Report

December 1, 2023

1 Homework 4: Spectral clustering

1.1 Data Mining

1.2 Group 56

1.3 Abdelrahman Saleh, Farmaki Athanasia

[1]:

```
import numpy as np
import networkx as nx
import scipy
from networkx import DiGraph
import scipy.linalg as la
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
from sklearn.metrics import silhouette_score
```

[2]:

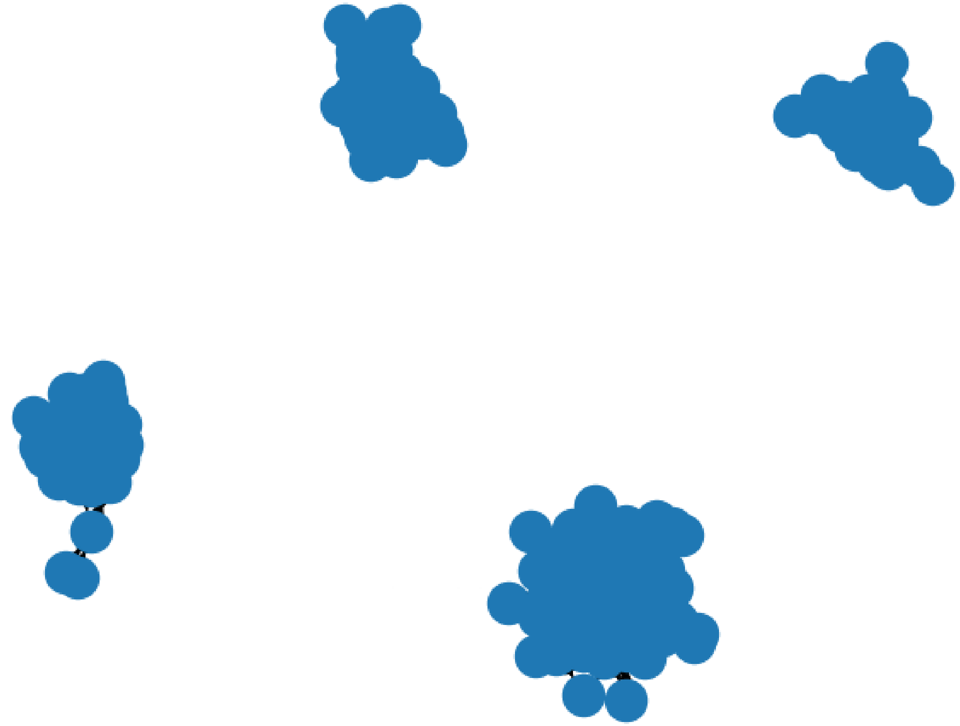
```
def loadGraph(path):
    if path == "data/example1.dat":
        G = nx.read_edgelist(path, delimiter=",", create_using=DiGraph)
    elif  path == "data/example2.dat":
        G = nx.read_weighted_edgelist(path, delimiter=",", create_using=DiGraph)
    else:
        raise NameError("can't fine the correct data.")
    return G

def transform_similarity_matrix(A, k):
    D = np.diag(np.sum(A, axis=1))
    D_inv = np.linalg.inv(np.sqrt(D))
    L = np.dot(np.dot(D_inv, A), D_inv)

    # # Calculate eigenvalues and eigenvectors
    eigenvalues, eigenvectors = scipy.linalg.eigh(L)

    X = eigenvectors[:, -k:]
    norm = np.linalg.norm(X)
    Y = X / norm
    return Y
```

```
def run_kmeans(data, k):
    kmeans_model = KMeans(n_clusters=k, random_state=1)
    kmeans_model.fit(data)
    cluster_labels = kmeans_model.labels_
    return cluster_labels

def retrieve_clusters(nodes, cluster_labels):
    clusters = {}
    for (i, cluster) in enumerate(cluster_labels):
        try:
            clusters[cluster].append(nodes[i])
        except:
            clusters[cluster] = [nodes[i]]
    return clusters

def evaluate_clustering(data, cluster_labels):
    silhouette_score_result = silhouette_score(data, cluster_labels)
    return silhouette_score_result
```

1.4 Load Graph example1.dat

[3]:

```
path = "data/example1.dat"
graph = loadGraph(path)
nx.draw(graph)
```



1.4.1 Find Best k by trying many settings

[4]:

```
best_k = None
max_score = -1
for k in [2, 3 , 4, 5, 6, 7]:
    sim_matrix = np.asarray(nx.adjacency_matrix(graph).todense())
    Y = transform_similarity_matrix(sim_matrix, k)
    cluster_labels = run_kmeans(Y, k)
    score = evaluate_clustering(Y, cluster_labels)
    if score >= max_score:
        max_score = score
        best_k = k
print(f"best k is: {best_k} with silhouette score: {max_score}")
```

```
/Users/athanasiapharmake/Desktop/workspace/MSC/data-
mining/HW4/venv/lib/python3.11/site-packages/sklearn/cluster/_kmeans.py:1416:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
/Users/athanasiapharmake/Desktop/workspace/MSC/data-
mining/HW4/venv/lib/python3.11/site-packages/sklearn/cluster/_kmeans.py:1416:
```

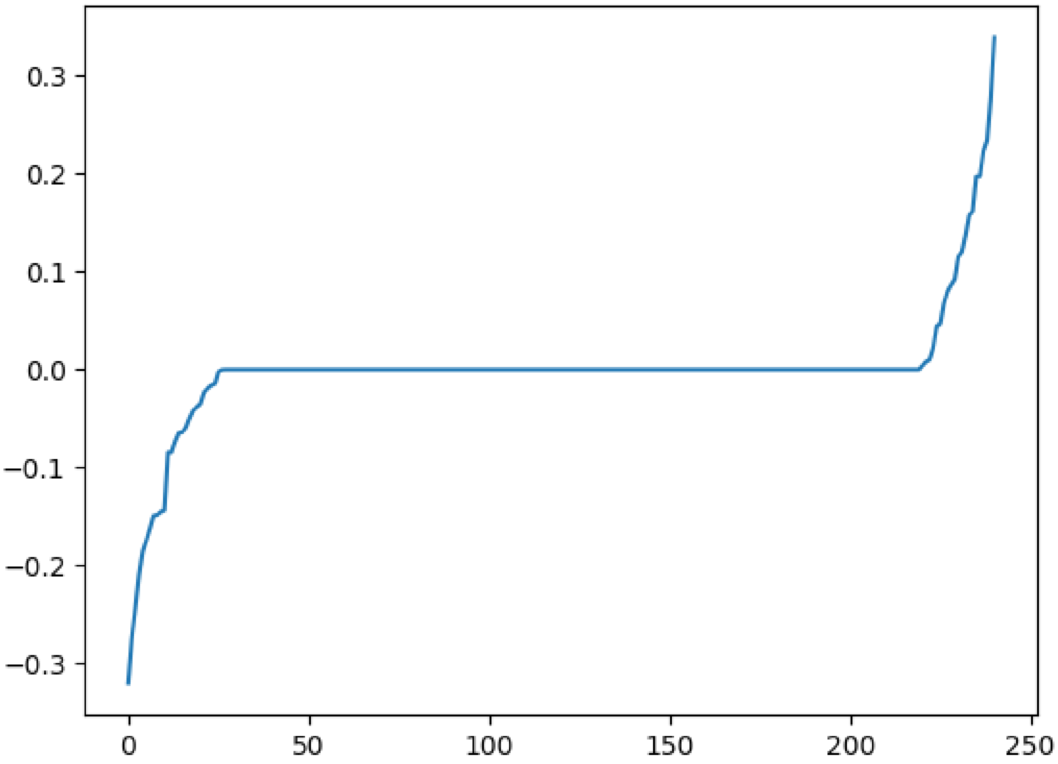
```
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
/Users/athanasiapharmake/Desktop/workspace/MSC/data-
mining/HW4/venv/lib/python3.11/site-packages/sklearn/cluster/_kmeans.py:1416:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
/Users/athanasiapharmake/Desktop/workspace/MSC/data-
mining/HW4/venv/lib/python3.11/site-packages/sklearn/cluster/_kmeans.py:1416:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
/Users/athanasiapharmake/Desktop/workspace/MSC/data-
mining/HW4/venv/lib/python3.11/site-packages/sklearn/cluster/_kmeans.py:1416:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
/Users/athanasiapharmake/Desktop/workspace/MSC/data-
mining/HW4/venv/lib/python3.11/site-packages/sklearn/cluster/_kmeans.py:1416:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
```

```
best k is: 4 with silhouette score: 0.7996277385453353
```

1.5 Fiedler Vector

[5]:

```python
def find_fiedler_vector(A):
    D = np.diag(np.sum(A, axis=1))
    D_inv = np.linalg.inv(np.sqrt(D))
    L = np.dot(np.dot(D_inv, A), D_inv) # Laplacian matrix

    # # Calculate eigenvalues and eigenvectors
    eigenvalues, eigenvectors = scipy.linalg.eigh(L)
    return eigenvectors[:, 2]

sim_matrix = np.asarray(nx.adjacency_matrix(graph).todense())
fiedler_vec = find_fiedler_vector(sim_matrix)
p = plt.plot(range(len(fiedler_vec)), sorted(fiedler_vec))
plt.show()
```



1.6 Run spectral clustering

[6]:
```
k = 4
sim_matrix = np.asarray(nx.adjacency_matrix(graph).todense())
Y = transform_similarity_matrix(sim_matrix, k)
cluster_labels = run_kmeans(Y, k)
```

```
/Users/athanasiapharmake/Desktop/workspace/MSC/data-
mining/HW4/venv/lib/python3.11/site-packages/sklearn/cluster/_kmeans.py:1416:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
```

1.7 Evaluate Clusters

[7]:
```
evaluate_clustering(Y, cluster_labels)
```

[7]:
```
0.7996277385453353
```

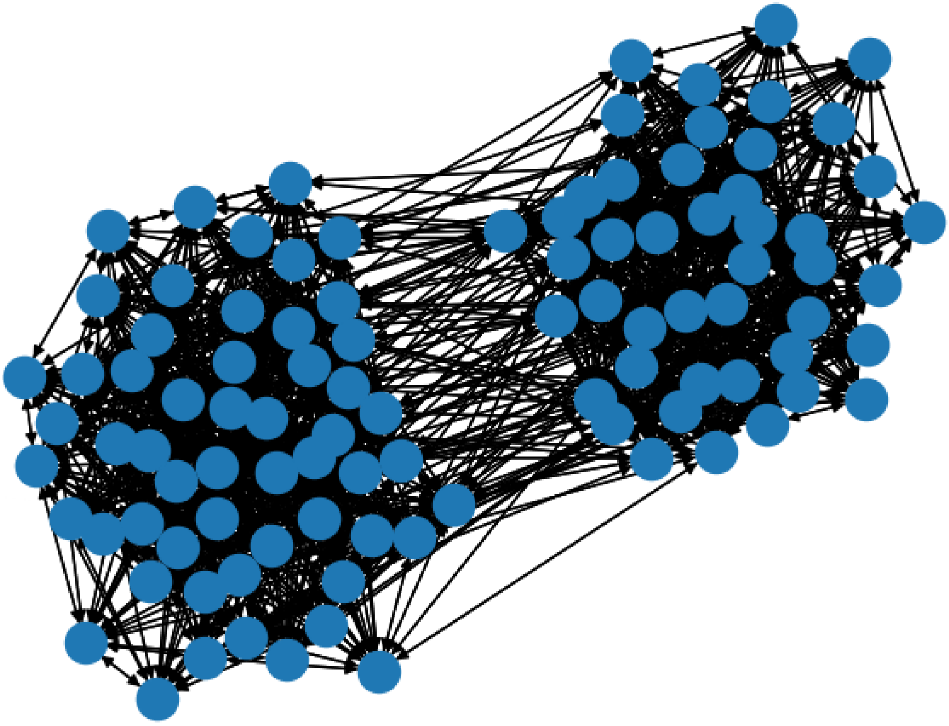
[8]:
```
nodes = list(graph.nodes)
clusters_to_nodes_map = retrieve_clusters(nodes, cluster_labels)
for k, v in clusters_to_nodes_map.items():
    print(f"Num points on cluster {k}: {len(v)}")
```

```
Num points on cluster 0: 117
Num points on cluster 1: 48
Num points on cluster 2: 41
Num points on cluster 3: 35
```

1.8 Load Graph example2.dat

[9]:

```
path = "data/example2.dat"
graph = loadGraph(path)
nx.draw(graph)
```



1.9 Find Best k by trying many settings

[10]:

```
best_k = None
max_score = -1
for k in [2, 3]:
    sim_matrix = np.asarray(nx.adjacency_matrix(graph).todense())
    Y = transform_similarity_matrix(sim_matrix, k)
    cluster_labels = run_kmeans(Y, k)
    score = evaluate_clustering(Y, cluster_labels)
    if score >= max_score:
        max_score = score
        best_k = k
print(f"best k is: {best_k} with silhouette score: {max_score}")
```

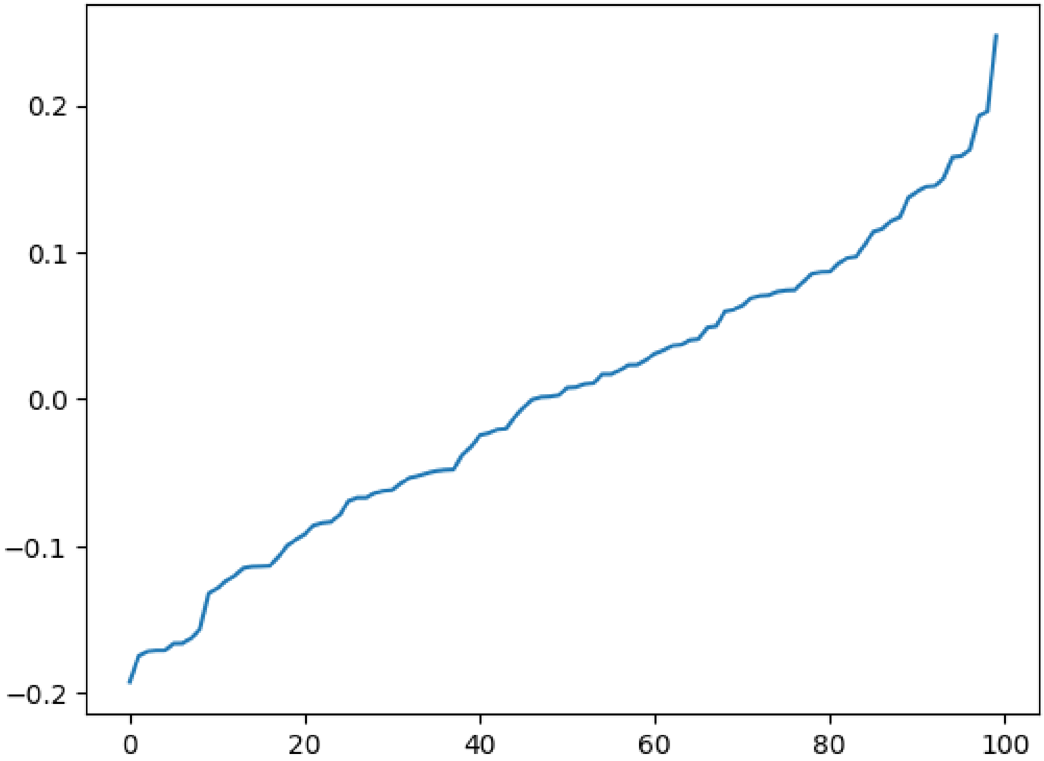
```
/Users/athanasiapharmake/Desktop/workspace/MSC/data-
mining/HW4/venv/lib/python3.11/site-packages/sklearn/cluster/_kmeans.py:1416:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
```

```
best k is: 2 with silhouette score: 0.8950388262484047
```

```
/Users/athanasiapharmake/Desktop/workspace/MSC/data-
mining/HW4/venv/lib/python3.11/site-packages/sklearn/cluster/_kmeans.py:1416:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
```

[11]:

```
def find_fiedler_vector(A):
    D = np.diag(np.sum(A, axis=1))
    D_inv = np.linalg.inv(np.sqrt(D))
    L = np.dot(np.dot(D_inv, A), D_inv) # Laplacian matrix

    # # Calculate eigenvalues and eigenvectors
    eigenvalues, eigenvectors = scipy.linalg.eigh(L)
    return eigenvectors[:, 2]

sim_matrix = np.asarray(nx.adjacency_matrix(graph).todense())
fiedler_vec = find_fiedler_vector(sim_matrix)
p = plt.plot(range(len(fiedler_vec)), sorted(fiedler_vec))
plt.show()
```



1.10 Run spectral clustering

[12]:
```
k = 2
sim_matrix = np.asarray(nx.adjacency_matrix(graph).todense())
Y = transform_similarity_matrix(sim_matrix, k)
cluster_labels = run_kmeans(Y, k)
evaluate_clustering(Y, cluster_labels)
```

```
/Users/athanasiapharmake/Desktop/workspace/MSC/data-
mining/HW4/venv/lib/python3.11/site-packages/sklearn/cluster/_kmeans.py:1416:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
```

[12]:
```
0.8950388262484047
```

[13]:
```
nodes = list(graph.nodes)
clusters_to_nodes_map = retrieve_clusters(nodes, cluster_labels)
for k, v in clusters_to_nodes_map.items():
    print(f"Num points on cluster {k}: {len(v)}")
```

```
Num points on cluster 1: 55
Num points on cluster 0: 45
```

[ ]:

```
```