CS564 Foundations of Machine Learning
Assignment: 2

Mukuntha N S (1601CS27)
R J Srivatsa (1601CS35)
Shikhar Jaiswal (1601CS44)

August 27, 2019

1 Assignment Description

In the computational biology field, identifying the correct set of clusters helps to analyze the characteristics of the genes that helps to understand the unknown gene characteristics. Also, finding the correct cluster centers is one of the important steps for getting good results. We try to find the correct number of clusters for the **B-Chronic Lymphocytic Leukemia(BCLL) Dataset** .

2 Procedure

Installation

Install the following dependencies either using pip or through conda in a Python 3.5+ environment:
• jupyter
• pandas
• sklearn
• matplotlib

```
python3 -m pip install jupyter pandas sklearn matplotlib
```

or alternatively,

```
conda install -c anaconda jupyter pandas sklearn matplotlib
```

Running The Notebook

To run the program, head to the directory $Assignment2/Q1$. Please note that the **dataset should be present in the same directory** as the jupyter notebook. Use the following command to run the notebook:

```
jupyter notebook Q1.ipynb
```

3 Discussion

The primary objective of the assignment is to understand the change of silhouette score along with cluster centers while running K-Means algorithm. The following sub-sections contain the explanation for individual code snippets. For a detailed look at the output, please refer the notebook and the individual files provided.

Notebook Code

Pre-processing and Visualization

Import the Python dependencies, and check the data samples and their values.

```
# Import the required libraries.
import re
import math
import random
import collections
import numpy as np
import pandas as pd
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
import matplotlib.pyplot as plt

random.seed(15)
np.random.seed(15)

df_list = []

# Function for loading the data into a pandas dataframe.
with open('./preprocessed_BCLL.txt') as f:
    for line in f:
        line = line.strip()
        columns = line.split('\t')

        for i, column in enumerate(columns):
            if i >= 2:
                columns[i] = float(column)

        df_list.append(columns)

data = pd.DataFrame(df_list)

print(data)
```

Part A

We run the K-Means algorithm 10 times where each time the number of clusters is randomly chosen between 2 to sqrt(N), where N represents the number of data points in the dataset. For each iteration, the Silhouette

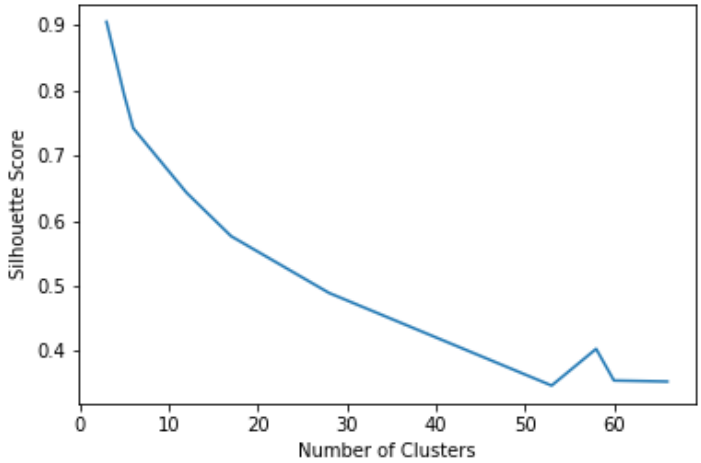
score is reported. Finally, we plot a graph, to understand the change of Silhouette score along with the number of cluster centers.

```
# Get all the numeric data from the original dataframe.
X = data._get_numeric_data().reset_index(drop = True)
# Normalize the dataframe.
X = X.apply(lambda column: (column - column.mean()) / column.std(), axis = 0)

SSE = {}
max_silhouette_score = float('-inf')
max_silhouette_score_cluster = None

# Sample 10 integers from 2 to sqrt(N) for the number of clusters, to prevent repetition
num_clusters_list = random.sample(range(2, math.floor(math.sqrt(X.shape[0]))), 10)

# Run loop for every possible value of num_clusters
for itn, num_clusters in enumerate(num_clusters_list):
    # Get the labels for each sample.
    labels = KMeans(n_clusters = num_clusters, random_state = 42).fit_predict(X)
    # Get the silhouette score for each sample.
    score = silhouette_score(X, labels, metric = 'euclidean', sample_size = None)

    # Update the maximum score seen, and the cluster number which gives that score.
    if score > max_silhouette_score:
        max_silhouette_score = score
        max_silhouette_score_cluster = num_clusters

    # Store the number of clusters which give a particular score.
    SSE[num_clusters] = score
    print('Iteration: ', itn, ' Number of Clusters: ', num_clusters, ' Silhouette Score: ', score)

# Order the cluster numbers.
sorted_SSE = sorted(SSE.items(), key = lambda kv: kv[0])
sorted_SSE = collections.OrderedDict(sorted_SSE)
# Plot the graph for silhouette score vs number of clusters.
fig, ax = plt.subplots()
ax.plot(list(sorted_SSE.keys()), list(sorted_SSE.values()))
plt.xlabel('Number of Clusters')
plt.ylabel('Silhouette Score')
fig.savefig('silhouette_scores.png')
print('Saved plot successfully.')
plt.show()
```

```
Output:
Iteration:  0  Number of Clusters:  28  Silhouette Score:  0.48935530619143564
Iteration:  1  Number of Clusters:  3  Silhouette Score:  0.9047188602880597
Iteration:  2  Number of Clusters:  6  Silhouette Score:  0.741793800010191
Iteration:  3  Number of Clusters:  12  Silhouette Score:  0.6427498275659183
```

```
Iteration:  4  Number of Clusters:  60  Silhouette Score:  0.35520084100725413
Iteration:  5  Number of Clusters:  17  Silhouette Score:  0.5762882703193228
Iteration:  6  Number of Clusters:  66  Silhouette Score:  0.3536492578742264
Iteration:  7  Number of Clusters:  5  Silhouette Score:  0.7919273249611707
Iteration:  8  Number of Clusters:  58  Silhouette Score:  0.4036976080795247
Iteration:  9  Number of Clusters:  53  Silhouette Score:  0.3475067719213223
Saved plot successfully.
```

Plot



Part B

For the iteration, with the highest Silhouette score, we show the gene names for each cluster in individual files, where each file represents a particular cluster.

```python
print('Maximum Silhouette Score: ', max_silhouette_score)
print('Number of Clusters: ', max_silhouette_score_cluster)

# Get the labels for each sample for the case where we see the maximum silhouette score.
labels = KMeans(n_clusters = max_silhouette_score_cluster, random_state = 42).fit_predict(X)

# Append the labels to the respective sample.
data['23'] = labels

# Output the clusters to individual files.
for label in set(labels.tolist()):
    cluster_data = data.loc[data['23'] == label, 1]
    output_filename = 'Cluster_' + str(label + 1) + '.csv'
    cluster_data.to_csv(output_filename, header = False, index = False)
    print("Wrote output to {}".format(output_filename))
```

```
Output:
Maximum Silhouette Score:  0.9047188602880597
Number of Clusters:  3

Wrote output to Cluster_1.csv
Wrote output to Cluster_2.csv
Wrote output to Cluster_3.csv
```

Additionally please refer $Q1/Cluster\_1.csv$ and similar files for the output.