UNIT-5
Clustering

Syllabus:

Clustering: Problem Definition, Clustering overview, Evaluation of clustering algorithms, Partitioning clustering K-Means Algorithm, K-Means Additional Issues, PAM Algorithm, Hierarchical Clustering-Algorithm- Agglomerative Methods and Divisive Methods, Basic Agglomerative Hierarchical Clustering Algorithm, DBSCAN Algorithm, Strengths and Weaknesses.

Clustering

- A cluster is a collection of data objects that are similar to one another within the same cluster and are dissimilar to the objects in other clusters.
- The process of grouping a set of physical or abstract objects into classes of similar objects is called clustering.
- The objects are clustered or grouped based on the principle of maximizing the intra class similarity and minimizing the interclass similarity.
- Unlike classification and prediction, which analyze class-labeled data objects, clustering analyzes data objects without consulting a known class label.
- In machine learning, clustering is an example of **unsupervised learning.**
- Unlike classification, clustering and unsupervised learning do not rely on predefined classes and class-labeled training examples. For this reason, clustering is a form of **learning by observation**, rather than learning by examples.
- Cluster analysis has been widely used in numerous applications, including market research, pattern recognition, data analysis, and image processing.

Requirements of Clustering in Data Mining:

- ***Scalability*:**
  Many clustering algorithms work well on small data sets. Clustering on only a sample of a given large data set may lead to biased results. Hence we require highly scalable clustering algorithms to work with large databases.
- ***Ability to deal with different kinds of attributes*:**
  Many algorithms are designed to cluster numeric data. However applications may require clustering other data types such as categorical, numerical, and binary data or mixture of these types. Recently more applications require clustering techniques on data types such as graphs, sequences, images and documents.
- ***Discovery of clusters with arbitrary shape*:**
  Many clustering algorithms determine clusters based on Euclidean or Manhattan distance measures. Algorithms based on these measures find spherical clusters with similar size and density. However clusters could be of any shape. So we need algorithms to detect clusters in arbitrary shape and it should not be bounded to distance measures.
- ***Requirements for domain knowledge to determine input parameters*:** Many clustering algorithms require users to provide domain knowledge in the form of input parameters. The clustering results can be quite sensitive to input parameters. Parameters are often difficult to determine, especially for data sets containing high-dimensional objects. This not only burdens users, but it also makes the quality of clustering difficult to control.

- ***Ability to deal with noisy data*:** Most real-world databases contain outliers or missing, unknown, or erroneous data. Some clustering algorithms are sensitive to such data and may lead to clusters of poor quality. Therefore we need clustering methods that are robust to noise.
- ***Incremental clustering and insensitivity to the order of input records*:** In many applications incremental updates may arrive at any time. Some clustering algorithms cannot incorporate incremental updates into existing clustering structures. Clustering algorithms may also be sensitive to the input data order It is important to develop incremental clustering algorithms and algorithms that are insensitive to the order of input
- ***Capability of clustering High- dimensional data* :** A data set can contain numerous dimensions or attributes **Most cl**ustering algorithms are good at handling low-dimensional data such as data sets involving only two or three dimensions. Finding clusters of data objects in a high dimensional space is challenging, especially considering that such data can be very sparse and highly skewed
- ***Constraint-based clustering***: Real-world applications may need to perform clustering under various kinds of constraints. A challenging task is to find data groups with good clustering behavior that satisfy specified constraints
- ***Interpretability and Usability***: Users want clustering results to be interpretable, comprehensible, and usable. That is, clustering may need to be tied in with specific semantic interpretations and applications. It is important to study how an application goal may influence the selection of clustering features and clustering

Categories of Clustering Methods

- Partitioning Methods: **k-means, k-medoids** [PAM(Partitioning Around Medoids)]
- Hierarchical Methods: **AGNES**(AGglomerative NESting), **DIANA**
- (DIvisive ANAlysis) **BIRCH, ROCK**
- Density-Based Methods: **DBSCAN**
- Grid-Based Methods
- Model-Based Methods
- Methods for High-Dimensional Data
- Constraint-Based Methods

***Partitioning Methods*:**

- Given a databases of n objects or data tuples, a partitioning methods constructs k partitions of data, where each partition represents a cluster and $k \leq n$.
- The partitioning methods conduct one-level partitioning on data sets.
- The basic partitioning methods typically adopt **exclusive cluster separation**. That is, each object must belong to exactly one group
- Most partitioning methods are distance-based.Given k, the number of partitions to construct, partitioning methods creates an initial partitioning. It then uses an iterative relocation technique that attempts to improve the partitioning by moving objects from one group to another.
- The general criterion of a good partitioning is that objects in the same cluster are "close" or related to each other, whereas objects of different clusters are "far apart".
- Achieving global optimality in partitioning-based clustering is often computationally prohibitive, so applications adopt heuristic methods such as greedy approaches like K-means and K-Medoids algorithms.

***Hierarchical Methods*:**

- A hierarchical method creates a hierarchical decomposition of the given set of data objects.
- It can be either agglomerative or divisive based on hoe the hierarchical decomposition is formed.
- The agglomerative (bottom-up) approach starts with each object forming a separate group. It successively merges the objects that are close to one another, until all of the groups are merged into one, or until a termination condition holds.
- The divisive (top-down) approach starts with all of the objects in the same cluster. In each successive iteration, a cluster is split up into smaller clusters, until eventually each object forms its own cluster or until a termination condition holds.
- Hierarchical clustering methods can be distance-based or density and continuity based.
- Hierarchical methods suffer from the fact that once a step is done it can never be undone.
- AGNES and DIANA are examples of hierarchical clustering.
- BIRCH integrates hierarchical clustering with iterative (distance-based) relocation.

***Density-Based Methods*:**

- These methods are based on the notion of density.
- The main idea is to continue growing a given cluster as long as the density in its "neighborhood" exceeds some threshold. That is, for each data point within a given cluster, the neighborhood of a given radius has to contain at least a minimum number of points.
- This method can be used to filter out noise and discover clusters of arbitrary shape. DBSCAN and OPTICS are typical examples of density-based clustering.

Grid-Based Methods:

- These methods quantize the object space into a finite number of cells that form a grid structure. All of the clustering operations are performed on the grid structure.
- The main advantage of this approach is its fast processing time, which is typically independent of the number of data objects and dependent only on the number of cells in each dimension in the quantized space.
- STING is an example of grid-based clustering.

Model-Based Methods:

- This approach hypothesizes a model for each of the clusters and finds the best fit of the data to the given model.
- A model-based algorithm may locate clusters by constructing a density function that reflects the spatial distribution of the data points. It also leads to a way of automatically determining the number of clusters based on standard statistics.
- It takes "noise" or outliers into account, therein contributing to the robustness of the approach.
- COBWEB and self organizing feature maps are examples of model-based clustering.

Methods for High-Dimensional Data:

- High-dimensional data can typically have many irrelevant dimensions.
- As the dimensionality increases, the data usually become increasingly sparse because the data points are likely located in different dimensional subspaces.
- The distance measurement between pairs of points becomes meaningless and the average density of points anywhere in the data is likely to be low. Distance- and density-based clustering methods are therefore ineffective for clustering high dimensional data.
- Alternative approaches have been proposed, such as subspace clustering methods, which search for clusters in subspaces (or subsets of dimensions) of the data,

rather than over the entire data space.

- CLIQUE and PROCLUS are examples of subspace clustering methods.
- Frequent pattern-based clustering is another clustering methodology, which extracts distinct frequent patterns among subsets of dimensions that occur frequently. **pCluster** is an example of frequent pattern-based clustering that groups objects based on their pattern similarity.

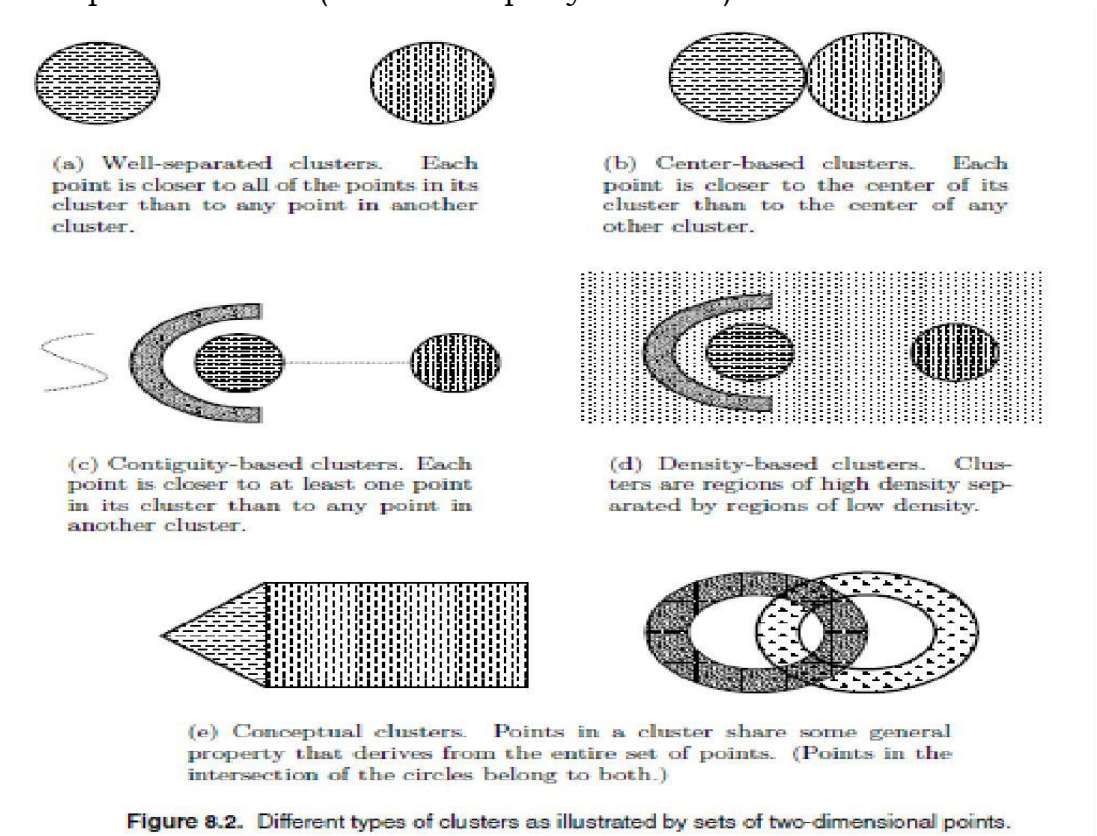
Constraint-Based Methods:

- These perform clustering by incorporating user-specified or application oriented constraints.
- A constraint can express a user's expectation or describe "properties" of the desired clustering results, and provides an effective means for communicating with the clustering process.
- Constraint-based methods are used in spatial clustering for clustering with obstacle objects (e.g., considering obstacles such as rivers and highways when planning the placement of automated banking machines) and user-constrained cluster analysis (e.g, considering specific constraints regarding customer groups when determining the best location for a new service station, such as "must serve at least 100 high-value customers").
- In addition, semi-supervised clustering employs, for example, pair wise constraints (such as pairs of instances labeled as belonging to the same or different clusters) in order to improve the quality of the resulting clustering.

Different Types of Clusters

Different notions of a cluster are:

- Well-Separated Clusters
- Center-Based Clusters (Prototype-Based Clusters)
- Contiguity-Based Clusters(Graph-Based Clusters)
- Density-Based Clusters
- Conceptual Clusters (Shared-Property Clusters)



(a) Well-separated clusters. Each point is closer to all of the points in its cluster than to any point in another cluster.

(b) Center-based clusters. Each point is closer to the center of its cluster than to the center of any other cluster.

(c) Contiguity-based clusters. Each point is closer to at least one point in its cluster than to any point in another cluster.

(d) Density-based clusters. Clusters are regions of high density separated by regions of low density.

(e) Conceptual clusters. Points in a cluster share some general property that derives from the entire set of points. (Points in the intersection of the circles belong to both.)

Figure 8.2. Different types of clusters as illustrated by sets of two-dimensional points.

***Well-Separated Cluster*:**

- 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
- Figure 8.2(a) gives an example of well separated clusters that consists of two groups of points in a two-dimensional the distance between any two points within a group.
- Well-separated clusters do not need to be globular, but can have any shape.

***Prototype-Based Clusters*:**

- 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 often a centroid, i.e., the average (mean) of all the points in the cluster.
- When a centroid is not meaningful, such as when the data has categorical attributes, the prototype is often a medoid, i.e., the most representative point of a cluster.
- For many types of data, the prototype can be regarded as the most central point, and in such instances, we commonly refer to prototype-based clusters as **center-based clusters**. Not surprisingly, such clusters tend to be globular.
- Figure 8.2(b) shows an example of center-based clusters.

Graph-Based Clusters:

- If the data is represented as a graph, where the nodes are objects and the links represent connections among objects, then a cluster can be defined as a **connected component**; i.e., a group of objects that are connected to one another, but that have no connection to objects outside the group.
- An important example of graph-based clusters is **contiguity-based clusters**, where two objects are connected only if they are within a specified distance of each other. This implies that each object in a contiguity-based cluster is closer to some other object in the cluster than to any point in a different cluster. Figure 8.2(c) shows an example of such clusters for two-dimensional points.
- Figure 8.2(c), shows an example of contiguity-based cluster where a small bridge of points can merge two distinct clusters.
- Other types of graph-based clusters are also possible. One such approach defines a cluster as a **clique**; i.e., a set of nodes in a graph that are completely connected to each other.
- Like prototype-based clusters, these clusters tend to be globular.

***Density-Based Cluster*:**

- A cluster is a dense region of objects that is surrounded by a region of low density.
- Figure 8.2(d) shows some density-based clusters for data created by adding noise to the data of Figure 8.2(c).
- A density-based definition of a cluster is often employed when the clusters are irregular or intertwined, and when noise and outliers are present. By contrast, a contiguity-based definition of a cluster would not work well for the data of Figure 8.2(d) since the noise would tend to formbridges between clusters.

Shared-Property (Conceptual Clusters) *:*

- A Cluster is a set of objects that share some property.
- Figure 8.2(e) shows a conceptual cluster where a triangular area (cluster) is adjacent to a rectangular one, and there are two intertwined circles (clusters).

Partitioning clustering

K-means

- It is Centroid-Based Technique
- The k-means algorithm takes the input parameter, k, and partitions a set of n objects into k clusters so that the resulting intra cluster similarity is high but the inter cluster similarity is low.
- Cluster similarity is measured in regard to the mean value of the objects in a cluster, which can be viewed as the cluster's centroid or center of gravity.

Working of the k-means algorithm:

- First, it randomly selects k of the objects, each of which initially represents a cluster mean or center.
- For each of the remaining objects, an object is assigned to the cluster to which it is the most similar, based on the distance between the object and the cluster mean.
- It then computes the new mean for each cluster.
- This process iterates until the criterion function converges. Typically, the square-

$$E = \sum_{i=1}^{k} \sum_{p \in C_i} |p - m_i|^2,$$

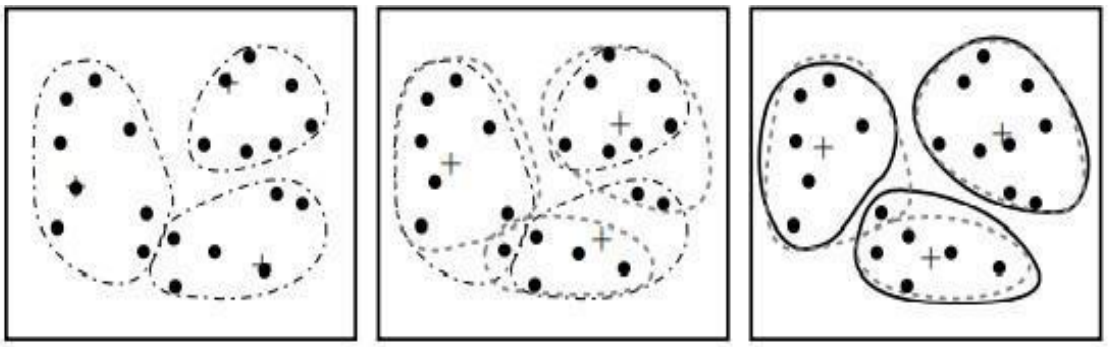
error criterion is used, defined as

  - where E is the sum of the square error(SSE) for all objects in the data setp is the point in space representing a given object
    - mi is the mean of cluster Ci
  - In other words, for each object in each cluster, the distance from theobject to its cluster center is squared, and the distances are summed.

K-Means Algorithm:

(1) arbitrarily choose k objects from D as the initial cluster centers;
(2) repeat
(3) (re) assign each object to the cluster to which the object is the mostsimilar, based on the mean value of the objects in the cluster;
(4) update the cluster means, i.e., calculate the mean value of the objects foreach cluster;
(5) **until** no change;

Example 1:



Clustering of a set of objects based on the *k*-means method. (The mean of each cluster is marked by a "+".)

Example 2: Given are the 1-dimensional points A = 2, B = 3, C = 4, D = 10, E = 11, F = 12, G=20, H=25 and I=30. Suppose initially we assign C and F as the centroid of each cluster, respectively. Use the k-means algorithm to find two clusters.

Solution:

Iteration 1:

Initially centroids are C_1=C and C_2=F

For A: Distance from centroid C_1 is 4-2=2

Distance from centroid C_2 is 12-2=10

This point is nearest to C_1 and assign to K_1

For B: Distance from centroid C_1 is 4-3=1

Distance from centroid C_2 is 12-4=8

This point is nearest to C_1 and assign to K_1

For D: Distance from centroid C_1 is 10-4=6

Distance from centroid C_2 is 12-10=2

This point is nearest to C_2 and assign to K_2

For E: Distance from centroid C_1 is 11-4=7

Distance from centroid C_2 is 12-11=1

This point is nearest to C_2 and assign to K_2

For G: Distance from centroid C_1 is 20-4=16

Distance from centroid C_2 is 20-12=8

This point is nearest to C_2 and assign to K_2

For H: Distance from centroid C_1 is 25-4=21

Distance from centroid C_2 is 25-12=13

This point is nearest to C_2 and assign to K_2

For I: Distance from centroid C_1 is 30-4=26

Distance from centroid C_2 is 30-12=18

This point is nearest to C_2 and assign to K_2

Therefore after fist iteration K_1 = {A, B, C} and K_2 = {D, E, F, G, H, I}

Iteration 2:

New Centroids are:

C1 = (2 + 3+ 4)/3 = 9 / 3 = 3 and C2 = (10 +11 +12 + 20 + 25 + 30)/6 =108 / 6 =18

For A: Distance from centroid C_1 is 3-2=1
Distance from centroid C_2 is 18-2=16
This point is nearest to C_1 and assign to K_1

For B: Distance from centroid C1 is 3-3=0
Distance from centroid C_2 is 18-3=15
This point is nearest to C_1 and assign to K_1

For C: Distance from centroid C_1 is 4-3=1
Distance from centroid C_2 is 18- 4=14

This point is nearest to C_1 and assign to K_1

For D: Distance from centroid C_1 is 10-3=7

Distance from centroid C_2 is 18-10=8
This point is nearest to C_1 and assign to K_1

For E: Distance from centroid C_1 is 11-3=8
Distance from centroid C_2 is 18-11=7
This point is nearest to C_2 and assign to K_2

For F: Distance from centroid C_1 is 12-3=8
Distance from centroid C_2 is 18-12=6
This point is nearest to C_2 and assign to K_2

For G: Distance from centroid C_1 is 20-3=17
Distance from centroid C_2 is 20-18=2
This point is nearest to C_2 and assign to K_2

For H: Distance from centroid C_1 is 25-3=22
Distance from centroid C_2 is 25-18=7
This point is nearest to C_2 and assign to K_2

For I: Distance from centroid C_1 is 30-3=27
Distance from centroid C_2 is 30-18=12
This point is nearest to C_2 and assign to K_2

Therefore after second iteration
K_1 = {A, B, C, D} and K_2 = {E, F, G, H, I}

Iteration 3: New Centroids are:

C_1 =(2+3+4+10)/4=19/4=4.75 C2=(11+12+20+25+30)/5=98/5=19.6

For A: Distance from centroid C_1 is 4.75-2=2.75
Distance from centroid C_2 is 19.6-2=17.6
This point is nearest to C_1 and assign to K_1

For B: Distance from centroid C_1 is 4.75-3=1.75
Distance from centroid C_2 is 19.6-3=16.6
This point is nearest to C_1 and assign to K_1

For C: Distance from centroid C_1 is 4.75-4=0.75
Distance from centroid C_2 is 19.6-4=15.6
This point is nearest to C_1 and assign to K_1

For D: Distance from centroid C_1 is 10-4.75=5.25
Distance from centroid C_2 is 19.6-10=9.6
This point is nearest to C_1 and assign to K_1

For E: Distance from centroid C_1 is 11-4.75=6.25
Distance from centroid C_2 is 19.6-11=8.6
This point is nearest to C_1 and assign to K_1

For F: Distance from centroid C_1 is 12-4.75=7.25
Distance from centroid C_2 is 19.6-12=7.6
This point is nearest to C_1 and assign to K_1

For G: Distance from centroid C_1 is 20-4.75=15.25
Distance from centroid C_2 is 20-19.6=0.4
This point is nearest to C_2 and assign to K_2

For H: Distance from centroid C_1 is 25-4.75=20.25
Distance from centroid C_2 is 25-19.6=5.4
This point is nearest to C_2 and assign to K_2

For I: Distance from centroid C_1 is 30-4.75=25.25
Distance from centroid C_2 is 19.6-18=1.6
This point is nearest to C_2 and assign to K_2

Therefore after third iteration:

K_1 = {A, B, C, D, E, F} and K_2 = {G, H, I}

Iteration 4: New Centroids are:

C_1 = (+3+4+10+11+12)/6=42/6=7 C2=(20+25+30)/3=75/3=25

For A: Distance from centroid C_1 is 7-2=5
Distance from centroid C_2 is 25-2=23
This point is nearest to C_1 and assign to K_1

For B: Distance from centroid C_1 is 7-3=4
Distance from centroid C_2 is 25-3=22
This point is nearest to C_1 and assign to K_1

For C: Distance from centroid C_1 is 7-4=3
Distance from centroid C_2 is 25-4=21
This point is nearest to C_1 and assign to K_1

For D: Distance from centroid C_1 is 10-7=3
Distance from centroid C_2 is 25-10=15

This point is nearest to C_1 and assign to K_1

For E: Distance from centroid C_1 is 11-7=4
Distance from centroid C_2 is 25-11=14
This point is nearest to C_1 and assign to K_1

For F: Distance from centroid C_1 is 12-7=5
Distance from centroid C_2 is 25-12=13
This point is nearest to C_1 and assign to K_1

For G: Distance from centroid C_1 is 20-7=13
Distance from centroid C_2 is 25-20=5
This point is nearest to C_2 and assign to K_2

For H: Distance from centroid C_1 is 25-7=18
Distance from centroid C_2 is 25-25=0
This point is nearest to C_2 and assign to K_2

For I: Distance from centroid C_1 is 30-7=23
Distance from centroid C_2 is 30-25=5
This point is nearest to C_2 and assign to K_2

Therefore after forth iteration:

K_1 = {A, B, C, D, E, F} and K_2 = {G, H, I}

Iteration 5:

C1=(2+3+4+10+11+12)/6=42/6=7 C2=(20+25+30)/3=75/3=25

Thus we are getting same centroid, we have to stop

Therefore final clusters are:

K_1 = {A, B, C, D, E, F} and K_2 = {G, H, I}

Strengths of k-means:

- K-means is simple and can be used for a wide variety of data types.
- It is also quite efficient, even though multiple runs are often performed.

Weaknesses of k-means:

- K-means is not suitable for all types of data.
- It cannot handle non-globular clusters or clusters of different sizes and densities.
- K-means also has trouble clustering data that contains noise & outliers.
- K-means is restricted to data for which there is a notion of a center (centroid).

K-means Additional Issues

Issues have to consider while k-means algorithm are:

- Handling Empty Clusters
- Handling Outliers
- Reducing the SSE with Post processing
- Updating Centroids Incrementally

Handling Empty Clusters

- One of the problems with the basic K-means algorithm given earlier is that empty clusters can be obtained if no points are allocated to a cluster during the assignment step. If this happens, then a strategy is needed to choose a replacement centroid, since otherwise, the squared error will be larger than necessary.
- One approach is to choose the point that is farthest away from any current centroid. If nothing else, this eliminates the point that currently contributes most to the total squared error.
- Another approach is to choose the replacement centroid from the cluster that has the highest SSE. This will typically split the cluster and reduce theoverall SSE of the clustering. If there are several empty clusters, then this process can be repeated several times.

Handling Outliers

- When the squared error criterion is used, outliers can unduly influence the clusters that are found.
- In particular, when outliers are present, the resulting cluster centroids (prototypes) may not be as representative as they otherwise would be and thus, the SSE will be higher as well. Because of this, it is often useful to discover outliers and eliminate them beforehand.
- Alternatively, outliers can also be identified in a post processing step..

Reducing the SSE with Post processing

An obvious way to reduce the SSE is to find more clusters, i.e., to use a larger *K*.

Two strategies that decrease the total SSE by increasing the number of clusters are the following:

- **Split a cluster:** The cluster with the largest SSE is usually chosen, but we could also split the cluster with the largest standard deviation for one particular attribute.
- **Introduce a new cluster centroid:** Often the point that is farthest from any cluster center is chosen. We can easily determine this ifwe keep track of the SSE contributed by each point. Another approach is to choose randomly from all points or from the points with the highest SSE.

Two strategies that decrease the number of clusters, while trying to minimize the increase in total SSE, are the following:

- **Disperse a cluster:** This is accomplished by removing the centroid that corresponds to the cluster and reassigning the points to other clusters. Ideally, the cluster that is dispersed should be the one that increases the total SSE the least.
- **Merge two clusters:** The clusters with the closest centroids are typically chosen, although another, perhaps better, approach is to merge the two clusters that result in the smallest increase in total SSE.

Updating Centroids Incrementally

- Instead of updating cluster centroids after all points have been assigned to a cluster, the centroids can be updated incrementally, after each assignment of a point to a cluster.
- Notice that this requires either zero or two updates to cluster centroids at each step, since a point either moves to a new cluster

(two updates) or stays in its current cluster (zero updates).

- Using an incremental update strategy guarantees that empty clusters arenot produced since all clusters start with a single point, and if a cluster ever has only one point, then that point will always be reassigned to the same cluster.
- On the negative side, updating centroids incrementally introduces an order dependency. In other words, the clusters produced may depend on the order in which the points are processed. Although this can be addressed by randomizing the order in which the points are processed**,** the basic K-means approach of updating the centroids after all points have been assigned to clusters has no order dependency.
- Also, incremental updates are slightly more expensive. However, K-means converges rather quickly, and therefore, the number of points switching clusters quickly becomes relatively small.

Time and Space Complexity of K-means

- *The space requirements for K-means* are modest because only the data points and centroids are stored. Specifically, the storage required is
  - ***O*((*m* + *K*)*n*),**where *m* is the number of points and *n* is the number of attributes.
- The time requirements for K-means are also modest—basically linear in the number of data points. In particular, the time required is ***O*(*I* ∗*K*∗*m*∗*n*),** wherem = number of points, K = number of clusters, I = number of iterations, n = number of attributes

K-Medoids or (PAM: Partitioning Around Medoids)

- It is Representative Object-Based Technique.
- Instead of taking the mean value of the objects in a cluster as a reference point, we can pick actual objects to represent the clusters, using one representative objectper cluster.
- Each remaining object is clustered with the representative object to which it is the most similar.
- The partitioning method is then performed based on the principle of minimizing the sum of the dissimilarities between each object and its corresponding reference point.i.e., an absolute-error criterion is used, defined as

$$E = \sum_{j=1}^{k} \sum_{p \in C_j} |p - o_j|,$$

K-Medoid Algorithm

(1) arbitrarily choose k objects in D as the initial representative objects or seeds;
(2) repeat
(3) assign each remaining object to the cluster with the nearest representative object;
(4) randomly select a non representative object, Orandom;
(5) compute the total cost, S, of swapping representative object, oj , with Orandom;
(6) if S < 0 then swap oj with Orandom to form the new set of k representativeobjects;
(7) **until** no change;

Advantages of k-Medoids:

- It is simple to understand and easy to implement.
- K-Medoid Algorithm is fast and converges in a fixed number of steps.
- PAM is less sensitive to outliers than other partitioning algorithms.

Disadvantages of k-Medoids:

- The main disadvantage of K-Medoid algorithms is that it is not suitable for clustering non-spherical (arbitrary shaped) groups of objects. This is because it relies on minimizing the distances between the non-medoid objects and the medoid (the cluster centre) – briefly, it uses compactness as clustering criteria instead of connectivity.
- It may obtain different results for different runs on the same dataset because the first k medoids are chosen randomly.

K-Means vs k-Medoids

k-Means	k-Medoids
1.It is centroid-based technique	It is representative object based technique
2.a centroid almost never corresponds to an actual data	Chooses actual data points as centers
3. K-means defines a prototype in terms of a centroid, which is usually the mean of a group of points.	K-medoid defines a prototype in terms of a medoid, which is the most representative point for a group of points.
4.attempts to minimize sum of square error(SSE)	attempts to minimize sum of absolute error(SAE)
5.It is more efficient than k-Medoids	It is more robust than k-Means
6.sensitive to outliers	Less influenced by outliers
7.Time complexity: O(nkt), where n is the total number of objects, k is the number of clusters, and t is the number of iterations	$O(k(n-k)^2)$

Hierarchical Clustering Methods

- A hierarchical clustering method works by grouping data objects into a tree of clusters. i.e, creates a hierarchical decomposition of the given set of data objects.
- In general, there are two types of hierarchical clustering methods:

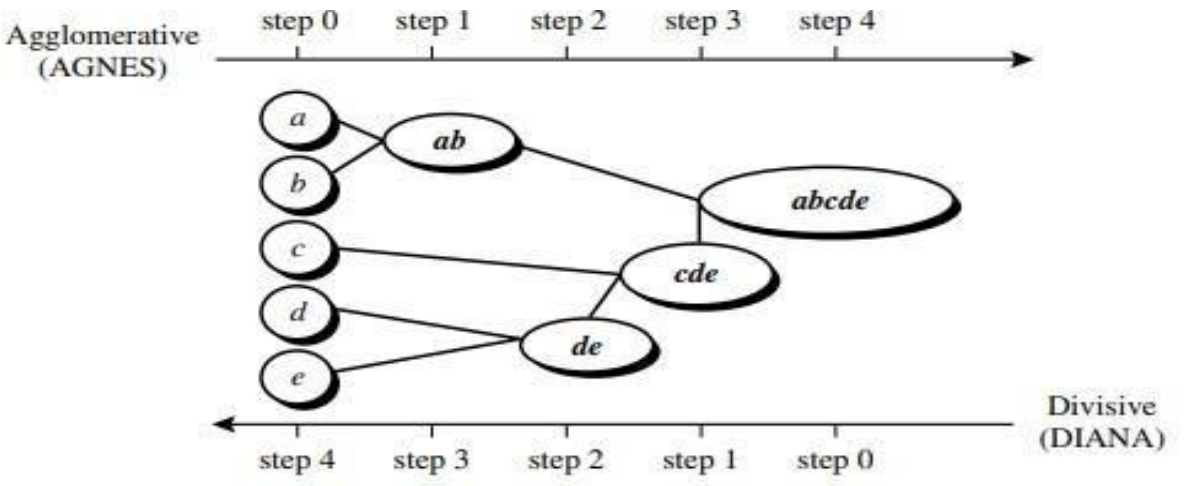
Agglomerative Hierarchical Clustering:

- This bottom-up strategy starts by placing each object in its own cluster and then merges these atomic clusters into larger and larger clusters, until all of the objects are in a single cluster or until certain termination conditions are satisfied.
- Most hierarchical clustering methods belong to this category.
- They differ only in their definition of inter cluster similarity.

Divisive Hierarchical Clustering:

- This top-down strategy does the reverse of agglomerative hierarchical clusteringby starting with all objects in one cluster.
- It subdivides the cluster into smaller and smaller pieces, until each object forms a cluster on its own or until it satisfies certain termination conditions, such as a desired number of clusters is obtained or the diameter of each cluster is within a certain threshold.

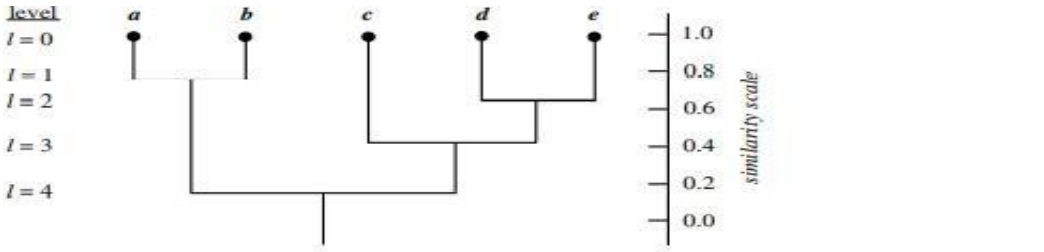
Example:



Agglomerative and divisive hierarchical clustering on data objects $\{a, b, c, d, e\}$.

Dendrogram

- A tree structure commonly used to represent the process of hierarchical clustering.
- It shows how objects are grouped together step by step.
- We can also use a vertical axis to show the similarity scale between clusters.



Dendrogram representation for hierarchical clustering of data objects $\{a, b, c, d, e\}$.

Basic Agglomerative Hierarchical Clustering Algorithm

- Many agglomerative hierarchical clustering techniques are variations on a single approach: starting with individual points as clusters, successively merge the two closest clusters until only one cluster remains.
- This approach is expressed more formally in following algorithm.

Algorithm 8.3 Basic agglomerative hierarchical clustering algorithm.

1: Compute the proximity matrix, if necessary.
2: **repeat**
3: Merge the closest two clusters.
4: Update the proximity matrix to reflect the proximity between the new cluster and the original clusters.
5: **until** Only one cluster remains.

Defining Proximity between Clusters (Linkage Metrics)

(Common Agglomerative Hierarchical Clustering Techniques)

- ❖ Single Link or MIN
- ❖ Complete Link or MAX or CLIQUE
- ❖ Average Link or Group Average

Single Link or MIN

- **MIN** defines cluster proximity as the proximity between the **closest** two points that are in different clusters.
- For the single link or MIN version of hierarchical clustering, 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.
- Using graph terms, the shortest edge between two nodes in different subsets of nodes.
- The single link technique is good at handling non-elliptical shapes, but is sensitive to noise and outliers.

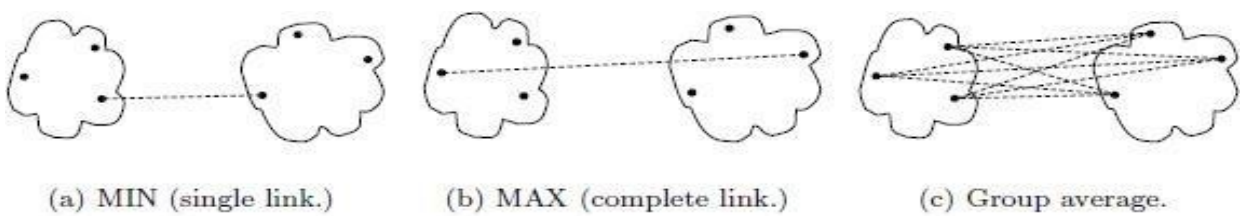
Complete Link or MAX or CLIQUE

- MAX takes the proximity between the farthest two points in different clusters to be the cluster proximity.
- For the complete link or MAX version of hierarchical clustering, 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.
- Using graph terms, the longest edge between two nodes in different subsets of nodes.
- Complete link is less susceptible to noise and outliers, but it can break large clusters and it favors globular shapes.

Group average

- This technique, defines cluster proximity to be the average pairwise proximities (average length of edges) of all pairs of points from differentclusters.
- This is an intermediate approach between the single and complete linkapproaches.

Following figures illustrates these three approaches.



(a) MIN (single link.) (b) MAX (complete link.) (c) Group average.

Figure 8.14. Graph-based definitions of cluster proximity

DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

- DBSCAN is a density-based clustering algorithm that produces a partitional clustering, in which the number of clusters is automatically determined by the algorithm.
- Density-based clustering locates regions of high density that are separated from one another by regions of low density.
- In the center-based approach, density is estimated for a particular point in the data set by counting the number of points within a specified radius, Eps, of that point.

DBSCAN Parameters:

- Eps: Maximum radius of neighbourhood around a particular point (User specifieddistance parameter)
- MinPts: A user-specified threshold that indicate minimum number of neighbourswithin Eps radius
- Eps-Neighborhood(p) ={q/dist(p,q) ≤ Eps}

Example:

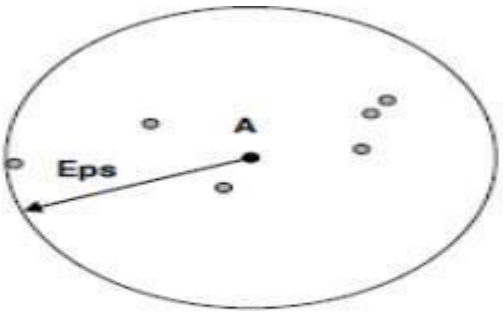


Figure 8.20. Center-based density.

In above figure the number of points within a radius of Eps of point A is 7, including Aitself. i.e., Eps-Neighborhood(A) = 7

Density = Number of points within a specified radius Eps

Classification of points

1. ***Core Point*:**
   - 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.
2. ***Border Point*:**
   - A border point has fewer than MinPts within Eps, but is in the neighbor hood of acore point.
   - These points are on the edge of a dense region
3. ***Noise Point:***
   - A noise point is any point that is neither a core point nor a border point.
   - These points are in a sparsely occupied region.

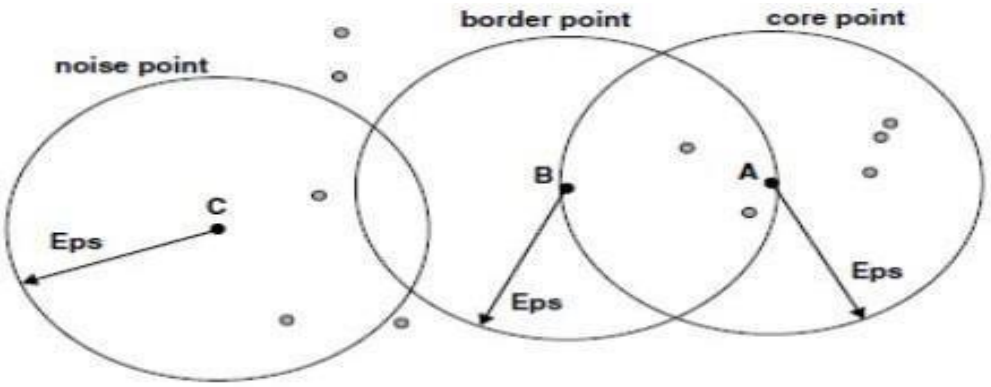


Figure 8.21. Core, border, and noise points.

In above Figure:

Point A is a core point,for the indicated radius (Eps) if MinPts ≤ 7

Point B is a border point.

Point C is a noise point.

The DBSCAN Algorithm:

- Given the core points, border points, and noise points, the DBSCAN algorithmcan be informally described as follows.
  - Any two core points that are close enough—within a distance Eps of oneanother—are put in the same cluster.
  - Likewise, any border point that is close enough to a core point is put in thesame cluster as the core point.
  - Noise points are discarded.

- The formal details are given in following algorithm:

Algorithm 8.4 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.

Example: If Eps is 2 and MinPts is 2, what are the clusters that DBSCAN would discover with the following 8 data points: A1=(2,10), A2=(2,5), A3=(8,4), A4=(5,8), A5=(7,5), A6=(6,4), A7=(1,2), A8=(4,9).The distance matrix is given below. Draw the 10 by 10 space and illustrate the discovered clusters. What if Eps is increased to $\sqrt{10}$?

	A1	A2	A3	A4	A5	A6	A7	A8
A1	0	$\sqrt{25}$	$\sqrt{36}$	$\sqrt{13}$	$\sqrt{50}$	$\sqrt{52}$	$\sqrt{65}$	$\sqrt{5}$
A2		0	$\sqrt{37}$	$\sqrt{18}$	$\sqrt{25}$	$\sqrt{17}$	$\sqrt{10}$	$\sqrt{20}$
A3			0	$\sqrt{25}$	$\sqrt{2}$	$\sqrt{2}$	$\sqrt{53}$	$\sqrt{41}$
A4				0	$\sqrt{13}$	$\sqrt{17}$	$\sqrt{52}$	$\sqrt{2}$
A5					0	$\sqrt{2}$	$\sqrt{45}$	$\sqrt{25}$
A6						0	$\sqrt{29}$	$\sqrt{29}$
A7							0	$\sqrt{58}$
A8								0

Solution:

Epsilon neighborhoods of each point are:

Eps-Neighborhood(A1) = $N_\varepsilon(A1) = \{\ \}$

$N_\varepsilon(A2) = \{\ \}$

$N_\varepsilon(A3) = \{A5, A6\}$

$N_\varepsilon(A4) = \{A8\}$

$N_\varepsilon(A5) = \{A3, A6\}$

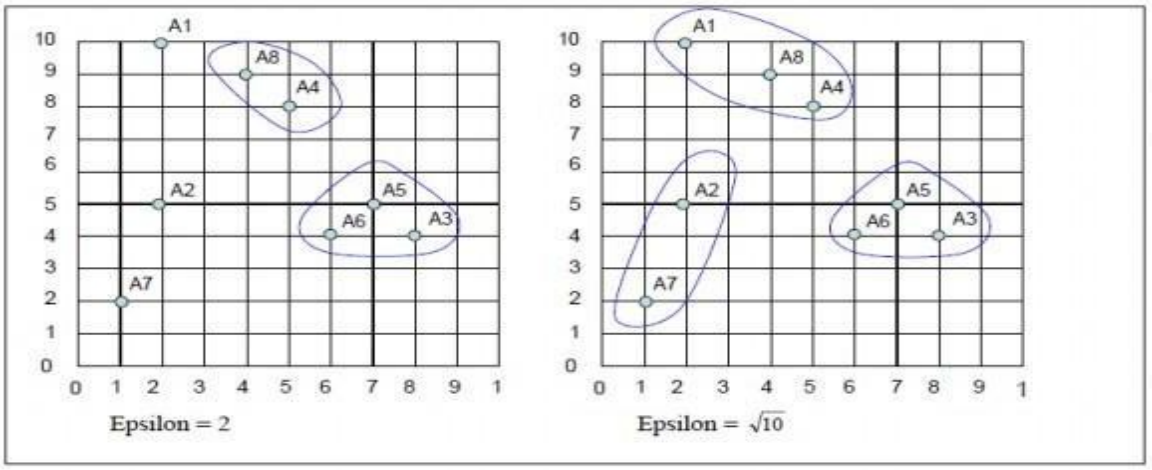
$N_\varepsilon(A6) = \{A3, A5\}$

$N_\varepsilon(A7) = \{\ \}$

$N_\varepsilon(A8) = \{A4\}$

So A1, A2, and A7 are outliers, while we have two clusters C1= {A4, A8} and C2= {A3, A5, A6}

If Epsilon is $\sqrt{10}$ then the neighborhood of some points will increase:
A1 would join the cluster C1 and A2 would joint with A7 to form cluster C3= {A2, A7}.



Strengths and Weaknesses of DBSCAN

Strengths:

- Because DBSCAN uses a density-based definition of a cluster, it is relatively resistant to noise
- DBSCAN can handle clusters of arbitrary shapes and sizes.

- DBSCAN can find many clusters that could not be found using K-means.

Weaknesses:

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