DBSCAN Clustering Density based clustering

Density-based Spatial Clustering of Applications with Noise (DBSCAN) clustering method.

https://www.geeksforgeeks.org/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 specific batches or groups, such that the 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.

Importing Libraries

```
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
import numpy as np
import csv
```

Loading the data

```
# Load data in X
with open("dataset.csv", 'r') as f:
    X = list(csv.reader(f, delimiter=","))

X = np.array(X[1:], dtype=np.float)
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

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

print(labels)
```

```
[ 0  1  2  1  0  0  2  1  1  0  0  0  2  0  1 -1  0  0  2  2  2  2  2  0
  0  2  1  1  2  1  0  0  1  0  1  2  1  1 -1  2  0  0  0  0  0  1  2  1
  0  2  2  0  0  2  2  0  1  2  0  2  2  2  2  2  1  2  2  1  1  1  2  1
  1  2  0 -1  0  1  2  0  0  1  1  1  1  0  2  0  2  2  1  0  1  0 -1  0
  0  1  1  2  0  2  1  2  2  2  2 -1  1 -1  0  0  0  0  1  1  0  1  0  2
  0  1  1  0  2  0  1  1  2  1  2  2  2  1 -1  2  2  1  0  1  2  1  1  2
  2 -1  2  0 -1  2  0  0  2  2  2 -1  0  1  0  1  0  1  2  2 -1  0  2  2
  0  1  0  2  2  2  0  0  2  2  1  0  2  1  1  2  1  1  0  1  0  1  0  0
  2  2  1  1  0  0  2  0  2  2  2  2  1  2  1  2  2  1  2  2  2  1  1  0
  0  0  2  2  2  2  0  2  2  1  1  2  1  1  1  0  1  0  0  0  2  0  0  1
  0  2  2  0  2  2  0  1  1  0  0  0  1  0  1  2  1  2  2  2  2  2  0  0
  1  1  0  0  1  1  2  0 -1  2  0  0  2  0  2  1  2 -1  1  0  2  2  1  2
  2  1  1  2  1  2  1  2  0  1  1  1  0  2  0  2  2  1  2  2  1  1  2  0
  0  0  0  0  1  0  0  0  0  1  1  0  0  0  1  2  1  0  2  2  1  1  2  1
  2  0  1  2  1  2  1  2  2  1  0  1  0  1  2  2  0  0  0  2  1  2  1  2
  0  2  2  1  0  1  0  1  1  1  1  2  1  2  1  0  1  0  2  0  0  0  1  0
  0  1  2  0  1  2  2  0  0  2  2  2  0  2  0  2  1  2  0  2  0  1  0  1
  0  0  1  0  2 -1  0  1  1  2  0  2  2  2  2  0  1  1  1  1  0  1  2  0
  1  0  2  1  1  0  1  0  0  1 -1  1  2  2  2  0  0  2  1  0  1  1  0  1
  0  0  2  2 -1  1  0  2  2  0  0  0  0  1  1  1  2  2  0  2  0  1  1  0
  2  0  1  1  2  1  0  1  2  0  1  2  2  0  1  1  1  2  0  0  1  2  1  1
  0  0  0  0  1  0  1  0  1  1  2  1  0  0  2  0  0  1  0  1  2  0  1  1
  0  1  0  0  2  2  0  2  2  0  2  0  0  0  0  2  1  1  1  0  2  2  1  2
  1  2  0  1  0  0  1  1  0  2  0  2  2  1  2  0  0  0  2  1  1  2  1  2
  2  1  2  1  0  0  0  0  1  1  1  2  0  0  0  0  2  2  2  1  2  0  0  1
  1  0  1  2  0  2  0  1  2  2  1  1  0  1  1  2  1  1  1  2  1  2  1  1
  0  0  1  1  0  2  2  1  1  1  1  2 -1  0  0  2  0  1  1  2  2  1  0  2
  1  0  2  2  0  1  1 -1 -1  2  1  1  1  2 -1  2  1  0  0  0  0  0  1  1
  2  0  2  1  0  0  0  1  2  0  0 -1  2  0  2  1  2  2  0  1  1  1  0  0
  2  1  1  2  2  0  2  2  2  1  2  0  2  0  0  0  2  1  2  1  2  2  1  1
  2  0 -1  1  2  1  1  1  0  1  2  0  2  1  0  1  1  2  1  2  0  0  2  0
  1  0  2  0  2]
```

```
# 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',
```

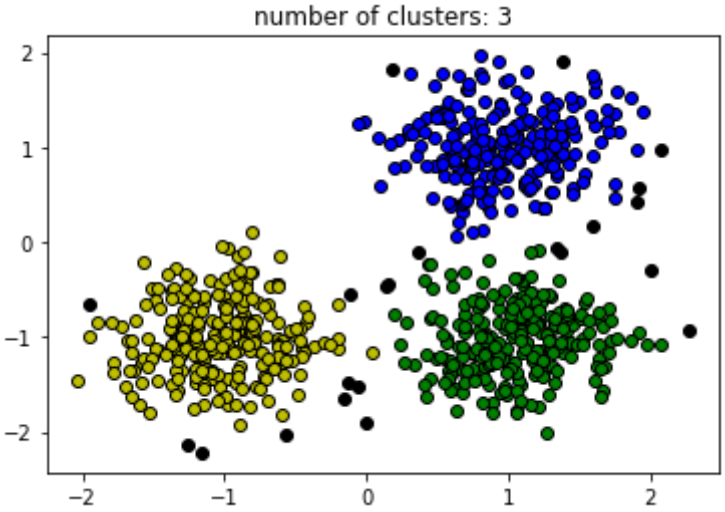
```
                    markersize=6)

  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()

```

```
['y', 'b', 'g', 'r']
```




```
1
```