Homework 1

Jiaxin XU

September 4, 2022

Problem 1: Katz centrality

$$\mathbf{c}_{Katz} = \beta(\mathbf{I} - \alpha\mathbf{A})^{-1}\mathbf{1}.$$

α is a positive constant. When we let $\alpha \to 0$, then all the vertices have the same centrality β As we increase α from 0, the centrality calculated will increase and then comes to a divergence point, where $(\mathbf{I} - \alpha\mathbf{A})^{-1}$ diverges. That is when

$$\det(\mathbf{I} - \alpha\mathbf{A}) = \det(\mathbf{A} - \alpha^{-1}\mathbf{I}) = 0.$$

As α increases, the determinant first crosses 0 when $\alpha = 1/k_1$, where k_1 is the largest eigenvalue of $\mathbf{A}$. Therefore, when α is less than $1/k_1$, the expression for Katz centrality will converge.

Problem 2: Use "walk" to compute the total number of common neighbors $|N(v_i) \cap N(v_j)|$ between nodes v_i and v_j.

$$|N(v_i) \cap N(v_j)| = N_{ij}^{(2)},$$

where $N_{ij}^{(2)}$ is the number of walks of length 2 from v_i to v_j,

$$N_{ij}^{(2)} = \sum_{k=1}^{n} A_{ik} A_{kj} = [A^2]_{ij},$$

and A is the adjacency matrix.

Problem 3: See Appendix for code. The similarity plot is shown as follows:

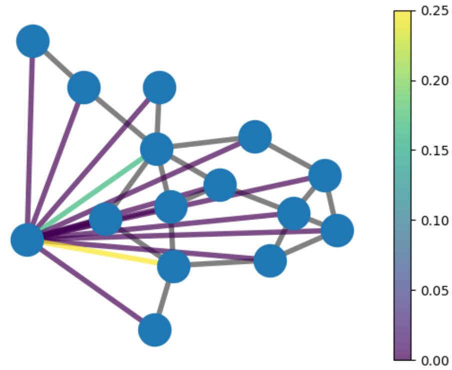

Figure 1: Jaccard's similarity between "Ginori" family and other families in the Florentine Families graph, edge colored by the corresponding similarity values.

Appendix

```python

"""
ACMS 80770-03: Deep Learning with Graphs
Instructor: Navid Shervani-Tabar
Fall 2022
University of Notre Dame

Homework 1: Programming assignment
"""

from operator import le
from platform import node
import networkx as nx
import matplotlib.pyplot as plt
import numpy as np
from networkx.algorithms import bipartite
from networkx.generators.random_graphs import erdos_renyi_graph
import copy


# -- Initialize graphs
seed = 30
G = nx.florentine_families_graph()
nodes = G.nodes()
layout = nx.spring_layout(G, seed=seed)


# -- compute jaccard's similarity
"""
    This example is using NetwrokX's native implementation to compute similarities.
    Write a code to compute Jaccard's similarity and replace with this function.
"""
# pred = nx.jaccard_coefficient(G)
def my_jaccard_similarity(G):
    nodes = list(G.nodes()) # the node names list
    A = nx.to_numpy_array(G)
    # matrix of total number of shared neighbors (intersection)
    A_cap = np.matmul(A,A)
    # matrix of total number of neighbors (union)
    A_cup = np.zeros_like(A)
    for i in range(len(A)):
        for j in range(len(A)):
            A_cup[i][j] = sum(A[i])+sum(A[j])-A_cap[i][j]

    # Jaccard's similarity matrix
    S = A_cap/A_cup

    return ((nodes[i],nodes[j],S[i][j]) for i in range(len(A)) for j in range(len(A))
    )

pred = my_jaccard_similarity(G)

```

```python

# -- keep a copy of edges in the graph
old_edges = copy.deepcopy(G.edges())

# -- add new edges representing similarities.
new_edges, metric = [], []
for u, v, p in pred:
    G.add_edge(u, v)
    print(f"({u}, {v}) -> {p:.8f}")
    new_edges.append((u, v))
    metric.append(p)

# -- plot Florentine Families graph
nx.draw_networkx_nodes(G, nodelist=nodes, label=nodes, pos=layout, node_size=600)
nx.draw_networkx_edges(G, edgelist=old_edges, pos=layout, edge_color='gray', width=4)

# -- plot edges representing similarity
"""
    This example is randomly plotting similarities between 8 pairs of nodes in the graph.
    Identify the    Ginori
"""
## Identify the    Ginori
Ginori_edge_ls = []
Ginori_metric_ls = []
for i in range(len(new_edges)):
    if new_edges[i][0] == 'Ginori' and new_edges[i][1] != 'Ginori':
        Ginori_edge_ls.append(new_edges[i])
        Ginori_metric_ls.append(metric[i])
## plot
ne = nx.draw_networkx_edges(G, edgelist=Ginori_edge_ls, pos=layout, edge_color=np.asarray(Ginori_metric_ls), width=4, alpha=0.7)
plt.colorbar(ne)
plt.axis('off')
plt.show()
```

Listing 1: Python code