Tutorial 2 for Chapter 3

Case study 10: Mining data on user's frequent areas of activity

Reference: 数据挖掘原理与应用

For the course AMA546 Statistical Data Mining
Lecturer: Dr. Catherine Liu
AMA, PolyU, HKSAR

Contents

Objectives of the analysis

'DBSCAN.csv' is a 288×3 table, which consists of the geolocation data (x,y) of a user recorded every five minutes in a day. Every sample represents a geolocation point at a specific time.

Our **object** is to utilize the geolocations provided below to identify the areas by DBSCAN where the user is most active and summarize our findings accordingly.

Description of the data

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt# Description of the data
from sklearn.cluster import DBSCAN
```

In [2]:
```
# Read the dataset
data = pd.read_csv('/Users/ranjiang/Library/CloudStorage/OneDrive-T
                   encoding='utf8', engine='python')
```

Notes: In this tutorial, we choose the *absolute path* to import csv file. Absolute path is a hierarchical path that locates a file or folder in a file system starting from the root such as '/root/.../xxx.csv', it is accurate.

We also can choose *relative path*. Relative path describes the location of a file relative to the current (working) directory

- './' means that the current directory and the file directory are in the same directory.
- '../' means go up one level, which means the target file is at a higher directory than the current file.
- '../../' means go up two level, just equal to implement '../' twice.

In [3]:
```
data.head()
```

Out[3]:

	time	x	y
0	00:00	1.010065	1.015373
1	00:05	1.007142	1.005767
2	00:10	1.010765	1.005684
3	00:15	1.008393	1.008145
4	00:20	1.004085	1.015046

Let's see the dimension of this dataset.

In [4]:
```
data.shape
```

Out[4]:
```
(288, 3)
```

In [15]:
```
# data.rename(columns={'时间': 'time'},inplace=True)
# data.head()
```

See types of attributes.

In [6]:
```
data.dtypes
```

Out[6]:
```
time     object
x       float64
y       float64
dtype: object
```

Exploratory data analysis (based on original dataset)

In [7]:
```
data.describe()
```

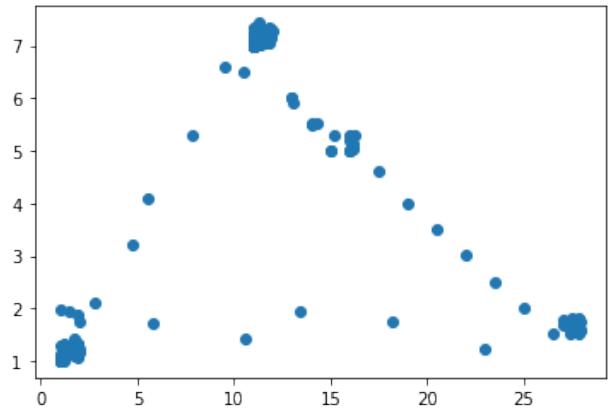
Out[7]:

	x	y
count	288.000000	288.000000
mean	8.923721	3.731517
std	8.142161	2.793233
min	1.000965	1.001381
25%	1.015438	1.013569
50%	11.003421	1.776322

75%	11.601037	7.005497
max	27.965653	7.434636

In [8]:
```
plt.figure() # Create a blank canvas
plt.scatter(
    data['x'],
    data['y'])
```

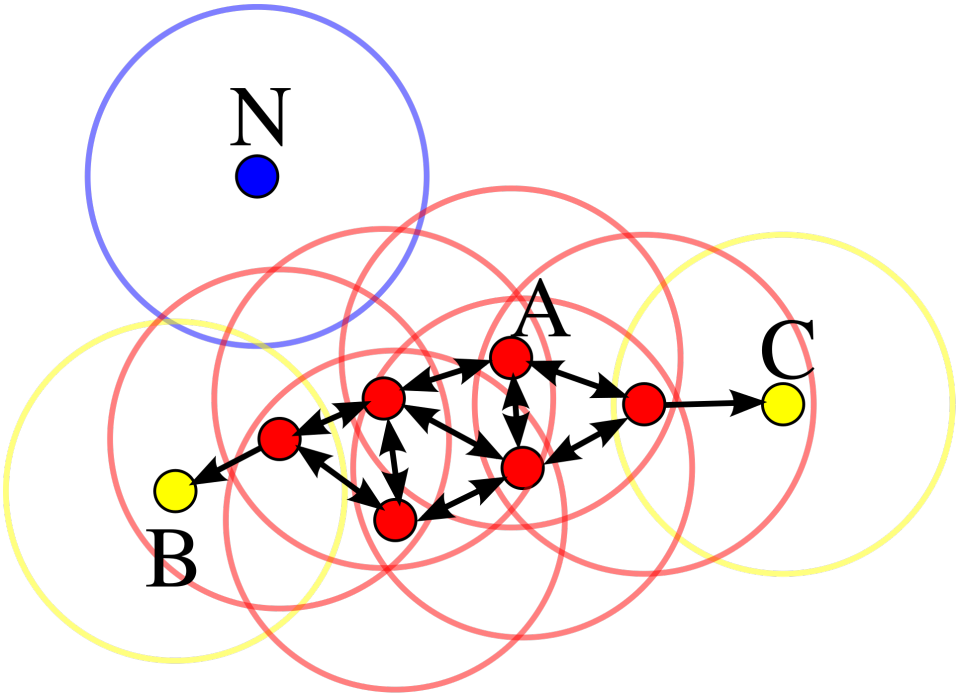
Out[8]: <matplotlib.collections.PathCollection at 0x7fe0f0c9b6d0>



By analyzing the scatter plot, we can identify the geographic locations that the user has visited, with larger clusters indicating areas that the user has frequented more often. Based on this analysis, we have found that the user has primarily visited four distinct areas, which could provide valuable insights into their travel patterns and preferences.

DBSCAN algorithm

Recall this figure:



- **MinPts**: a certain threshold that identifies different kinds of points
- **Core points**: These points are in the interior of a density-based cluster. A point is a

core point if there are at least MinPts within a distance of Eps, where MinPts and Eps are userspecified parameters.

- **Border points**: A border point is not a core point, but falls within the neighbourhood of a core point.
- **Noise points**: A noise point is any point that is neither a core point nor a border point.

In [9]:
```
#Set parameters eps and min_samples
eps = 0.5
min_samples = 5 #MinPts
```

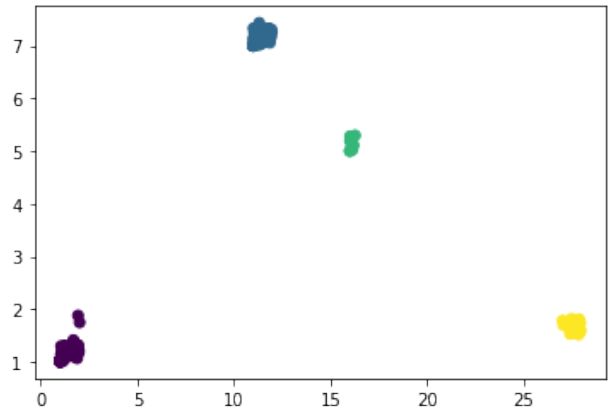
In [10]:
```
model = DBSCAN(eps=eps, min_samples = min_samples)
```

In [11]:
```
data['type'] = model.fit_predict(
    data[['x','y']]
)
data.head()
```

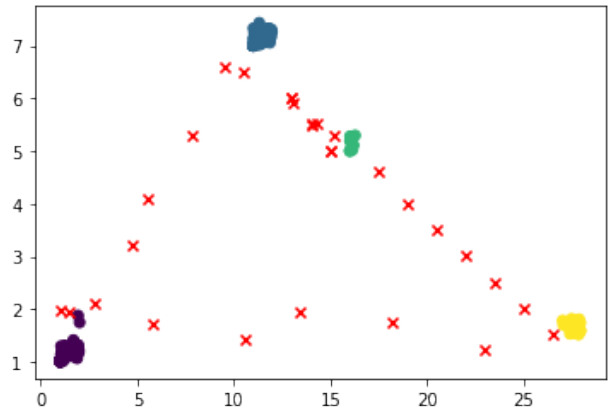
Out[11]:

	time	x	y	type
0	00:00	1.010065	1.015373	0
1	00:05	1.007142	1.005767	0
2	00:10	1.010765	1.005684	0
3	00:15	1.008393	1.008145	0
4	00:20	1.004085	1.015046	0

In [12]:
```
# Draw the clusters
plt.figure()
cluster1 = plt.scatter(
    data[data.type != -1]['x'],
    data[data.type != -1]['y'],
    c = data[data.type!=-1]['type']
)
```



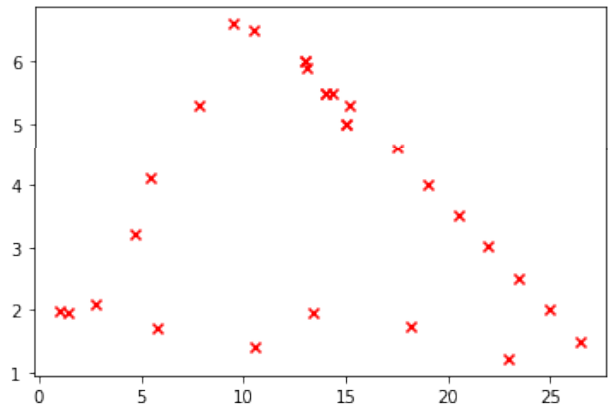
In [13]:
```
# Draw the clusters
plt.figure()
cluster1 = plt.scatter(
    data[data.type != -1]['x'],
    data[data.type != -1]['y'],
    c = data[data.type!=-1]['type']
)
# Add the noise points in red
# plt.figure()
plt.scatter(
    data[data.type == -1]['x'],
    data[data.type == -1]['y'],
    c = 'red', marker = 'x'
)
plt.show()
```



Notes: If we split cluster scatter plot part and noise scatter plot part into two cells, then noise cannot add the cluster plot directly, i.e.

In [14]:
```
plt.scatter(
    data[data.type == -1]['x'],
    data[data.type == -1]['y'],
    c = 'red', marker = 'x'
)
```

Out[14]: `<matplotlib.collections.PathCollection at 0x7fe100c2f7f0>`



Therefore, if we want to combine them, we need to put two scatter plots together in one cell.

Summary report

Objectives

The case study cluster a specific person's active regions, where we applied the **DBSCAN** model on slides '*Density and spectral*' p3-14.

Organisation of the data

The data set contains a total of 287 samples and 3 attributes/variables/features time and locations (x, y).

Exploratory data analysis´

The user's travel patterns and preferences can be deduced by analyzing the scatter plot, which displays the geographic locations they visited. Larger clusters on the scatter plot indicate frequent visits to certain areas. Our analysis shows that the user has visited mainly four distinct areas, which could provide significant insights into their travel preferences.

Model specification

The analysis objective suggested a clustering model that cluster a specific person's active regions by a common density clustering algorithm DBSCAN. There is an existing package sklearn.cluster, so we directly use this package to impelement clustering. We also can realize it manually, which has shown before.