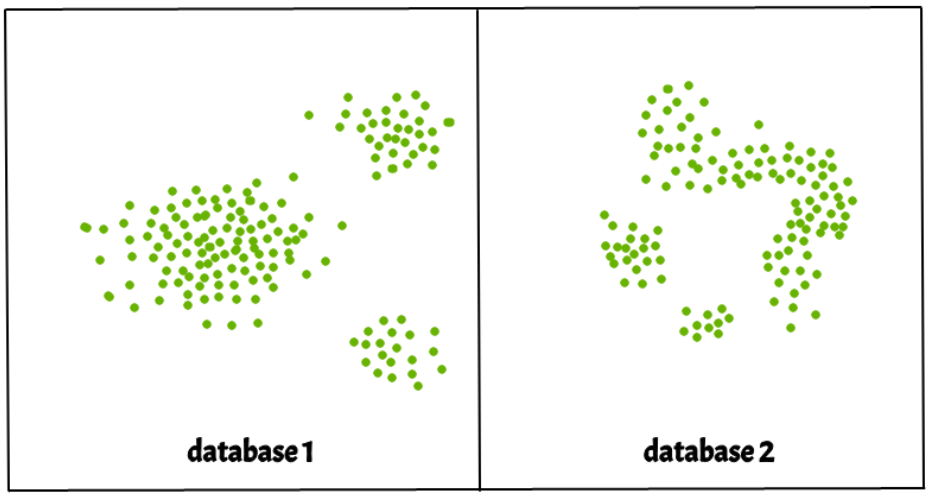
DBSCAN Clustering in ML | Density based clustering

Clustering analysis or simply Clustering is basically an Unsupervised learning method that divides the data points into a number of e data points in the same groups have similar properties and data points in different groups have different properties in some sense. It comprises of many different methods based on different evolution.

E.g. K-Means (distance between points), Affinity propagation (graph distance), Mean-shift (distance between points), DBSCAN (distance between nearest points), Gaussian mixtures (Mahalanobis distance to centers), Spectral clustering (graph distance) etc.

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.



Why DBSCAN ?

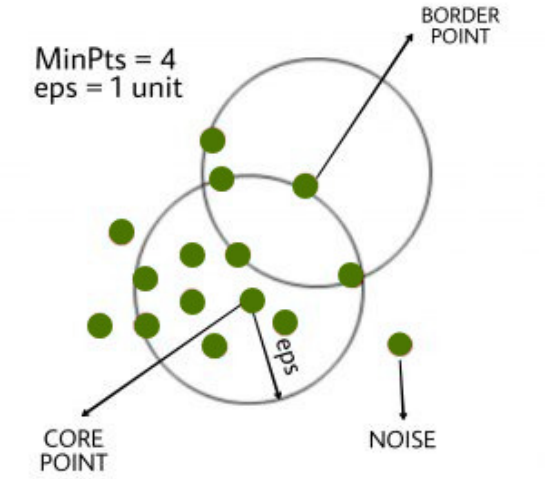
Partitioning methods (K-means, PAM clustering) and hierarchical clustering work for finding spherical-shaped clusters or convex clusters. In other words, they are suitable only for compact and well-separated clusters. Moreover, they are also severely affected by the presence of noise and outliers in the data.

Real life data may contain irregularities, like –

i) Clusters can be of arbitrary shape such as those shown in the figure below.

ii) Data may contain noise.

***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.*



DBSCAN algorithm can be abstracted in the following steps –

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.

Below is the DBSCAN clustering algorithm in pseudocode:

```
DBSCAN(dataset, eps, MinPts){
# cluster index
C = 1
for each unvisited point p in dataset {
         mark p as visited
         # find neighbors
         Neighbors N = find the neighboring points of p

         if |N|>=MinPts:
             N = N U N'
             if p' is not a member of any cluster:
```

The figure below shows a data set containing nonconvex clusters and outliers/noises. Given such data, k-means algorithm has difficulties for identifying these clusters with arbitrary shapes.

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.

```
                add p' to cluster C
 }
```

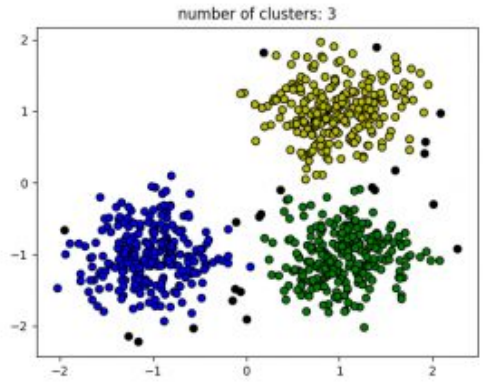
Implementation of above algorithm in Python :

Here, we'll use the Python library `sklearn` to compute DBSCAN. We'll also use the `matplotlib.pyplot` library for visualizing clusters.

The dataset used can be found **here**.

```
import numpy as np
from sklearn.cluster import DBSCAN
from sklearn import metrics
from sklearn.datasets.samples_generator import make_blobs
from sklearn.preprocessing import StandardScaler
from sklearn import datasets

# Load data in X
db = DBSCAN(eps=0.3, min_samples=10).fit(X)
core_samples_mask = np.zeros_like(db.labels_, dtype=bool)
core_samples_mask[db.core_sample_indices_] = True
labels = db.labels_

# Number of clusters in labels, ignoring noise if present.
n_clusters_ = len(set(labels)) - (1 if -1 in labels else 0)

print(labels)

# Plot result
import matplotlib.pyplot as plt

# Black removed and is used for noise instead.
unique_labels = set(labels)
colors = ['y', 'b', 'g', 'r']
print(colors)
for k, col in zip(unique_labels, colors):
    if k == -1:
        # Black used for noise.
        col = 'k'

    class_member_mask = (labels == k)

    xy = X[class_member_mask & core_samples_mask]
    plt.plot(xy[:, 0], xy[:, 1], 'o', markerfacecolor=col,
                                      markeredgecolor='k',
                                      markersize=6)

    xy = X[class_member_mask & ~core_samples_mask]
    plt.plot(xy[:, 0], xy[:, 1], 'o', markerfacecolor=col,
                                      markeredgecolor='k',
                                      markersize=6)

plt.title('number of clusters: %d' %n_clusters_)
plt.show()
```

Output:

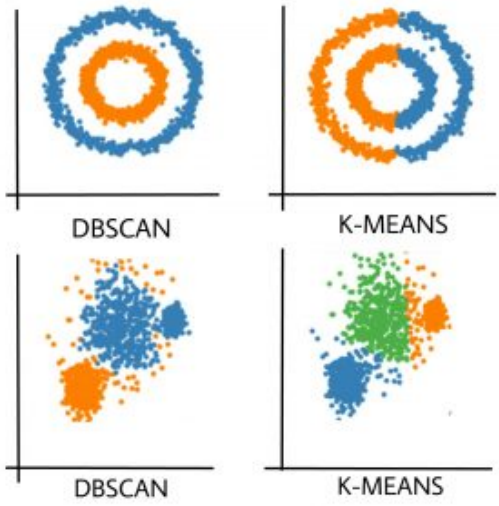


Black points represent outliers. By changing the *eps* and the *MinPts* , we can change the cluster configuration.

Now the question should be raised is – *Why should we use DBSCAN where K-Means is the widely used method in clustering analysis?*

Disadvantage Of K-MEANS:

1. K-Means forms spherical clusters only. This algorithm fails when data is not spherical (i.e. same variance in all directions).



2. K-Means algorithm is sensitive towards outlier. Outliers can skew the clusters in K-Means in very large extent.

ML | Mean-Shift Clustering

ML | Fuzzy Clustering

Clustering in R Programming

ML | BIRCH Clustering

ML | K-Medoids clustering with example

ML | Spectral Clustering

K means Clustering - Introduction

Different Types of Clustering Algorithm

Clustering in Machine Learning

Criterion Function Of Clustering

ML | OPTICS Clustering Explanation

K-Means Clustering in R Programming

Hierarchical Clustering in R Programming

ML | Types of Linkages in Clustering

ML | Unsupervised Face Clustering Pipeline

Implementing Agglomerative Clustering using Sklearn

Debomit Dey
Check out this Author's contributed articles.

If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please Improve this article if you find anything incorrect by clicking on the "Improve Article" button below.

Article Tags : Machine Learning Python data-science

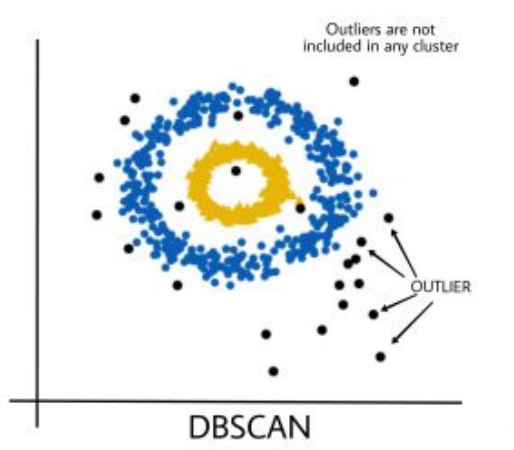
Practice Tags : Machine Learning

2

3.6

Based on **3** vote(s)

☐ To-do ☐ Done



3. K-Means algorithm requires one to specify the number of clusters a priory etc.

Basically, DBSCAN algorithm overcomes all the above-mentioned drawbacks of K-Means algorithm. DBSCAN algorithm identifies the dense region by grouping together data points that are closed to each other based on distance measurement.

Python implementation of above algorithm without using the sklearn library can be found here dbscan_in_python.

References :
https://en.wikipedia.org/wiki/DBSCAN
https://scikit-learn.org/stable/auto_examples/cluster/plot_dbscan.html

Recommended Posts:

Difference between CURE Clustering and DBSCAN Clustering
DBScan Clustering in R Programming
ML | Hierarchical clustering (Agglomerative and Divisive clustering)
ML | Classification vs Clustering

Feedback/ Suggest Improvement | Improve Article

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments