Lab_5_CLustering_Part_2

September 6, 2021

1 Lab 5 : Clustering Part 2

[1]:
```
import numpy as np
import pandas as pd
import math
import matplotlib.pyplot as plt
import matplotlib
```

2 DBSCAN Algorithm

DBSCAN(Density-Based Spatial Clustering of Applications with Noise) is a commonly used unsupervised clustering algorithm. DBSCAN does not need to specify the number of clusters. It can automatically detect the number of clusters based on your input data and parameters. More importantly, DBSCAN can find arbitrary shape clusters that k-means are not able to find.

3 Algorithm:

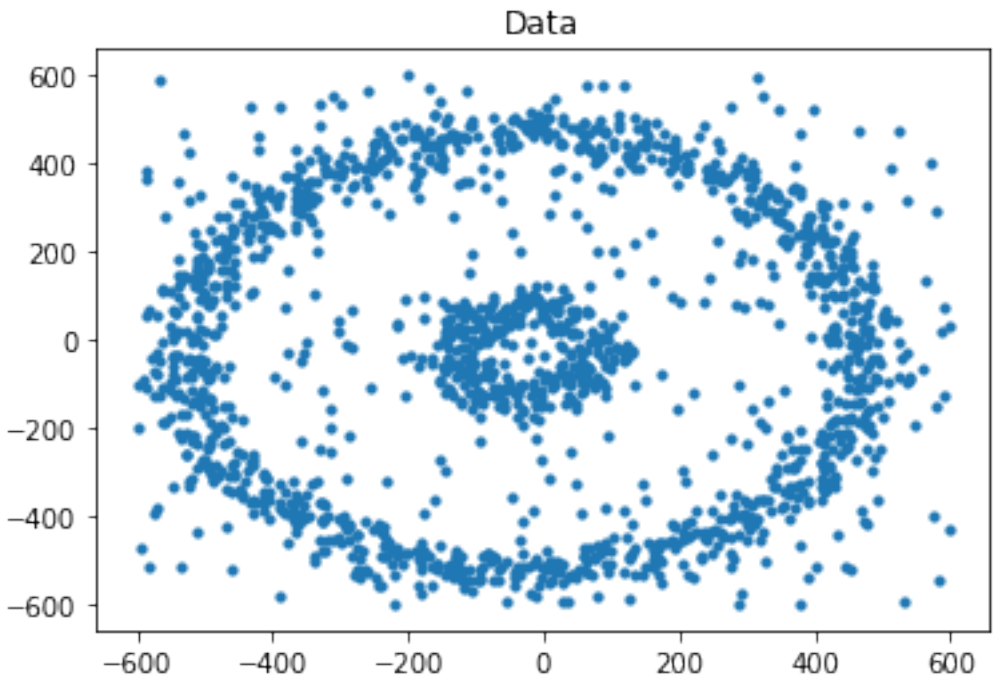
a. The algorithm proceeds by arbitrarily picking up a point in the dataset (until all points have been visited).

b. If there are at least 'minPoint' points within a radius of '' to the point then we consider all these points to be part of the same cluster.

c. The clusters are then expanded by recursively repeating the neighborhood calculation for each neighboring point

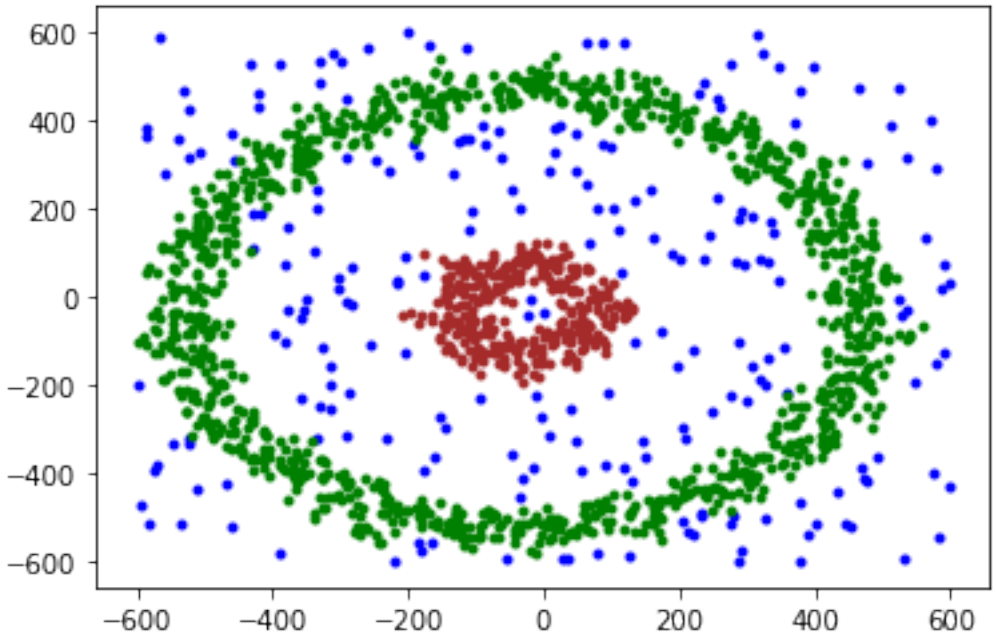
A. **Generate "N" spherical training data points.**

[4]:
```
## write your code here
```



B. **Perform DBSCAN Algorithm on the above generated data to obtain clusters**

[5]:
```
## Write your code here
```



```
number of cluster found: 2
Counter({1: 1063, 2: 311, 0: 226})
numbrer of outliers found: 226
```

C. **Experiment by varying the number of min points and epsilon radius and plot your observations**

```
[ ]: ## write your code here
```

D. **Compare your model with the built in DBSCAN in Sci-kit Learn. Also compare you results with GMM and the K-means Algorithm**

```
[ ]: from sklearn.cluster import DBSCAN
## write your code here


#########################################################################
from sklearn.mixture import GaussianMixture
## write your code here


#########################################################################
from sklearn.cluster import KMeans
## write your code here
```

4 Hierarchical Clustering

Hierarchical clustering is an unsupervised clustering technique which groups together the unlabelled data of similar characteristics.

There are two types of hierarchical clustering:

- Agglomerative Clustering
- Divisive Clustering

Agglomerative Clustering:

In this type of hierarchical clustering all data set are considered as indivisual cluster and at every iterations clusters with similar characteristics are merged to give bigger clusters. This is repeated untill one single cluster is reached. It is also called bottem-top approach.

4.1 Agglomerative Clustering:

Lets start with some dummy example :

X=$[x_1, x_2, ..., x_5]$, with

$x_1 = \begin{bmatrix}1\\1\end{bmatrix}, x_2 = \begin{bmatrix}2\\1\end{bmatrix}, x_3 = \begin{bmatrix}5\\4\end{bmatrix}, x_4 = \begin{bmatrix}6\\5\end{bmatrix}, x_5 = \begin{bmatrix}6.5\\6\end{bmatrix}$

Steps to perform Agglomerative Clustering:

1. Compute Distance matrix ($N \times N$ matrix, where N number of vectors present in the dataset): $D(a, b) = ||x_a - x_b||_2$

2. Replace the diagonal elements with inf and find the index of the minimum element present in the distance matrix (suppose we get the location (l,k)).
3. Replace $x_{min(l,k)} = .5 \times [x_l + x_m]$ and delete $x_{max(l,m)}$ vector from X(i.e now $(N = N-1)$),

repeat from step 1 again untill all the vectors combined to a single cluster.

[6]:
```
def Euclidian_Dist(x,y):

  return ## write your code here

def Dist_mat(X):
 ## write your code here

  return dist_mat

def combine(X):
  ## write your code here

  return newX
```

[7]:
```
X=np.array([[1,1],[2,1],[5,4],[6,5],[6.5,6]])
X=X.transpose()

## write your code here

## validate from inbuilt Dendogram
import plotly.figure_factory as ff

lab=np.linspace(1,X.shape[1],X.shape[1])
fig = ff.create_dendrogram(X.T, labels=lab)
fig.update_layout(width=800, height=300)
fig.show()
```

```
[[1.  2.  5.  6.  6.5]
 [1.  1.  4.  5.  6. ]]
[[inf 1.  5.  6.4 7.4]
 [1.  inf 4.2 5.7 6.7]
 [5.  4.2 inf 1.4 2.5]
 [6.4 5.7 1.4 inf 1.1]
 [7.4 6.7 2.5 1.1 inf]]
Vector of X to be combined:  [1, 2]
Mean of clusters after every iteration:
 [[1.5 5.  6.  6.5]
 [1.  4.  5.  6. ]]
[[inf 4.6 6.  7.1]
 [4.6 inf 1.4 2.5]
 [6.  1.4 inf 1.1]
 [7.1 2.5 1.1 inf]]
Vector of X to be combined:  [3, 4]
```

```
Mean of clusters after every iteration:
 [[1.5  5.   6.25]
 [1.   4.   5.5 ]]
[[inf 4.6 6.5]
 [4.6 inf 2. ]
 [6.5 2.  inf]]
Vector of X to be combined:  [2, 3]
Mean of clusters after every iteration:
 [[1.5   5.625]
 [1.    4.75 ]]
[[inf 5.6]
 [5.6 inf]]
Vector of X to be combined:  [1, 2]
Mean of clusters after every iteration:
 [[3.5625]
 [2.875 ]]
```

5 Clustering Algorithms on MNIST Digit dataset

Perform Kmeans and gmm clustering on MNIST dataset

1. Load MNIST data from the given images and labels
2. Consider any 2 classes

[12]:
```
!pip install idx2numpy
```

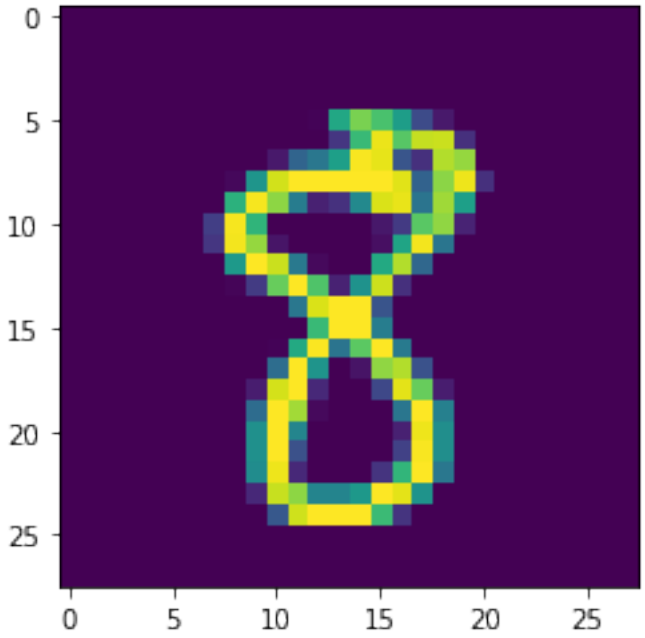
```
Collecting idx2numpy
  Downloading idx2numpy-1.2.3.tar.gz (6.8 kB)
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages
(from idx2numpy) (1.19.5)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages
(from idx2numpy) (1.15.0)
Building wheels for collected packages: idx2numpy
  Building wheel for idx2numpy (setup.py) ... done
  Created wheel for idx2numpy: filename=idx2numpy-1.2.3-py3-none-any.whl
size=7919
sha256=d62daa4e4917a1dd1f07c03e601c5693d6dcd43320a9ae08d8865794ad9a15d5
  Stored in directory: /root/.cache/pip/wheels/1a/ce/ad/d5e95a35cfe34149aade5e50
0f2edd535c0566d79e9a8e1d8a
Successfully built idx2numpy
Installing collected packages: idx2numpy
Successfully installed idx2numpy-1.2.3
```

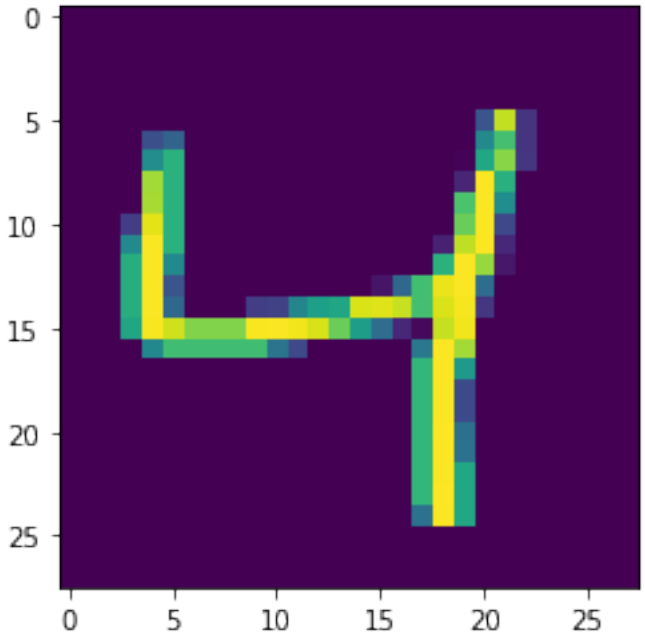
[21]:
```
import idx2numpy
from keras.utils import np_utils
img_path = ## write your code here
label_path = ## write your code here
```

```
Images = idx2numpy.convert_from_file(img_path)
labels = idx2numpy.convert_from_file(label_path)

## write your code here
```

```
(11693, 784)
(11693,)
[[0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]]
```





Use the K-means clustering algorithm from the last lab to form the clusters

[ ]:
```
## write your code here
```

Use the GMM clustering algorithm from the last lab to form the clusters

[ ]:
```
## write your code here
```