Birla Institute of Technology & Science, Pilani
First Semester 2023-2024
Graph Mining (CS F426)
Mid-Semester Exam-Part A

Date : Oct 12, 2023 Duration: 45 minutes
Nature of Exam : Closed Book Maximum Marks: 40

Q1. For a given vector [2, 1, 3], compute its unit vector in the same direction as the original vector. **[1 marks]**

Q2. In a web graph, if all the hub and authority scores are initialized to 1, what would be the hub/authority score of a node after one iteration? Prove your answer. **[2 marks]**

Q3. Which of the following facts about the Laplacian and adjacency matrices are True/False? **[5 marks]**

a. The all-1s vector is always an eigenvector of LG of eigenvalue 0.
b. The eigenvectors of the largest k eigenvalues of Laplacian (LG) provide a lot of information about graph. In contrast, the eigenvectors of the adjacency (AG) correspond to the smallest eigenvalues are useful.
c. The multiplicity of 0 as an eigenvalue of LG is equal to the number of connected components of LG.
d. For a regular graph, if every vertex has the same degree, AG and LG will have the same eigenvectors.
e. The unnormalized graph Laplacian does not depend on the diagonal elements of the adjacency matrix.

Q4. Below is an example of a utility matrix, representing relationship between users' and movies ratings on a 1-5 scale, with 5 is the highest rating. Blanks represent the situation where user has not seen that movie. The movie names are HP1, HP2, and HP3 for harry potter I, II and III, TW for twilight and SW 1, SW2 and SW3 for star wars episodes 1, 2 and 3. Users are represented as A, B, C, and D. **[4+2=6 marks]**

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

a. Compute the Jaccard similarity and Jaccard distance between A & B and A & C and identify whether A is closer to B than C or vice-versa.
b. Do you observe any pitfalls in comparing the closeness using Jaccard measure? If yes, identify?

Q5. Using above utility matrix, create a directed and weighted bipartite graph between users and the movies. **[1+4+1+2=8 marks]**
(a) Calculate the user-movie matrix X.
(b) Using X, computes the user-user similarity (M) and movie-movie similarity (N) matrices.
c) How similar each user is to user B compared to each other?
d) What does it mean for a matrix to be symmetric? Are the user-user and movie-movie matrices symmetric? Justify your answer.

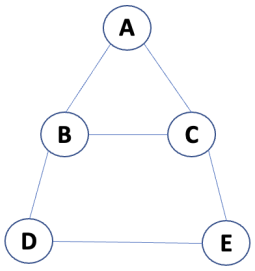
Q6. a) For any symmetric 3x3 matrix **[2+6=8 marks]**

$$[a-\lambda \quad b \quad c\ b \quad d-\lambda \quad e \quad c \quad e \quad f-\lambda]$$

There is cubic equation in λ that says the determinant of this matrix is zero. Find this equation in terms of a through f.
b) Using above equation find the eigenpair of the following matrix:

$$[1 \quad 1 \quad 1\ 1 \quad 2 \quad 3\ 1 \quad 3 \quad 5]$$

Q7. For the below graph with (A-I) nodes, use the Girvan-Newman approach to compute the following **[2+6+2 =10 marks]**



(a) Find the total number of shortest paths for every nodes from nodes ***i)*** C and ***ii)*** E.
(b) Compute the betweenness of each edge in the given graph.
(c) Draw the dendrogram to represent the communities after removing edges with high betweenness until no edge is left.
**********END**********

Graph Mining - CSF426

Mid-semester Exam 23-24: Part-B

Duration: 45 mins

Total Marks: 30

Open book

Instructions:

1. Rename code file with your "firstname and student id".

3. Students are required to fill the blanks with code syntax in the sections provided below TO-DO comments.

Objective:

This programming exercise is designed to solve two tasks: **A. Feature diffusion B. Label prediction**.

It uses karate club graph with 34 nodes, 156 weighted edges. Nodes are divided into 2 classes: "Mr.Hi" and "Officer". Deatils of tasks within A and B are provided in below sections

1. Packages

```
import matplotlib.pyplot as plt
import networkx as nx
import numpy as np
import pandas as pd
from sklearn.metrics import accuracy_score
```

2. Loading karate club graph with 34 nodes and 78 edges with weights (TO-DO) [1+2=3 marks]

```
G = nx.karate_club_graph()
print("Nodes in G are:", ______)
print("Edges with weights in G are:", ____________)
```

```
# Expected output:
Nodes in G are: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31
Edges with weights in G are: {(0, 1): 4, (0, 2): 5, (0, 3): 3, (0, 4): 3, (0, 5): 3, (0, 6): 3, (0, 7): 2, (0, 8): 2, (0, 10): 2, (0,
```

2.1 Convert Graph G into adjacency matrix A (TO-DO) [2 marks]

```
A = nx.adjacency_matrix(G).todense() # Weighted A
____________ # Coverting weighted adjacency A into unweighted A
print(A)
```

```
# Expected output:
[[0 1 1 ... 1 0 0]
 [1 0 1 ... 0 0 0]
 [1 1 0 ... 0 1 0]
 ...
 [1 0 0 ... 0 1 1]
 [0 0 1 ... 1 0 1]
 [0 0 0 ... 1 1 0]]
```

2.2 Compute A_hat by adding self edges (TO-DO) [2 mark]

```
A_hat = _______
print(A_hat)
```

```
# Expected output
[[1. 1. 1. ... 1. 0. 0.]
 [1. 1. 1. ... 0. 0. 0.]
 [1. 1. 1. ... 0. 1. 0.]
 ...
 [1. 0. 0. ... 1. 1. 1.]
 [0. 0. 1. ... 1. 1. 1.]
 [0. 0. 0. ... 1. 1. 1.]]
```

2.3 Normalize A_hat using below method (TO-DO) [2 marks]

$A\_tilde = D^{-1/2}.A\_hat.D^{-1/2}A$ where D is diagonal matrix of A_hat

```
d_inv_sqrt = np.diag(np.sqrt(np.sum(A_hat, axis = 1)**-1))
A_tilde = ____________________
print(A_tilde)
```

```
Expected output:
array([[0.05882353, 0.0766965 , 0.07312724, ..., 0.09166985, 0.        ,
        0.        ],
       [0.0766965 , 0.1       , 0.09534626, ..., 0.        , 0.        ,
        0.        ],
       [0.07312724, 0.09534626, 0.09090909, ..., 0.        , 0.0836242 ,
        0.        ],
       ...,
       [0.09166985, 0.        , 0.        , ..., 0.14285714, 0.10482848,
        0.08908708],
       [0.        , 0.        , 0.0836242 , ..., 0.10482848, 0.07692308,
        0.06537205],
       [0.        , 0.        , 0.        , ..., 0.08908708, 0.06537205,
        0.05555556]])
```

3. Compute degree of each node and list top 4 nodes with highest degree (TO-DO) [3 marks]

```
_____________________________ # degree of each node in descending order
_____________________________ # Ids of Top-4 nodes based on degree
```

```
[(33, 17), (0, 16), (32, 12), (2, 10)]
```

3.1 Initialize a point feature of top four nodes with their respective normalized degree and others with zero (TO-DO) [2 mark]

```
_________________# Initialize feature vector
```

4 (Feature Diffusion:)

Aim is to learn 1-dim feature for each node in graph G. Initally we assume features of a few nodes and learn for other through diffusion.

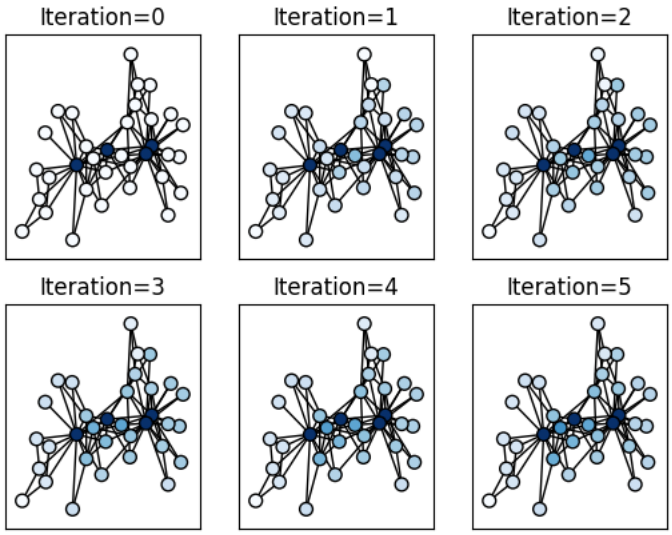
4.1 Perform feature diffusion using F=X(t+1)=A.X(t) and preserve the original features in every iteration and Plot results before and after each diffusion step (TO-DO) [5 marks]

```
iter = 5 # Number of diffusion steps.
F = X # copy initial signals
results = [X] # Record the inital signal of nodes
for i in range(iter):
  # F =   _________  # Compute signals in each diffusion step
  # F = __________ # Preserve initial signals in F
  results.append(F) # Record signals after each diffusion step

# Plot results before and after each diffusion step i.e. X0,X1,X2,X3,X4,X5

pos = nx.spring_layout(G)
fig, axes = plt.subplots(nrows=2, ncols=3)
ax = axes.flatten()

for i in range(iter+1):
    ax[i].set_title(f'Iteration={i}')
    nodes = nx.draw_networkx_nodes(G, ax= ax[i] ,pos = pos, node_size=50, node_color=results[i], cmap=plt.cm.Blues)
    nodes.set_edgecolor("#000000")
    nx.draw_networkx_edges(G, pos = pos, ax=ax[i])
```



4.2 Plot normalized degrees and features learned for each node after diffusion (TO-DO) [4 marks]

```
    # Draw line plot on for degree and features vs node ids
```

5. (Label propagation)

Aim is to utilize the labels of top-k {2, 3, and 4} highly connected nodes and predict for others.

5.1 One hot encoding

Converting node class labels into one hot encoding e.g. Two class Problem: labels = {"Mr. Hi", "officer"} 1-hot encoding will be: Mr. Hi =[1, 0] Officer =[0, 1]

```
labels = nx.get_node_attributes(G, "club")
y = []
for key in labels.keys() :
    y.append(labels[key])
y = pd.get_dummies(np.array(y)).values
```

5.2 Perform label propagation using below steps:

a. Split data into train and test: Consider Top-4 highest nodes as train nodes (train_n) and others as test nodes.

b. Using labels of train_n, predict the labels of test nodes.

c. Repeat the same analysis for Top-3 and Top-2 nodes and compute the accuracy in all three cases. (TO-DO) [7 marks]

Perform label prediction using $Y\_pred=(I-\alpha D^{-1/2}A D^{-1/2})^{-1}.Y\_train$

```
alpha = 0.2
train_n = _________ # indices of training nodes
y_train = ________ # labels vector used for training

y_pred = _________ # perdicted labels after diffusion

score = accuracy_score(_____, _______)
```