CS40003
(Data Analytics)

Term Project
Group ID - 14
Project ID - 14

Topic - Unsupervised learning technique with hierarchical clustering using **DIANA** algorithm

Team Members :
Aryan Singh 17MT10010
Saikat Mondal 17MT3FP04

Introduction :

Clustering is defined as the method used for grouping of the data in such a way that the data belonging to a particular cluster has more similar properties or features to the points in that cluster in comparison to the points in different clusters.
Clustering is a type of unsupervised learning where we try to find information from the input data without labels.

There are several types of clustering which are mentioned below :

1. Density Based Methods - DBSCAN, STING
2. Partitioning Methods - k-Means, k-Medoids
3. Hierarchical Based Methods - Divisive (DIANA), Agglomerative (AGNES), BIRCH, ROCK etc.
4. Graph Based Methods - MST Clustering, SSN Similarity Clustering
5. Model Based Methods - Autoclass, ANN Clustering, EM Algorithm

Here we will discuss Hierarchical Clustering and in particular **Di**visive **Ana**lysis algorithm.

In any type of hierarchical clustering method, our main goal is to create the clusters that have predominant ordering from top to bottom or vice versa which is accomplished by splitting down a large cluster into small clusters or merging the small clusters into one single large cluster.

DIVISIVE HIERARCHICAL CLUSTERING :

Divisive Analysis Clustering is a top-down clustering method in which all the observations initially is considered as a single cluster and then it is partitioned into two least similar clusters. Then the clusters are further split recursively until there is one single cluster for each observation or any other termination condition is achieved. For example, the algorithm terminates after getting desired number of clusters, achieving particular size of cluster etc.

Let’s say we have dataset, G={g1, g2, g3, …, gn} having n points/examples on which we want to do clustering. By definition, G is a single large cluster. We calculate the distance matrix for the dataset using some similarity measures/metrics.

The similarity measures which we have used for the calculation of distance matrix are **L2 norm** (Euclidean Norm) and **Cosine similarity.** Some other similarity metrics are Manhattan distance, Mahalanobis distance, Hamming distance.
The distance matrix a symmetric matrix whose elements C(i,j) represents the distance or similarity between the ith and jth points.

After getting the distance matrix, we find the average distance between di and the rest of the points in the cluster. The point having the greatest average distance is then moved from the cluster G and added to a new cluster initially empty cluster, G1. This new cluster is also referred as “Splinter Group”. After the first split, we now have 2 clusters. The object in G1 has the highest order of dissimilarity with G which is the main aim of clustering.

After obtaining two entirely different cluster, we have to check for other points in the bigger cluster whether to move them to a different cluster or not. For this, we have to calculate the d(G1) where d(G1) is the average distance between the point and the splinter cluster, H where the distance is measured according to a particular metric and similarity of the point with the remaining objects, say di, in the cluster other than splinter cluster is measured. The difference of di and d(G1) is then calculated. If this quantity is positive then the point is shifted to the splinter cluster else it remains in that cluster.
The process of removing the points from the larger cluster terminates when no single point is left which has to be moved i.e it halts when every point remaining has negative difference and at the end we obtain two different clusters.

Further splitting of the 2 clusters obtained is done on the basis of diameter. Diameter of a cluster is defined as the maximum distance between any 2 points in a cluster.
The one having the larger is selected first for splitting and the process which was done for a single large cluster is repeated for this cluster until any of the convergence criteria is satisfied.

Pseudo Code for the algorithm :

1. Given a dataset G(g1, g2, g3…, gn) of size n. The dataset is initialized as a single cluster.
2. Distance matrix is calculated for a similarity metric.
3. The cluster split on the basis of the largest average distance into 2 clusters and the remaining points are moved if the difference between average distance and the distance with other cluster is positive.
4. The diameter of individual clusters are calculated. The one with maximum diameter is selected for the further splitting.
5. The above steps are repeated for every cluster until singleton clusters are not obtained or the convergence is reached.

Exponential number of ways are possible to split a cluster, (2^(n-1) - 1) to be particular. Since the method follows a set of rules to determine the split, it is not exhaustive and hence the running time is polynomial instead of exponential. The running time of this algorithm is O(n^2).

DIANA gives more accurate results in comparison to AGNES. The reason behind it is that the global distribution of the dataset is considered in case of DIANA while local patterns are identified initially by AGNES. Due to these reasons, AGNES identifies small clusters and DIANA identifies larger accurately. However, in DIANA once a cluster is formed, we can not improve the cluster quality by backtracking.

DIANA is not preferred for large dataset due to being computationally expensive.

PYTHON CODE :

Implementation of Similarity Measures -

```
# L2 and Cosine similarity measurement
import math

def l2_norm(x): #function to calculate L2 norm of a vector
  ''' x - the vector
  returns - L2 norm of the vector '''

  sum = 0
  for i in range(len(x)):
    sum += x[i]**2
  return math.sqrt(sum)

def SimilarityMeasure(data1, data2, type):
  ''' data1, data2 - vectors, between which we are going to calculate
similarity
  returns - distance between the two vectors '''

  if type=='L2':
    # for L2 norm or pythagorean distance
    dist = 0
    for i in range(4):
      dist += (data1[i] - data2[i] )**2
    dist = math.sqrt(dist)
    return dist

  if type=='Cosine':
    # form cosine similarity = a.b/|a|.|b|
    dot_prod = 0
    data1_mod = l2_norm(data1)
    data2_mod = l2_norm(data2)
    for x in range(4):
      dot_prod += data1[x]*data2[x]
    return (dot_prod/(data1_mod*data2_mod))

  else:
    print('Please provide proper similarity measurement type')
    return 0
```

Implementation of Distance/Similarity Matrix -

```
import numpy as np
from pathlib import Path
import pandas as pd
import pickle

#this function calculates Distance Matrix or Similarity matrix

def DistanceMatrix(data =None):
  ''' data - the dataset whose Similarity matrix we are going to
calculate
  returns - the distance matrix by loading the pickle file '''
  pickleFilePath = Path('SimMat.pkl') #checking if the distance matrix
was saved from last run to save processing

  Data_list = []
  for index, rows in data.iterrows():
    my_data = [rows.Hobby, rows.Age, rows.Educational_Level,
rows.Marital_Status]
    Data_list.append(my_data)

  if pickleFilePath.is_file():
    temp_file=open(pickleFilePath, 'rb')
    return pickle.load(temp_file)

  else:
    N = len(data)
    similarity_mat = np.zeros([N, N]) #for cosine np.ones
    for i in range(N):
      for j in range(N):

similarity_mat[i][j]=SimilarityMeasure(Data_list[i],Data_list[j],type="
Cosine")
      with open('SimMat.pkl', 'wb') as file:
        pickle.dump(similarity_mat, file)

    temp_file=open(pickleFilePath, 'rb')
    return pickle.load(temp_file)
```

Implementation of DIANA -

```
import numpy as np
import copy
import pandas as pd

#this is the main class that computes the clusters using diana
algorithm

class DianaClustering:
  def __init__(self,data):
 ''' constructor of the class, it takes the main data frame as input'''
    self.data = data
    self.n_samples, self.n_features = data.shape

  def fit(self,n_clusters):
 ''' this method uses the main Divisive Analysis algorithm to do the
clustering
    Arguments - n_clusters - number of clusters we want (integer)
    returns -
    cluster_labels - numpy array
             an array where cluster number of a sample corresponding to
             the same index is stored
    '''
    similarity_matrix = DistanceMatrix(self.data) # similarity matrix
of the data

    clusters = [list(range(self.n_samples))]      # list of clusters,
initially the whole dataset is a single cluster

    while True:
      c_diameters = [np.max(similarity_matrix[cluster][:, cluster]) for
cluster in clusters]  #cluster diameters

      max_cluster_dia = np.argmax(c_diameters)  #maximum cluster
diameter
      max_difference_index =
np.argmax(np.mean(similarity_matrix[clusters[max_cluster_dia]][:,
clusters[max_cluster_dia]], axis=1))
      splinters = [clusters[max_cluster_dia][max_difference_index]]
#sprinter group
      last_clusters = clusters[max_cluster_dia]
      del last_clusters[max_difference_index]
      while True:
        split = False
        for j in range(len(last_clusters))[::-1]:
          splinter_distances = similarity_matrix[last_clusters[j],
splinters]
          last_distances = similarity_matrix[last_clusters[j],
np.delete(last_clusters, j, axis=0)]
          if np.mean(splinter_distances) <= np.mean(last_distances):
            splinters.append(last_clusters[j])
            del last_clusters[j]
            split = True
            break
        if split == False:
          break
      del clusters[max_cluster_dia]
      clusters.append(splinters)
      clusters.append(last_clusters)
      if len(clusters) == n_clusters:
        break

    cluster_labels = np.zeros(self.n_samples)
    for i in range(len(clusters)):
      cluster_labels[clusters[i]] = i

    return cluster_labels

if __name__ == '__main__':
  data = pd.read_csv('HAYES_ROTH.csv') #reading the dataset csv file
  data = data.drop(columns="Name") # dropping the unnecessary features
  data = data.drop(columns="Class")

  diana = DianaClustering(data) #applying the Diana Clustering
algorithm

  clusters = diana.fit(3) #as there are 3 classes we chose to divide
the dataset in 3 clusters

  print(clusters)
```

*** The libraries such as numpy, pandas, pickle and math have **NOT** been used in building DIANA, instead they have been used for manipulating data and ease calculation steps. For clustering purpose, applied algorithm is written from scratch and libraries like **ScikitLearn, Scipy have NOT been used**. ***

Experimental Results :

We applied DIANA to HAYES_ROTH dataset. The dataset consists of 132 objects and 5 different attributes. All the attributes are nominal other than 'Name' and these attributes are Hobby(1-3), Age(1-4), Educational Level(1-4), Marital Status(1-4).

Our task was to form the clusters on the basis of given attributes.

Original Clusters in the dataset are

C1 : [36, 105, 81, 94, 20, 50, 68, 89, 19, 118, 16, 91, 30, 57, 114, 66, 74, 106, 130, 54, 67, 69, 127, 96, 121, 123, 42, 5, 95, 119, 93, 132, 108, 120, 35, 112, 59, 1, 28, 97, 51, 103, 7, 15, 126, 45, 131, 17, 40, 9, 92]

C2 : [10, 113, 80, 60, 85, 52, 79, 23, 25, 37, 116, 88, 77, 82, 84, 86, 6, 115, 33, 39, 53, 70, 78, 129, 73, 26, 104, 2, 41, 62, 98, 109, 31, 34, 63, 65, 117, 56, 76, 29, 111, 49, 58, 32, 99, 24, 124, 14, 71, 90, 21]

C3 : [83, 61, 107, 125, 122, 8, 3, 110, 64, 11, 128, 4, 48, 102, 75, 47, 46, 18, 27, 22, 87, 72, 55, 101, 100, 13, 38, 43, 12, 44]

Clusters identified by DIANA are

G1 : [10, 83, 61, 113, 80, 36, 105, 81, 94, 60, 85, 50, 68, 89, 52, 118, 91, 79, 25, 30, 57, 3, 66, 110, 116, 88, 77, 82, 64, 84, 86, 74, 106, 130, 54, 67, 69, 39, 53, 127, 96, 121, 70, 123, 78, 11, 129, 5, 4, 95, 73, 26, 104, 102, 2, 41, 119, 47, 93, 132, 108, 62, 120, 27, 98, 31, 34, 63, 117, 56, 59, 76, 1, 28, 29, 111, 97, 49, 51, 87, 58, 32, 103, 7, 99, 15, 126, 45, 100, 124, 13, 14, 71, 17, 40, 90, 21, 9, 92]

G2 : [107, 122, 20, 33, 128, 75, 18, 22, 24, 44]

G3 : [125, 8, 19, 16, 23, 114, 37, 6, 115, 42, 48, 46, 35, 109, 112, 65, 72, 55, 101, 38, 43, 131, 12]

The list contains the Names which are most similar.

CLUSTER QUALITY ESTIMATION :

For measuring the quality of clusters obtained from the algorithm, we use ‘Divisive Coefficient’. It is defined as the average of the diameter of previous clusters to which the point once belonged. It is a measure of the amount of clustering structure found.