NOTES: MODULE-5

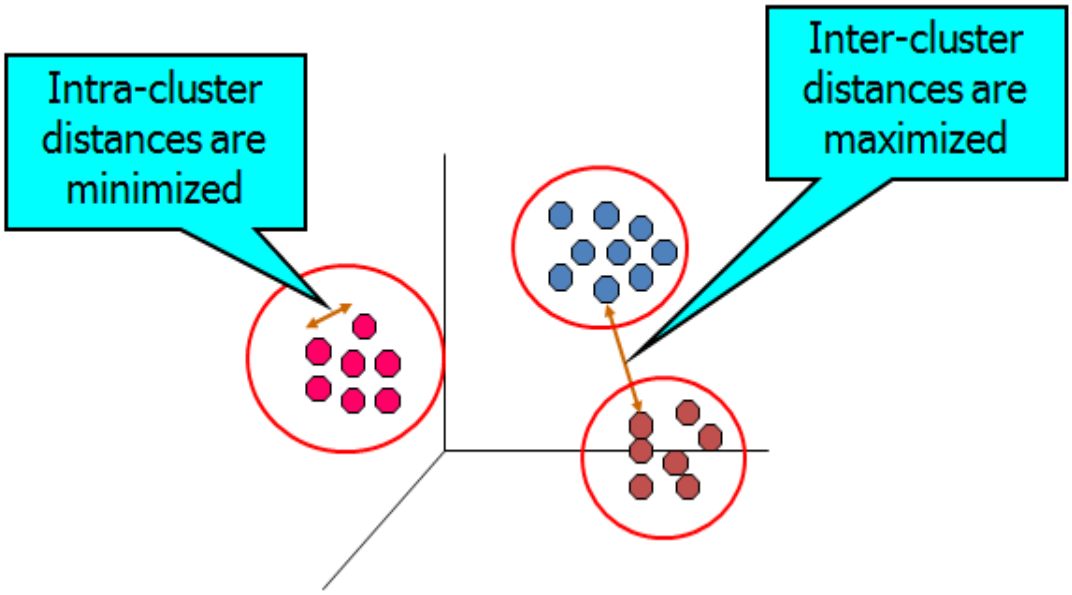
Subject: DATA MINING and DATA WAREHOUSING
SEM; 6the sem CSE
Prepared By: Prof.Abdul Majeed KM
College: PA College of Engineering Mangalore

Clustering Analysis:

- ✓ Overview
- ✓ K-Means
- ✓ Agglomerative Hierarchical Clustering
- ✓ DBSCAN
- ✓ Cluster Evaluation
- ✓ Density-Based Clustering
- ✓ Graph Based Clustering
- ✓ Scalable Clustering Algorithms

What is a Clustering?

- In general a grouping of objects such that the objects in a group (cluster) are similar (or related) to one another and different from (or unrelated to) the objects in other groups



Applications of Cluster Analysis

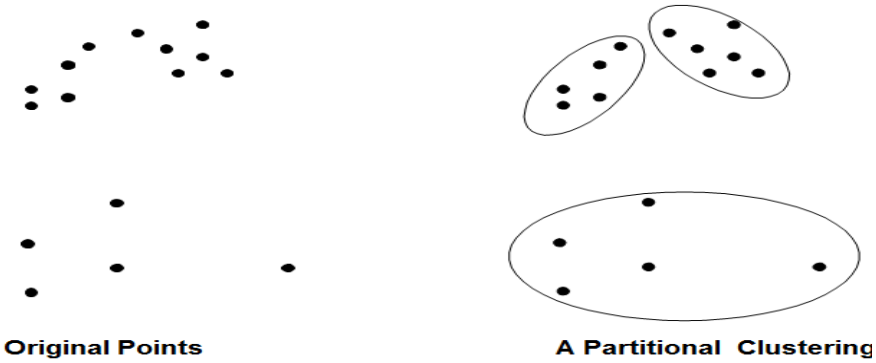
- Understanding - Group related documents for browsing, group genes and proteins that have similar functionality, or group stocks with similar price fluctuations
- Summarization - Reduce the size of large data sets

What is not Cluster Analysis?

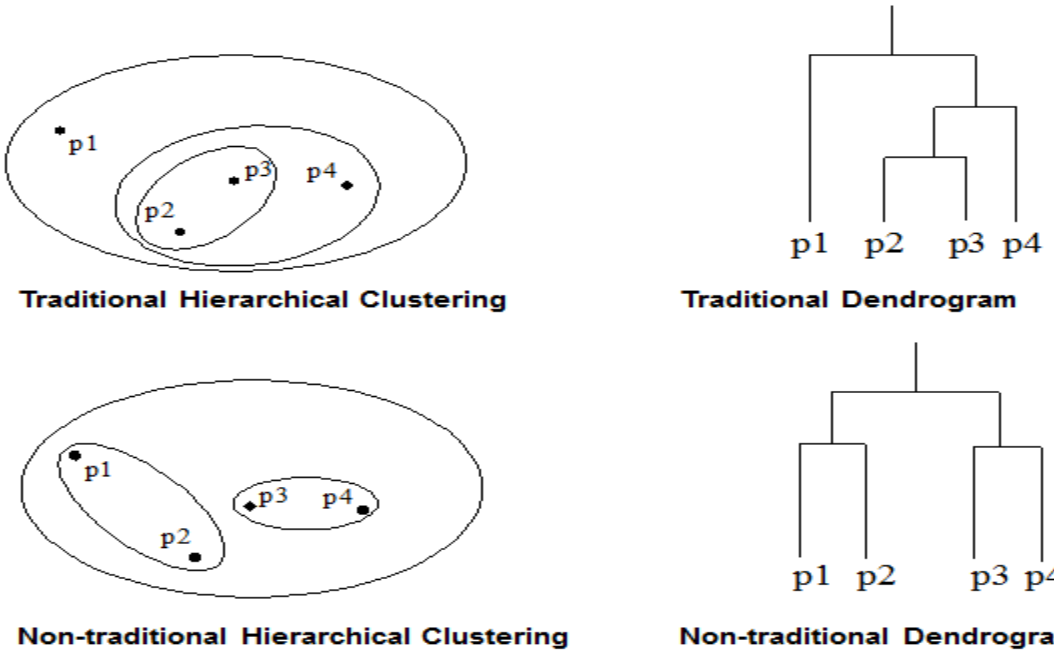
- Supervised classification - Uses class label information
- Simple segmentation - Dividing students into different registration groups alphabetically, by last name
- Results of a query - Groupings are a result of an external specification
  → Clustering uses only the data

Types of Clustering

- A clustering is a set of clusters
- Important distinction between **hierarchical and partitioned** sets of clusters
- **Partitioned Clustering**
  - A division data objects into non-overlapping subsets (clusters) such that each data object is in exactly one subset
- **Hierarchical clustering**
  - A set of nested clusters organized as a hierarchical tree



Partitioned Clustering



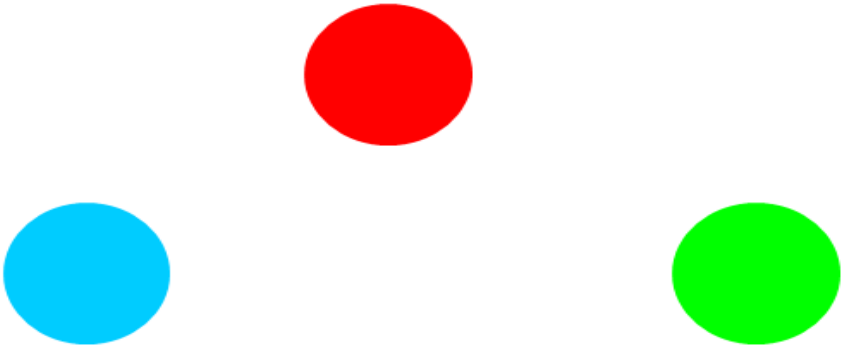
Hierarchical Clustering

- **Exclusive versus non-exclusive**
  - In non-exclusive clusterings, points may belong to multiple clusters.
  - Can represent multiple classes or 'border' points
- **Fuzzy versus non-fuzzy**
  - In fuzzy clustering, a point belongs to every cluster with some weight between 0 and 1
  - Weights must sum to 1
  - Probabilistic clustering has similar characteristics
- **Partial versus complete**
  - In some cases, we only want to cluster some of the data
- **Heterogeneous versus homogeneous**
  - Cluster of widely different sizes, shapes, and densities

Types of Clusters

[1] Well-separated clusters:

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



3 well-separated clusters

[2] Center-based clusters

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

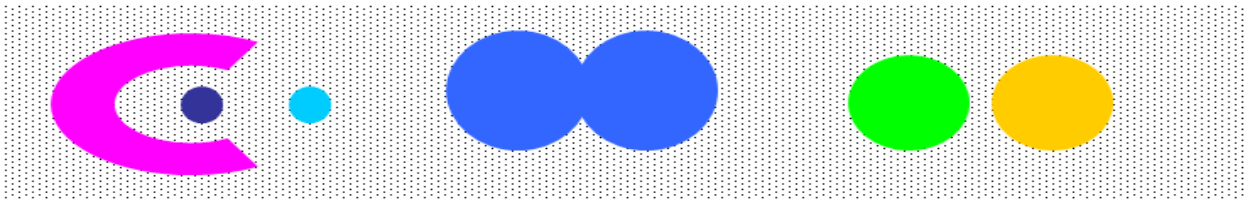
4 center-based clusters

[3] Contiguous clusters: A cluster is a set of points such that a point in a cluster is closer (or more similar) to one or more other points in the cluster than to any point not in the cluster.

8 contiguous clusters

[4] Density-based clusters: A cluster is a dense region of points, which is separated by low-density regions, from other regions of high density.

- Used when the clusters are irregular or intertwined, and when noise and outliers are present.



6 density-based clusters

[5] Property or Conceptual: Finds clusters that share some common property or represent a particular concept.

2 Overlapping Circles

[6] Described by an Objective Function: Finds clusters that minimize or maximize an objective function.

- Enumerate all possible ways of dividing the points into clusters and evaluate the `goodness' of each potential set of clusters by using the given objective function. (NP Hard)
- Can have global or local objectives.
- Hierarchical clustering algorithms typically have local objectives
- Partitional algorithms typically have global objectives
- A variation of the global objective function approach is to fit the data to a parameterized model.
- Parameters for the model are determined from the data.
- Mixture models assume that the data is a 'mixture' of a number of statistical distributions.

K-means Clustering Algorithm

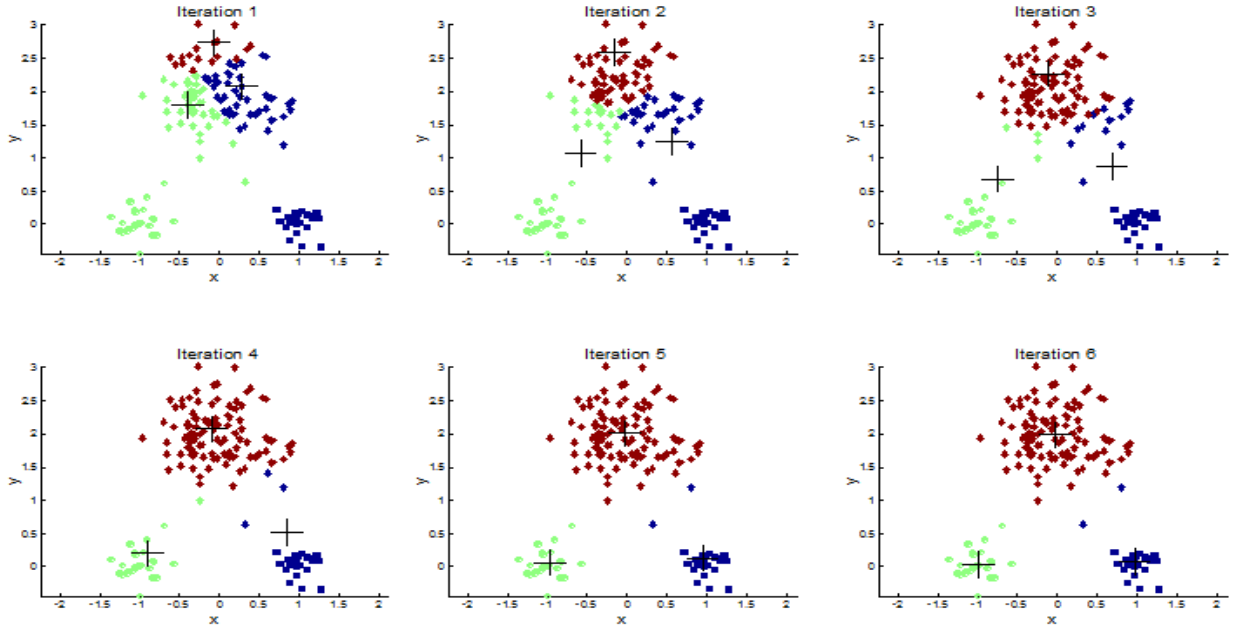
- Partitional clustering approach
- Each cluster is associated with a centroid (center point)
- Each point is assigned to the cluster with the closest centroid
- Number of clusters, K, must be specified
- The basic algorithm is very simple

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

Details:

- Initial centroids are often chosen randomly: Clusters produced vary from one run to another.
- The centroid is (typically) the mean of the points in the cluster.
- 'Closeness' is measured by Euclidean distance, cosine similarity, correlation, etc.
- K-means will converge for common similarity measures mentioned above.
- Most of the convergence happens in the first few iterations: Often the stopping condition is changed to 'Until relatively few points change clusters'
- Complexity is O(n * K * I * d): n = number of points, K = number of clusters, I = number of iterations, d = number of attributes

Importance of Choosing Initial Centroids



Evaluating K-means Clusters

Most common measure is Sum of Squared Error (SSE)

- For each point, the error is the distance to the nearest cluster
- To get SSE, we square these errors and sum them.
- x is a data point in cluster C_i and m_i is the representative point for cluster C_i
  - can show that m_i corresponds to the center (mean) of the cluster
- Given two clusters, we can choose the one with the smallest error
- One easy way to reduce SSE is to increase K, the number of clusters
  - A good clustering with smaller K can have a lower SSE than a poor clustering with higher K

Problems with Selecting Initial Points

If there are K 'real' clusters then the chance of selecting one centroid from each cluster is small.

- Chance is relatively small when K is large
- If clusters are the same size, n, then

$$P = \frac{\text{number of ways to select one centroid from each cluster}}{\text{number of ways to select } K \text{ centroids}} = \frac{K! n^K}{(Kn)^K} = \frac{K!}{K^K}$$

- For example, if K = 10, then probability = $10!/10^{10} = 0.00036$

– Sometimes the initial centroids will readjust themselves in 'right' way, and sometimes they don't

– Consider an example of five pairs of clusters

Solutions to Initial Centroids Problem

- Multiple runs
  - Helps, but probability is not on your side
- Sample and use hierarchical clustering to determine initial centroids
- Select more than k initial centroids and then select among these initial centroids
  - Select most widely separated
- Pre-processing
  - Normalize the data
  - Eliminate outliers
- Postprocessing
  - Eliminate small clusters that may represent outliers
  - Split 'loose' clusters, i.e., clusters with relatively high SSE
  - Merge clusters that are 'close' and that have relatively low SSE
  - Can use these steps during the clustering process
  - ISODATA
- Bisecting K-means
  - Not as susceptible to initialization issues

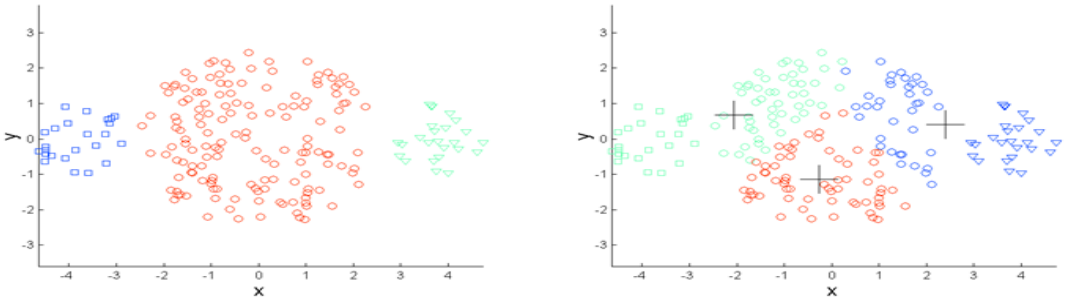
Bisecting K-means Algorithm

– Variant of K-means that can produce a partitional or a hierarchical clustering

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

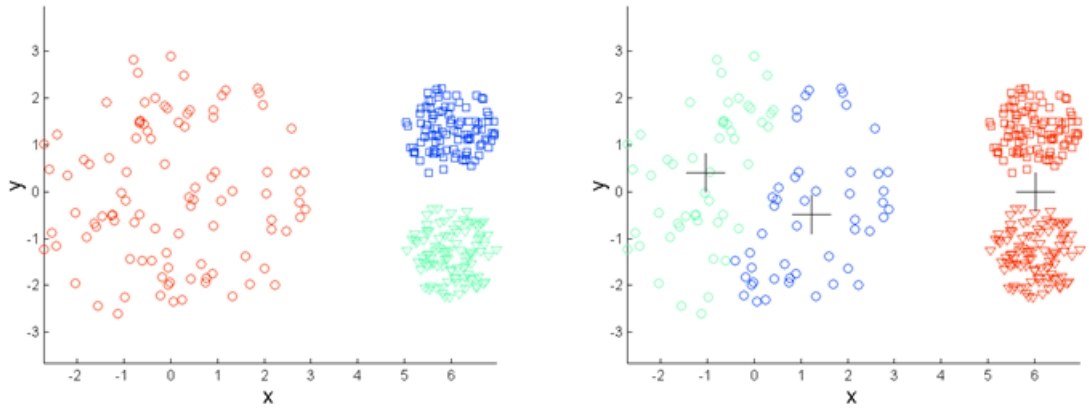
Limitations of K-means

- K-means has problems when clusters are of differing
  - **Sizes**



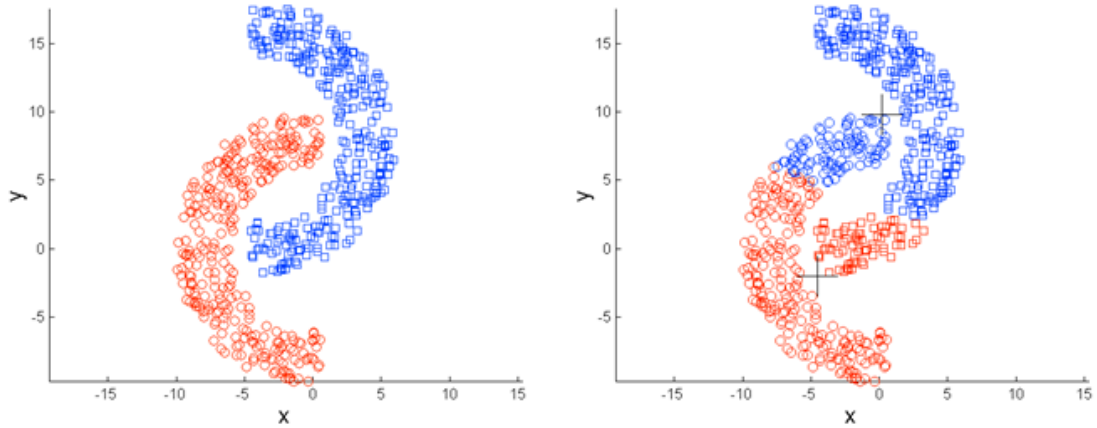
Original Points K-means (3 Clusters)

- **Densities**



Original Points K-means (3 Clusters)

- **Non-globular shapes**

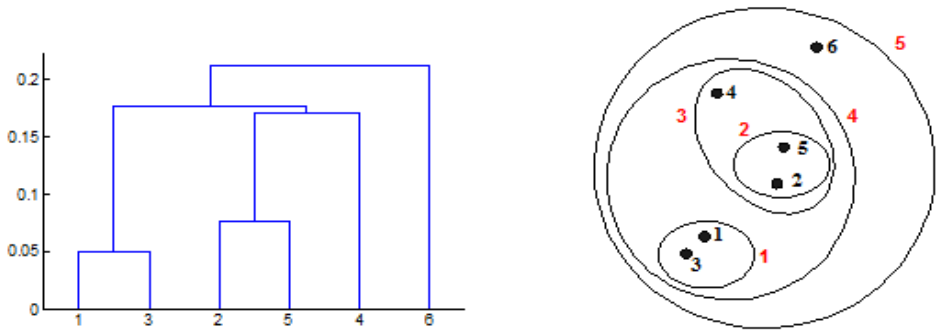


Original Points K-means (2 Clusters)

- K-means has problems when the data contains outliers.

Hierarchical Clustering

- Produces a set of nested clusters organized as a hierarchical tree
- Can be visualized as a dendrogram
  - ✓ A tree like diagram that records the sequences of merges or splits
- Two main types of hierarchical clustering
  - ✓ **Agglomerative:**
    - ❖ Start with the points as individual clusters
    - ❖ At each step, merge the closest pair of clusters until only one cluster (or k clusters) left
  - ✓ **Divisive:**
    - ❖ Start with one, all-inclusive cluster
    - ❖ At each step, split a cluster until each cluster contains a point (or there are k clusters)
- Traditional hierarchical algorithms use a similarity or distance matrix
  - ✓ Merge or split one cluster at a time



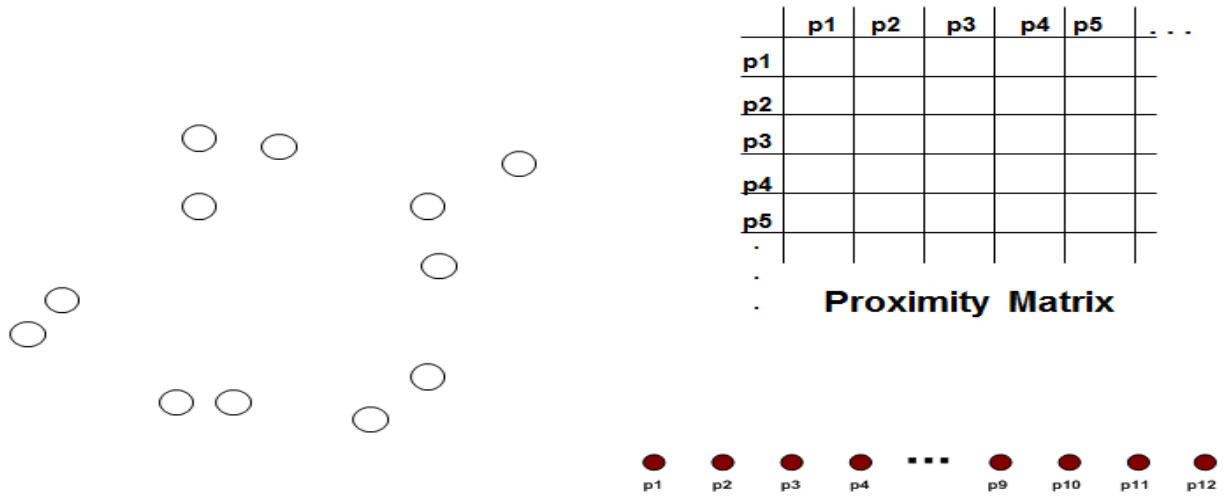
Hierarchical clustering

Agglomerative Clustering Algorithm: More popular hierarchical clustering technique

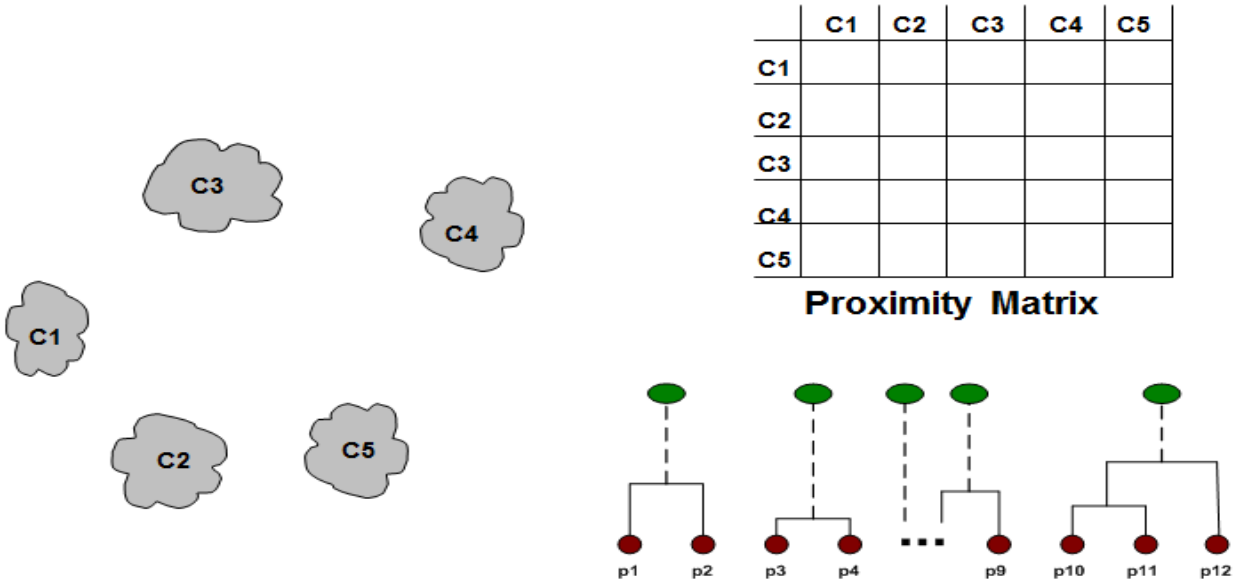
- Basic algorithm is straightforward
  1. *Compute the proximity matrix*
  2. *Let each data point be a cluster*
  3. ***Repeat***
     - a. *Merge the two closest clusters*
     - b. *Update the proximity matrix*
  4. ***Until*** *only a single cluster remains*

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

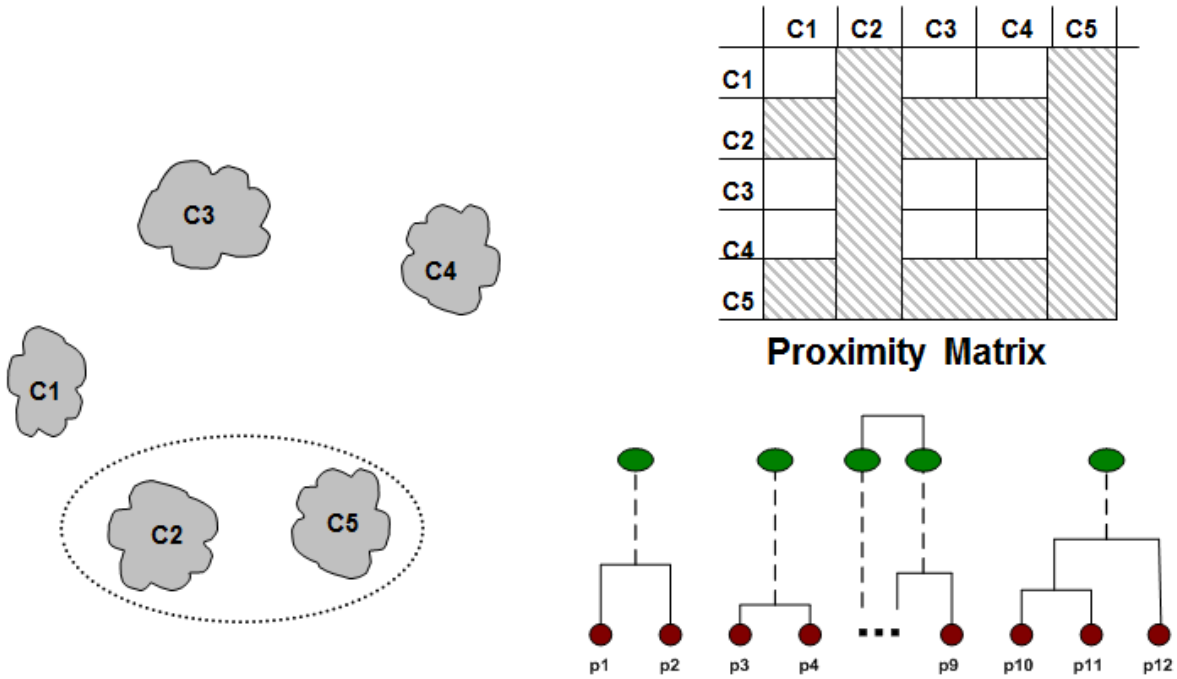
1. Starting Situation: Start with clusters of individual points and a proximity matrix



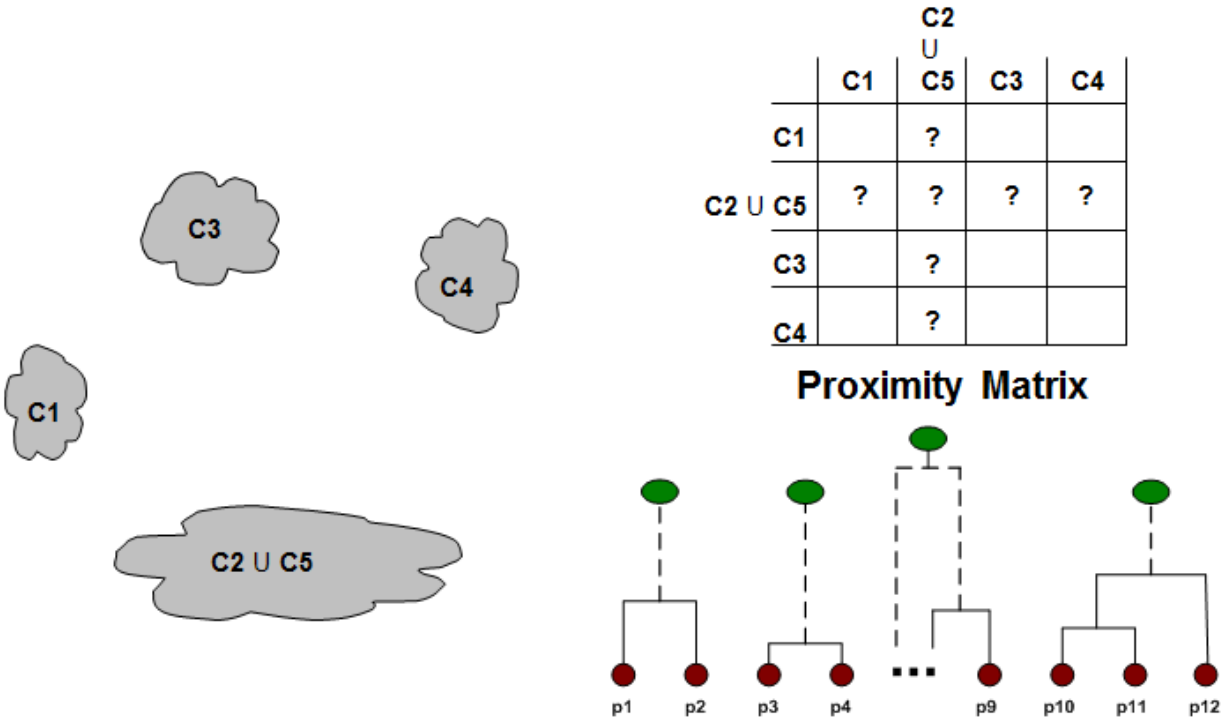
2. Intermediate Situation: After some merging steps, we have some clusters



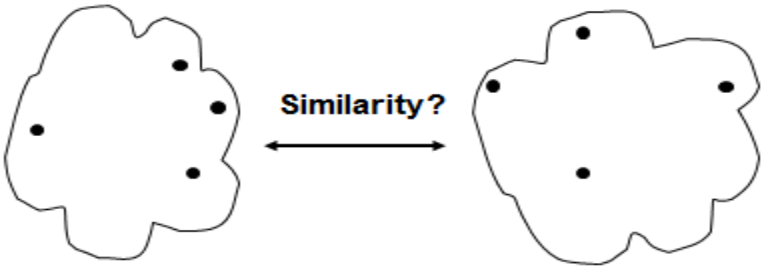
We want to merge the two closest clusters (C2 and C5) and update the proximity matrix.



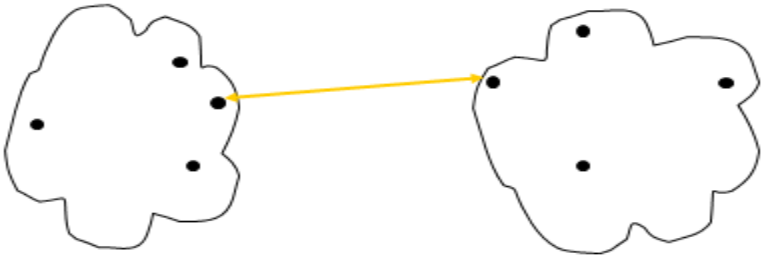
3. **After Merging:** The question is "How do we update the proximity matrix?"



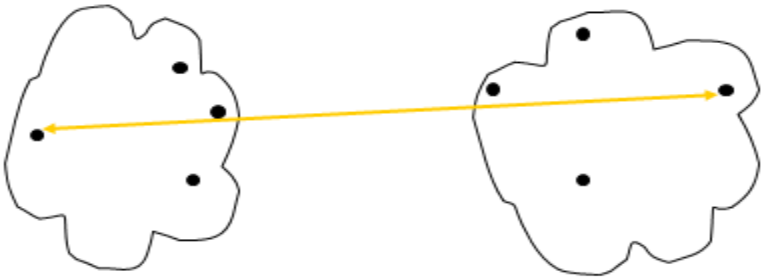
How to Define Inter-Cluster Similarity



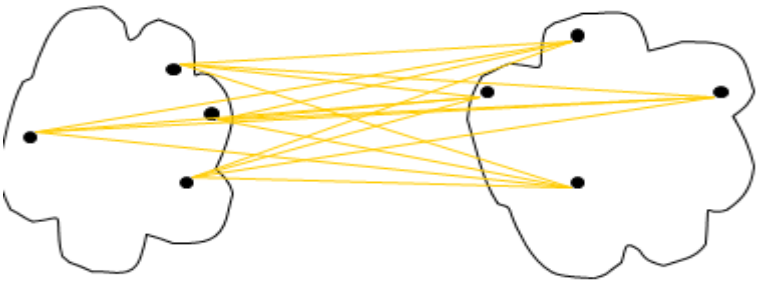
1. **MIN:** Similarity of two clusters is based on the two most similar (closest) points in the different clusters. Determined by one pair of points, i.e., by one link in the proximity graph.



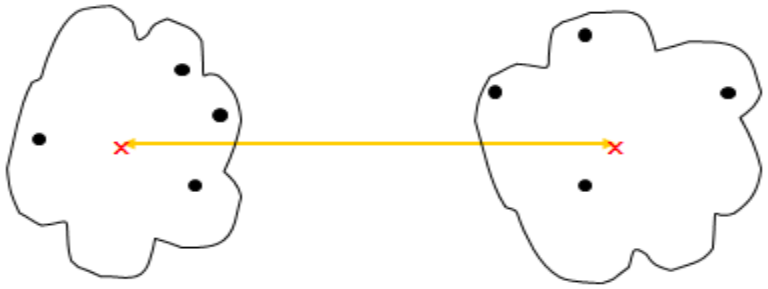
2. **MAX:** Similarity of two clusters is based on the two least similar (most distant) points in the different clusters. Determined by all pairs of points in the two clusters.



3. **Group Average:** Proximity of two clusters is the average of pairwise proximity between points in the two clusters.



4. **Distance Between Centroids**



5. **Other methods driven by an objective function**

✓ **Ward's Method** uses squared error: Similarity of two clusters is based on the increase in squared error when two clusters are merged

- ✓ Similar to group average if distance between points is distance squared
  - Less susceptible to noise and outliers
  - Biased towards globular clusters
  - Hierarchical analogue of K-means
- ✓ Can be used to initialize K-means

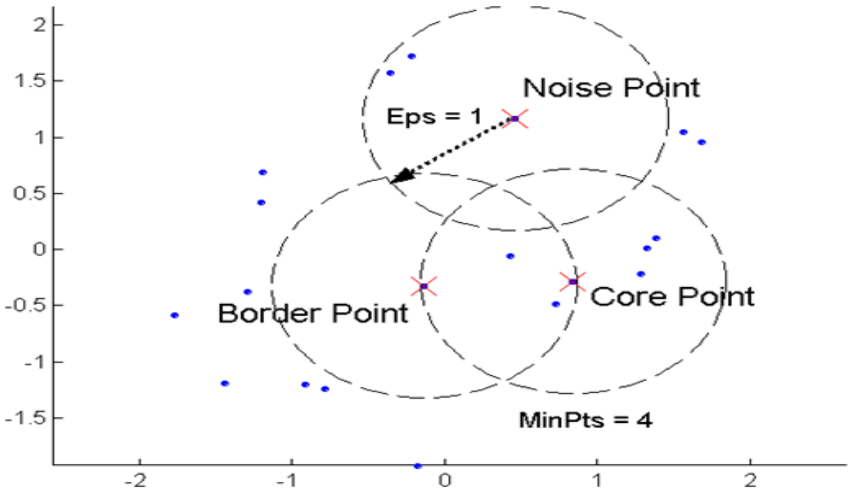
Hierarchical Clustering: Problems and Limitations

- Once a decision is made to combine two clusters, it cannot be undone
- No objective function is directly minimized
- Different schemes have problems with one or more of the following:
  - Sensitivity to noise and outliers
  - Difficulty handling different sized clusters and convex shapes
  - Breaking large clusters

DBSCAN Algorithm

Density-based clustering locates regions of high density that are separated from one another by regions of low density. DBSCAN is a simple and effective density-based clustering algorithm that illustrates a number of important concepts that are important for any density-based clustering approach. In this section, we focus solely on DBSCAN after first considering the key notion of density.

- DBSCAN is a density-based algorithm.
- Density = number of points within a specified radius (Eps)
- 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.



- **Strength:**
  - **Resistant to Noise**
  - **Can handle clusters of different shapes and sizes**
- **Weakness:**
  - **Does not work well with clusters with Varying densities and High-dimensional data**

8.4.2 The DBSCAN Algorithm

Given the previous definitions of core points, border points, and noise points, the DBSCAN algorithm can be informally described as follows. Any two core points that are close enough—within a distance *Eps* of one another—are put in the same cluster. Likewise, any border point that is close enough to a core point is put in the same cluster as the core point. (Ties may need to be resolved if a border point is close to core points from different clusters.) Noise points are discarded. The formal details are given in Algorithm 8.4. This algorithm uses the same concepts and finds the same clusters as the original DBSCAN, but is optimized for simplicity, not efficiency.

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.

Cluster Validity and Evaluation

Determine the Number of Clusters:

- Empirical method
  - # of clusters: $k \approx \sqrt{n/2}$ for a dataset of n points, e.g., n = 200, k = 10
- Other methods:
  - Elbow method
  - Cross validation method

Measuring Clustering Validity

- Numerical measures that are applied to judge various aspects of cluster validity, are classified into the following three types.
  - External Index: Used to measure the extent to which cluster labels match externally supplied class labels.
    - Entropy
  - Internal Index: Used to measure the goodness of a clustering structure *without* respect to external information.
    - Sum of Squared Error (SSE)
  - Relative Index: Used to compare two different clusterings or clusters.
    - Often an external or internal index is used for this function, e.g., SSE or entropy
- Sometimes these are referred to as criteria instead of indices
  - However, sometimes criterion is the general strategy and index is the numerical measure that implements the criterion.

Measuring Cluster Validity Via Correlation

- Two matrices
  - Proximity Matrix
  - "Incidence" Matrix
    - One row and one column for each data point
    - An entry is 1 if the associated pair of points belong to the same cluster
    - An entry is 0 if the associated pair of points belongs to different clusters
- Compute the correlation between the two matrices
  - Since the matrices are symmetric, only the correlation between $n(n-1) / 2$ entries needs to be calculated.
- High correlation indicates that points that belong to the same cluster are close to each other.
- Not a good measure for some density or contiguity based clusters.

Using Similarity Matrix for Cluster Validation

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

Internal Measures: SSE

- Clusters in more complicated figures aren't well separated
- Internal Index: Used to measure the goodness of a clustering structure without respect to external information
  - SSE
- SSE is good for comparing two clustering or two clusters (average SSE).
- Can also be used to estimate the number of clusters

Framework for Cluster Validity

- Need a framework to interpret any measure.
  - For example, if our measure of evaluation has the value, 10, is that good, fair, or poor?
- Statistics provide a framework for cluster validity
  - The more "atypical" a clustering result is, the more likely it represents valid structure in the data
  - Can compare the values of an index that result from random data or clusterings to those of a clustering result.
    - If the value of the index is unlikely, then the cluster results are valid
  - These approaches are more complicated and harder to understand.
- For comparing the results of two different sets of cluster analyses, a framework is less necessary.
  - However, there is the question of whether the difference between two index values is significant

Internal Measures: Cohesion and Separation

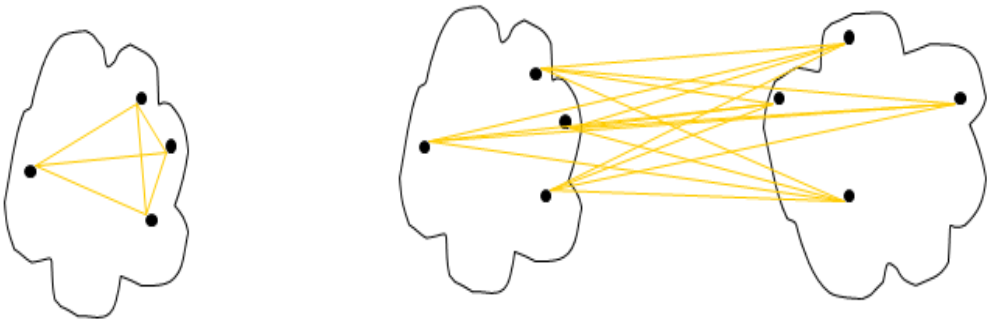
- Cluster Cohesion: Measures how closely related are objects in a cluster
  - Example: SSE
- Cluster Separation: Measure how distinct or well-separated a cluster is from other clusters
- Example: Squared Error
  - Cohesion is measured by the within cluster sum of squares (SSE)

$$WSS = \sum_i \sum_{x \in C_i} (x - m_i)^2$$

  - Separation is measured by the between cluster sum of squares

$$BSS = \sum_i |C_i| (m - m_i)^2$$

- A proximity graph based approach can also be used for cohesion and separation.
  - Cluster cohesion is the sum of the weight of all links within a cluster.
  - Cluster separation is the sum of the weights between nodes in the cluster and nodes outside the cluster.



Internal Measures: Silhouette Coefficient

- Silhouette Coefficient combine ideas of both cohesion and separation, but for individual points, as well as clusters and clusterings

- For an individual point, *i*
  - Calculate ***a*** = average distance of *i* to the points in its cluster
  - Calculate ***b*** = min (average distance of *i* to points in another cluster)
  - The silhouette coefficient for a point is then given by
    s = 1 – a/b if a < b, (or s = b/a - 1 if a ≥ b, not the usual case)
  - Typically between 0 and 1.
  - The closer to 1 the better.
- Can calculate the Average Silhouette width for a cluster or a clustering

External Measures of Cluster Validity: Entropy and Purity

- Assume that the data is labeled with some class labels
  - E.g., documents are classified into topics, people classified according to their income, senators classified as republican or democrat.
- In this case we want the clusters to be homogeneous with respect to classes
  - Each cluster should contain elements of mostly one class
  - Also each class should ideally be assigned to a single cluster
- This does not always make sense
  - Clustering is not the same as classification
- But this is what people use most of the time

Measures

- n = number of points
- m_i = points in cluster i
- c_j = points in class j
- m_{ij}= points in cluster i coming from class j
- $p_{ij} = m_{ij}/m_i$= prob of element from class j in cluster i
- Entropy:
  - Of a cluster i: $e_i = -\sum_{j=1}^{L} p_{ij} \log p_{ij}$
    - ✓ Highest when uniform, zero when single class
  - Of a clustering: $e = \sum_{i=1}^{K} \frac{m_i}{n} e_i$
- Purity:
  - Of a cluster i: $p_i = \max_j p_{ij}$
  - Of a clustering: $purity = \sum_{i=1}^{K} \frac{m_i}{n} p_i$

	Class 1	Class 2	Class 3	
Cluster 1	m_{11}	m_{12}	m_{13}	m_1
Cluster 2	m_{21}	m_{22}	m_{23}	m_2
Cluster 3	m_{31}	m_{32}	m_{33}	m_3
	c_1	c_2	c_3	n

Precision:

Of cluster i with respect to class j: $Prec(i,j) = p_{ij}$

For the precision of a clustering you can take the maximum

Recall:

Of cluster i with respect to class j: $Rec(i,j) = \frac{m_{ij}}{c_j}$

For the precision of a clustering you can take the maximum

F-measure:

Harmonic Mean of Precision and Recall:

$$F(i,j) = \frac{2 * Prec(i,j) * Rec(i,j)}{Prec(i,j) + Rec(i,j)}$$

Density-Based Clustering Methods

- DBSCAN, a simple, but effective algorithm for finding density-based clusters, i.e., dense regions of objects that are surrounded by low-density regions
- This section examines **additional density-based clustering techniques** that address issues of efficiency, finding clusters in subspaces, and more accurately modeling density
    - **Grid-based clustering**
    - **Subspace clustering**
    - **CLIQUE and DENCLUE**,

Grid based clustering:

- Grid-Based Clustering, which breaks the data space into grid cells and then forms clusters from cells that are sufficiently dense. Such an approach can be efficient and effective, at least for low-dimensional data.
- A grid is an efficient way to organize a set of data, at least in low dimensions. The idea is to split the possible values of each attribute into a number of contiguous intervals, creating a set of grid cells
- Each object falls into a grid cell whose corresponding attribute intervals contain the values of the object. Objects can be assigned to grid cells in one pass through the data, and information about each cell, such as the number of points in the cell, can also be gathered at the same time.
- There are a number of ways to perform clustering using a grid, but most approaches are based on density, at least in part, and thus, in this section, we will use grid-based clustering to mean density-based clustering using a grid.

Basic Grid-based Algorithm

1. Define a set of grid-cells
2. Assign objects to the appropriate grid cell and compute the density of each cell.
3. Eliminate cells, whose density is below a certain threshold t.
4. Form clusters from contiguous (adjacent) groups of dense cells (usually minimizing a given objective function)

Example 9.8 (Grid-Based Density). Figure 9.10 shows two sets of two-dimensional points divided into 49 cells using a 7-by-7 grid. The first set contains 200 points generated from a uniform distribution over a circle centered at (2, 3) of radius 2, while the second set has 100 points generated from a uniform distribution over a circle centered at (6, 3) of radius 1. The counts for the grid cells are shown in Table 9.2. Since the cells have equal volume (area), we can consider these values to be the densities of the cells. ■

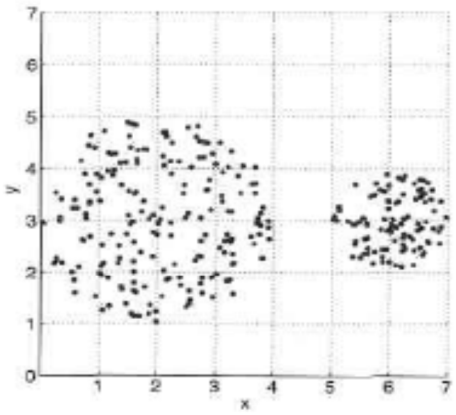


Figure 9.10. Grid-based density.

0	0	0	0	0	0	0
0	0	0	0	0	0	0
4	17	18	6	0	0	0
14	14	13	13	0	18	27
11	18	10	21	0	24	31
3	20	14	4	0	0	0
0	0	0	0	0	0	0

Table 9.2. Point counts for grid cells.

- Density is the number of points per amount of space

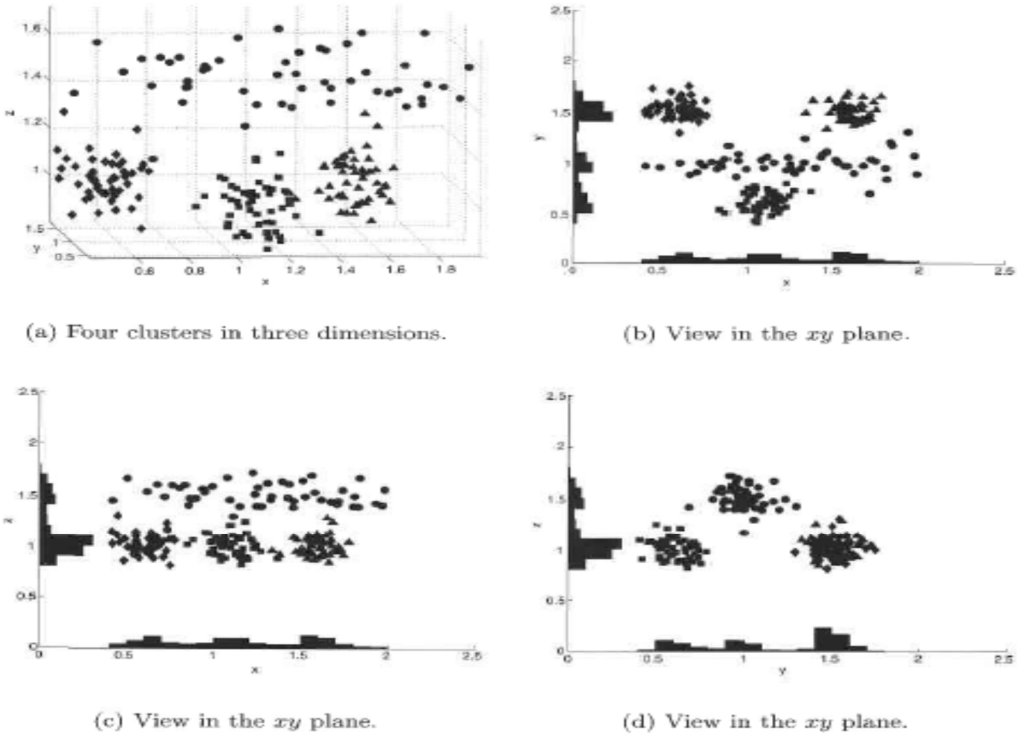
Advantages of Grid-based Clustering Algorithms

✓ **fast**:
- No distance computations
- Clustering is performed on summaries and not individual objects; complexity is usually O(#-populated-grid-cells) and not O(#objects)
- Easy to determine which clusters are neighboring

✓ Shapes are limited to union of grid-cells

Subspace clustering:

- Subspace clustering, which looks for clusters (dense regions) in subsets of all dimensions. For a data space with n dimensions, potentially 2" - 1 subspaces need to be searched, and thus an efficient technique is needed to do this.
- However, if only subsets of the features are considered, i.e., subspaces of the data, then the clusters that we find can be quite different from one subspace to another
- There are two reasons that subspace clusters may be interesting. First, the data may be clustered with respect to a small set of attributes, but randomly distributed with respect to the remaining attributes. Second, there are cases in which different clusters exist in different sets of dimensions.



(a) Four clusters in three dimensions. (b) View in the xy plane.

(c) View in the xy plane. (d) View in the xy plane.

Figure 9.11. Example figures for subspace clustering.

CLIQUE (CLustering In QUEst):

- CLIQUE is a grid-based clustering algorithm that methodically finds subspace clusters. It is impractical to check each subspace for clusters since the number of such subspaces is exponential in the number of dimensions. Instead, CLIQUE relies on the following property;
- *Monotonicity property of density-based clusters:* If a set of points forms a density-based cluster in k dimensions (attributes), then the same set of points is also part of a density-based cluster in all possible subsets of those dimensions.

Algorithm 9.5 CLIQUE.

1: Find all the dense areas in the one-dimensional spaces corresponding to each attribute. This is the set of dense one-dimensional cells.
2: $k \leftarrow 2$
3: **repeat**
4: Generate all candidate dense k-dimensional cells from dense $(k-1)$-dimensional cells.
5: Eliminate cells that have fewer than ξ points.
6: $k \leftarrow k+1$
7: **until** There are no candidate dense k-dimensional cells.
8: Find clusters by taking the union of all adjacent, high-density cells.
9: Summarize each cluster using a small set of inequalities that describe the attribute ranges of the cells in the cluster.

DENCLUE (DENsity CLUstEring)

- DENCLUE: A Kernel-Based Scheme for Density-Based Clustering
- DENCLUE is a density-based clustering approach that models the overall density of a set of points as the sum of influence functions associated with each point
- The resulting overall density function will have local peaks, i.e., local density maxima, and these local peaks can be used to define clusters in a natural way.
- Specifically, for each data point, a hill climbing procedure finds the nearest peak associated with that point, and the set of all data points associated with a particular peak (called a **local density attractor**) becomes a cluster
- However, if the density at a local peak is too low, then the points in the associated cluster are classified as noise and discarded.
- Also, if a local peak can be connected to a second local peak by a path of data points, and the density at each point on the path is above the minimum density threshold, then the clusters associated with these local peaks are merged.
- Therefore, **clusters of any shape can be discovered.**

Algorithm 9.6 DENCLUE algorithm.

1: Derive a density function for the space occupied by the data points.
2: Identify the points that are local maxima.
(These are the density attractors.)
3: Associate each point with a density attractor by moving in the direction of maximum increase in density.
4: Define clusters consisting of points associated with a particular density attractor.
5: Discard clusters whose density attractor has a density less than a user-specified threshold of ξ.
6: Combine clusters that are connected by a path of points that all have a density of ξ or higher.

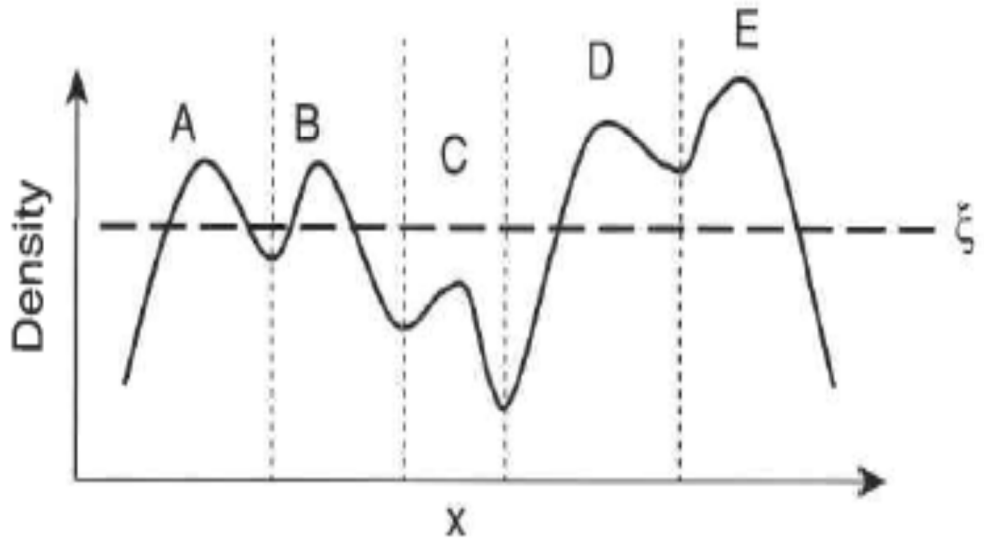


Figure 9.13. Illustration of DENCLUE density concepts in one dimension.

Graph Based Clustering

This section considers some graph-based clustering algorithms that use a number of key properties and characteristics of graphs. The following are some key approaches, different subsets of which are employed by these algorithms.

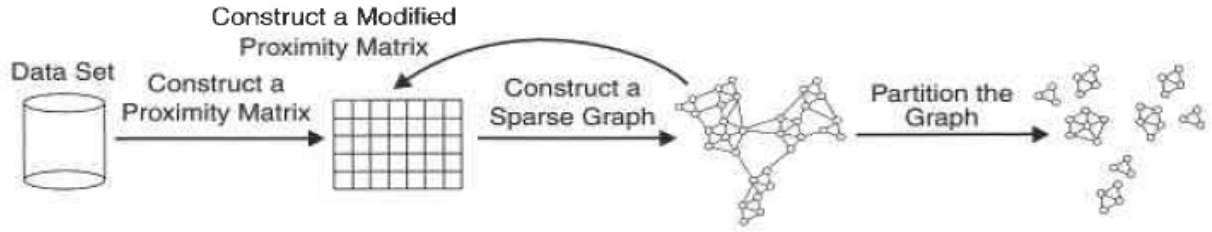
- **MST, Opossum, Chameleon, Jarvis-Patrick, and SNN Clustering algorithm**

1. Sparsify the proximity graph to keep only the connections of an object with its nearest neighbors. This sparsification is useful for handling noise and outliers. It also allows the use of highly efficient graph partitioning algorithms that have been developed for sparse graphs.

2. Define a similarity measure between two objects based on the number of nearest neighbors that they share. This approach, which is based on the observation that an object and its nearest neighbors usually belong to the same class, is useful for overcoming problems with high dimensionality and clusters of varying density.
3. Define core objects and build clusters around them. To do this for graph based clustering, it is necessary to introduce a notion of density-based on a proximity graph or a sparsified proximity graph.
4. Use the information in the proximity graph to provide a more sophisticated evaluation of whether two clusters should be merged. Two clusters are merged only if the resulting cluster will have characteristics similar to the original two clusters.

Sparsification

The m by m proximity matrix for rn data points can be represented as a dense graph in which each node is connected to all others and the weight of the edge between any pair of nodes reflects their pairwise proximity. The sparsification may be performed, for example, by breaking all links that have a similarity (dissimilarity) below (above) a specified threshold or by keeping only links to the k nearest neighbors of point. This latter approach creates what is called a k-nearest neighbor graph.



Ideal process of clustering using sparsification.

A. Minimum Spanning Tree (MST) Clustering:

MST starts with the minimum spanning tree of the proximity graph and can be viewed as an application of sparsification for finding clusters. A minimum spanning tree of a graph is a subgraph that (1) has no cycles, i.e., is a tree, (2) contains all the nodes of the graph, and (3) has the minimum total edge weight of all possible spanning trees.
The first step is to find the MST of the original dissimilarity graph. Note that a minimum spanning tree can be viewed as a special type of sparsified graph. Step 3 can also be viewed as graph sparsification. Hence, MST can be viewed as a clustering algorithm based on the sparsification of the dissimilarity graph.

MST divisive hierarchical clustering algorithm.

1: Compute a minimum spanning tree for the dissimilarity graph.
2: **repeat**
3: Create a new cluster by breaking the link corresponding to the largest dissimilarity.
4: **until** Only singleton clusters remain.

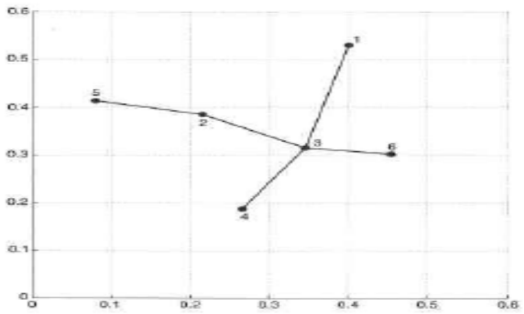


Figure 9.16. Minimum spanning tree for a set of six two-dimensional points.

OPOSSUM: Optimal Partitioning of Sparse Similarities Using METIS:

OPOSSUM is a clustering technique that was specifically designed for clustering sparse, high dimensional data, such as document or market basket data. Like MST, it performs clustering based on the sparsification of a proximity graph. However, OPOSSUM uses the METIS algorithm, which was specifically created for partitioning sparse graphs. The steps of OPOSSUM are given below:

OPOSSUM clustering algorithm.

1: Compute a sparsified similarity graph.
2: Partition the similarity graph into k distinct components (clusters) using METIS.

The METIS graph partitioning program partitions a sparse graph into k distinct components, where k is a user-specified parameter, in order to (1) minimize the weight of the edges (the similarity) between components and (2) fulfiIl a balance constraint. OPOSSUM uses one of the following two balance constraints: (1) the number of objects in each cluster must be roughly the same, or (2) the sum of the attribute values must be roughly the same.

Strengths

OPOSSUM is simple and fast. It partitions the data into roughly equal-sized clusters.

Weaknesses

if OPOSSUM is used to generate a large number of clusters, then these clusters are typically relatively pure pieces of larger clusters.

Chameleon: Hierarchical Clustering with Dynamic Modeling:

- Adapt to the characteristics of the data set to find the natural clusters
- Use a dynamic model to measure the similarity between clusters
  - Main property is the **relative closeness and relative inter-connectivity** of the cluster
  - ***Two clusters are combined if the resulting cluster shares certain properties with the constituent clusters***
  - The merging scheme preserves *self-similarity*
- One of the areas of application is spatial data

Relative Closeness (RC) is the absolute closeness of two clusters normalized by the internal closeness of the clusters. Two clusters are combined only if the points in the resulting cluster are almost as close to each other as in each of the original clusters.

$$RC = \frac{\bar{S}_{EC}(C_i, C_j)}{\frac{m_i}{m_i+m_j}\bar{S}_{EC}(C_i) + \frac{m_j}{m_i+m_j}\bar{S}_{EC}(C_j)},$$

where m_i and m_j are the sizes of clusters C_i and C_j, respectively.

$\bar{S}_{EC}(C_i, C_j)$	is the average weight of the edges (of the k-nearest neighbor graph) that connect clusters C_i and C_j
$\bar{S}_{EC}(C_i)$	is the average weight of edges if we bisect cluster C_i
$\bar{S}_{EC}(C_j)$	is the average weight of edges if we bisect cluster Cj. (EC stands for edge cut.)

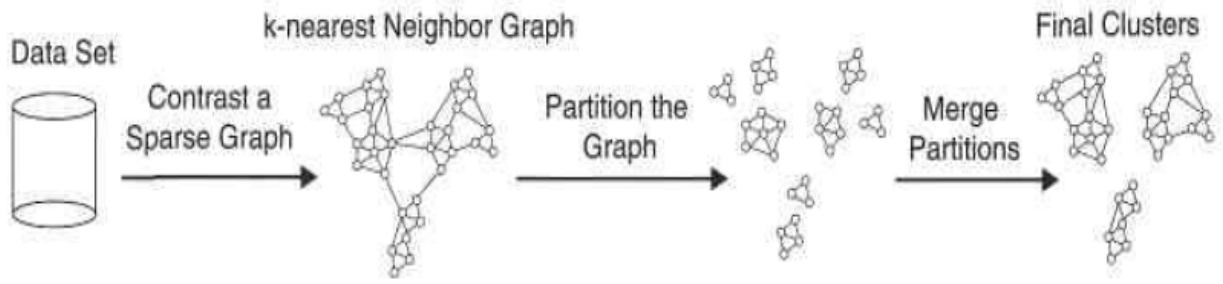
Relative Interconnectivity (RI) is the absolute interconnectivity of two clusters normalized by the internal connectivity of the clusters. Two clusters are combined if the points in the resulting cluster are almost as strongly connected as points in each of the original clusters.

$$RI = \frac{EC(C_i, C_j)}{\frac{1}{2}(EC(C_i) + EC(C_j))}$$

where $EC(C_i,C_j)$ is the sum of the edges (of the k-nearest neighbor graph) that connect clusters C_i and C_j; $EC(C_i)$ is the minimum sum of the cut edges if we bisect cluster Ci; and EC(Cj) is the minimum sum of the cut edges if we bisect cluster C_j.

Chameleon algorithm.

1: Build a *k*-nearest neighbor graph.
2: Partition the graph using a multilevel graph partitioning algorithm.
3: **repeat**
4: Merge the clusters that best preserve the cluster self-similarity with respect to relative interconnectivity and relative closeness.
5: **until** No more clusters can be merged.



Overall process by which Chameleon performs clustering.

The first step in Chameleon is to generate a k-nearest neighbor graph. Conceptually, such a graph is derived from the proximity graph, and it contains links only between a point and its k nearest neighbors, i.e., the points to which it is closest.
Once a sparsified graph has been obtained, an efficient multilevel graph partitioning algorithm, such as METIS can be used to partition the data set.
Agglomerative Hierarchical Clustering: Chameleon merges clusters based on the notion of self-similarity. Chameleon can be parameterized to merge more than one pair of clusters in a single step and to stop before all objects have been merged into a single cluster.
Strengths: Chameleon can effectively cluster spatial data, even though noise and outliers are present and the clusters are of different shapes, sizes, and density.
Limitations: Chameleon has problems when the partitioning process does not produce subclusters, as is often the case for high-dimensional data.

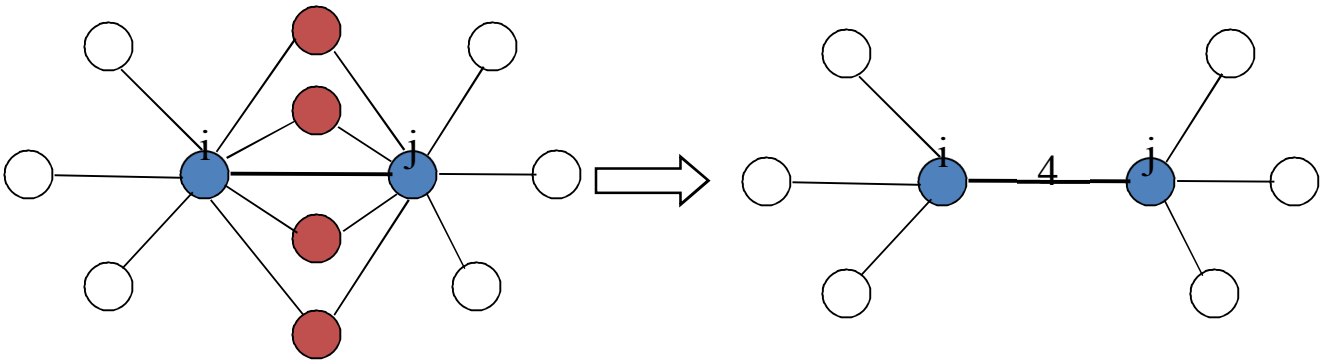
Shared Nearest Neighbor (SNN) Similarity:

- In some cases, clustering techniques that rely on standard approaches to similarity and density do not produce the desired clustering results. This section examines the reasons for this and introduces an indirect approach to similarity that is based on the following principle:
- If two points are similar to many of the same points, then they are similar to one another, even if a direct measurement of similarity does not indicate this.

Algorithm 9.10 Computing shared nearest neighbor similarity

1: Find the k-nearest neighbors of all points.
2: **if** two points, $\mathbf{x}$ and $\mathbf{y}$ are *not* among the k-nearest neighbors of each other **then**
3: $similarity(\mathbf{x}, \mathbf{y}) \leftarrow 0$
4: **else**
5: $similarity(\mathbf{x}, \mathbf{y}) \leftarrow$ number of shared neighbors
6: **end if**

-



SNN graph: the weight of an edge is the number of shared neighbors between vertices given that the vertices are connected

Jarvis-Patrick(JP) Clustering Algorithm:

- The JP clustering algorithm replaces the proximity between two points with the SNN similarity.
- A threshold is then used to sparsify this matrix of SNN similarities.
- In graph terms, an SNN similarity graph is created and sparsified.
- Clusters are simply the connected components of the SNN graph.

Algorithm 9.11 Jarvis-Patrick clustering algorithm.

1: Compute the SNN similarity graph.
2: Sparsify the SNN similarity graph by applying a similarity threshold.
3: Find the connected components (clusters) of the sparsified SNN similarity graph.

- Strength:
  - Because JP clustering is based on the notion of SNN similarity, it is good at dealing with noise and outliers and can handle clusters of different sizes shapes, and densities.
  - The algorithm works well for high-dimensional data and is particularly good at finding tight clusters of strongly related objects.
- Limitation: Jarvis-Patrick clustering is too brittle and not all objects are clustered

SNN Density based Clustering Algorithm (SSN + DBSCAN):

1. **Compute the similarity matrix** This corresponds to a similarity graph with data points for nodes and edges whose weights are the similarities between data points
2. **Sparsify the similarity matrix by keeping only the *k* most similar neighbors** This corresponds to only keeping the *k* strongest links of the similarity graph
3. **Construct the shared nearest neighbor graph from the sparsified similarity matrix.** At this point, we could apply a similarity threshold and find the connected components to obtain the clusters (Jarvis-Patrick algorithm)
4. **Find the SNN density of each Point.** Using a user specified parameters, *Eps*, find the number points that have an SNN similarity of *Eps* or greater to each point. This is the SNN density of the point
5. **Find the core points** Using a user specified parameter, *MinPts*, find the core points, i.e., all points that have an SNN density greater than *MinPts*
6. **Form clusters from the core points** If two core points are within a radius, *Eps*, of each other they are place in the same cluster
7. **Discard all noise points** All non-core points that are not within a radius of *Eps* of a core point are discarded
8. **Assign all non-noise, non-core points to clusters** This can be done by assigning such points to the nearest core point

(Note: Step 1 to 4:Compute SSN Similarity Graph and steps 4-8 are DBSCAN)

Algorithm 9.12 SNN density-based clustering algorithm.

1: Compute the SNN similarity graph.
2: Apply DBSCAN with user-specified parameters for *Eps* and *MinPts*.

Scalable Clustering Algorithms

- Even the best clustering algorithm is of little value if it takes an unacceptably long time to execute or requires too much memory.
- This section examines clustering techniques that place significant emphasis on scalability to the very large data sets that are becoming increasingly common.
- General strategies for scalability, including approaches for reducing the number of proximity calculations, sampling the data, partitioning the data, and clustering a summarized representation of the data.
- Two specific examples of scalable clustering algorithms:
  - **CURE and BIRCH.**

BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies):

BIRCH is a highly efficient clustering technique for data in Euclidean vector spaces i.e. data for which mean value make sense. BIRCH can efficiently cluster such data with one pass and can improve that clustering with additional passes. BIRCH can also deal effectively with outliers.

BIRCH is based on the notion of a Clustering Feature (CF) and a CF tree. The idea is that a cluster of data points (vectors) can be represented by a triple of numbers (N, LS, SS), where N is the number of points in the cluster, LS is the linear sum of the points, and SS is the sum of squares of the points. A CF tree is a height-balanced tree. Leaf nodes consist of a sequence of clustering features, CF_i, where each clustering feature represents a number of points that have been previously scanned. Leaf nodes are subject to the restriction that each leaf node must have a diameter that is less than a parameterized threshold, T. By adjusting the threshold parameter T, the height of the tree can be controlled.

A CF tree is built as the data is scanned. As each data point is encountered, the CF tree is traversed, starting from the root and choosing the closest node at each level. When the closest leaf cluster for the current data point is finally identified, a test is performed to see if adding the data item to the candidate cluster will result in a new cluster with a diameter greater than the given threshold, T. If not, then the data point is added to the candidate cluster by updating the CF information. The cluster information for all nodes from the leaf to the root is also updated.

BIRCH follows each split with a merge step. At the interior node where the split stops, the two closest entries are found. If these entries do not correspond to the two entries that just resulted from the split, then an attempt is made to merge these entries and their corresponding child nodes.

Algorithm BIRCH.

1: **Load the data into memory by creating a CF tree that summarizes the data.**
2: **Build a smaller CF tree if it is necessary for phase 3.** T is increased, and then the leaf node entries (clusters) are reinserted. Since T has increased, some clusters will be merged.
3: **Perform global clustering.** Different forms of global clustering (clustering that uses the pairwise distances between all the clusters) can be used. However, an agglomerative, hierarchical technique was selected. Because the clustering features store summary information that is important to certain kinds of clustering, the global clustering algorithm can be applied as if it were being applied to all the points in a cluster represented by the CF.
4: **Redistribute the data points using the centroids of clusters discovered in step 3, and thus, discover a new set of clusters.** This overcomes certain problems that can occur in the first phase of BIRCH. Because of page size constraints and the T parameter, points that should be in one cluster are sometimes split, and points that should be in different clusters are sometimes combined. Also, if the data set contains duplicate points, these points can sometimes be clustered differently, depending on the order in which they are encountered. By repeating this phase multiple times, the process converges to a locally optimum solution.

CURE (Clustering Using REpresentatives):

- **CURE** is a clustering algorithm that uses a variety of different techniques to create an approach that can handle large data sets, outliers, and clusters with non-spherical shapes and non-uniform sizes.
- CURE represents a cluster by using multiple representative points from the cluster.
- The first representative point is chosen to be the point farthest from the center of the cluster, while the remaining points are chosen so that they are farthest from all the previously chosen points.
- In this way, the representative points are naturally relatively well distributed.
- Once the representative points are chosen, they are shrunk toward the center by a factor, α

Algorithm CURE.

1: **Draw a random sample from the data set.** The CURE paper is notable for explicitly deriving a formula for what the size of this sample should be in order to guarantee, with high probability, that all clusters are represented by a minimum number of points.
2: **Partition the sample into p equal-sized partitions.**
3: **Cluster the points in each partition into $\frac{m}{pq}$ clusters using CURE's hierarchical clustering algorithm to obtain a total of $\frac{m}{q}$ clusters.** Note that some outlier elimination occurs during this process.
4: **Use CURE's hierarchical clustering algorithm to cluster the $\frac{m}{q}$ clusters found in the previous step until only K clusters remain.**
5: **Eliminate outliers.** This is the second phase of outlier elimination.
6: **Assign all remaining data points to the nearest cluster to obtain a complete clustering.**

.