CS550: Massive Data Mining and Learning Spring 2022

Problem Set 3

Due 11:59pm Monday, Apr 18, 2022

Submission Instructions

Assignment Submission: Include a signed agreement to the Honor Code with this assignment. Assignments are due at 11:59pm. Students should submit their homework via Canvas. Students can typeset or scan their homework. Students also need to include their code in the final submission zip file. Put all the code for a single question into a single file. Finally, put your PDF answer file and all the code files in a folder named as your Name and NetID (i.e., Firstname-Lastname-NetID.pdf), compress the folder as a zip file (e.g., Firstname-Lastname-NetID.zip), and submit the zip file via Canvas.

Late Policy: The homework is due on 4/18 (Monday) at 11:59pm. We will release the solutions of the homework on Canvas on 4/22 (Friday) 11:59pm. If your homework is submitted to Canvas before 4/18 11:59pm, there will no late penalty. If you submit to Canvas after 4/18 11:59pm and before 4/22 11:59pm, your score will be penalized by 0.9^k, where k is the number of days of late submission. For example, if you submitted on 4/21, and your original score is 80, then your final score will be $80\times 0.9^3=58.32$ for $22-18=3$ days of late submission. If you submit to Canvas after 4/22 11:59pm, then you will earn no score for the homework.

Honor Code: Students may discuss homework problems with peers. However, each student must write down their solutions independently to show they understand the solution well enough in order to reconstruct it by themselves. Students should clearly mention the names of all the other students with whom they discussed the homework. Directly using code or solutions obtained from others or from the web is considered an honor code violation. We check all the submissions for plagiarism. We take the honor code seriously and expect students to do the same.

Discussion Group (People with whom you discussed ideas used in your answers):

On-line or hardcopy documents used as part of your answers:

I acknowledge and accept the Honor Code.

(Signed)_____KK_____

If you are not printing this document out, please type your initials above.

Answer to Question 1(a)
Modularity of a network divided into two components:

$$Q = \frac{1}{4m}\sum_{i,j}\left(A_{ij} - \frac{k_i k_j}{2m}\right)s_i s_j$$

After partitioning the graph G after removing edge A-G gives:
Community label vector S = [1,1,1,1,-1,-1,-1,-1]

Adjacency Matrix A =

0	1	1	1	0	0	0	0
1	0	1	1	0	0	0	0
1	1	0	1	0	0	0	0
1	1	1	0	0	0	0	0
0	0	0	0	0	1	1	0
0	0	0	0	1	0	1	0
0	0	0	0	1	1	0	1
0	0	0	0	0	0	1	0

Degree distribution k = [3,3,3,3,2,2,3,1]
Nodes m = 10
Code:

```
def modularity(A,k,s):
  res = 0
  for i in range(len(A)):
    for j in range(len(A)):
      res += (A[i][j] - (k[i]*k[j])/(2*m))*s[i]*s[j]
  return res/(4*m)
```

A-G removed:

```
# A-G removed
A =
[[0,1,1,1,0,0,0,0],[1,0,1,1,0,0,0,0],[1,1,0,1,0,0,0,0],[1,1,1,0,0,0,0,0],[0,0,0,0,0,1,1,0
],[0,0,0,0,1,0,1,0],[0,0,0,0,1,1,0,1],[0,0,0,0,0,0,1,0]]
k =  [3,3,3,3,2,2,3,1]
s = [1,1,1,1,-1,-1,-1,-1]
m = 10
modularity(A,k,s)
```

0.48

From the above values we get $\mathbf{Q_{A\text{-}G\ removed} = 0.48}$

```
# original graph
A =
[[0,1,1,1,0,0,1,0],[1,0,1,1,0,0,0,0],[1,1,0,1,0,0,0,0],[1,1,1,0,0,0,0,0],[0,0,0,0,0,1,1,0
],[0,0,0,0,1,0,1,0],[1,0,0,0,1,1,0,1],[0,0,0,0,0,0,1,0]]
k =  [3,3,3,3,2,2,3,1]
s = [1,1,1,1,-1,-1,-1,-1]
m = 11
modularity(A,k,s)
```

0.3925619834710744

For the original graph, k = [3,3,3,3,2,2,3,1] and m = 11

The value of Q if A-G is not removed, i.e., the modularity of the original graph is $\mathbf{Q_{original} = 0.392}$

Answer to Question 1(b)

Adding E-H to the original graph changes the adjacency matrix A,k,m. S remains the same because the partitioning is still on the edge A-G but it is not removed from the graph.

New values of A,k,m:

A =

$$\begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

k = [4,3,3,3,3,2,4,2]

m= 12

Code:

```
# E-H added
A =
[[0,1,1,1,0,0,1,0],[1,0,1,1,0,0,0,0],[1,1,0,1,0,0,0,0],[1,1,1,0,0,0,0,0],[0,0,0,0,0,1,1,1
],[0,0,0,0,1,0,1,0],[1,0,0,0,1,1,0,1],[0,0,0,0,1,0,1,0]]
k =  [4,3,3,3,3,2,4,2]
s = [1,1,1,1,-1,-1,-1,-1]
m = 12
modularity(A,k,s)
```

0.4131944444444444

From the above new values, we get $\mathbf{Q_{E\text{-}H\ added} = 0.413}$

Adding an edge inside the same community increases the intra-community connectivity. SInce E,H belong to same community, the product of their community label vectors is 1. This translates to an increase in Q when compared to the value of Q when there was no edge E-H in the original graph, i.e., 1(a): Q= 0.39256. The modularity increases when compared to the original graph.

Answer to Question 1(c)
We are adding an edge between F-A in the original graph. Let's see how this affects the modularity.
We recalculate A,k,m:

A =

0	1	1	1	0	1	1	0
1	0	1	1	0	0	0	0
1	1	0	1	0	0	0	0
1	1	1	0	0	0	0	0
0	0	0	0	0	1	1	0
1	0	0	0	1	0	1	0
1	0	0	0	1	1	0	1
0	0	0	0	0	0	1	0

k = [5,3,3,3,2,3,4,1]
m= 12

Code:

```
# F-A added to original
A =
[[0,1,1,1,0,1,1,0],[1,0,1,1,0,0,0,0],[1,1,0,1,0,0,0,0],[1,1,1,0,0,0,0,0],[0,0,0,0,0,1,1,0
],[1,0,0,0,1,0,1,0],[1,0,0,0,1,1,0,1],[0,0,0,0,0,0,1,0]]
k =  [5,3,3,3,2,3,4,1]
s = [1,1,1,1,-1,-1,-1,-1]
m = 12
modularity(A,k,s)
```

0.3194444444444445

From the above values, we get Q = 0.319
The modularity of the graph decreases when compared to the original graph because the edge is added between two communities.

Answer to Question 2(a)

The adjacency matrix(A) =

$$\begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

The degree matrix(D) =

$$\begin{bmatrix} 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Laplacian = D - A =

$$\begin{bmatrix} 4 & -1 & -1 & -1 & 0 & 0 & -1 & 0 \\ -1 & 3 & -1 & -1 & 0 & 0 & 0 & 0 \\ -1 & -1 & 3 & -1 & 0 & 0 & 0 & 0 \\ -1 & -1 & -1 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & -1 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 2 & -1 & 0 \\ -1 & 0 & 0 & 0 & -1 & -1 & 4 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \end{bmatrix}$$

Answer to Question 2(b)

```
import numpy as np
from scipy import linalg
L =
np.asmatrix([[4,-1,-1,-1,0,0,-1,0],[-1,3,-1,-1,0,0,0,0],[-1,-1,3,-1,0,0,0,0],[-1,-1,-1,3,
0,0,0,0],[0,0,0,0,2,-1,-1,0],[0,0,0,0,-1,2,-1,0],[-1,0,0,0,-1,-1,4,-1],[0,0,0,0,0,0,-1,1]
])
w,v = linalg.eigh(L)
sorted_indexes = (-w).argsort()
w = w[sorted_indexes]
v  = v[:,sorted_indexes]
print("Eignen Values of L:\n",w,"\nEigen vectors of L:\n",v)
```

Eignen Values of L:
[5.64575131e+00 4.00000000e+00 4.00000000e+00 4.00000000e+00
3.00000000e+00 1.00000000e+00 3.54248689e-01 2.14038189e-16]
Eigen vectors of L:
[[6.62557346e-01 -7.96411860e-02 0.00000000e+00 6.07171542e-01
0.00000000e+00 0.00000000e+00 -2.47017739e-01 -3.53553391e-01]
[-1.42615758e-01 -5.60530944e-01 5.62206567e-01 -2.79396081e-01
-2.70599246e-17 -3.18493382e-17 -3.82527662e-01 -3.53553391e-01]
[-1.42615758e-01 8.02836107e-01 2.31676233e-01 -1.00566604e-01
-1.12776339e-16 8.59836280e-17 -3.82527662e-01 -3.53553391e-01]
[-1.42615758e-01 -1.62663977e-01 -7.93882800e-01 -2.27208857e-01
1.50488323e-16 -5.79022479e-17 -3.82527662e-01 -3.53553391e-01]
[1.42615758e-01 2.65470620e-02 2.16840434e-16 -2.02390514e-01
7.07106781e-01 -4.08248290e-01 3.82527662e-01 -3.53553391e-01]
[1.42615758e-01 2.65470620e-02 -2.82759927e-16 -2.02390514e-01
-7.07106781e-01 -4.08248290e-01 3.82527662e-01 -3.53553391e-01]
[-6.62557346e-01 -7.96411860e-02 -1.11022302e-16 6.07171542e-01
-1.06520593e-17 3.76795815e-18 2.47017739e-01 -3.53553391e-01]
[1.42615758e-01 2.65470620e-02 -1.71737624e-16 -2.02390514e-01
1.12242936e-16 8.16496581e-01 3.82527662e-01 -3.53553391e-01]]

Each column in the eigenvectors matrix is the eigenvector of the corresponding value in the eigenvalues list.

Answer to Question 2(c)
The second smallest eigenvalue is **0.354248689**
The corresponding eigenvector:
[-0.247017739, -0.382527662, -0.382527662, -0.382527662, 0.382527662, 0.382527662, 0.247017739, 0.382527662]

Partition of the graph using 0 as boundary:

Community 1:

Node	Eigenvector
A	-0.247017739
B	-0.382527662
C	-0.382527662
D	-0.382527662

Community 2:

Node	Eigenvector
E	0.382527662
F	0.382527662
G	0.247017739
H	0.382527662

Answer to Question 3(a)

Given, any two nodes are connected if they have a common factor other than 1. The question states that "the set of nodes C_i of G are divisible by i", this implies that every node in C_i has i in its factors. This is a satisfying condition on which the community is built. So every node in C_i will be connected to each other, hence making this a clique.

Answer to Question 3(b)

C_i is a maximal clique for every prime i where $i \in [2, 1000000]$

If i is greater than 1000000, then C_i is an empty clique, and adding any node to C_i will produce a 1-clique. Thus C_i is not maximal.

If i is less than or equal to 1000000, but is not a prime, let j be a factor of i, with $1 < j < i$. Node j is not in C_i, yet it has an edge to every member of C_i, because it has j as a common factor. Thus C_i is not maximal.

if i is prime, and less than or equal to 1000000, then there is no node outside C_i that has an edge to the node i. suppose j were such a node. Then i and j have a common factor other than 1, which can only be i, since i is prime. However, if j has i as a factor, then j is a multiple of i and therefore already in C_i.

Answer to Question 3(c)

There are no cliques other than C_2 that have 500,000 nodes. In any other clique, there need to be exactly 500000 nodes to compete with C_2 and the rest of the primes take portions of the rest of 500000 nodes to make their cliques. So none of them are larger than C_2.

Also, from our previous argument, 2 is a prime and C_2 is maximal. Out of all the prime numbers, i=2 has the maximum number of multiples in the set of nodes, hence C_2 has the maximum number of elements.

Link to codes: https://colab.research.google.com/drive/1U-dY3rOrh2ueMAHsGWiNgMMT22V4jUO-?usp=sharing