Advanced Clustering Algorithms.

1. DBSCAN (Density - based spatial clustering of applications with noise).

2.Graph Based Clustering Algorithms.

3.Grid Based Clustering Algorithms.

4.CLARA (Clustering LARge Applications).

5.CLARANS (CLARA Based on Randomized Search).

6.STING (Grid Based Multi-resolution Clustering).

1.Density-based spatial clustering of applications with noise (DBSCAN).

This is a density based clustering algorithm given a set of points in some space, it groups together points that are closely packed together (points with many nearby neighbors), marking as outliers points that lie alone in low-density regions (whose nearest neighbors are too far away).

Suppose we have a dataset of n-dimensional data points.

1. For each point in the dataset we make an n-dimensional sphere of radius **epsilon** around the point and count the number of data points within the sphere.
2. If the number of points within the sphere are more than **min_points** then we mark the center of the sphere to be belonging to a cluster.
3. We also mark the points inside the sphere to be belonging to the same cluster. We then recursively expand the cluster by applying the same criteria to the points inside the sphere, except the center.
4. Incase the number of points inside the sphere are less than **min_points**, we ignore it and proceed to the next point in the dataset.
5. But if it's isolated cannot form the cluster cannot be reachable by other clusters, we call this one **outlier** or **noise**.

The parameters **epsilon** and **min_points** can be determined for the best possible clustering using the dataset itself. This method is called adaptive DBSCAN, which I'm not going to deal with over here. However, for practical purposes we may initialize the values manually if we know the kind of data we will run it on.

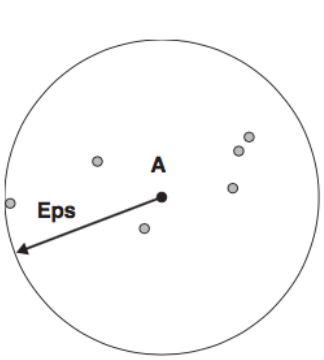


Figure (a) Center-based

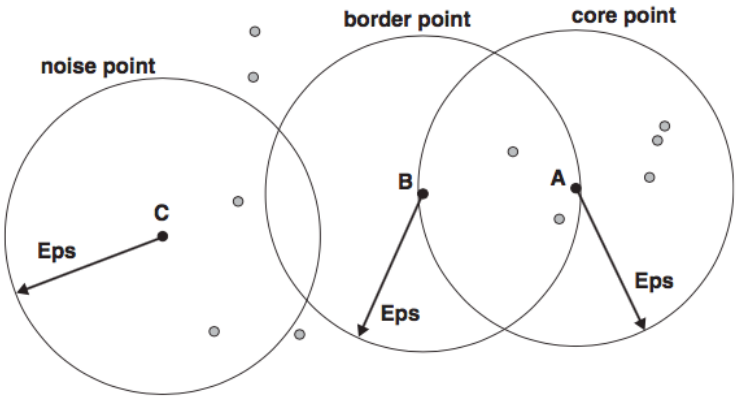


Figure (b) Core, border, and noise points.

Core points: These points are in the interior of a density-based cluster. A point is a core point if the number of points within a given neighborhood around the point as determined by the distance function and a user- specified distance parameter, **epsilon**, exceeds a certain threshold, **min_points**, which is also a user-specified parameter. In Figure 8.21, **point A is a core point**, for the indicated radius **epsilon**.(if min_points ≤ 7).

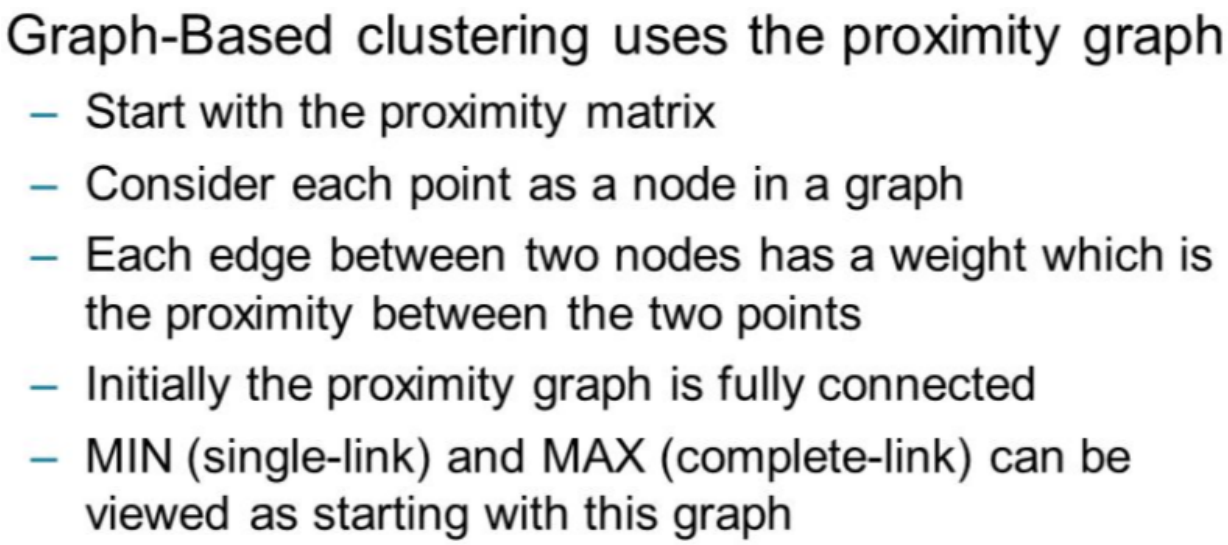
Border points: A border point is not a core point, but falls within the neighborhood of a core point. In Figure 8.21, point B is a border point. A border point can fall within the neighborhoods of several core points.

Noise points: A noise point is any point that is neither a core point nor a border point. In Figure 8.21, point C is a noise point.

2. Graph Based Clustering Algorithms.

A graph-based clustering algorithm will first construct a graph or hypergraph and then apply a clustering algorithm to partition the graph or hypergraph. A link-based clustering algorithm can also be considered as a graph-based one, because we can think of the links between data points as links between the graph nodes.

Graph-based clustering algorithm is based on graph theory.



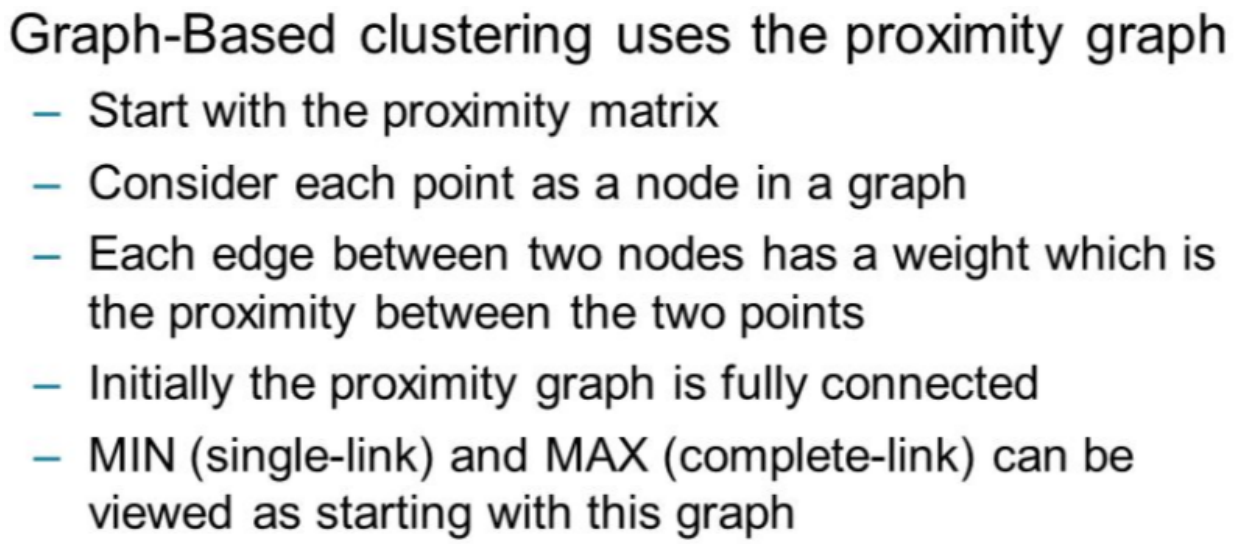
3. Grid Based Clustering Algorithms.

that it is concerned not with the data points but with the value space that surrounds the data points. In general, a typical grid-based clustering algorithm consists of the following five basic steps :

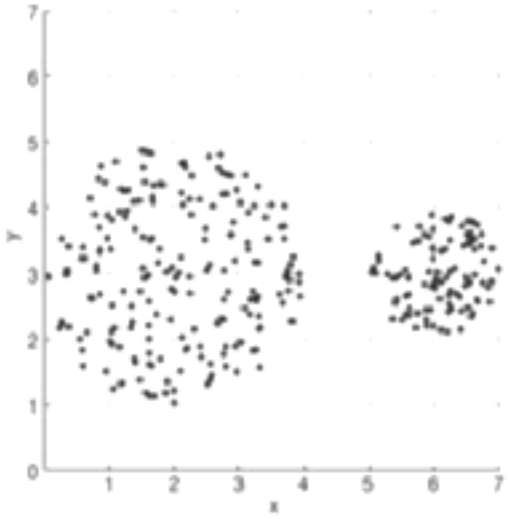
1. Creating the grid structure, i.e., partitioning the data space into a finite number of cells.
2. Calculating the cell density for each cell.
3. Sorting of the cells according to their densities.
4. Identifying cluster centers.
5. Traversal of neighbor cells.

The computational complexity of most clustering algorithms is at least linearly proportional to the size of the data set.The computational complexity depends on number of populated grid cells and not on the data set.

The grid-based clustering approach differs from the conventional clustering algorithms in



The computational complexity of most clustering algorithms is at least linearly proportional to the size of the data set.The computational complexity depends on number of populated grid cells and not on the data set.



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

The great advantage of grid-based clustering is its significant reduction of the computational complexity, especially for clustering very large data set. Another advantage is that there is no distance computation between the data cells.

4. CLARA (Clustering LARge Applications).

An obvious way of clustering larger datasets is to try and extend existing methods so that they can cope with a larger number of objects. The focus is on clustering large numbers of objects rather than a small number of objects in high dimensions. CLARA extends their k-medoids approach for a large number of objects. It works by clustering a sample from the dataset and then assigns all objects in the dataset to these clusters.

CLARA (CLustering LARge Applications) relies on the sampling approach to handle large data sets. Instead of finding medoids for the entire data set, CLARA draws a small sample from the data set and applies the PAM algorithm to generate an optimal set of medoids for the sample. The quality of resulting medoids is measured by the average dissimilarity between every object in the entire data set D and the medoid of its cluster, defined as the following cost function:

D - Data Structure to be clustered.
O_i - Object i in D
S - Sample of D.
n - number of objects in D.
k - number of clusters.
s - Size of S.

$$Cost(M, D) = \frac{\sum_{i=1}^{n} dissimilarity(O_i, rep(M, O_i))}{n}$$

where, M - set of selected medoids,
dissimilarity(Oi, Oj) - dissimilarity between objects Oi and Oj.
rep(M, Oi) - returns a medoid in M which is closest to Oi.

CLARA draws multiple samples of data set, applies **PAM** on each sample and gives the best clustering output.

Partitioning Around Medoids (PAM) algorithm :

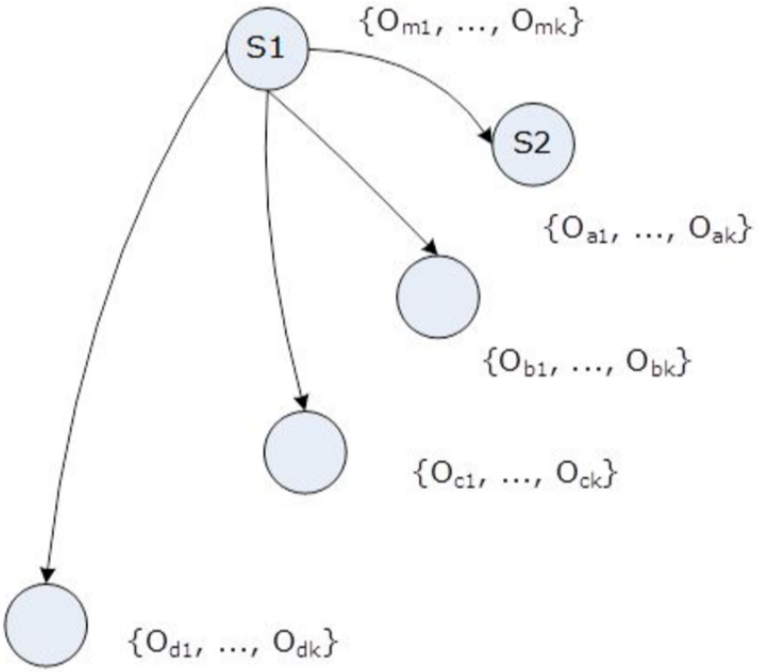
1. Initialize : select $\mathbf{k}$ of the $\mathbf{n}$ data points as the medoids
2. Associate each data point to the closest medoid.
3. While the cost of the configuration decreases:
    1. For each medoid $\mathbf{m}$, for each non-medoid data point $\mathbf{o}$:
        1. Swap $\mathbf{m}$ and $\mathbf{o}$, recompute the cost (sum of distances of points to their medoid).
        2. If the total cost of the configuration increased in the previous step, undo the swap.

Since CLARA adopts a sampling approach, the quality of its clustering results depends greatly on the size of the sample. When the sample size is small, CLARA's efficiency in clustering large data sets comes at the cost of clustering quality.

5. CLARANS (CLARA Based on Randomized Search).

In this section, we will present our clustering algorithm– CLARANS (Clustering Large Applications based on RANdomized Search). We will first give a graph-theoretic framework within which we can compare PAM and CLARA, and motivate the development of CLARANS. Then, after describing the details of the algorithm, we will present experimental results showing how to fine tune CLARANS and that CLARANS outperforms CLARA and PAM in terms of both efficiency and effectiveness.

Given n objects, the process described above of finding k medoids can be viewed abstractly as searching through a certain graph. In this graph, denoted by $G_{n,k}$, a node is represented by a set of k objects { O_{m1} , ... , O_{mk} }, intuitively indicating that { O_{m1} , ... , O_{mk} } are the selected medoids. The set of nodes in the graph is the set { { O_{m1} , ... O_{mk} } | O_{m1} ; ... ; O_{mk} are objects in the data set }



.

It is easy to see that each node has k (n - k) neighbors. Since a node represents a collection of k medoids, each node corresponds to a clustering. Thus, each node can be assigned a cost that is defined to be the total dissimilarity between every object and the medoid of its cluster.

Like CLARA, CLARANS does not check every neighbor of a node. But, unlike CLARA, it does not restrict its search to a particular subgraph. In fact, it searches the original graph $G_{n,k}$. One key difference between CLARANS and PAM is that the former only checks a sample of the neighbors of a node. But, unlike CLARA, each sample is drawn dynamically in the sense that no nodes corresponding to particular objects are eliminated outright. In other words, while CLARA draws a sample of nodes at the beginning of a search, **CLARANS draws a sample of neighbors in each step of a search**. This has the benefit of not confining a search to a localized area.

Algorithm CLARANS :

Step 1. Input parameters **num_local** and **max_neighbor.** Initialize i to 1, and mincost to a large number.

Step 2. Set current to an arbitrary node in $G_{n,k}$.

Step 3 . Set j to 1.

Step 4 . Consider a random neighbor S of current, and based on 5, calculate the cost differential of the two nodes.

Step 5 . If S has a lower cost, set current to S, and go to Step 3 else to Step 6.

Step 6 . Otherwise, increment j by 1. If j max_neighbor, go to Step 4.

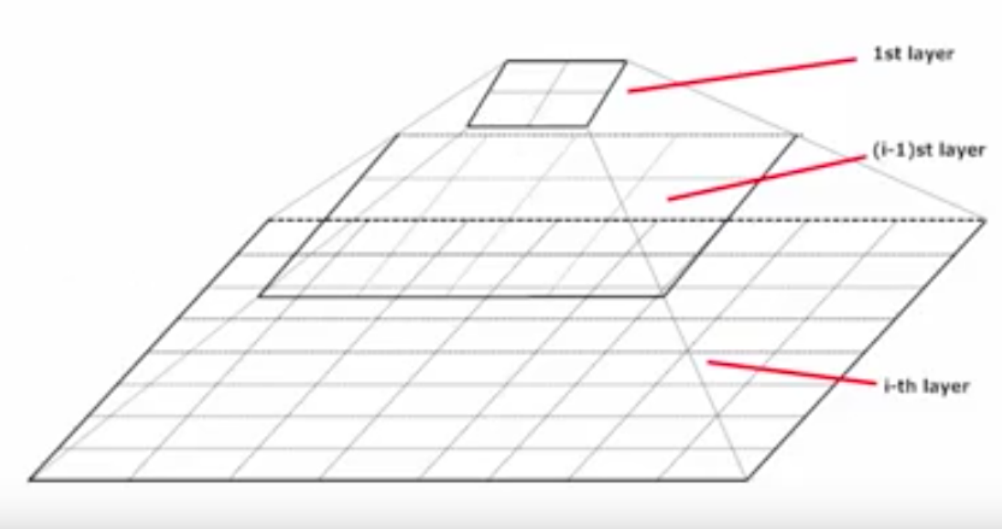
Step 7. Otherwise, when j > max_neighbor, compare the cost of current with mincost. If the former is less than mincost, set mincost to the cost of current and set best_node to current.

Step 8 . Increment i by 1. If i > num_local, output best_node and halt. Otherwise, go to Step 2.

Steps 3 to 6 above search for nodes with progressively lower costs. But, if the current node has already been compared with the maximum number of the neighbors of the node (specified by max_neighbor) and is still of the lowest cost, the current node is declared to be a “local” minimum. Then, in Step 7, the cost of this local minimum is compared with the lowest cost obtained so far. The lower of the two costs above is stored in mincost. Algorithm CLARANS then repeats to search for other local minima, until num_local of them have been found.

6. STING (Grid Based Multi-resolution Clustering).

In grid-based approach called STING (STatistical INformation Grid) to spatial data mining. The spatial area is divided into rectangular cells. We have several different levels of such rectangular cells corresponding to different resolution and these cells form a hierarchical structure. Each cell at a high level is partitioned to form a number of cells of the next lower level. Statistical information of each cell is calculated and stored beforehand and is used to answer queries.



For each cell, we have **attribute-dependent** and **attribute-independent parameters**. The **attribute independent parameter** is:

- **n** number of objects (points) in this cell.

As for the **attribute-dependent parameters**, we assume that for each object, its attributes have numerical values.

For each numerical attribute, we have the following five parameters for each cell:

- **m** mean of all values in this cell
- **s** standard deviation of all values of the attribute in this cell
- **min** the minimum value of the attribute in this cell
- **max** the maximum value of the attribute in this cell
- **distribution** the type of distribution that the attribute value in this cell follows.

The parameter distribution is of enumeration type. Potential distribution types are: normal, uniform, exponential, and so on. The value NONE is assigned if the distribution type is unknown. The distribution type will determine a “kernel” calculation in the generic algorithm.

Parameter Generation:

We generate the hierarchy of cells with their associated parameters when the data is loaded into the database. Parameters n, m, s, min, and max of bottom level cells are calculated directly from data.

The value of distribution could be either assigned by the user if the distribution type is known beforehand or obtained by hypothesis tests such as χ 2 -test. Parameters of higher level cells can be easily calculated from parameters of lower level cell.

Let n, m, s, min, max, dist be parameters of current cell and n_i , m_i , s_i , min_i , max_i , and $dist_i$ be parameters of corresponding lower level cells, respectively.

The n, m, s, min, and max can be calculated as follows :

$$n = \sum_i n_i$$

$$m = \frac{\sum_i m_i n_i}{n}$$

$$s = \sqrt{\frac{\sum_i (s_i^2 + m_i^2) n_i}{n} - m^2}$$

$$min = \min_i (min_i)$$

$$max = \max_i (max_i)$$

The determination of dist for a parent cell is a bit more complicated. First, we set dist as the distribution type followed by most points in this cell. This can be done by examining $dist_i$ and n_i . Then, we estimate the number of points, say confl, that conflict with the distribution determined by dist, m, and s according to the following rule:

1. If $dist_i \neq dist$, $m_i \approx m$ and $s_i \approx s$, then confl is increased by an amount of n_i.
2. If $dist_i \neq dist$, but either $m_i \approx m$ or $s_i \approx s$ is not satisfied, then set confl to n.
   (This enforces dist will be set to NONE later);
3. If $dist_i = dist$, $m_i \approx m$ and $s_i \approx s$, then confl is not changed.
4. If $dist_i = dist$, but either $m_i \approx m$ or $s_i \approx s$ is not satisfied, then confl is set to n.

Finally, if **confl/n** is greater than a threshold **t** (This threshold is a small constant, say 0.05, which is set before the hierarchical structure is built), then we set dist as NONE; otherwise, we keep the original type.

Starting with the root, we calculate the likelihood that this cell is relevant to the query at some confidence level using the parameters of this cell (exactly how this is computed is described later). This likelihood can be defined as the proportion of objects in this cell that satisfy the query conditions. (If the distribution type is NONE, we estimate the likelihood using some distribution-free techniques instead.) After we obtain the confidence interval, we label this cell to be relevant or not relevant at the specified confidence level. When we finish examining the current layer, we proceed to the next lower level of cells and repeat the same process. The only difference is that instead of going through all cells, we only look at those cells that are children of the relevant cells of the previous layer. This procedure continues until we finish examining the lowest level layer (bottom layer).

Statistical Information Grid-based Algorithm:

Step 1 : Determine a layer to begin with.

Step 2 : For each cell of this layer, we calculate the confidence interval (or estimated range) of probability that this cell is relevant to the query.

Step 3 : From the interval calculated above, we label the cell as relevant or not relevant.

Step 4 : If this layer is the bottom layer, go to Step 6; otherwise, go to Step 5.

Step 5 : We go down the hierarchy structure by one level. Go to Step 2 for those cells that form the relevant cells of the higher level layer.

Step 6 : If the specification of the query is met, go to Step 8; otherwise, go to Step 7.

Step 7 : Retrieve those data fall into the relevant cells and do further processing. Return the result that meet the requirement of the query. Go to Step 9.

Step 8 : Find the regions of relevant cells. Return those regions that meet the requirement of the query. Go to Step 9.

Step 9 : Stop.

Refrences :

1. Paliwal, Priyamvada, and Meghna Sharma. "Enhanced DBSCAN Algorithm." International Journal of Advanced Research in Computer Science and Software Engineering (2013).
2. Ng, Raymond T., and Jiawei Han. "CLARANS: A method for clustering objects for spatial data mining." IEEE transactions on knowledge and data engineering 14.5 (2002): 1003-1016.
3. Wang, Wei, Jiong Yang, and Richard Muntz. "STING: A statistical information grid approach to spatial data mining." *VLDB*. Vol. 97. 1997.
4. Saha, Moumita. "A Graph Based Approach to Multiview Clustering." *International Conference on Pattern Recognition and Machine Intelligence*. Springer, Berlin, Heidelberg, 2013.