经济管理学院

课　程　报　告

（复杂网络与社会计算）

题　　目：week8 课程作业

课程教师：赵吉昌

学院/专业：信息管理与信息系统

学生姓名：范春

学　　号：21377061

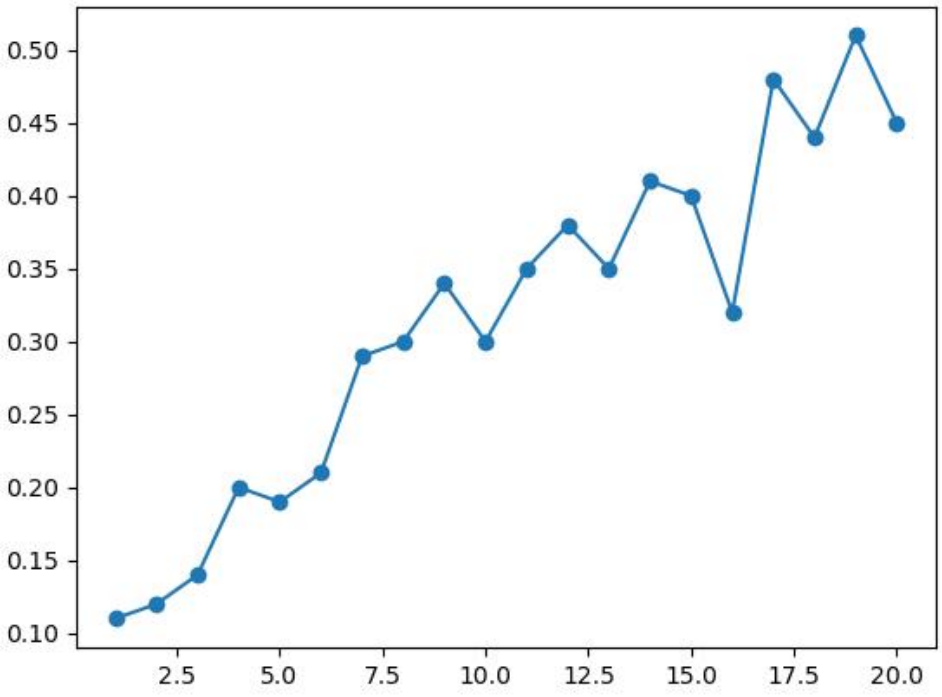
2024 年 4 月 16 日

北京航空航天大學
BEIHANG UNIVERSITY

作业要求如下：

2. 1中的参考资料中提到，当网络稀疏化后，临界阈值会明显变小。但需要注意的是，其设定仍然是均匀网络，即所有节点度是一样的。现考虑在真实的网络（稀疏，不均匀）中对其理论模型进行仿真，并观察不同的M下，是否会出现所谓的“社会习惯”变化，即得到临界阈值与M的关系（如1中的图Fig1(B)）。真实的社交网络可使用http://snap.stanford.edu/data/gemsec-Deezer.html中提到的数据。另外，仿真轮数等参数可以按1做类似设定。

运行结果如下：



完整代码如下：

```
import networkx as nx
import pandas as pd
import numpy as np
import random
import matplotlib.pyplot as plt
import copy

def generateNet(filePath):
    df = pd.read_csv(filePath)
    G = nx.Graph()
    for index, row in df.iterrows():
        G.add_edge(int(row['node_1']), int(row['node_2']))

    # 选择一个随机的初始节点
    initial_node = random.choice(list(G.nodes))

    # 执行随机游走直到子图包含 1000 个节点
    visited = set([initial_node])
    current_node = initial_node
    while len(visited) < 1000:
        neighbors = list(G.neighbors(current_node))
```

```
        if not neighbors:
            break
        current_node = random.choice(neighbors)
        visited.add(current_node)

    # 创建子图
    subgraph = G.subgraph(visited)
    return subgraph

# 代理类
class Agent:
    def __init__(self, memory_length):
        self.memory_length = memory_length
        self.memory = ['A'] * memory_length

    def update_memory(self, info):
        self.memory.pop(0)
        self.memory.append(info)

    def update_allmemory(self,info):
        self.memory = [info]*self.memory_length

    def get_most_common_info(self):
        return max(set(self.memory), key=self.memory.count)

    def check_B_majority(self):
        count_B = self.memory.count('B')
        half_length = len(self.memory) /2
        if count_B > half_length:
            return 1
        else:
            return 0

def simulate(G):
    results = {}
    for M in range(1,21):
        agents = {node: Agent(M) for node in G.nodes()}
        for C in np.arange(0.1,0.61,0.01):
            total_B_majority = 0
            agents_copy = copy.deepcopy(agents)
            minority_agents = random.sample(list(agents_copy.keys()), int(C*100))
            #将选择的少数的记忆更新
            for node in minority_agents:
                agents_copy[node].update_allmemory('B')
```

```
66.
67.             for T in range(1000):
68.                 for N in range(1000):
69.                     selected_edge = random.choice(list(G.edges()))
70.                     speaker, hearer = random.sample(selected_edge, 2)
71.
72.                     if hearer not in minority_agents:
73.                         if speaker in minority_agents:
74.                             agents_copy[hearer].update_memory('B')
75.                         else:
76.                             agents_copy[hearer].update_memory(agents_copy[speaker].get_most_common
    _info())
77.
78.             total_B_majority += sum(agents_copy[node].check_B_majority() for node in agents_copy)-
    C*1000
79.             if total_B_majority/(1000-C*1000)>=0.9: #由于随机性可能存在边始终未选中，因此此处设比例大于
    0.9 即实现社会习惯转变
80.                 results[M]=C
81.                 break
82.     return results
83.
84. def main():
85.     filePath = "C:\\Users\\ 范 春
    \\Desktop\\week8\\gemsec_deezer_dataset\\deezer_clean_data\\HR_edges.csv"
86.     G = generateNet(filePath)
87.     results = simulate(G)
88.     M = list(results.keys())
89.     tipping_point = list(results.values())
90.
91.     plt.plot(M, tipping_point, marker='o')
92.     plt.show()
93.     df = pd.DataFrame({'M':M,'tipping_point':tipping_point})
94.     df.to_excel("C:\\Users\\范春\\Desktop\\week8\\results.xlsx",index=False)
95.
96. if __name__=='__main__':
97.     main()
```