CEN429 - INTRODUCTION TO DATA SCİENCE

MIDTERM PROJECT RAPORT

STUDENT NUMBER : 2018555006

NAME : MUSTAFA MAZHAR MURAT

SURNAME : BAĞCI

DEPARTMENT: COMPUTER ENGINEERING

SUBJECT : CLUSTRING WITH DUNCLUE ALGORITHM

DATASET: veriler.txt

ABSTRACT

Clustering algorithms play an important role in data analysis. DENCLUE is a clustering approach which is a Density-based Clustering algorithm. In this study, first of all DENCLUE algorithm is introduced. Secondly, DENCLUE algorithm implemented in python and an application is run with an example data set. Finally, findings about usage of DENCLUE algorithm in example data is mentioned.

CONTENT

INTRODUCTION

Clustering algorithms play an important role in data analysis. These unsupervised learning, exploratory data analysis tools provide systems for knowledge discovery by categorizing data points into distinct groups based on shared characteristics. This allows for the identification of relationships and trends that may be hard to see in the raw data. They facilitate more informed decision making by systematically adding more understanding to complex and intricate datasets.

DENCLUE represents Density-based Clustering. It is a clustering approach depends on a group of density distribution functions. The DENCLUE algorithm use a cluster model depends on kernel density estimation. A cluster is represented by a local maximum of the predicted density function. The algorithm operates based on the idea that clusters in a dataset can be defined as areas of high density separated by areas of low density.

DENCLUE doesn't operate on records with uniform distribution. In high dimensional space, the data always look like uniformly distributed because of the curse of dimensionality. Hence, DENCLUDE doesn't operate well on high-dimensional records in general.

FEATURES OF DENCLUE CLUSTRING ALGORITHM

The method is built on the following ideas which are as follows

- The influence of each data point can be formally modeled using a mathematical function, called an influence function, which describes the impact of a data point within its neighbourhood.

- The complete density of the data area can be modeled analytically as the sum of the influence function used to some data points.

- Clusters can be determined numerically by recognizing density attractors, where density attractors are local maxima of the complete density function.

Major features

- Solid mathematical foundation.

- Good for data sets with large amounts of noise.

- Allows a compact mathematical description of arbitrarily shaped clusters in high shaped clusters in high-dimensional data sets dimensional data sets.

- Significant faster than existing algorithm (faster than Significant faster than existing algorithm (faster than DBSCAN by a factor of up to 45) DBSCAN by a factor of up to 45)

- But needs a large number of parameters.

PYTHON CODE

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings

warnings.filterwarnings("ignore")

data = pd.read_csv("veriler.txt", sep="\t")

def kernel_function(X, X_i, h):
    power = np.sqrt(np.square(X-X_i).sum())
    den = 2*h*h
    return np.square((0.399/h))*np.exp(-(power/den))
def gradient_ascent(new_feature, h):
    for i in range(10):
        new_values = [0, 0]
        sum_k = 0
        for j in range(data.shape[0]):
            k = kernel_function(new_feature, data.iloc[j][['X','Y']].values,
h)
            sum_k += k
            new_values += data.iloc[j][['X','Y']].values*k
        new_feature = new_values/sum_k
    return np.round(new_feature,2)
def Denclue(data, h, t):
    data['height'] = 0.000
    data['new_F1'] = 0.000
    data['new_F2'] = 0.000
    no = 20
    for i in range(data.shape[0]):
        if no==i+1:
            print('*', end="")
            no+=20
        feat_1 = 0.0
        new_Feature = [0, 0]
        for j in range(data.shape[0]):
            k = kernel_function(data.iloc[i][['X','Y']].values,
data.iloc[j][['X','Y']].values, h)
            feat_1 += k
            new_Feature += data.iloc[j][['X','Y']].values*k
        data['height'][i] = (feat_1/len(data)-t)
        new_Feature /= feat_1
        if data['height'][i]<0:
            data['height'][i] = 0
```

```
        else:
            g = gradient_ascent(new_Feature, h)
            data['new_F1'][i] = g[0]
            data['new_F2'][i] = g[1]
    centers= pd.DataFrame({
        'F1_center': data[data['new_F1']!=0]['new_F1'].unique(),
        'F2_center': data[data['new_F2']!=0]['new_F2'].unique(),
        'Cluster': np.arange(1,
len(data[data['new_F1']!=0]['new_F1'].unique())+1)
    })
    print('\n\nCluster Centers')
    print(centers)
    data['Cluster'] = -1
    for i in range(data.shape[0]):
        for j in range(centers.shape[0]):
            if data.iloc[i]['new_F1'] == centers.iloc[j]['F1_center'] and
data.iloc[i]['new_F2'] == centers.iloc[j]['F2_center']:
                data['Cluster'][i] = centers.iloc[j]['Cluster']
                break
    print('\nNo of Datapoints in Each Cluster')
    print(data.groupby('Cluster')['Cluster'].count())
h = 0.3
t = 0.01
Denclue(data, h, t)
plt.figure(figsize=(8, 8))
sns.scatterplot(data=data, x='X', y='Y', hue='Sınıf', palette="deep", s=130)
plt.grid(True)
plt.show()
```

PYTHON CODE REVIEW

There is an implementation of DENCLUE algorithm in python below. In this code, we could see there are blocks. Each block handles a different step in the DENCLUE algorithm are as follows

- Import Libraries: The code imports necessary libraries like NumPy, Pandas, Seaborn, and Matplotlib for data manipulation, visualization, and mathematical operations.

- Reading the Data: The code reads data from a file named "veriler.txt" into a Pandas DataFrame (data) using pd.read_csv() function, assuming the data is tab-separated (sep="\t").

- Defining Kernel Function: The kernel_function() is defined to compute the kernel function. It calculates the kernel value between two points based on the Euclidean distance and a bandwidth parameter h which initialized to 0.3 . This function helps in estimating density.

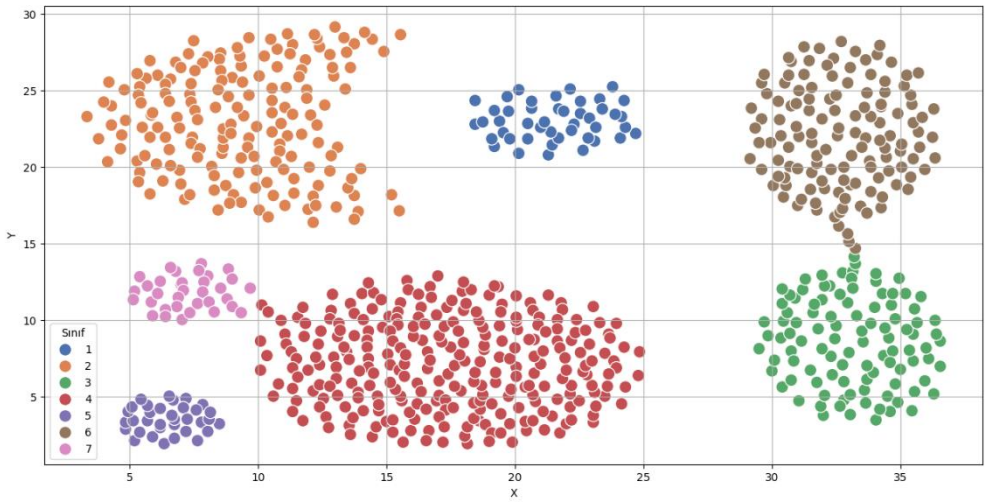
- Gradient Ascent Function: gradient_ascent() implements a gradient ascent method. It's used iteratively to find the attractor (cluster center) by moving towards the maximum density point in the data space.

- DENCLUE Algorithm Implementation (Denclue function): The Denclue() function is the core implementation of the DENCLUE algorithm. It initializes columns 'height', 'new_F1', and 'new_F2' in the DataFrame data for further computations. It iterates through each data point and computes its density and potential cluster center using kernel functions. The function uses gradient ascent (gradient_ascent()) to find the cluster center. It assigns a cluster number to each data point based on the identified centers.

- Cluster Centers and Assignment: The code identifies cluster centers based on the computed 'new_F1' and 'new_F2' values, storing them in a DataFrame called centers. It assigns cluster numbers to data points based on their proximity to these identified cluster centers.

- Displaying Cluster Information: The code prints the identified cluster centers and the number of data points in each cluster.

- Visualization: Finally data is visualized with sns.scatterplot() of seaborn library. Below figure is the scatter plot table of given data veriler.txt:



DENCLUE algorithm identified given data as expected and returned 7 different clusters. Clusters 1, 5 and 7 has a fewer variables whereas clusters 2 and 4 has the most variables. Finally clusters 6 and 3 has reasonable amount of variables which looks related and touches at some point.

CONCLUSION

In conclusion, the DENCLUE algorithm presents a robust density-based clustering approach that excels in identifying clusters of arbitrary shapes within datasets by leveraging density information. Its strength lies in its ability to handle irregularly shaped clusters and varying densities effectively. By iteratively computing attractors based on local density and assigning data points to these attractors, DENCLUE showcases adaptability in identifying clusters in complex datasets.

However, DENCLUE's performance can be influenced by the choice of parameters, such as kernel bandwidth, which requires careful tuning to achieve optimal results. Moreover, it may face challenges with high-dimensional data and datasets where clusters have significantly different densities.

Despite its limitations, DENCLUE remains a valuable clustering algorithm, especially in scenarios where detecting non-spherical or irregularly shaped clusters is essential. Its reliance on density estimation and attractor computation makes it a powerful tool in the realm of unsupervised learning and clustering techniques. Continued research and refinement of DENCLUE and its parameterization could further enhance its applicability and accuracy in real-world data analysis tasks.

SOURCES

Original python implementation code:
https://github.com/mgarrett57/DENCLUE/blob/master/denclue.py

Denclue – Density Based Clustering Algorithm – Research Sites:

https://users.wpi.edu/~yli15/courses/DS504Fall16/includes/denclue.pdf

https://www2.cs.uh.edu/~ceick/DM/DENCLUE.pdf

https://www.tutorialspoint.com/what-is-denclue

https://www.datamining365.com/2020/04/density-based-clustering-dbscan.html