Q2

Understand the working of BIRCH and DBSCAN clustering algorithms. Give a pseudocode and trace the same over a sample dataset of your choice.

1. BIRCH Clustering

Introduction:

**Balanced Iterative Reducing and Clustering using Hierarchies**, or **BIRCH** for short, deals with large datasets by first generating a more compact summary that retains as much distribution information as possible, and then clustering the data summary instead of the original dataset. BIRCH actually complements other clustering algorithms by virtue if the fact that different clustering algorithms can be applied to the summary produced by BIRCH. BIRCH can only deal with metric attributes (similar to the kind of features KMEANS can handle). A metric attribute is one whose values can be represented by explicit coordinates in an Euclidean space (no categorical variables).

Pseudo Code:

Take the initial Data,

* **Phase 1:** Load data into memory

Scan DB and load data into memory by building a CF tree. If memory is exhausted rebuild the tree from the leaf node.

* **Phase 2:** Condense data

Resize the data set by building a smaller CF tree

Remove more outliers

Condensing is optional

* **Phase 3:** Global clustering

Use existing clustering algorithm (e.g. KMEANS, HC) on CF entries

* **Phase 4:** Cluster refining

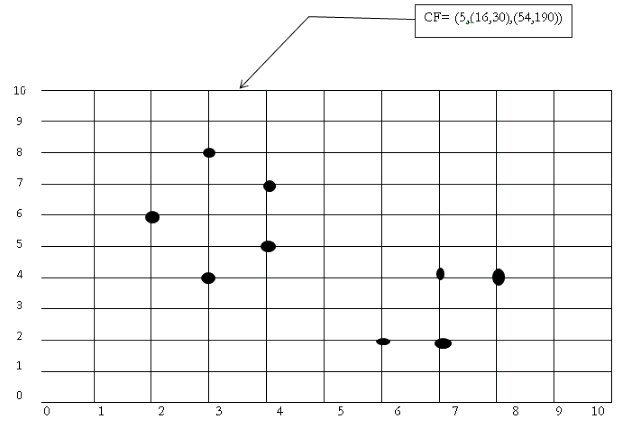
Refining is optional

Fixes the problem with CF trees where same valued data points may be assigned to different leaf entries.

Trace:

CF= (N, LS, SS)

Example Trace



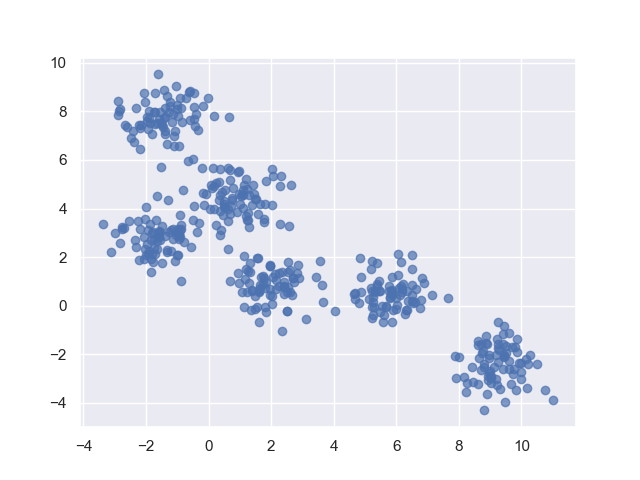
(3,4) (2,6)(4,5)(4,7)(3,8)

N=5

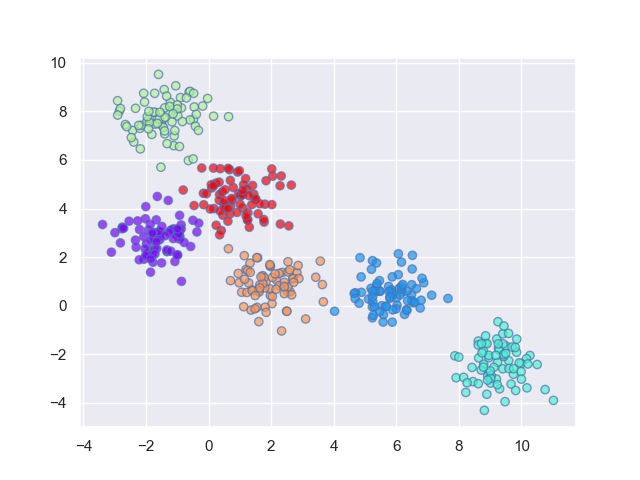
NS= (16, 30 ) i.e. 3+2+4+4+3=16 and 4+6+5+7+8=30

SS=(54,190)=32+22+42+42+32=54  and  42+62+52+72+82= 190

Input Data



After Clustering,



1. DBSCAN Clustering

Introduction:

Fundamentally, all clustering methods use the same approach i.e. first we calculate similarities and then we use it to cluster the data points into groups or batches. Here we will focus on **Density-based spatial clustering of applications with noise** (DBSCAN) clustering method.

Clusters are dense regions in the data space, separated by regions of the lower density of points. The ***DBSCAN algorithm*** is based on this intuitive notion of “clusters” and “noise”. The key idea is that for each point of a cluster, the neighborhood of a given radius has to contain at least a minimum number of points.

**DBSCAN algorithm requires two parameters –**

1. **eps** : It defines the neighborhood around a data point i.e. if the distance between two points is lower or equal to ‘eps’ then they are considered as neighbors. If the eps value is chosen too small then large part of the data will be considered as outliers. If it is chosen very large then the clusters will merge and majority of the data points will be in the same clusters. One way to find the eps value is based on the ***k-distance graph***.
2. **MinPts**: Minimum number of neighbors (data points) within eps radius. Larger the dataset, the larger value of MinPts must be chosen. As a general rule, the minimum MinPts can be derived from the number of dimensions D in the dataset as, MinPts >= D+1. The minimum value of MinPts must be chosen at least 3.

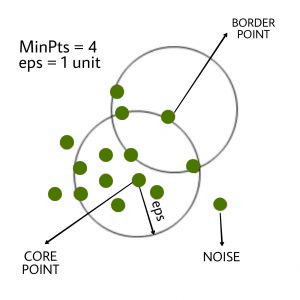
***In this algorithm, we have 3 types of data points.***

***Core Point****: A point is a core point if it has more than MinPts points within eps.****Border Point****: A point which has fewer than MinPts within eps but it is in the neighborhood of a core point.****Noise or outlier****: A point which is not a core point or border point.*

Pseudo Code:

1. Find all the neighbor points within eps and identify the core points or visited with more than MinPts neighbors.
2. For each core point if it is not already assigned to a cluster, create a new cluster.
3. Find recursively all its density connected points and assign them to the same cluster as the core point.  
   A point*a* and *b* are said to be density connected if there exist a point *c* which has a sufficient number of points in its neighbors and both the points*a* and *b* are within the *eps distance*. This is a chaining process. So, if *b* is neighbor of *c*, *c* is neighbor of*d*, *d* is neighbor of *e*, which in turn is neighbor of *a* implies that *b* is neighbor of*a*.
4. Iterate through the remaining unvisited points in the dataset. Those points that do not belong to any cluster are noise.

Trace:



Here, first we identify core points and as shown in diagram,

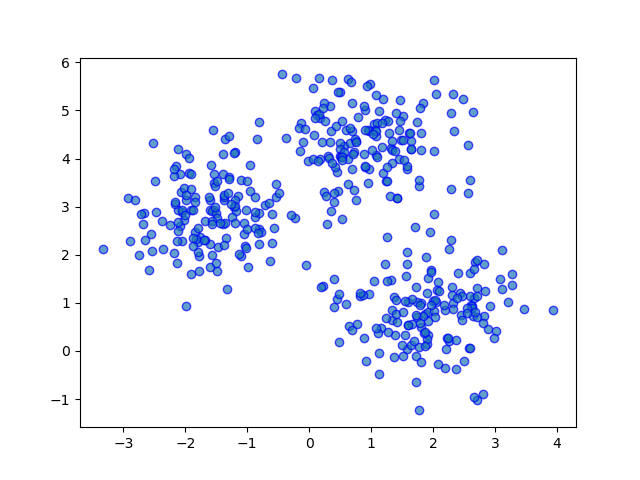
The marked point is a core point as it has > 4 neighbors within eps 1 unit.

Hence the core point is assigned a cluster A.

Similarly, we can identify that all the points are belonging to cluster A itself as neighbors of the central core point also have > 4 neighbors within 1-unit eps thereby connecting their neighbors under same cluster as them, i.e. Cluster A.

Only that 1 point is marked noise as it has no nearby points with > 4 neighbors within 1-unit eps. Hence it does not belong to any cluster, hence noise.

Input Data



Output Clusters

