CSCI446/946 Big Data Analytics - Week 4

Lab 3 – Clustering with Python

Introduction

This instruction is developed on Python programming language, and use Spyder to run `Lab3.py`. In Lab3, you will practice a set of clustering algorithms [1] which correlate to big data analytics lifecycle phase 3-5: Model Planning, Model Building and Communicate Results. To complete Tasks 1-2, you are required to

1. download `Lab3.py` and the Pokeman dataset from Moodle
2. implement the code shown in `blue` in a file called `Lab3.py`
3. create a Word document to 1) explain what you will do next according to **Describes**, and 2) answering **Questions**
4. format your report: title, heading, body of text, code, programming results, tables, figures, references, etc.

Import packages:

```
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans
from scipy.cluster.hierarchy import linkage, dendrogram, cut_tree
from scipy.spatial.distance import pdist
import matplotlib.pyplot as plt
```

Task 1: Clustering Algorithm 1 – K-means

The first task gives an overview of how to practice clustering algorithm K-means on the Pokeman dataset by using Python.

The Pokemon dataset that we used in Week 4 is a good example to use for clustering as it has a number of numerical fields giving the properties of each Pokemon. We can also imagine that there might be different types of Pokemon within the data, so using clustering methods to find the different types is a reasonable approach.

Describe 1: First we read the data from the CSV file, we'll use the Pokemon name as the data frame index and drop the first column which is a useless index number.

Code 1:

```
df = pd.read_csv('Pokemon.csv', index_col=1)
df.drop('Unnamed: 0', axis=1, inplace=True)
print("Pokeman dataset size:", df.shape)
print("Pokeman dataset head \n", df.head())
```

Question 1: What's the variable name which restores Pokemon data? What is the data structure, i.e., the meaning of rows and columns, data type of each column? Is there any missing data?

Now you should apply the kMeans clustering method to this data. The first step is to select just the numerical fields in the data. You can either drop the non-numerical fields or make a new data frame containing just the numerical ones (I suggest making a new data frame).

Then apply the kMeans clustering function to the data, following the steps in the lecture notes and text book. Since we don't have any real idea how many clusters there could be in the data, start with a small number of clusters (eg. 4) just to make it easier to understand the clusters.

Describe 2: Select the numerical fields in the data
Code 2:
```
print("column name and data types: \n", df.dtypes)
```
Question 2: What are the selection results of numerical fields?

Describe 3: Dropping non-numeric columns from the original dataframe and creating a new dataframe
Code 3:
```
pokemon = df.drop(columns=['Type 1','Type 2','Legendary'])
```
Question 3: What's the variable name which restores numerical Pokemon data? What is the data structure, i.e., the meaning of rows and columns, data type of each column? Is there any missing data?

Describe 4: Apply kMeans clustering function
Initial a small number of clusters n_clusters=2
Code 4:
```
km = KMeans(n_clusters=2)
km.fit(pokemon)
```
Question 4: Explain the km.fit() function.

Once you have applied kMeans you will have some results to explore. Your goal is to understand the clusters that have been produced. If you know something about Pokemon you might be able to recognise similarities between members of each cluster, if not (and even if you do) you need to understand what the members of each cluster have in common.

Describe 5: The first task is to find out what the members of each cluster are.
To do this, generate a set of cluster labels using km.predict() on your original data and add this to the data frame.
Code 5:
```
pokemon['label'] = km.predict(pokemon)
print("dfpokeman with cluster labels: \n", pokemon)
```
Question 5: Explain the labels and how are labels restored in dataframe pokemon.

Describe 6: Next exploration is the difference between each cluster
find the mean value for each column by cluster, you can do this by selecting the rows for each cluster and then taking the mean or by using the groupby method
Code 6:
```
pokeman_mean = pokemon.groupby(['label']).agg('mean')
print(pokeman_mean)
```

Question 6: Explain the findings. From these means, can you characterise the clusters that were found? eg. "Cluster 0 are early stage Pokemon with relatively low HP, Attack and Defence ratings". Provide descriptions like this of each cluster.

Describe 7: Identify two columns that seem to be more distinct between clusters, use these to plot the data with different colours for each cluster to visualise the result of clustering. Comment on the separation of clusters in your plot.

Code 7:

```
columns = list(pokeman_mean.columns)
for column in columns:
    dist = np.abs(pokeman_mean[column][0] -
pokeman_mean[column][1])/max(pokeman_mean[column])
    print('{}: {}'.format(column, dist))
plt.scatter(pokemon['Attack'], pokemon['Defense'],
c=pokemon['label'])
plt.scatter(pokemon['Sp. Atk'], pokemon['Sp. Def'],
c=pokemon['label'])
plt.scatter(pokemon['HP'], pokemon['Speed'], c=pokemon['label'])
plt.scatter(pokemon['Defense'], pokemon['Speed'],
c=pokemon['label'])

# using the variable axs for multiple Axes
fig, axs = plt.subplots(14, 2, figsize=(20,80))
columns = list(pokeman_mean.columns)
j2, i2 = 0, 0
for i in range(len(columns)-1):
    for j in range(i+1,len(columns)):
        if j2 > 1:
            j2 = 0
            i2 += 1
        axs[i2,j2].scatter(pokemon[columns[i]],
pokemon[columns[j]], c=pokemon['label'])
        axs[i2,j2].set_title('{} vs {}'.format(columns[i],
columns[j]))
        j2 += 1
```

Question 7: Repeat the experiment with a larger number of clusters. Do more clusters make it easier to distinguish the clusters?

Describe 8: apply k-means on suitable k clusters based on explorations above, for example n_clusters=3

```
km = KMeans(n_clusters=3)
km.fit(pokemon[columns])
pokemon['label_c'] = km.predict(pokemon[columns])
pokeman_mean = pokemon.drop('label',
axis=1).groupby(['label_c']).agg('mean')
print(pokeman_mean)
plt.scatter(pokemon['Total'], pokemon['HP'], c=pokemon['label_c'])
plt.title('Total vs HP, n_cluster=3')
plt.show()
```

Question 8: Drawn conclusions about kMeans results.

Task 2: Clustering Algorithm 2 – Hierarchical Clustering

In this task, we will keep using Pokeman dataset on Hierarchical clustering, learn from the breaking down steps provided in Task1 and practice the following codes.

Since we don't know how many clusters there should be in the data, a better approach is to use Hierarchical clustering and examine the dendograph to understand what natural clusters are present in the data.

Apply Hiearchical clustering to the data and plot the dendogram. From this diagram, how many clusters should the data be separated into? Can you characterise these clusters?

Code 9:

```
pokemon = df.drop(columns=['Type 1','Type 2','Legendary'])
dist = pdist(pokemon, 'euclidean')
linkage_matrix = linkage(dist, method = 'complete')
plt.figure(figsize=(15,7))
dendrogram(linkage_matrix)
plt.show()
```

Code 10:

```
labels = cut_tree(linkage_matrix, n_clusters=3)
pokemon['label'] = labels
print("describe cluster labeled 0: \n",
pokemon[pokemon['label']==0].describe())
print("describe cluster labeled 1: \n",
pokemon[pokemon['label']==1].describe())
```

Task 3: Clustering Algorithm 3 – Self-Organizing Maps for Clustering and Data Visualization

We will use an extract of a real world dataset. The dataset "A1_BC_SEER_data.csv" contains records of cancer patients and includes information on diagnostics, treatments, and outcomes. The samples are labelled according to whether a patient has survived the cancer, or whether the patient died from cancer. Your objective is to find out how the data is organized, and how the two classes are organized in the feature space. Prepare the data and train a SOM as shown in the code below. Explain how you can characterize clusters, and what insights are revealed by the plots produced by the code.
Download the files: `myminisom.py` and `A1_BC_SEER_data.csv`.

Code 11: (Prepare the data)

```
import pandas as pd
import numpy as np
from sklearn.utils import shuffle

df = pd.read_csv("A1_BC_SEER_data.csv", )
df = shuffle(df)
```

```
target = df['Survival months']  #Extract the target column

#Binarize target
target = np.where(df['Survival months'] < 60, 0, target)
target = np.where(df['Survival months'] >= 60, 1, target)
```

Code 12: Preprocess the data

```
from sklearn.model_selection import train_test_split

myseed=7   #Seed for the random number generator

#Remove irrelevant features, and targets from df
dropList = ['Patient ID', 'Survival months']
for item in dropList:
    df.drop(item, axis=1, inplace=True)

#Scale the data (think about whether the next three lines
# should be uncommented)?
#from sklearn import preprocessing
#scaling = preprocessing.MinMaxScaler()
#data = scaling.fit_transform(data)

#Create a train, test, and validation set
X, X_tst, Y, Y_tst = train_test_split(df, target, test_size=.333,
random_state=myseed)
X_trn, X_val, Y_trn, Y_val = train_test_split(X, Y, test_size=.5,
random_state=myseed)

X_trn = X_trn.to_numpy()
X_tst = X_tst.to_numpy()
X_val = X_val.to_numpy()
```

Code 13: Train the SOM

```
from myminisom import MiniSom   #see myminisom.py

#Create the SOM
som_shape = (100, 100)      #define the size of the som
som = MiniSom(som_shape[0], som_shape[1], X_trn.shape[1],
sigma=som_shape[0]/2, learning_rate=.9,
neighborhood_function='gaussian', random_seed=myseed)

#initialize the SOM, then train it
epochs=40
som.pca_weights_init(X_trn)
som.train_random(X_trn, epochs * len(X_trn), verbose=True)

#Find the BMU for each sample
BMU_trn = np.array([som.winner(x) for x in X_trn])
BMU_class0 = BMU_trn[Y_trn==0]
BMU_class1 = BMU_trn[Y_trn==1]
```

Code 14: Plot some results (density map of all samples)

```
import matplotlib.pyplot as plt
from copy import copy

densitymap = np.zeros(som_shape)
for row in range(0,BMU_trn.shape[0]):
    x,y = BMU_trn[row]
    densitymap[y,x] += 1

densitymap[densitymap==0]=np.nan  #mark zero values with nan
my_cmap = copy(plt.cm.jet)
my_cmap.set_bad(color=(1,1,1))    #plot nan in white color
plt.imshow(densitymap, cmap=my_cmap, interpolation="none",
origin="lower", aspect=0.75)
plt.colorbar()
plt.title('Mapping density')
plt.show()
```

Code 15: Plot the density map of all samples from class 1

```
import matplotlib.pyplot as plt

densitymap = np.zeros(som_shape)
for row in range(0,BMU_class1.shape[0]):
    x,y = BMU_class1[row]
    densitymap[y,x] += 1

densitymap[densitymap==0]=np.nan  #mask zero values
plt.imshow(densitymap, cmap=my_cmap, interpolation="none",
origin="lower", aspect=0.75)
plt.colorbar()
plt.title('Mapping density (class 1)')
plt.show()
```

Modify code 15 to plot the density map for the samples in class 0. Compare the density plot of class 0 with the density plot of class 1.

Describe: What can be seen in those plots. What insights about the data and class distribution can you derive from these plots?

Explain: How can you characterize clusters?

Reference

[1] E. E. Services, Data science and big data analytics: discovering, analyzing, visualizing and presenting data, Chapter 4, Wiley, 2015.