Data Mining

Lab 02

Introduction to Clustering Techniques

Department of Computer Science
Kristianstad University
Course: DT584C
Master's in computer science

Student: Hazrat Ali <react.dev.se@gmail.com>
Teacher: Dawit Mengistu

Date: 8 January 2020

Objective

In this lab, you shall

1. Learn how to apply various clustering techniques
2. Evaluate the performance of your implementation

- Note: You may implement your own classification code from scratch or build on available tools.
- A dataset containing 200K rows is given. It contains a two attribute dataset (x,y). Each row is a point in a 2D plane. Your task is to identify the data clusters.

Task1:

- You shall apply the k-means algorithm for k=2,3,....
- Repeat the above for k-medoid.

Task2:

- Repeat the above task using your favourite hierarchical algorithms. Compare your results (accuracy) with that you found in Task1.

Report:

- How many clusters can you see in this data? (What is the plausible number of clusters?). Motivate your answer using:
    i. Mathematical analysis.
    ii. Visual analysis (e.g. plotting XY graph on Excel)
- Present a comparison of your results in the two tasks by evaluating the execution results for 1-2 and by examining the implementation of the algorithm for 3-4 below:

    i. Accuracy results
    ii. Performance (execution time)
    iii. Resource efficiency (Memory consumption)
    iv. Parallelizability
- Show your comparisons in tabular form, present each task's result as a column.

Introduction

Clustering is a technique used to segregate data points into groups such that points in one group are more similar than points in the other groups. It is a kind of unsupervised learning which means we don't have any information about the kind of labels/class of the data points in advance.

The number of groups is represented by a constant k and it can be determined by several techniques like **Elbow** or can be determined by some initial insight into the dataset.

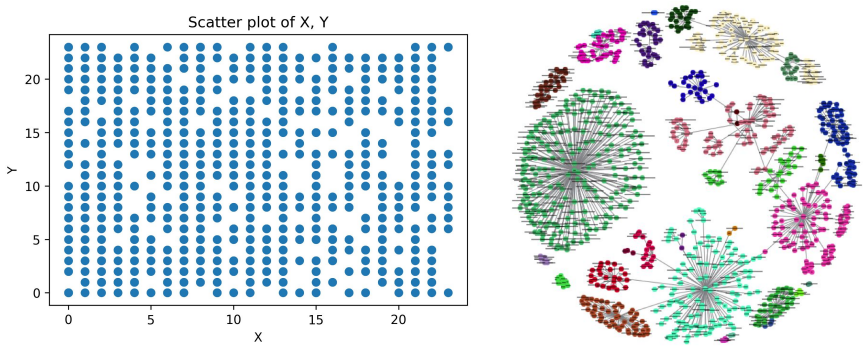


Fig. 1

In this lab we performed the two tasks and the python script can be found in the source code as task1.ipynb and task2.ipy nb respectively. In the first task we computed k-means and k-medoids for k=2,3...7. While in the second task we performed agglomerative hierarchical clustering and compared the results of the two.

Development enviroment

- Operating Systems
- Install Python3
- Anaconda
- Jupyter
- Conda Environments
- Conda install and conda forge
- Run Jupyter in a specific environment

K Means & K Medoids

In **K Means** we select the mean point to be the center or the cluster. This implies that if there are outliers then it will significantly affect the mean value.

In case of **K Medoids** we select a point among the points (with higher frequency distribution i.e mode).

The dataset has 200, 000 records and can be viewed as follows. This is a big dataset and therefore the data points are spread all over. For the sake of display we have truncated these data points and also we have scaled them.

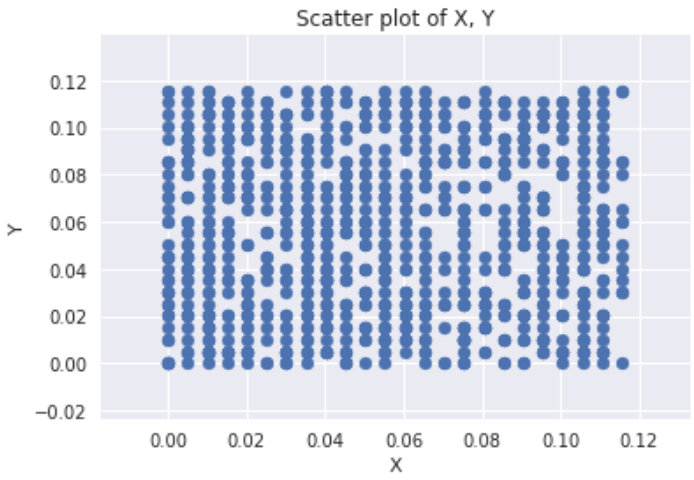


Fig. 2

Figure 3 below shows the K-Mean clustering for k values (2,3,4,5,6,7) respectively.

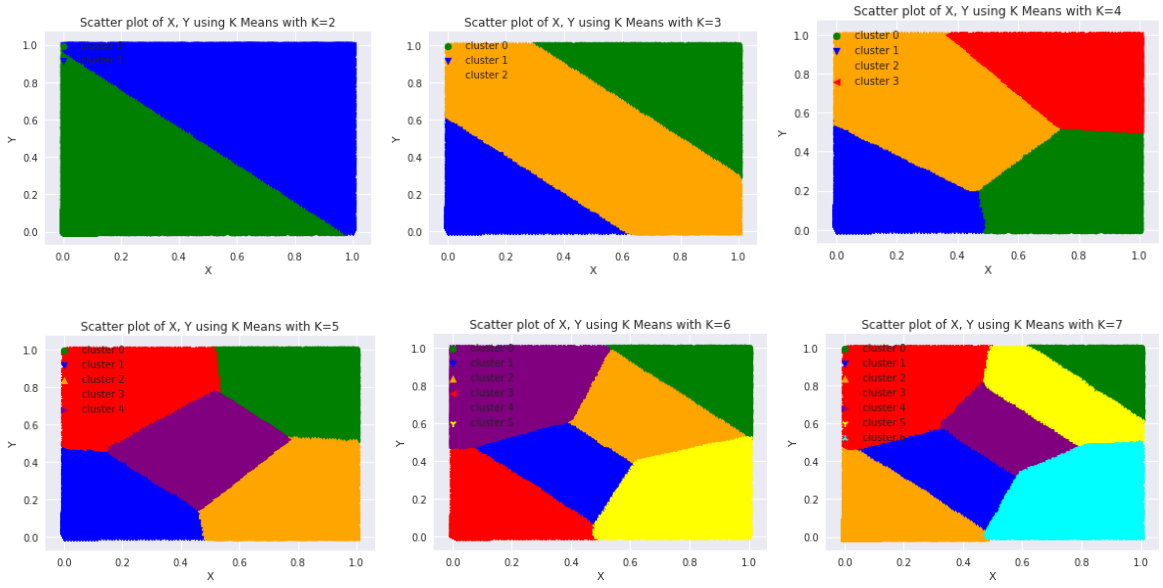


Fig: 3

Figure 4 below shows the K-Medoid clustering for k values (2,3,4,5,6,7) respectively.

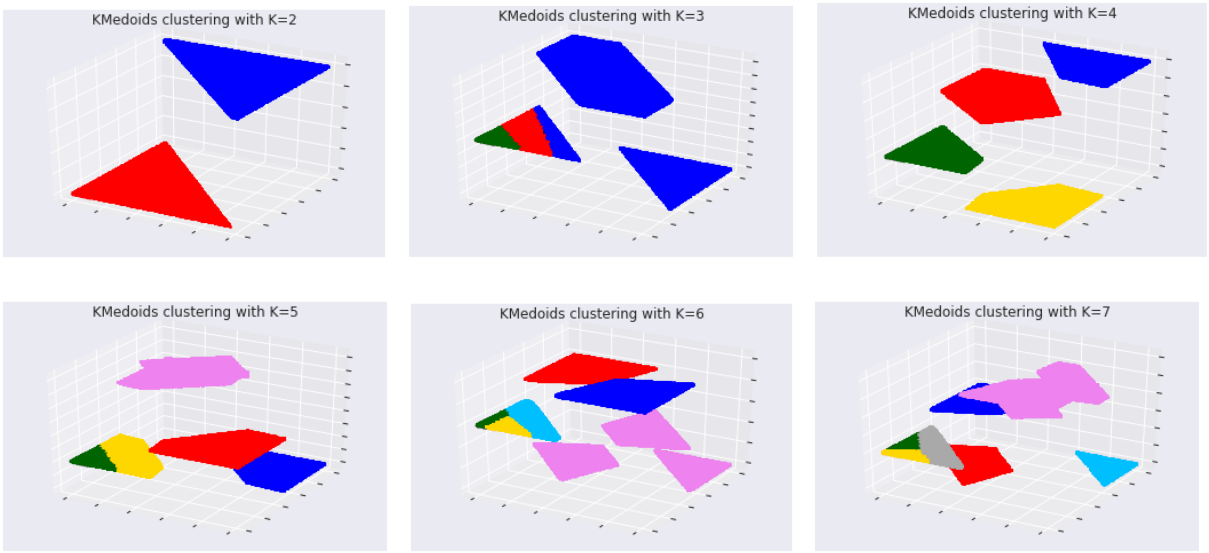


Fig: 4

The number of clusters depends on the value of k as shown in figure 3, and 4 above. However we can identify a reasonable k value by using Elbow method. We plot the SSE against the number of k values and find the k value where we have acceptable SSE value. In the fig below we can observe that **k values (3, 4)** are very reasonable to be chosen for this cluster.

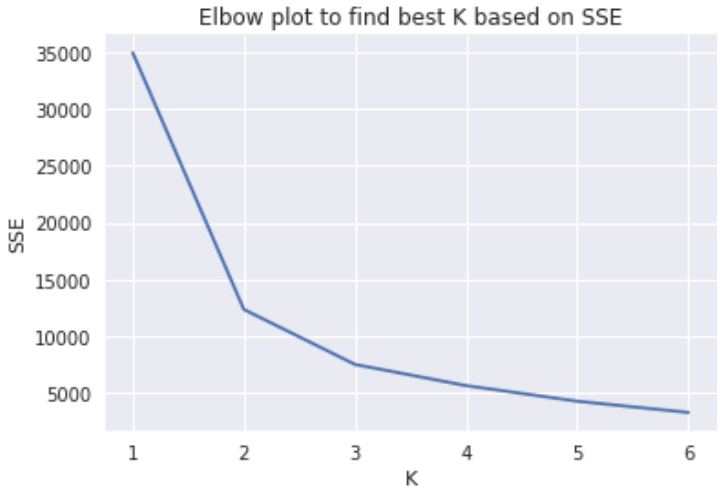


Fig: 5

As we increase the value of k also decreases and we see there are two turns one at 2 and other at 3. It means that at these k values the distance is minimum or the cluster are fully converged and might a best choice for k. There seems to be a significant change in SSE at these two points which gives a good guess for **k**.

However we can more finely observe the numerical values of the SSE as follows (k: 2-7).

```
Sum of squared errors:  [34924.36, 12365.0, 7533.18, 5669.97, 4300.31, 3310.36]
```

There is not much difference between **3 and 4** compared to the earlier difference between 1 and 2 k values.

Hierarchical clustering

Hierarchical clustering is a technique of clustering such that data points are segregated in and represented in the form of a tree. It means that there is a hierarchical (parent -child) relationship among the clusters.

There are two types of hierarchical algorithms.

1. Agglomerative

   It is a bottom up approach and it means that each data point is a cluster initially. With each iteration the similar clusters (nearest) are merged into one and so on until it merges into one big cluster or a certain threshold of clusters count is reached.

2. Divisive

   It is a top down approach in which there is only one big cluster initially which splits down into many with each iteration.

A cluster can be represented by a **centroid** which is the average data point of all the possessing data points in that cluster. A cluster can also be represented by a candidate data point such that is has the minimum SSE (sum of squared errors) and it is called a **clusteroid**.

Hierarchical clustering computes distances att each step for N number of data points givens a **complexity** of $O(N^3)$.

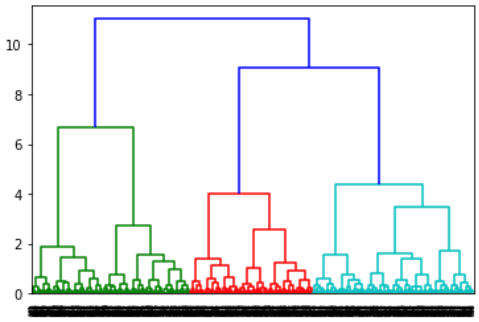
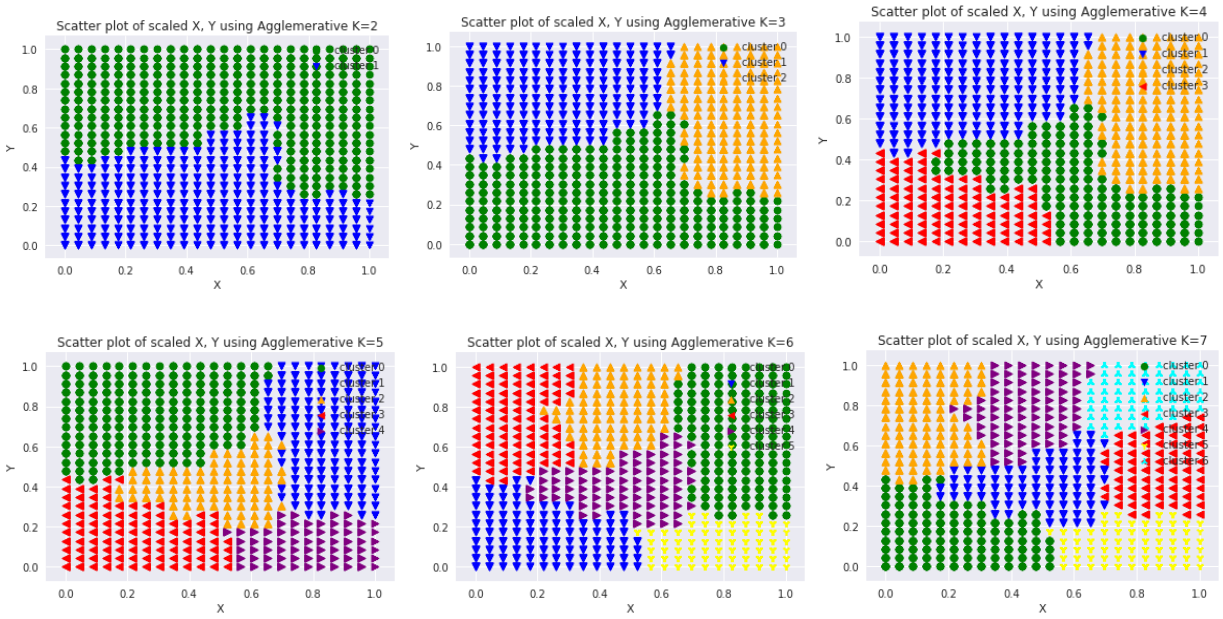


Fig: 6

Hierarchical clustering can be represented /plotted by a dendrogram which is a tree like structure as shown in Figure 6 above.

The plots for k =2-7 using agglomerative hierarchical clustering is shown in figure below.



Evaluation

- The following table (table 1) shows the evaluation of the three algorithms with respect 4 metrics i.e. accuracy, performance, memory consumption and parallelizability.

  Execution time(s)

K	K Means	K Medoids	Agglomerative HC(10k)
2	0.699	544.56	2.11
3	1.02	995.87	1.96
4	1.68	944.12	1.95
5	2.61	724.37	1.92
6	2.47	868.80	1.91
7	2.08	541.76	1.90

Table1

Please note that I used only 10 thousands data points for Agglomerative clustering because of very long time execution although I used google cloud computing with 16CPU(n1) and 16GB RAM.

Installation

1. `apt install git`
2. `git  clone https://github.com/iloveyii/data-mining-lab2.git`
3. `sudo apt install python3-pip`
4. Install packages using pip3

   `pip3 install -r requirements.txt`

Problems & Solutions

ERROR:

ModuleNotFoundError: No module named 'sklearn_extra'

SOLUTION:

Create conda environment

```
conda create --name my_env python=3.6
```

Then activate it

```
conda activate my_env
```

Install pyclustering using conda

```
conda install pyclustering
```

You can deactivate environment as

```
conda deactivate my_env
conda deactivate
```

To list the available / created environments

```
conda env list
```

To list all packages installed in the environment

```
conda list
```

ERROR:

ModuleNotFoundError: No module named 'sklearn_extra'

SOLUTION:

Use conda-forge

ERROR:

Add: forge channel

SOLUTION:

```
conda config --add channels conda-forge
conda config --set channel_priority strict
```

ERROR:

Install: sklearn, scikit-learn

SOLUTION:

```
conda install scikit-learn
```

ERROR:

Install: sklearn-extra from git

SOLUTION:

1. Activate your conda environment
   ```
   source activate myenv
   ```
2. ```
   conda install git pip
   ```
3. ```
   pip install
   git+git://github.com/scikit-learn-contrib/scikit-learn-extra@master
   ```

ERROR:

start: jupyter in a specific environment

SOLUTION:

1. `conda activate my_env`
2. `conda install nb_conda`
3. `conda deactivate`
4. `conda activate my_env`
5. `jupyter notebook`

ERROR:

Google Datalab: to install packages in datalab simply add the following at the start of notebook

SOLUTION:

```
%%bash
pip3 install sklearn pandas matplotlib pyclustering
```

ERROR:

Google Datalab: OSError: /usr/lib/x86_64-linux-gnu/libstdc++.so.6: version `CXXABI_1.3.11' not found

SOLUTION:

```
Conda  install

git+git@github.com:scikit-learn-contrib/scikit-learn-extra.git
```

Install Anaconda on Linux 18.04

curl -O https://repo.anaconda.com/archive/Anaconda3-2019.03-Linux-x86_64.sh

bash Anaconda3-2019.03-Linux-x86_64.sh

conda init

eval "$(/home/ali/anaconda3/bin/conda shell.ali)"

/home/ali/anaconda3/bin/conda init

/home/ali/anaconda3/bin/conda config --set auto_activate_base true

ln -s /home/ali/anaconda3/bin/conda /usr/bin/conda

source ~/.bashrc

conda list

/home/ali/anaconda3/bin/jupyter notebook

jupyter notebook

conda install pyclustering

conda config --add channels conda-forge

conda config --set channel_priority strict

conda install pyclustering

jupyter notebook

Useful links

Search condo-forge for a package which has more packages than coda channel:

https://anaconda.org/conda-forge

Helpful link for medoids

https://codedocs.xyz/annoviko/pyclustering/classpyclustering_1_1cluster_1_1kmedoids_1_1kmedoids.html

Sklearn site

https://scikit-learn-extra.readthedocs.io/en/latest/install.html