Analysis of Clustering Algorithms

-By Amey Patel and Gvs Akhil

1) Partitonal algorithms

K-Means:

The storage space required is modest , it is linear in the number of data points. Its space complexity is given by O((m+K)n) where m is the number of points and n is the number of attributes. The running time of the algorithm is O[ikmn] where i the number of iterations needed until convergence.

It cannot handle clusters of different sizes and densities if the no. of clusters chosen is small. It however fails to give effective results when there are outliers in the data sets. The performance of the algorithm can drop due to their presnece.Hence , it becomes essential to identify the outliers and eliminate them. Another shortcoming of the simple algorithm is its restriction to data having a notion of a centroid.

K-Medians:

The median is computed in each single dimension in the Manhattan-distance formulation of the *k*-medians problem.This ensures that the individual attributes come from the dataset.This makes the algorithm more reliable for discrete or even binary data sets.
The algorithm itself is similar to the K-Means algorithm save for the computation of the center of each cluster. The K-Means algorithm computes the mean of the samples in each cluster whereas the K-Medians opts to compute the median of the samples in a cluster.Hence, the center calculated in the K-Medians always is itself a memeber of the cluster.This is not always true for the K-Means algorithm.
K-Medians is robust to outliers and results in compact clusters.

K-Medoids:

K-means attempts to minimize the total suared error ,while *k*-medoids minimizes the sum of dissimilarities between points labeled to be in a cluster and a point designated as the center of that cluster.

The complexity of $O(k(n-k)^2)$.where n is the number of d-dimensional vectors, k the number of clusters .
It could be more robust to noise and outliers as compared to K-Means because it minimizes a sum of general pairwise dissimilarities instead of a sum of squared Euclidean distances.

2) Density-Based Algorithms

DBSCAN:

The space complexity of the DBSCAN algorithm is O(m).
The time complexity of the DBSCAN algorithm is $O(m^2)$ in the worst case.However using stuctures such as KD-Trees allows for a quicker retrieval of points that lie within a circle of certain radius around a specific chosen point.This can reduce the time complexity to O(mlogm),which is a drastic improvement on the initial complexity of $O(m^2)$.

The DBSCAN algorithm fails to give efficient results when if the density of the involved clusters vary widely.It is relatively resistant to noice and can handle outliers and clusters of arbitaty size and shape. It can segregate points in a data set into clusters that K-Means might find difficult to.It is usually not used in the computation of higher-deminsional data as in these cases, the definition of density of data points can become harder.Its results ,however, for noise handling with increasing dimensionality gets worse.

DENCLUE

It is a generalisarion of DBSCAN and K-Means.The clusters formed are of arbitrary shape ans size and hence provides more flexibility.
The time complexity of DENCLUE is given by O(log|D|).,where D is the number of the points within a specified radius r(epsilon).
The performance of DENCLUE is appealing in low dimensional space, however, it does not work well as the dimensionality increase or if noise is present.
Scenario noise and outliers are not of concern and still quality and time matters then DBSCAN is appreciable. Where run time is most important factor there DENCLUE will be the best option.

3) Hierarchial Algorithms:

BIRCH:

Scales linearly: It generally produces a good clustering in a single scan .However, a few more scans improve its quality drastically. The computation complexity of the algorithm is O(n), where n is number-objects.
The Birch algorithm handles only numeric data, and is sensitive to the order of the data record.

CURE:

The Cure algorithm when implemented using heaps and K-d trees require linear storage space i.e it's space complexity is)(n).
The worst-case complexity of our clustering algorithm is $O(n^2 \log n)$. When the dimensionality of the points is small , it reduces to $O(n^2)$.

Chameleon:

The computational cost for high-dimensional data may require $O(n^2)$time for n objects in the worst case.
Chameleon performances really well when it comes to discovering arbitrarily shaped clusters of high quality than several well-known algorithms such as BIRCH and density based DBSCAN.

4) Grid-Based Algorithms

Wavelet clustering:

It is a combination of grid-based and density-based clustering algorithm.It handles large data sets efficiently, discovers clusters with arbitrary shape, successfully handles outliers, is insensitive to the order of input, and does not require the

specification of input parameters such as the number of clusters or a neighborhood radius.
Wavelet-based clustering is very fast, with a computational complexity of O(n), where n is the number of objects in the database.
Hence, it is a very important algorithm and is often used.

STING:

It is a grid-based algorithm.It provides parallel processing and incremental updating.It has high efficiency.
The time complexity of generating clusters is O(n), where n is the total number of objects. After generating the hierarchical structure, the query processing time is O(g), where g is the total number of grid cells at the lowest level.

5) Model-Based Algorithms

COBWEB:

This is a model-based algorithm which is efficient in producing trees with reasonable number of classes. It is usually used on data sets that are small in size. They are inefficient in the handling of outliers. The time complexity of the algorithm is $O(n^2)$

Categorization of clustering algorithms

Category	Algorithm name	Size of Dataset	Type of Dataset	Run-Time Complexity
Partitional	K-Means	Large	Numerical	O(nkd)
	K-Medians	Large	Numerical	O(nkd)
	K-Mediods	Small	Categorical	$O(n^2dt)$
Density-Based	DBSCAN	Large	Numerical	O(nlogn)
	DENCLUE	Large	Numerical	O(log|D|)
Hierarchial	BIRCH	Large	Numerical	O(n)
	CURE	Large	Numerical	$O(n^2logn)$
	Chameleon	Large	All types of data	$O(n^2)$
Grid-Based	Wavelet	Large	Special data	O(n)
	STING	Large	Special data	O(k)
Model-Based	COBWEB	Small	Numerical	$O(n^2)$

SOURCES

- http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=D2A6731CE0FAD4066F51833BAD6BCD89?doi=10.1.1.278.5025&rep=rep1&type=pdf
- https://www.ijcsi.org/papers/IJCSI-12-2-62-85.pdf
- https://www.cise.ufl.edu/~jmishra/clustering/ClusteringPresentation
- http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.403.7600&rep=rep1&type=pdf
- https://www.cis.upenn.edu/~sudipto/mypapers/cure_final.pdf

- A Survey of Clustering Algorithms for Big Data: Taxonomy and Empirical Analysis