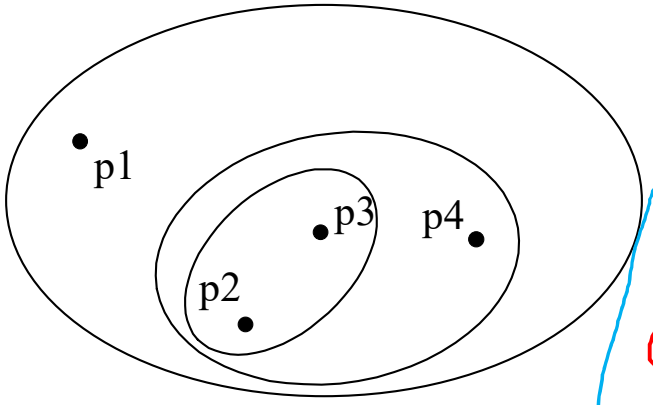


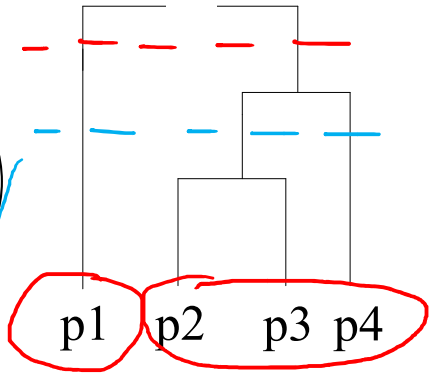
Original Points

A Partitional Clustering

Hierarchical Clustering

need 2 clusters





Nested cluster diagram

Tree (Dendrogram)

3 clusters

Types of Clustering

2. Exclusive versus Overlapping versus Fuzzy Clustering

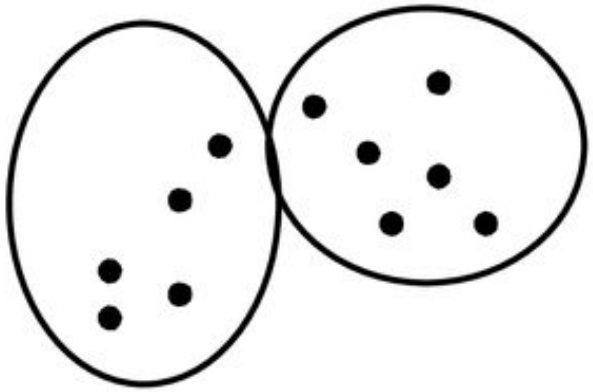
= partitional clustering

- **Exclusive clustering** assigns each object to a single cluster.
- **Overlapping** or **non-exclusive clustering** assigns each object to one or more clusters.

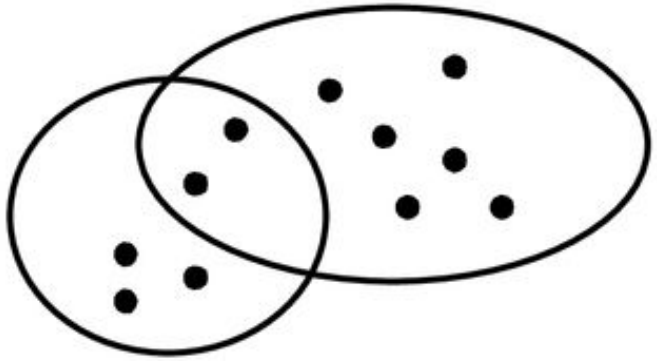
= Hierarchical clustering

- **Fuzzy** or **probabilistic clustering** assigns every object to every cluster with a membership weight that is between 0 (absolutely doesn't belong) and 1 (absolutely belongs). Usually, the sum of the weights for each object must equal to 1. The results from this clustering can be converted to an exclusive clustering by assigning each object to the cluster in which its membership weight or probability is highest.

Exclusive Clustering



Overlapping Clustering (Non-exclusive clustering)



Fuzzy Clustering (Probabilistic Clustering)

Object No.	Degree of Membership	
	Cluster 1	Cluster 2
1	0.8	0.2
2	0.3	0.7
3	0.6	0.4
⋮	⋮	⋮
n	0.9	0.1

cluster 1 = {1, 3, →, n}

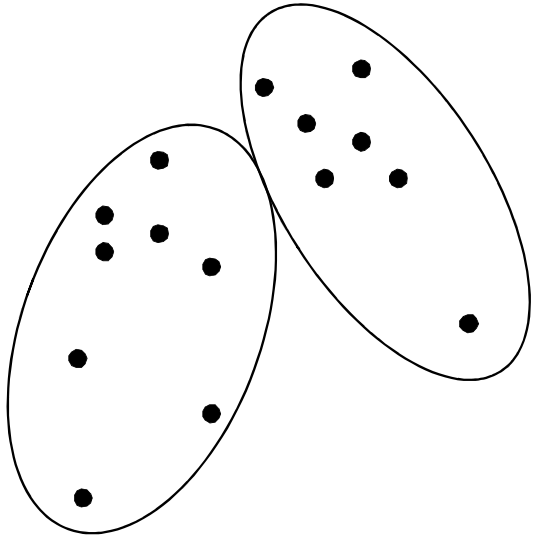
cluster 2 = {2, →}

Types of Clustering

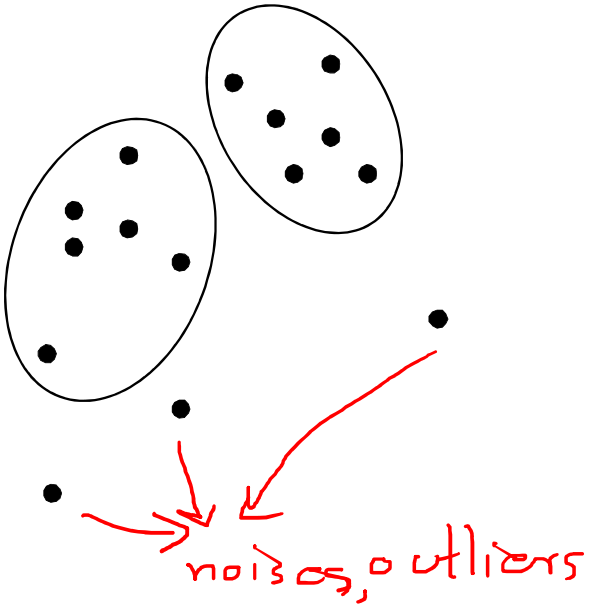
3. Complete versus Partial Clustering

- **Complete clustering** assigns every object to a cluster.
- **Partial clustering** does not assign every object to a cluster. The reason is that some objects in the data set may represent *noise, outliers*, or *uninterested background*.

Complete Clustering



Partial Clustering



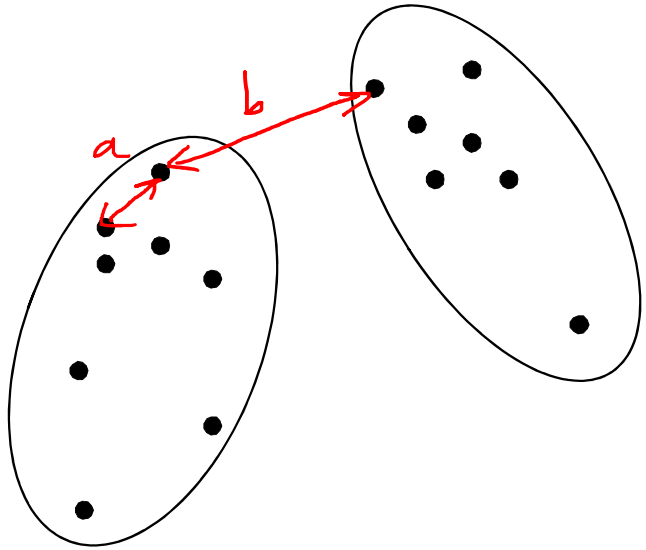
Topics

- Introduction
- Types of Clustering
- **Type of Clusters**
- Basic Clustering Algorithms
  - K-Means Clustering
    - Basic K-means Algorithm
    - Bisecting K-means Algorithm
  - Hierarchical Clustering
  - DBSCAN Algorithm

Types of Clusters

1. Well-Separated

- A cluster is a set of objects in which each object is closer (or more similar) to every other object in the cluster than to any object not in the cluster.
- The distance between any two points in different groups is a lot larger than the distance between any two points within a group.

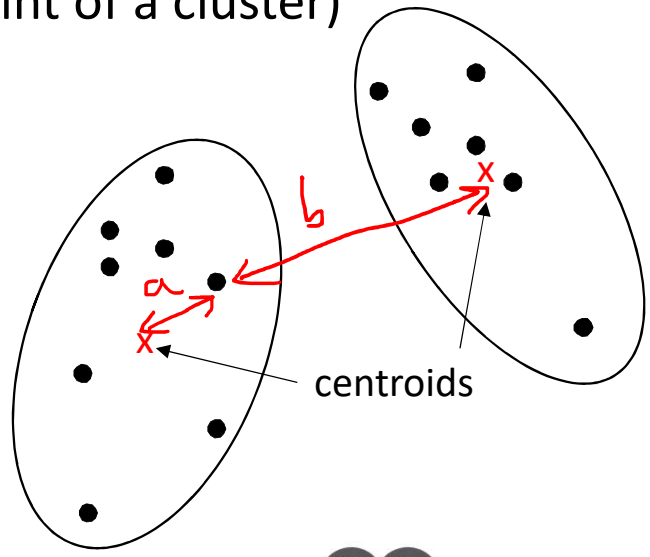


$a < b$

Types of Clusters

2. Prototype-Based

- A cluster is a set of objects in which each object is closer (more similar) to the prototype that defines the cluster than to the prototype of any other cluster.
- For data with continuous attributes, the prototype of a cluster is a **centroid** (the average (mean) of all the points in the cluster).
- For data with categorical attributes, the prototype of a cluster is a **medoid** (the most representative point of a cluster)

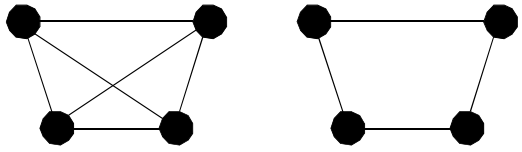


a < b

Types of Clusters

3. Graph-Based

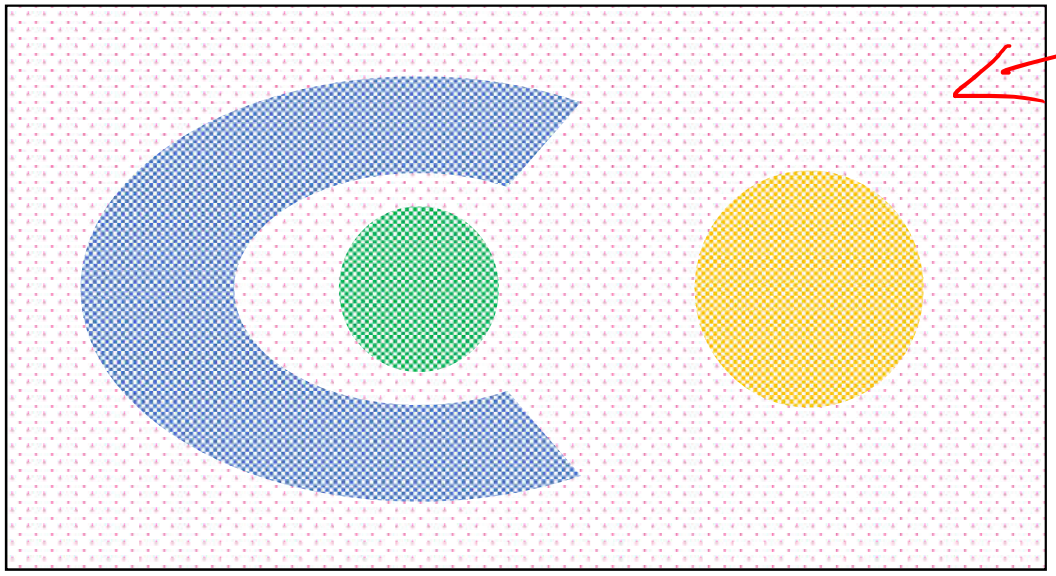
- If the data is represented as a graph, where nodes are objects and the links represent connections among objects then a cluster can be defined as a connected component.
- An example of graph-based clusters are **continuity-based clusters**, where two objects are connected only if they are within a specified distance of each other. This implied that each object in continuity-based clusters is closer to some other object in the cluster than to any point in a different cluster.
- It can have trouble of assigning a group to objects when noise is present.
- A cluster can be defined as a **clique** which is a set of nodes in a graph that are completely connected to each other.
- If we add connections between objects in the order of their distance from one another, a cluster is formed when a set of objects forms a clique.



Types of Clusters

4. Density-Based

- A cluster is a dense region of objects that is surrounded by a region of low-density.
- A density-based definition of a cluster is employed when the clusters are irregular or intertwined, and when noise and outliers are present.



Topics

- Introduction
- Types of Clustering
- Type of Clusters
- **Basic Clustering Algorithms**
  - **K-Means Clustering**
    - **Basic K-means Algorithm**
    - Bisecting K-means Algorithm
  - Hierarchical Clustering
  - DBSCAN Algorithm

Prototype-based Clustering Techniques

- **Prototype-based clustering techniques** create a one-level partitioning of the data objects.
- Two most well-known techniques are:

= partion data objects into different groups

1. **K-means:** It defines a prototype in terms of a **centroid**, which is usually the mean of a group of points, and is typically applied to objects in a continuous n-dimensional space.
2. **K-medoid:** It defines a prototype in terms of a **medoid**, which is the most representative point for a group of points, and can be applied to a wide range of data since it requires only a proximity measure for a pair of objects.

We will focus only on **K-means clustering** in this class!

K-means Clustering

- **K-means clustering** is a prototype-based, partitional clustering technique that attempts to find a user-specified number of clusters (K), which are represented by their centroids.
- **The basic K-means algorithm** is described as follows:

  Step 1: Choose K initial centroids randomly, where K is a user specified parameter (the number of clusters desired).

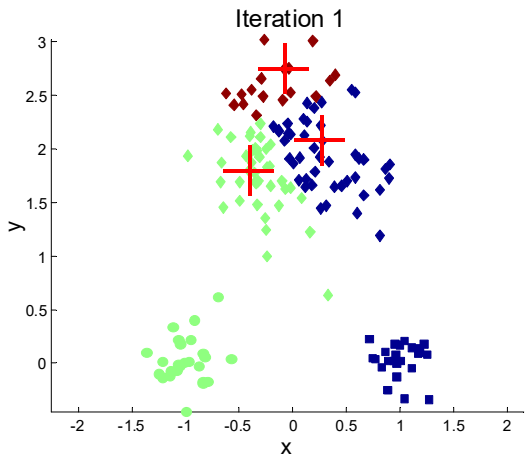
  Step 2: Assign each data object to the closest centroid, and each collection of points assigned to a centroid is a cluster.

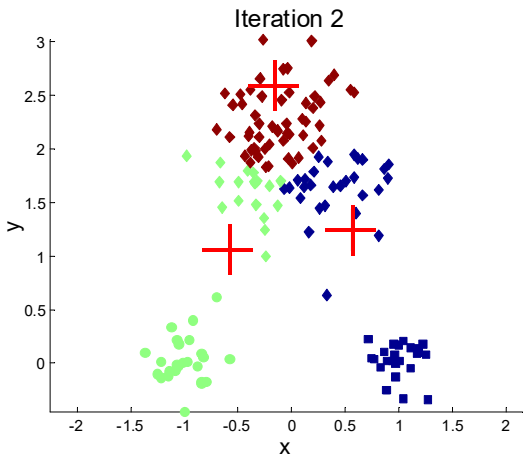
  Step 3: Update the centroid of each cluster based on the objects assigned to the cluster by determining the mean of the points in the cluster.

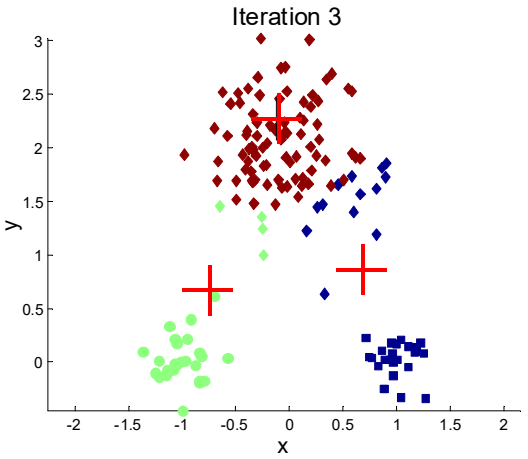
  Step 4: Repeat step 2 and 3 until no object changes clusters, or equivalently, until the centroids remain the same (no more changes occur)

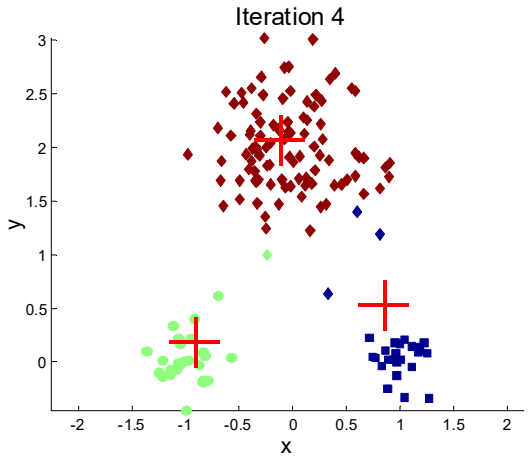
  Note: The stopping criterion could be replaced by a weaker condition, e.g., repeat until only 1% of the objects change clusters.

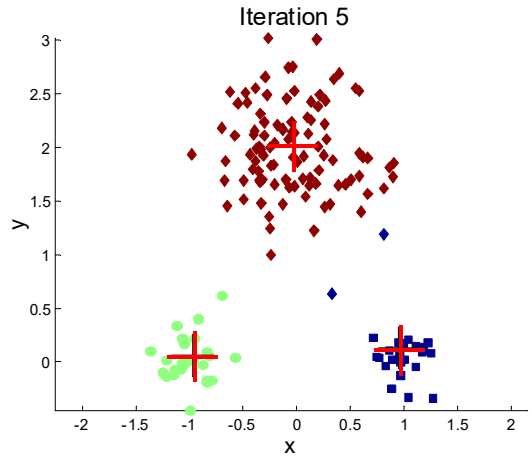
K-means Example

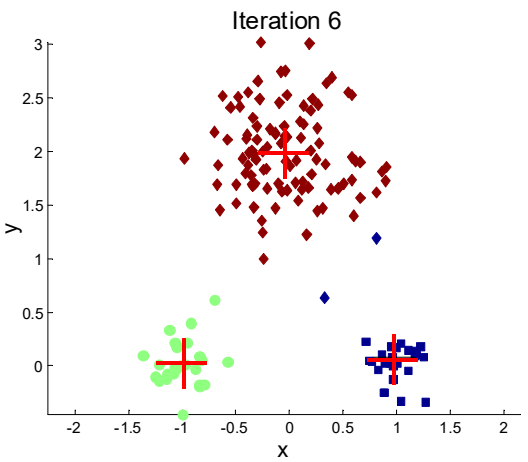






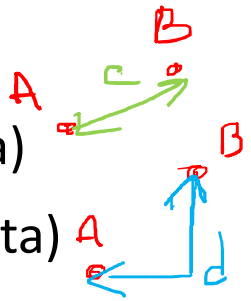






The Basic K-means Algorithm

- A **proximity measure** is needed to measure the closeness of an object to centroids in order to determine which centroid a particular object should be assigned to.
- Proximity measures are as follows:
  - **1) Euclidean (L_2) distance** is appropriate for Euclidean data (continuous data)
  - **2) Manhattan (L_1) distance** is appropriate for Euclidean data (continuous data)
  - **3) Cosine similarity** is appropriate for document-term data
  - **4) Jaccard measure** is appropriate for document-term data



- The centroid can vary depending on the proximity measure for the data and the goal of the clustering.
- The **goal** of the clustering is typically expressed by an **objective function** that depends on the distances of the objects to one another or to the cluster centroids.

The Basic K-means Algorithm

1. Continuous Data

1) Euclidean distance as our proximity measure, the objective function could be minimizing **the summation of the squared error (SSE)**.

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

where x is an object

C_i is the i^{th} cluster

c_i is the centroid of cluster C_i

K is the number of clusters

$dist$ is the standard Euclidean distance between two objects in Euclidean space

Note: The centroid that minimizes the SSE of the cluster is **mean**, thus we can recompute the centroid of each cluster (step 3) by finding the mean.

The Basic K-means Algorithm

2) Manhattan distance as our proximity measure, the objective function would be minimizing **the summation of error (SE)**.

$$SE = \sum_{i=1}^{K} \sum_{x \in C_i} dist(c_i, x)$$

where $dist$ is the Manhattan distance between two objects in Euclidean space

Note: The centroid that minimizes the SE of the cluster is **median**, thus we can recompute the centroid of each cluster (step 3) by finding the median.

The Basic K-means Algorithm

2. Document Data

Cosine similarity measure as our proximity measure, the objective function would be maximizing **the summation of the cosine similarity** of an object to its cluster centroid known as the **total cohesion**.

$$Total\ Cohesion = \sum_{i=1}^{K} \sum_{x \in C_i} cosine(c_i, x)$$

Note: The centroid that maximize the total cohesion is **mean**, thus we can recompute the centroid of each cluster (step 3) by finding the mean.

The Basic K-means Algorithm

Choosing Initial Centroids

- A common approach is to choose the initial centroids randomly; this approach has many disadvantages:

  - Different runs of K-means typically produce different clustering results.

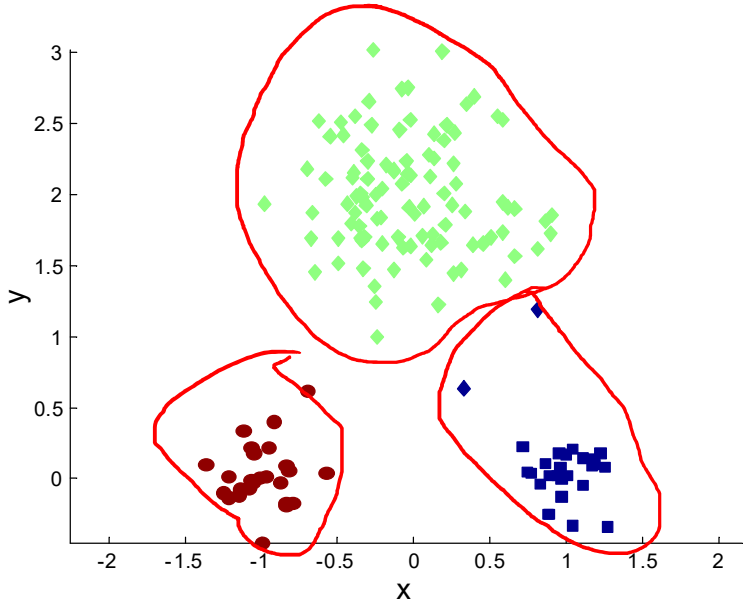
  -The resulting clusters are often poor.

  -The clustering algorithm takes much time until it is terminated.

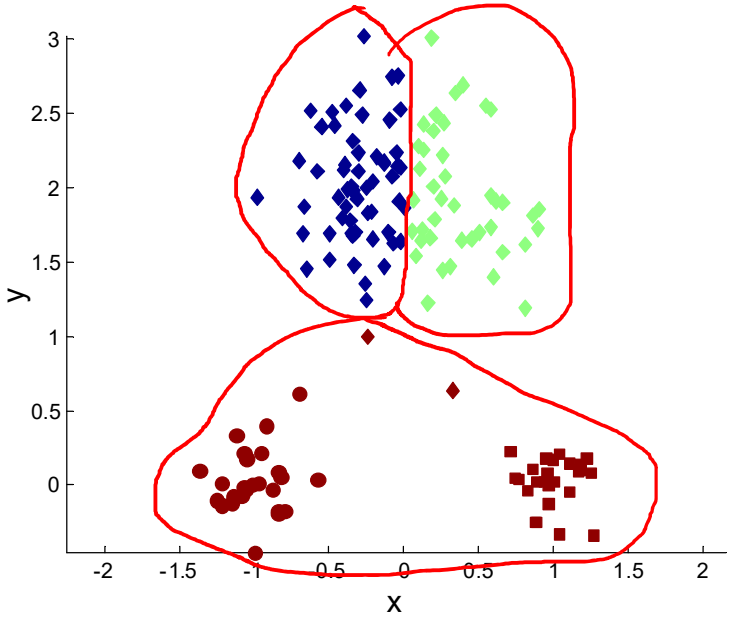
- If centroids are chosen carefully, then the clustering algorithm will be terminated after a few repeat loop.

The Basic K-means Algorithm

Two different K-means Clusterings with two different set of initial centroids

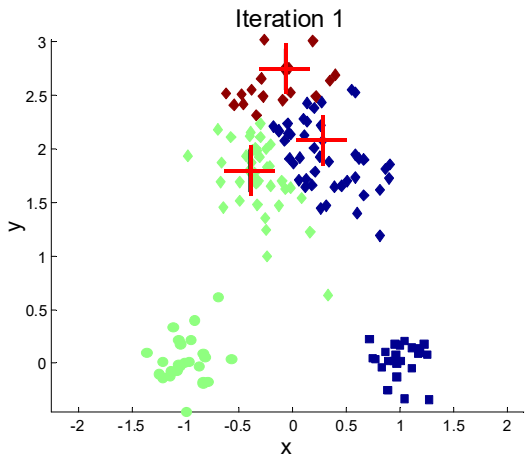


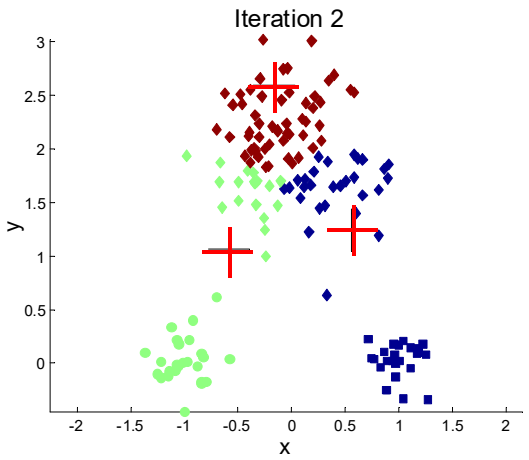
Optimal Clustering
(Global optimum)

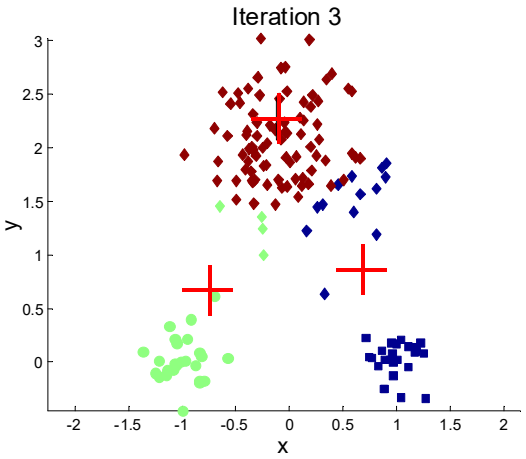


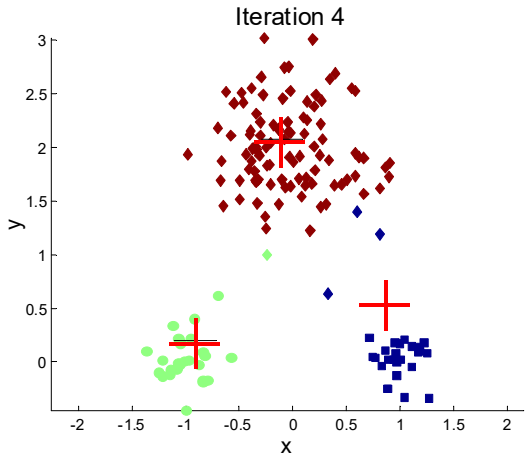
Sub-optimal Clustering
(Local optimum)

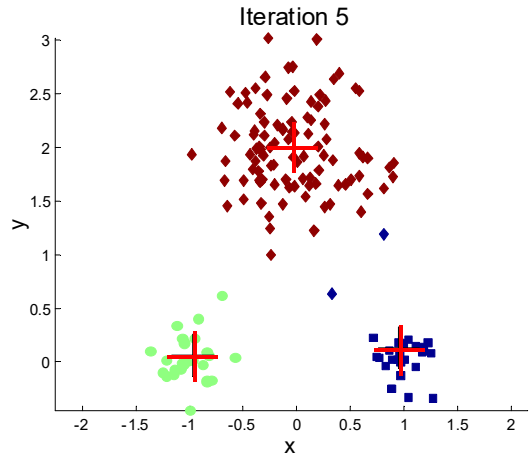
Example: The global optimum clustering result is found after 6 iterations.

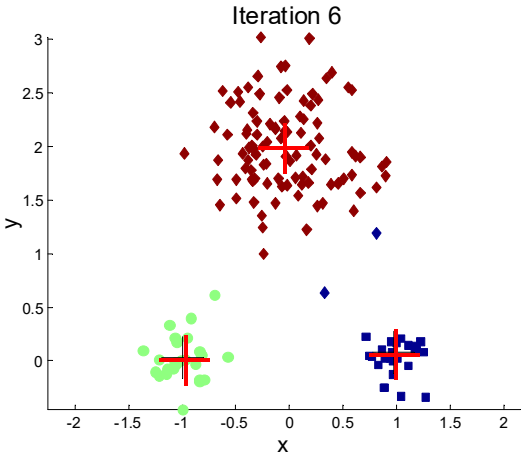












The Basic K-means Algorithm

Solutions to Initial Centroids Problem

1. Perform multiple runs: It works as follows

1) Each run starts with a different set of randomly chosen initial centroids

2) Select the set of clusters with the minimum SSE.

Note that this approach may not work very well, depending on the data set and the numbers of clusters sought.

2. Use a hierarchical clustering technique: It works as follows

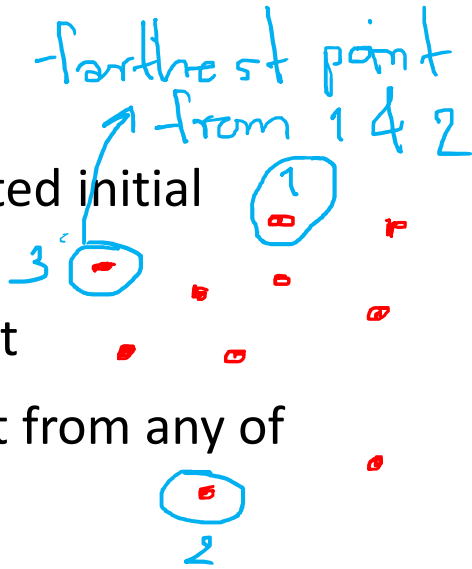
1) Take a sample of points

2) Cluster them using a hierarchical clustering technique

3) Extract K clusters from the hierarchical clustering

4) Use the centroid of those clusters as the initial centroids

The Basic K-means Algorithm



3. **Farthest approach:** This approach guarantee random and well-separated initial centroids. It works as follows

   1) Select the first centroid at random or take the centroid of all object

   2) For each successive initial centroid, select the point that is farthest from any of the initial centroids already selected.

4. **Farthest approach (sample objects):**

   Implement approach 3 to a sample of objects. This is because

   - Approach 3 can select outliers rather than objects in dense regions
   - It is expensive to compute the farthest object from the current set of initial centroids.

5. **Use a variant of K-means approach** which is less sensitive to initialization problems, such as **bisecting K-means**.

K-means Clustering: Additional Issues

1. Handling Empty Clusters

- Empty clusters can be obtained if no objects are allocated to a cluster during the assignment step.
- This problem can be solved by

    1) Choosing a replacement centroid that is farthest away from any current centroids

    2) Choosing a replacement centroid from the cluster that has the highest SSE; this will typically split the cluster and reduce the overall SSE of the clustering.

K-means Clustering: Additional Issues

2. Outliers

- Outliers can affect the resulting cluster centroids which will result in higher SSE.
- This problem can be solved by

  1) Discovering outliers and eliminating them beforehand when outliers are present.

  2) Keeping track of the SSE contributed by each point, and eliminating those points with usually high contributions, especially over multiple runs.

K-means Clustering: Additional Issues

3. Reducing the SSE with Postprocessing

- An obvious way to reduce the SSE is to find more cluster (larger K), but we can still reduce SSE without increasing the number of clusters.
- This approach is often possible to escape local SSE minima and still produce a cluster solution with the desired number of clusters.
- It fixes the set of cluster produced by implementing the approach that alternates cluster splitting and merging phases.

  Splitting phase → This can decrease the total SSE by increasing the number of clusters which can be done by

  1) Splitting a cluster with the largest cluster SSE into two clusters.

  2) Introducing a new cluster centroid:

     - Choose the point that is farthest from any cluster center
     - Choose a point randomly from all points or from the points with highest SSE

K-means Clustering: Additional Issues

Merging phase → This can decrease the number of clusters, while trying to minimize the increase in total SSE which can be done by

1) Removing the centroid that corresponds to the cluster and reassigning the points of that cluster to other clusters. The cluster that is dispersed should be the one that has lowest cluster SSE.
2) Merging the two clusters that have lowest cluster SSE

K-means Clustering: Additional Issues

4. Updating Centroids Incrementally

- Instead of updating cluster centroids after all points have been assigned to a cluster, the centroid can be updated incrementally, after each assignment of a point to a cluster.
- This strategy guarantees that empty clusters are not produced since all clusters start with a single point.
- The relative weight of the point being added may be adjusted; for example, the weight of points is often decreased as the clustering proceed which will result in better accuracy and faster convergence.
- The clusters produced may depend on the order in which the point are processed which can be addressed by randomizing the order in which the points are processed.
- This approach is slightly more expensive than the original approach.

Topics

- Introduction
- Types of Clustering
- Type of Clusters
- Basic Clustering Algorithms
  - K-Means Clustering
    - Basic K-means Algorithm
    - **Bisecting K-means Algorithm**
  - Hierarchical Clustering
  - DBSCAN Algorithm

Bisecting K-means

- **Bisecting K-means** is a straightforward extension of the basic K-means algorithm which is used to speed up the algorithm by reducing the number of times to compute proximity measure.
- It works by splitting the set of all points into two clusters, select one of these clusters to split, and so on, until K clusters have been produced.

1: Initialize the list of clusters to contain the cluster containing all points.
2: **repeat**
3: Select a cluster from the list of clusters
4: **for** $i = 1$ to *number_of_iterations* **do**
5: Bisect the selected cluster using basic K-means
6: **end for**
7: Add the two clusters from the bisection with the lowest SSE to the list of clusters.
8: **until** Until the list of clusters contains K clusters

Bisecting K-means

- There are a number of different ways to choose which cluster to split as follows:

  1) Choose the largest cluster at each step

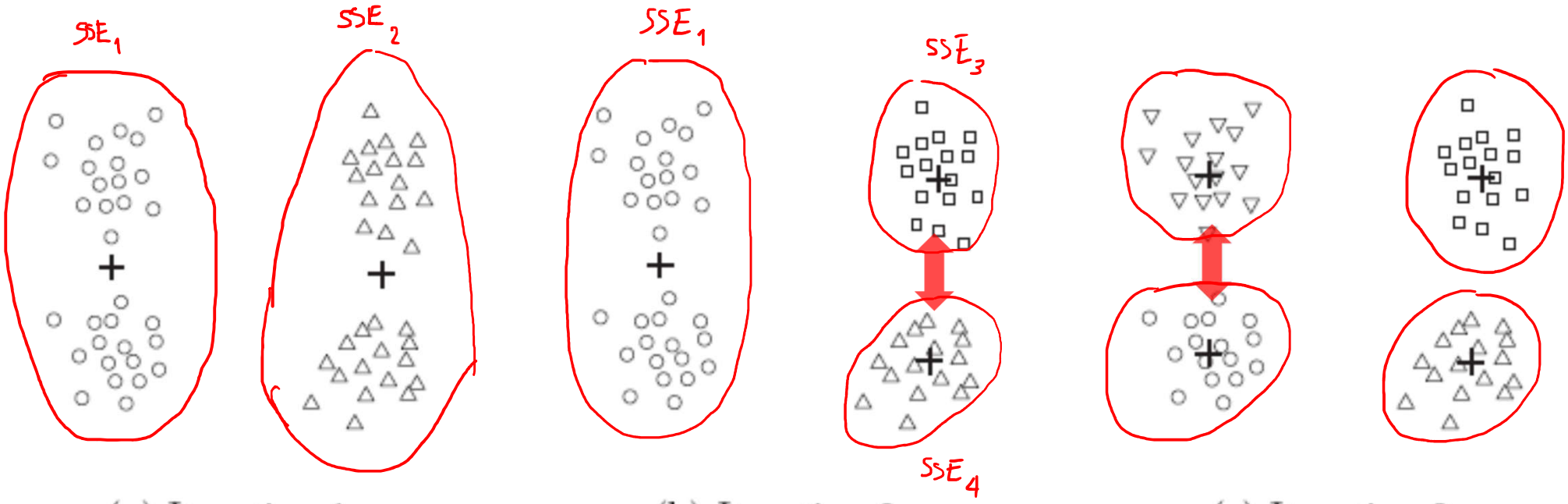
  2) Choose the one with the largest SSE

  3) Use other criteria based on both size and SSE

- We often refine the resulting clusters by using their centroids as the initial centroids for the basic K-means algorithm.

Bisecting K-means Example

Bisecting K-means on the four clusters example.



(a) Iteration 1. (b) Iteration 2. (c) Iteration 3.

$SSE_2 > SSE_1$

$SSE_1 > SSE_3, SSE_4$

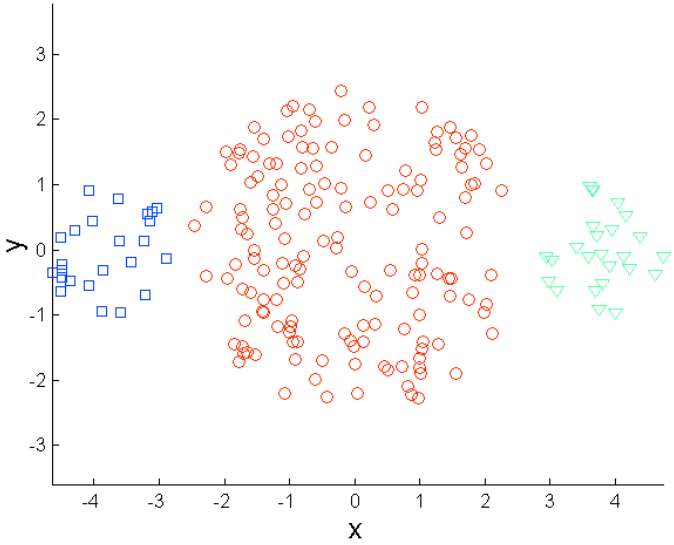
Limitations of K-means

Limitations of K-means

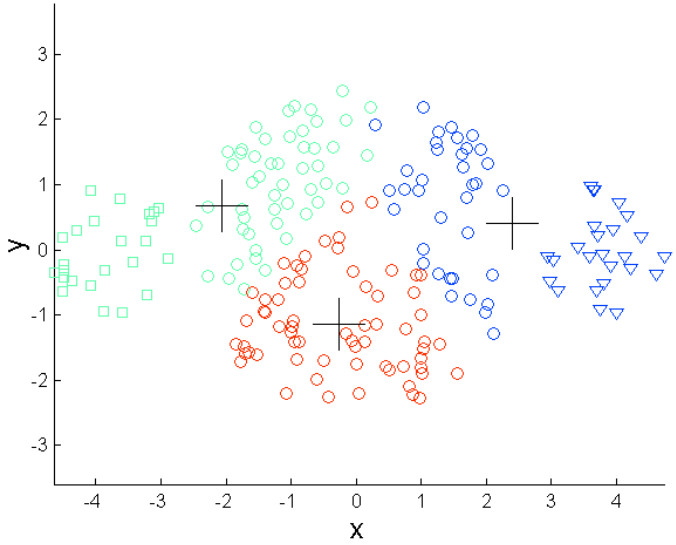
1. K-means is not suitable for all types of data.
2. K-means cannot handle non-globular clusters.
3. K-means has problems when clusters are of different sizes and densities.
4. K-means has problems when the data contains outliers, but this limitation could be resolved by implementing outliers detection approach.

Limitations of K-means: Differing Sizes

K-means with clusters of different sizes



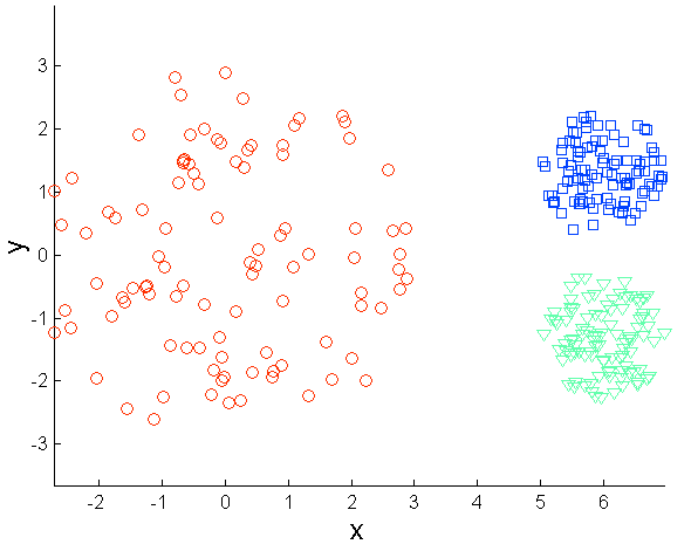
Original Points



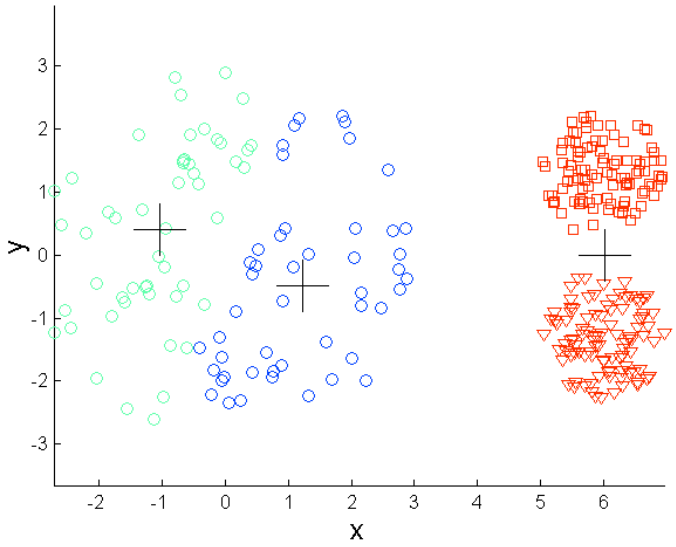
K-means (3 Clusters)

Limitations of K-means: Differing Density

K-means with clusters of different density



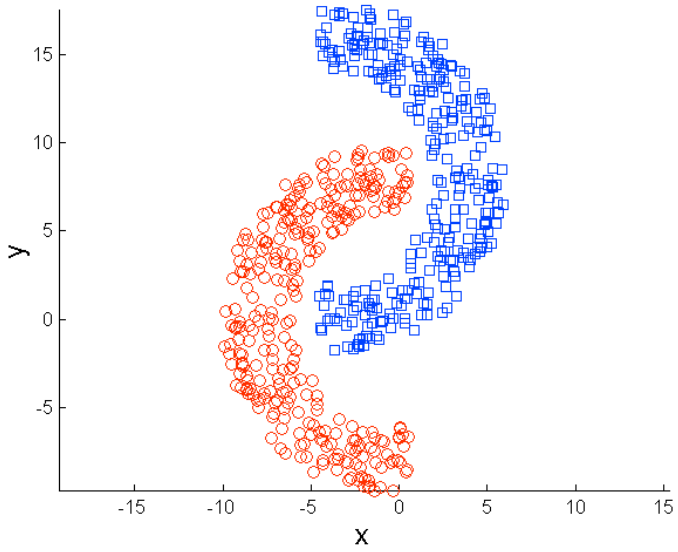
Original Points



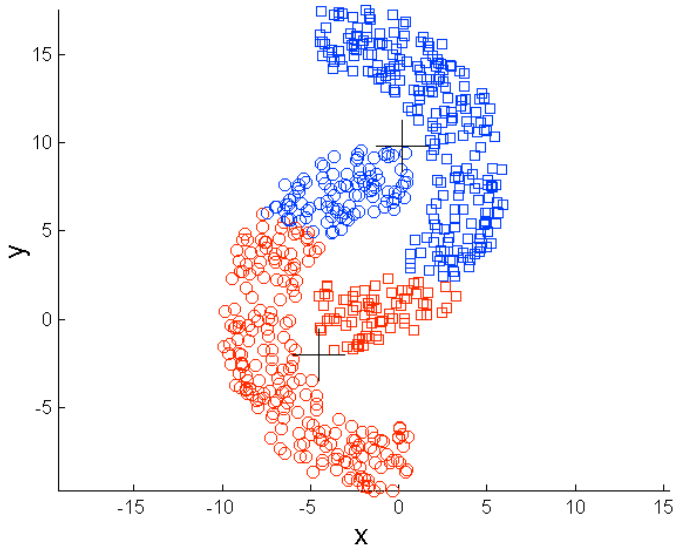
K-means (3 Clusters)

Limitations of K-means: Non-globular Shape

K-means with non-globular clusters



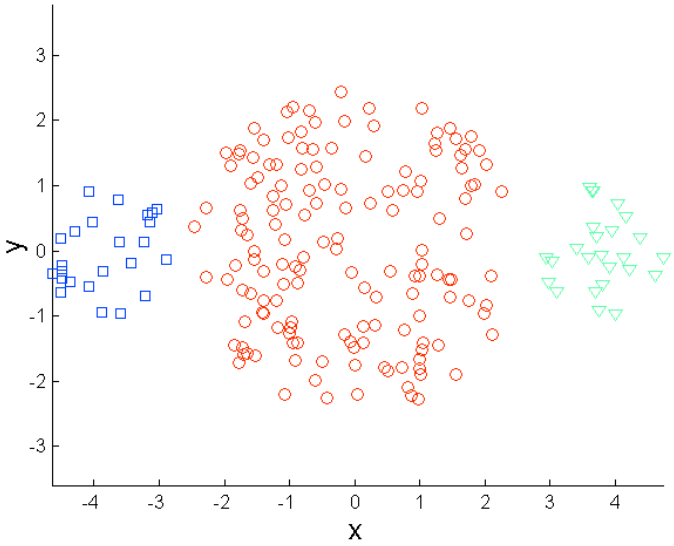
Original Points



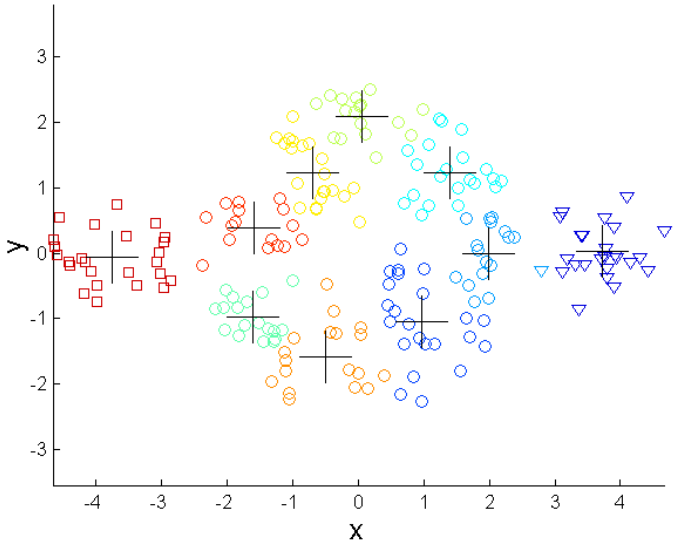
K-means (2 Clusters)

Overcoming K-means Limitations

One solution is to use a larger number of clusters which will find parts of clusters. However, we still need to put these clusters together to get the desired number of clusters.

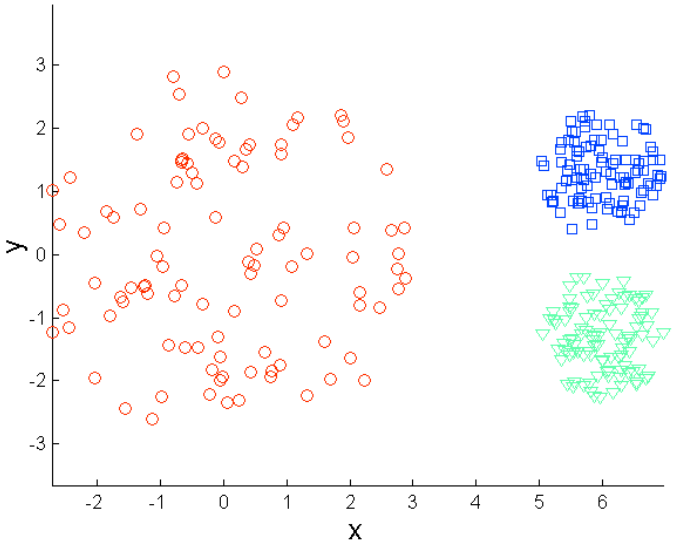


Original Points

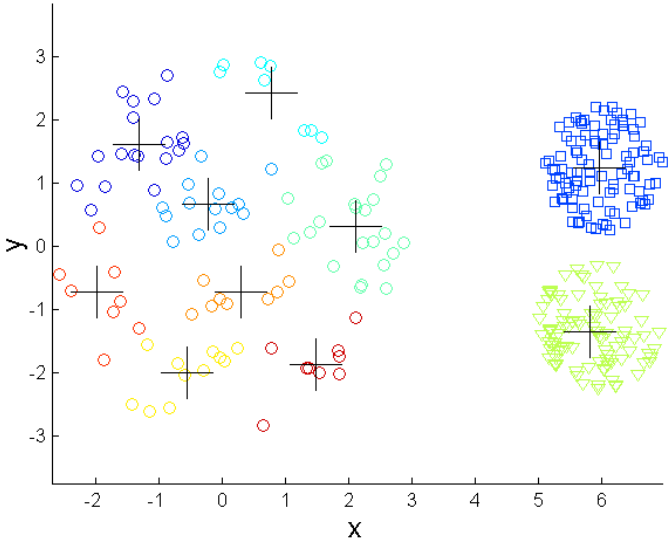


K-means Clusters

Overcoming K-means Limitations

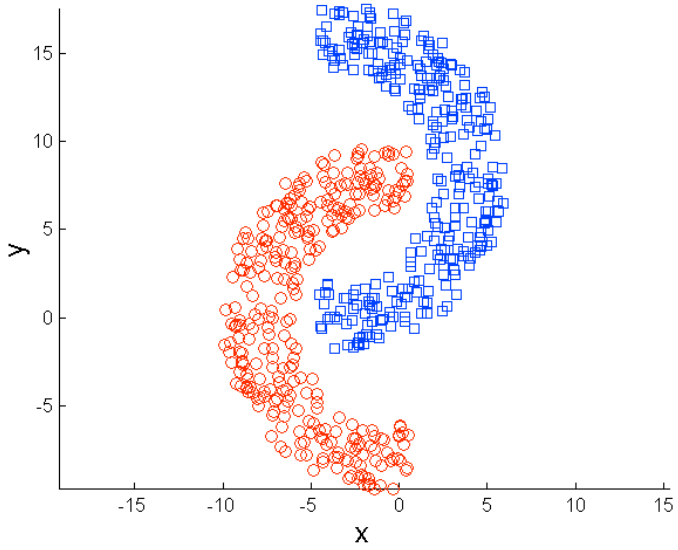


Original Points

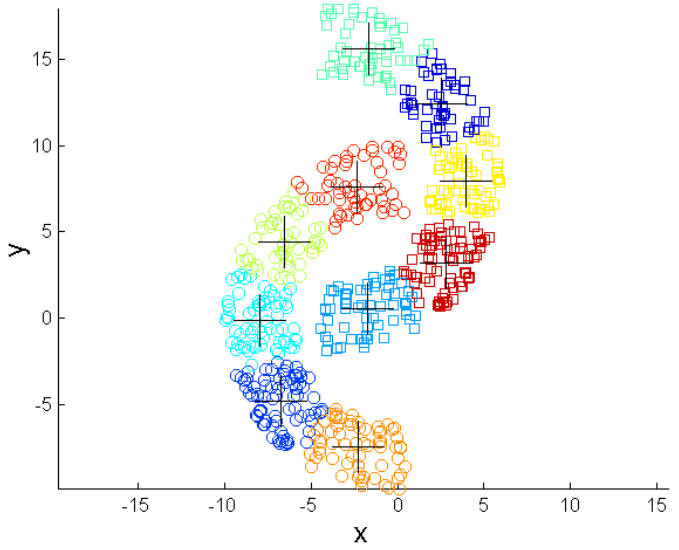


K-means Clusters

Overcoming K-means Limitations



Original Points



K-means Clusters

Topics

- Introduction
- Types of Clustering
- Type of Clusters
- Basic Clustering Algorithms
  - K-Means Clustering
    - Basic K-means Algorithm
    - Bisecting K-means Algorithm
  - **Hierarchical Clustering**
  - DBSCAN Algorithm

Hierarchical Clustering

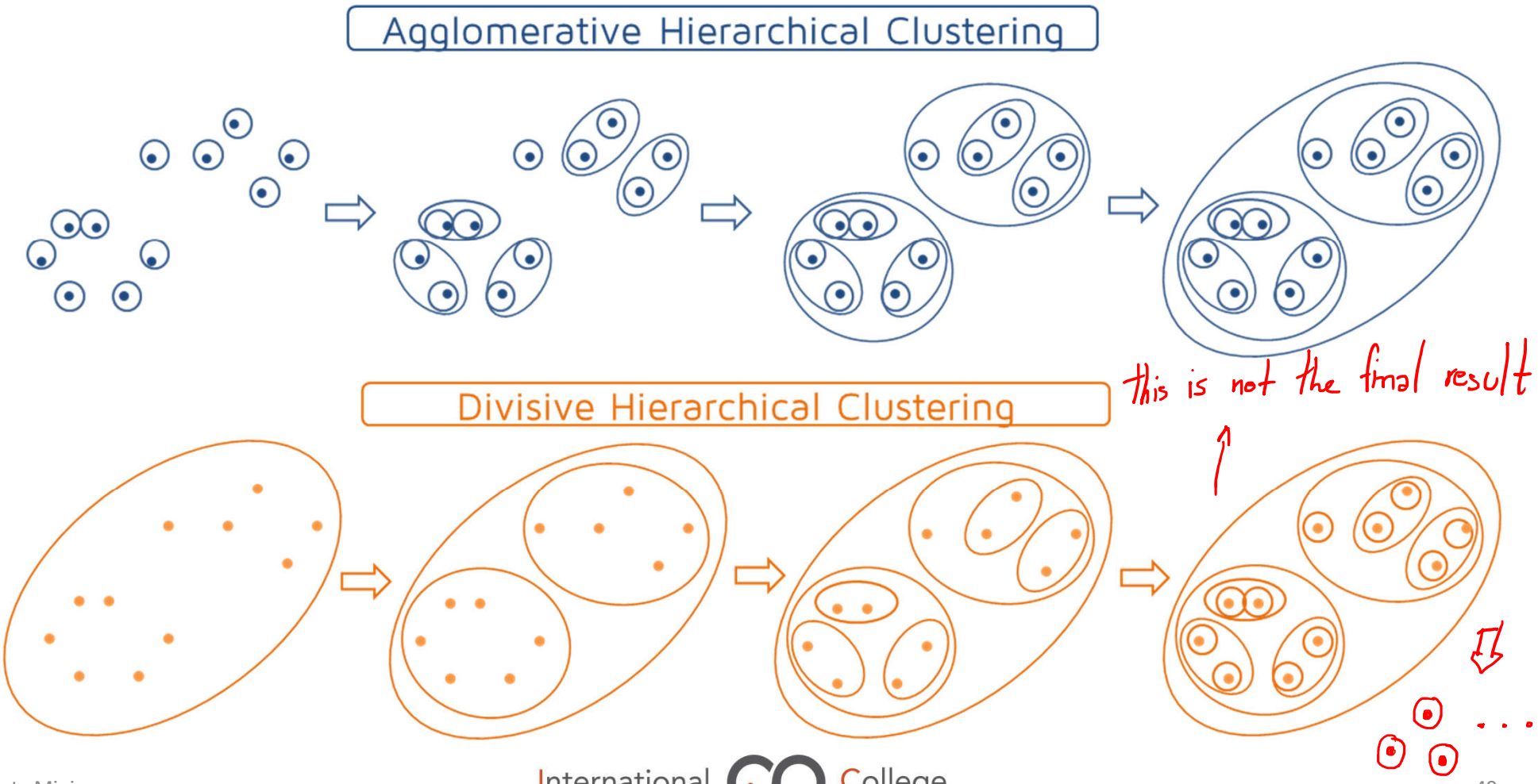
- **Hierarchical clustering** is a method of cluster analysis which seeks to build a hierarchy of clusters. Strategies for hierarchical clustering generally fall into two types:

  1. **Agglomerative:** Start with the points as individual clusters and, at each step, merge the closest pair of clusters. This requires defining a notion of cluster proximity.

  2. **Divisive:** Start with one, all-inclusive cluster and, at each step, split a cluster until only singleton clusters of individual points remain. In this case, we need to decide which cluster to split at each step and how to do the splitting.

We will focus only on agglomerative hierarchical clustering techniques!

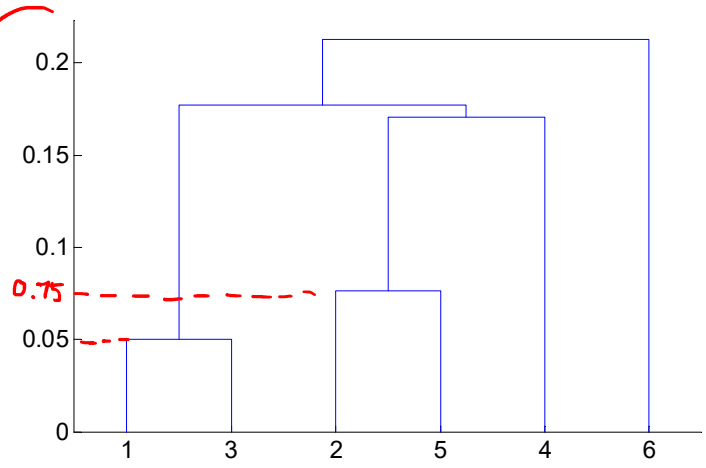
Hierarchical Clustering



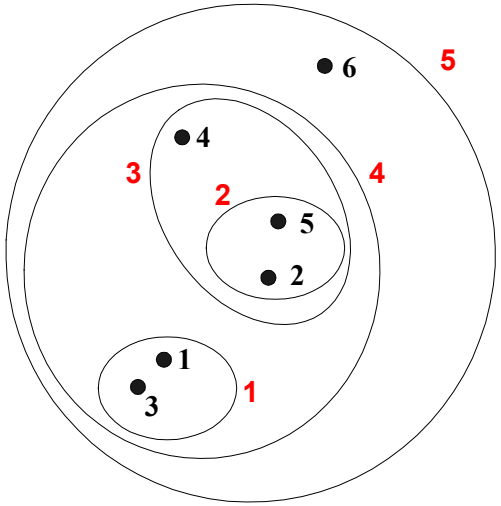
Hierarchical Clustering

- **Dendrogram** displays both the cluster-subcluster relationships and the order in which the clusters were merged (agglomerative) or split (divisive).
- **Nested cluster diagram** can also be graphically represented a hierarchical clustering.

distance of the two clusters



Dendrogram



Nested cluster diagram

Note: The **height** at which two clusters are merged in the dendrogram reflects the distance of the two clusters.

Agglomerative Clustering Algorithm

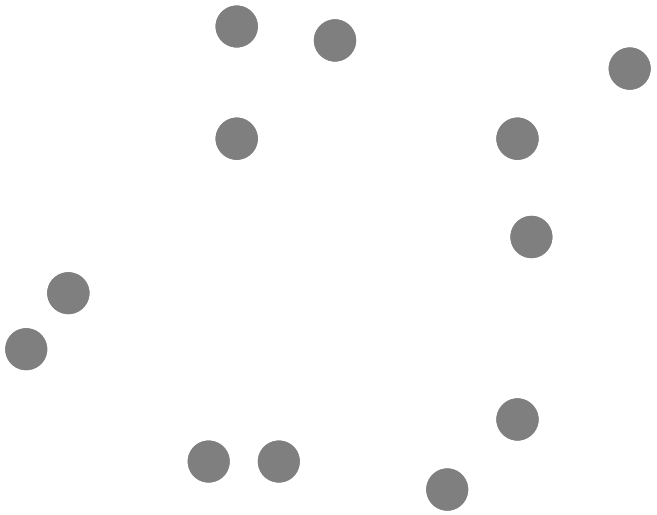
- The basic agglomerative hierarchical clustering algorithm:
  - Step 1: Compute the proximity matrix
  - Step 2: Let each individual object be a cluster
  - Step 3: Merge the two closest clusters
  - Step 4: Update proximity matrix
  - Step 5: Repeat steps 3 – 4 until only one cluster remains

---

1: Compute the proximity matrix
2: Let each data point be a cluster
3: **Repeat**
4: Merge the two closet clusters
5: Update the proximity matrix
6: **Until** only a single cluster remains

Starting Situation

- Start with clusters of individual objects and a proximity matrix

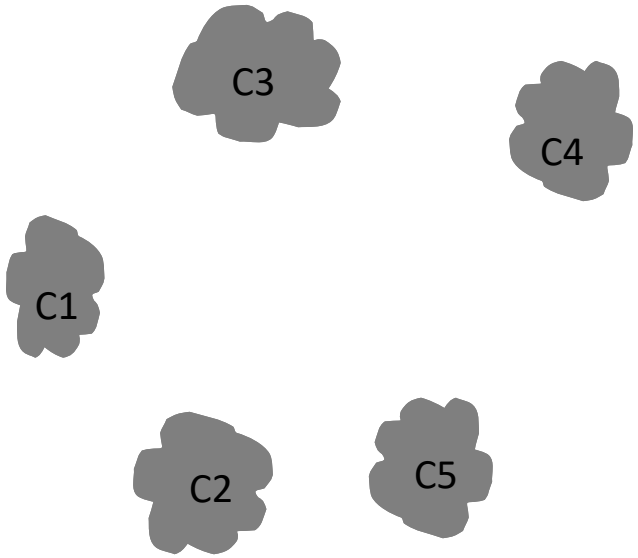


	p1	p2	p3	…	p11	p12
p1						
p2						
p3						
…						
p11						
p12						

Proximity Matrix

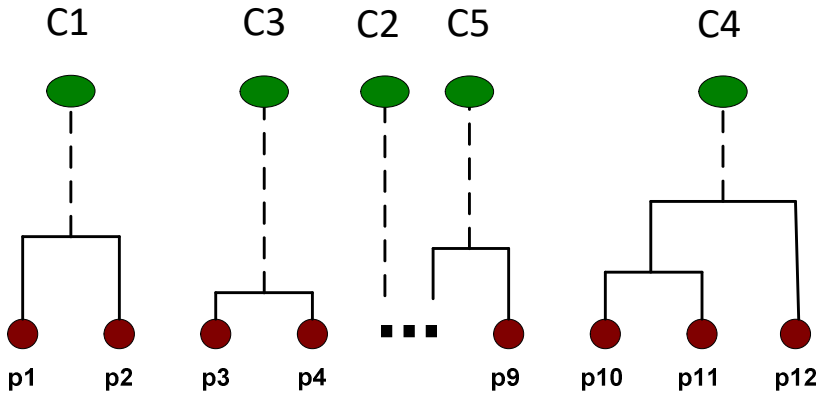
Intermediate Situation

- After some merging steps, we have some clusters

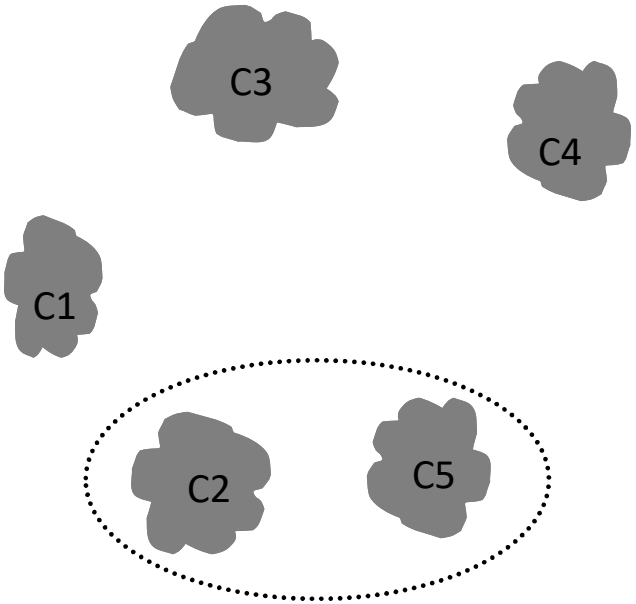


	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

Proximity Matrix

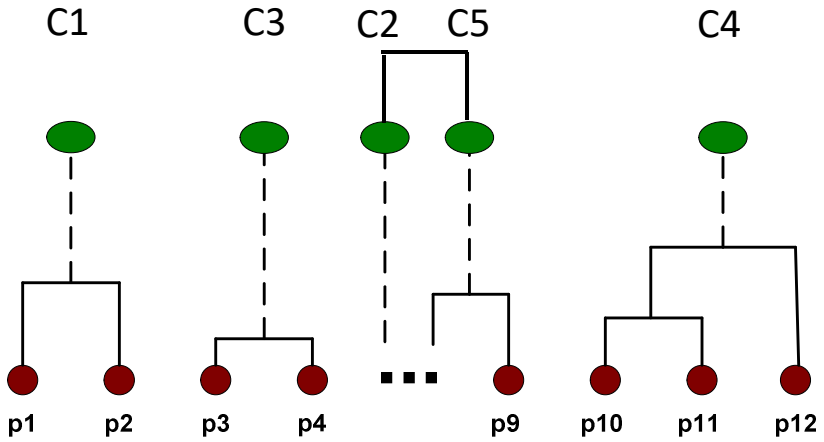


Intermediate Situation



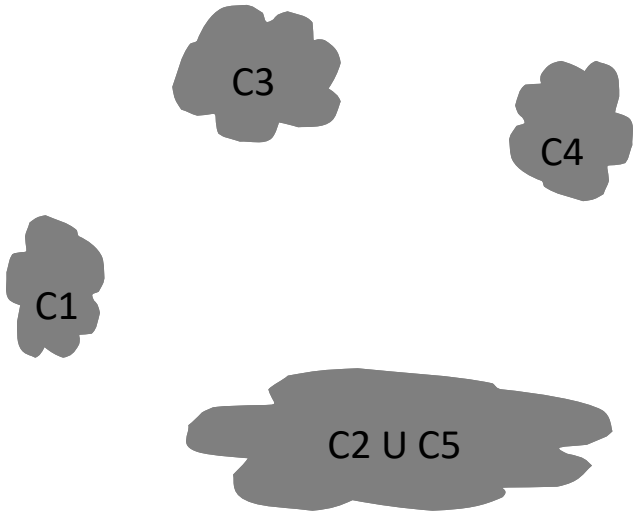
	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

Proximity Matrix



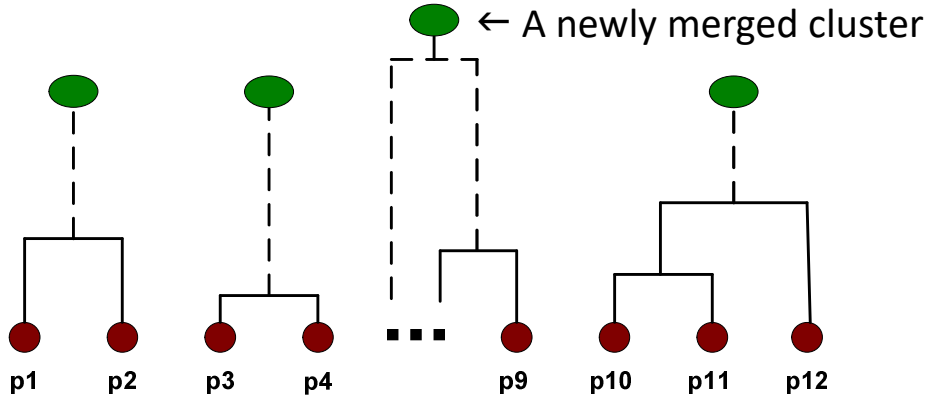
After Merging

- The question is **"How do we update the proximity matrix?"**



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

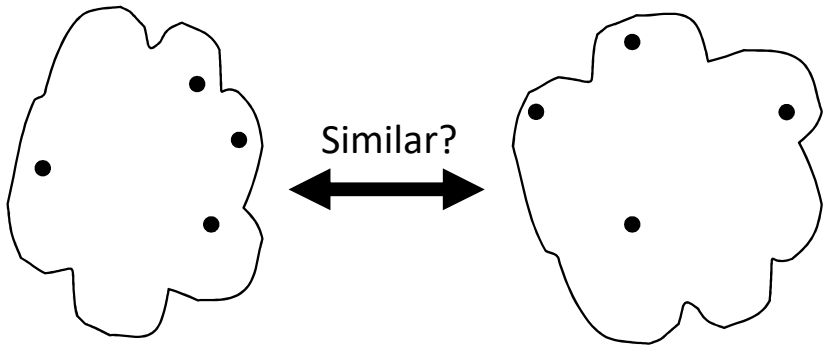
Proximity Matrix



Agglomerative Clustering Algorithm

Defining Proximity between Clusters

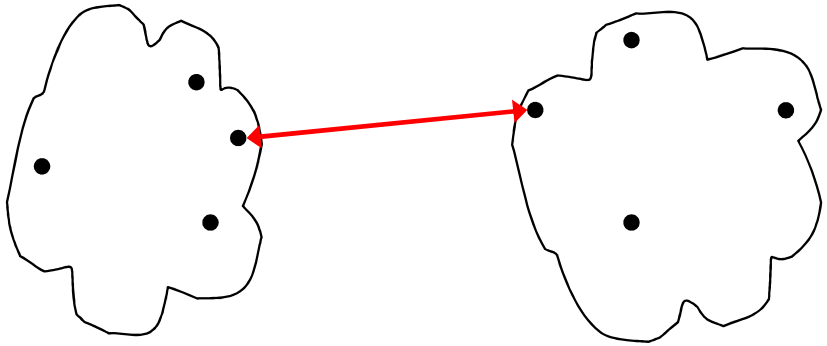
- The key operation of the algorithm is the computation of the proximity between two clusters.
- Cluster proximity is computed to indicate the similarity between two clusters.
- Cluster proximity could be defined using the following approaches:
  1. MIN (Single Link)
  2. MAX (Complete Link)
  3. Group Average
  4. Centroid Methods
  5. Ward's Method



Cluster Proximity

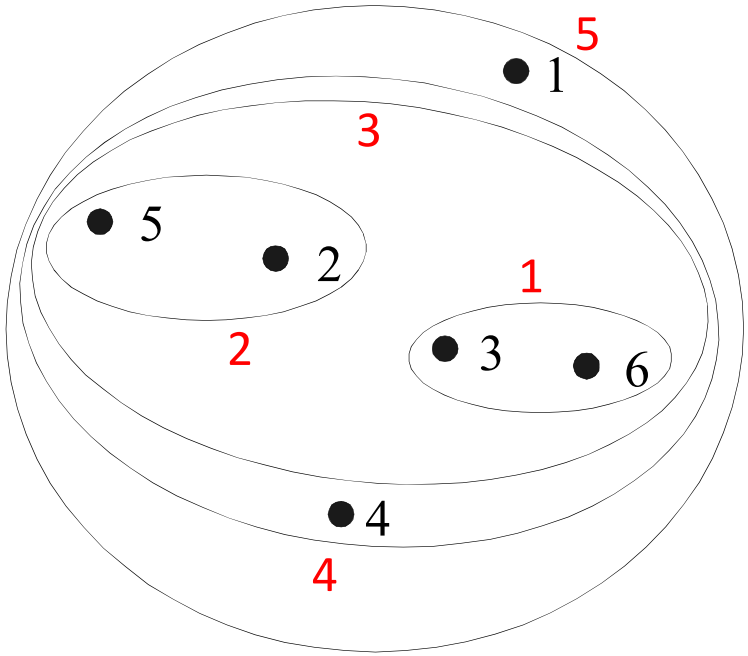
1. MIN or Single Link

- The proximity of two clusters is defined as the minimum of the distance (maximum of the similarity) between any two points in the two different clusters.
- Strength: It can handle non-elliptical shapes.
- Limitation: It sensitive to noise and outliers.

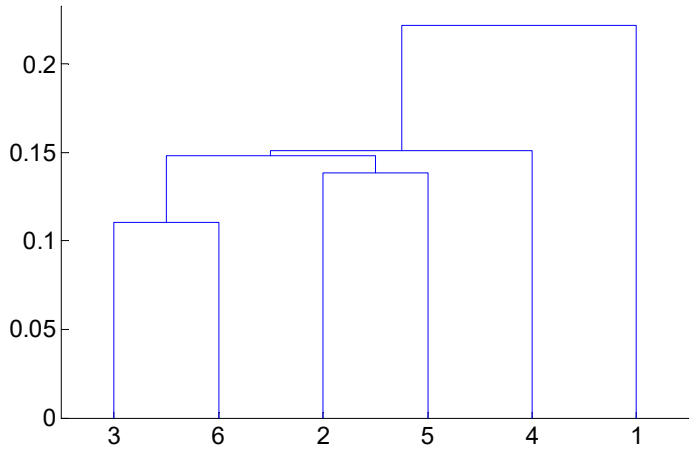


Cluster Proximity

Hierarchical Clustering: MIN (Single Link)



Nested Clusters

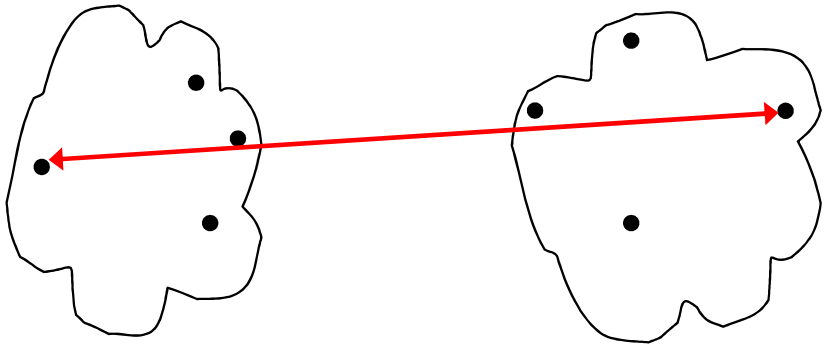


Dendrogram

Cluster Proximity

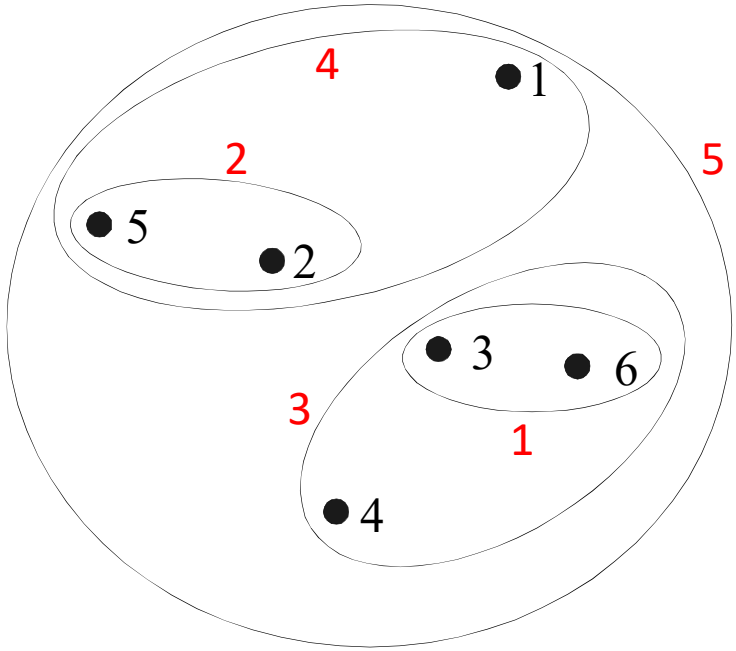
2. MAX or Complete Link or CLIQUE

- The proximity of two clusters is defined as the maximum of the distance (minimum of the similarity) between any two points in the two different clusters.
- Strength: It is less sensitive to noise and outliers.
- Limitations:
  - It can break large clusters.
  - It favors globular shapes.

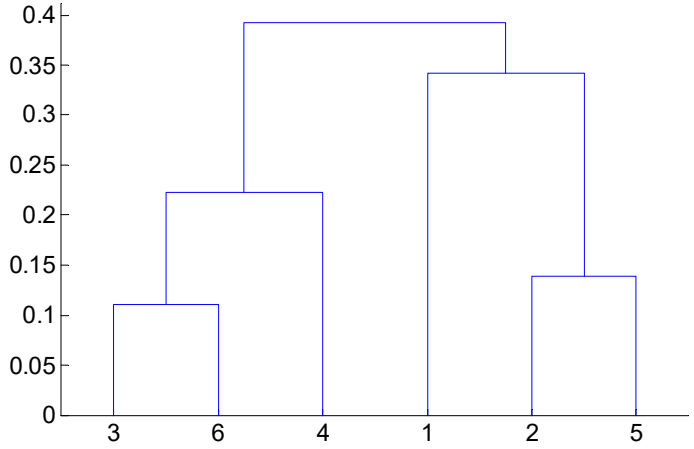


Cluster Proximity

Hierarchical Clustering: MAX (Complete Link)



Nested Clusters



Dendrogram

Cluster Proximity

3. Group Average

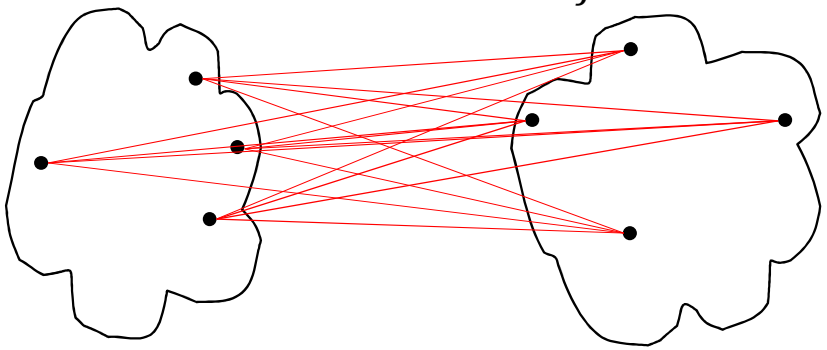
- The proximity of two clusters is defined as the average pairwise proximity among all pairs of points in the two different clusters.

$$proximity(C_i, C_j) = \frac{\sum_{x \in C_i} \sum_{y \in C_j} proximity(x, y)}{m_i m_j}$$

where C_i and C_j are clusters

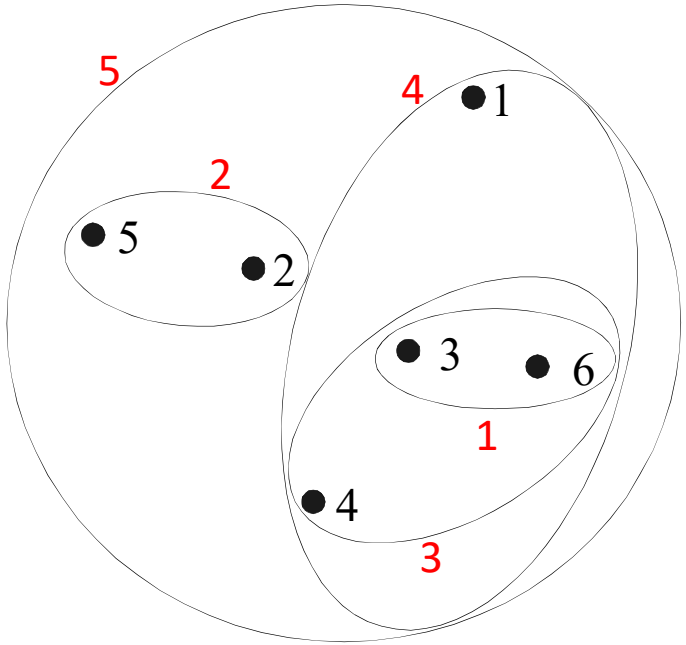
m_i and m_j are numbers of points in clusters C_i and C_j, respectively

x and y are objects in clusters C_i and C_j, respectively

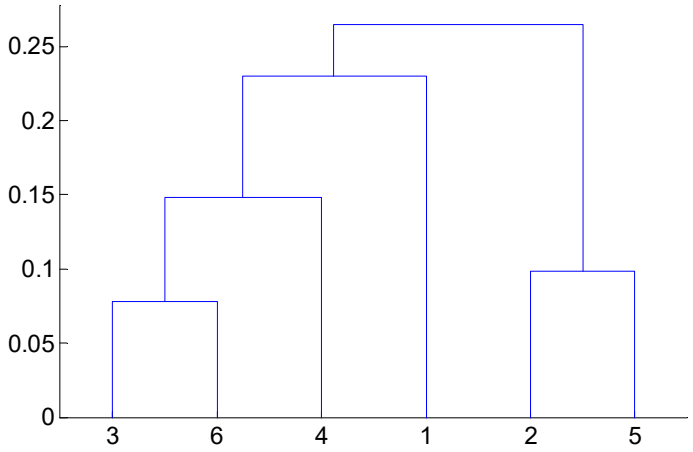


Cluster Proximity

Hierarchical Clustering: Group Average



Nested Clusters

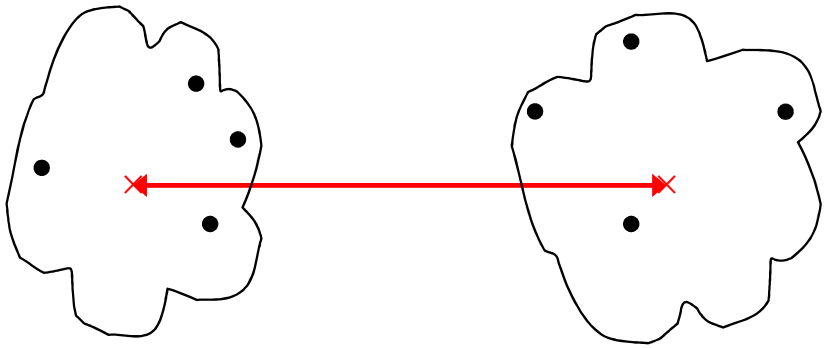


Dendrogram

Cluster Proximity

4. Centroid Methods

- The proximity of two clusters is obtained by calculating the distance between the centroids of clusters.



Cluster Proximity

5. Ward's Method

- The proximity of two clusters is defined as the increase in the squared error that results when two clusters are merged.
- Ward's method is very similar to the group average method when the proximity between two points is taken to be the square of the distance between them.

$$Increase\ in\ SSE = \sum_{x \in C^*} dist(c^*, x)^2 - \left[\sum_{x \in C_1} dist(c_1, x)^2 + \sum_{x \in C_2} dist(c_2, x)^2 \right]$$

where x is an object

C_1, C_2 and C^* are the first and the second clusters before merging and the merged cluster, respectively

c_1, c_2 and c^* are the centroids of cluster C_1, C_2 and C^*, respectively

$dist$ is the standard Euclidean distance between two objects in Euclidean space

Topics

- Introduction
- Types of Clustering
- Type of Clusters
- Basic Clustering Algorithms
  - K-Means Clustering
    - Basic K-means Algorithm
    - Bisecting K-means Algorithm
  - Hierarchical Clustering
  - **DBSCAN Algorithm**

DBSCAN

- **DBSCAN** is a density-based, partial clustering algorithm which works based on the **center-based approach** by locating regions of high density that are separated from regions of low density.
- **Center-based approach** estimates density for a particular point in the data set by counting the number of points within a **specified radius** (**Eps**) of that point which also includes the point itself, thus the density of any point will depend on the specified radius.

  Too small radius → All points will have a density of 1

  Large enough radius → All points will have a density of m
- Two most important characteristics of DBSCAN:

  - The number of cluster is automatically determined by the algorithm.

  - Points in low-density regions are classified as noise and omitted.

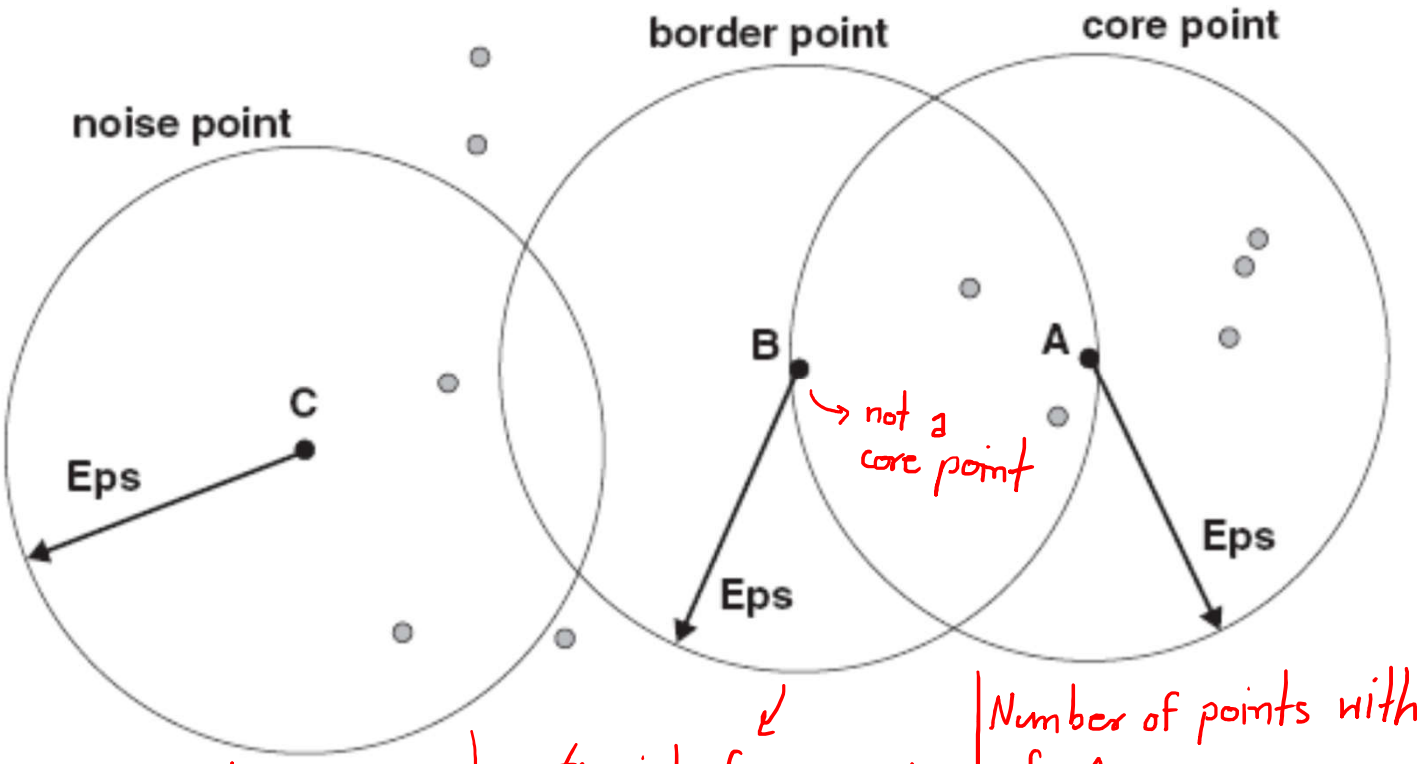
DBSCAN

Classification of Points According to Center-based Density

- There are three types of points in the center-based approach:
  1. **Core points:** These points are in the interior of a density-based cluster. A core point is a point that number of points within a given neighborhood around it exceeds a certain threshold (**MinPts**).
  2. **Border points:** A border point is not a core point, but falls within the neighborhood of a core point.
  3. **Noise points:** A noise point is any point that is neither a core point nor a border point.

DBSCAN

MinPts = 5



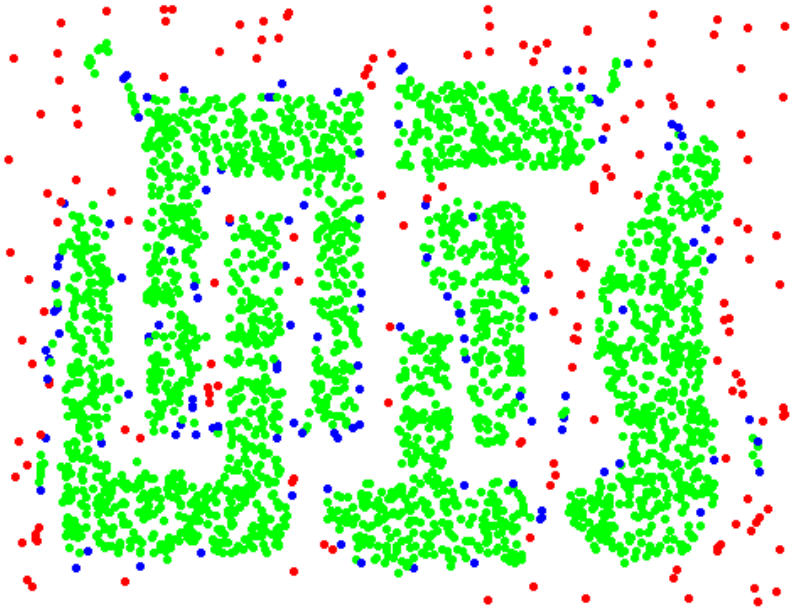
points for C = 3 < 5

points for B = 4 < 5

Number of points within radius Eps of A : 7 > 5 (MinPts)

DBSCAN: Core, Border, Noise Points

Original Points



Point types: core, border and noise

Eps = 10, MinPts = 4

DBSCAN Algorithm

- The DBSCAN algorithm can be informally described as follows:
  1) Any two core points will be put in the same cluster if they are within a distance Eps of one another.
  2) Any border point that is within a distance Eps of a core point will be put in the same cluster as that core point.
  3) Noise points are discarded.
- **DBSCAN algorithm**:

1: Label all points as core, border, or noise points.
2: Eliminate noise points.
3: Put an edge between all core points that are within Eps of each other.
4: Make each group of connected core points into a separate cluster.
5: Assign each border point to one of the clusters of its associated core points.

Selection of DBSCAN Parameters: Eps and $MinPts$

- The basic approach is to look at the behavior of the distance from a point to its k^{th} nearest neighbor called **k-dist**.
- **k-dist** can be done as follows:

  1) Compute the distance of k-dist (the distance from a point to its k^{th} nearest neighbor) for all the data points

  2) Sort them in increasing order

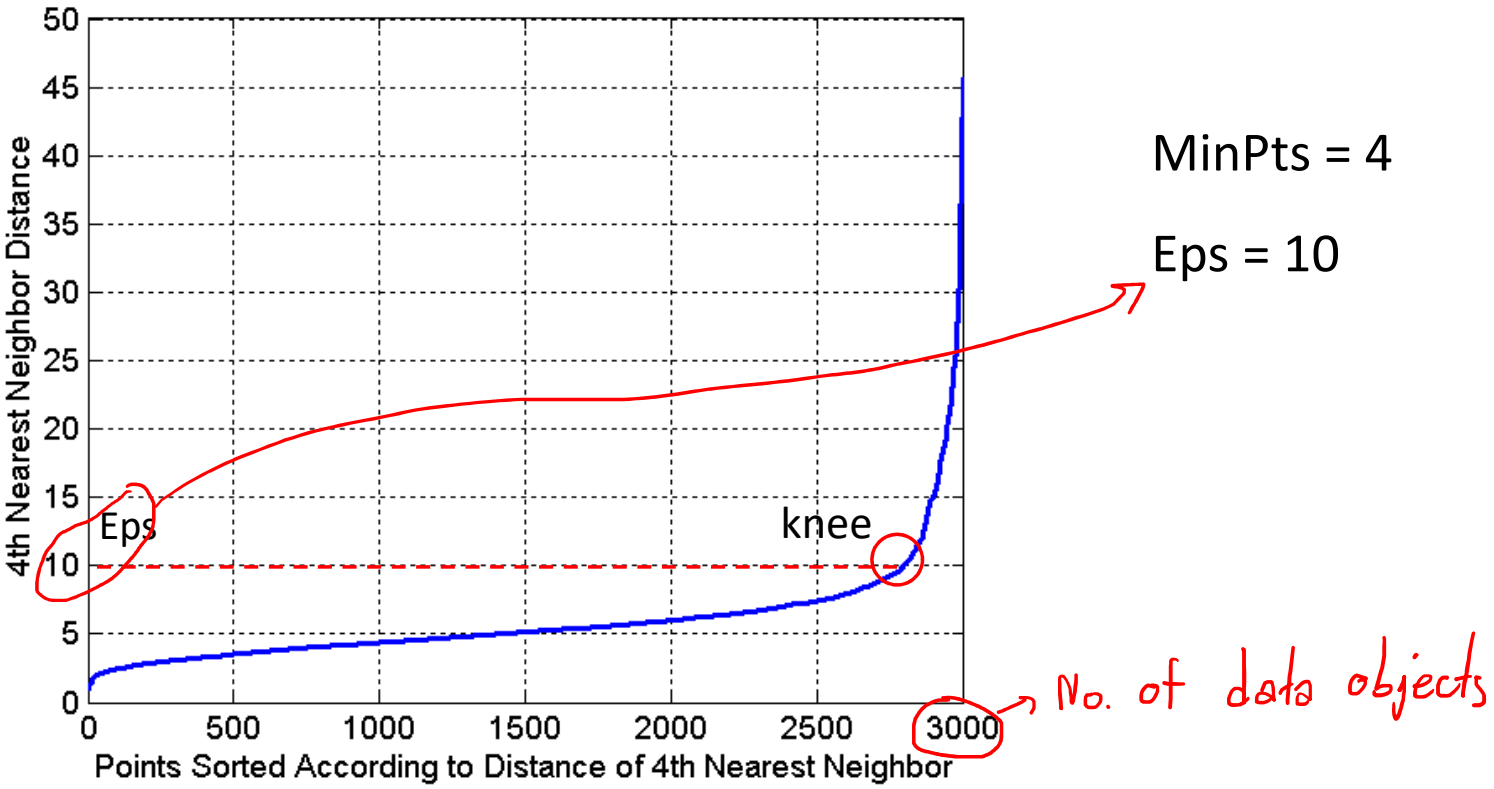
  3) Draw a line connecting the sorted values

  4) Look for a sharp edge (knee)

  5) Set the y value of a knee point found in step 4 as Eps, and set the value k as $MinPts$

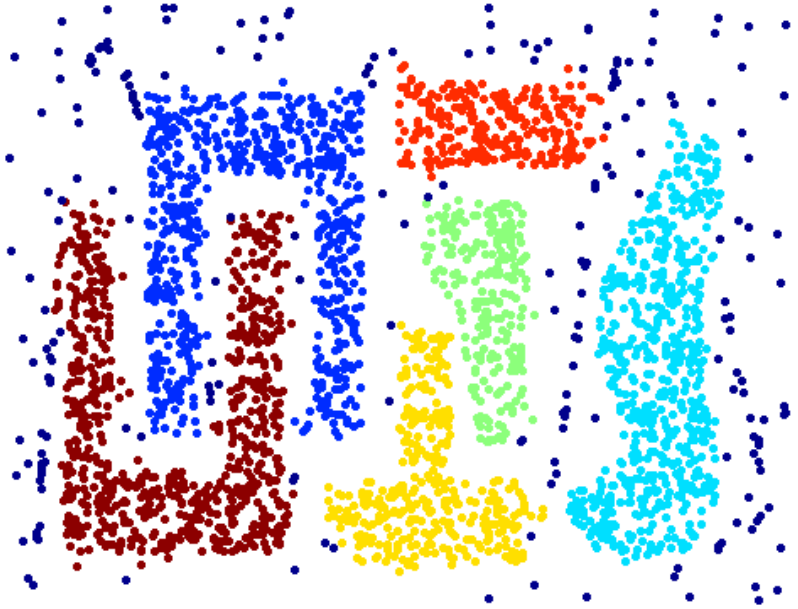
Selection of DBSCAN Parameters

4-dist plot



DBSCAN: Determine Clusters

Original Points



Clusters

Strengths and Weaknesses of DBSCAN

Strengths:

- It is relatively resistant to noise.
- It can handle clusters of arbitrary shapes and sizes.
- It can find many clusters that could not be found using K-means.

Weaknesses:

- It has trouble when the clusters have widely varying densities.
- It has trouble with high-dimensional data because density is more difficult to define for such data.
- It can be expensive when the computation of nearest neighbors requires computing all pairwise proximities.