西安电子科技大学
XIDIAN UNIVERSITY

数据挖掘第三次上机实验报告

基于层次聚类的复杂网络社团检测

姓名：刘臻劼

学号：19030700016

班级：1903071

导师：郭杏莉

目录

1 实验内容

1.1 实验目标

本次实验目标为：基于空手道网络数据，根据网络结构特征给出结点相似性度量指标并对网络数据进行聚类。同时计算模块性指标 Q 值，当 Q 值最大时输出聚类结果并可视化。

1.2 数据集介绍

Karate Club 数据集[1]描述了一个空手道俱乐部会员的社交关系，以 34 名会员作为节点，如果两位会员在俱乐部之外仍保持社交关系，则在节点间增加一条边。在社会学家 Zachary 收集数据的过程中，管理人员 John A 和教练 Mr. Hi 之间产生了冲突，会员们选择了站队，一半会员跟随 Mr. Hi 成立了新俱乐部，剩下一半会员找了新教练或退出了俱乐部。

俱乐部社交网络结构可视化如下:

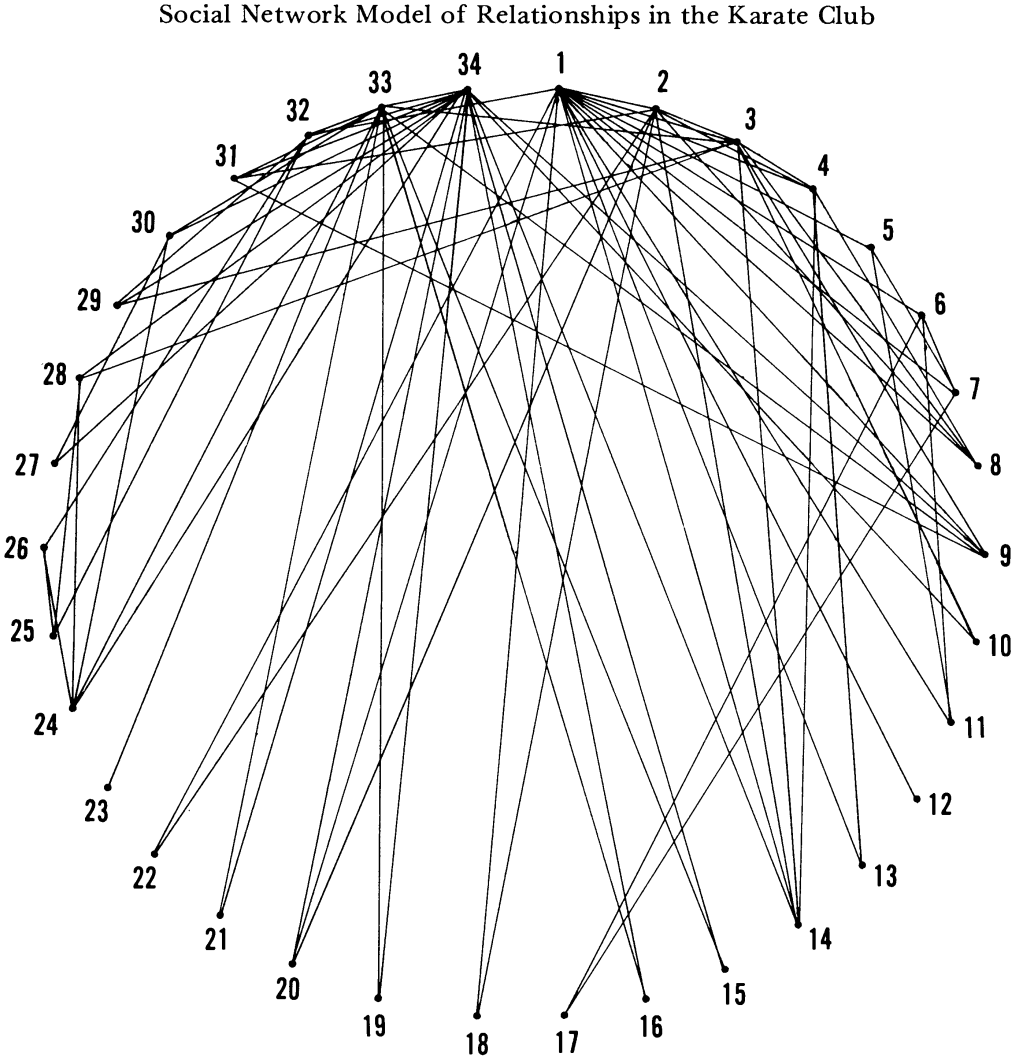

图 1: 社交网络结构图

2 实验分析

2.1 网络建模与结点相似度构造

对于图 1 所示的社交网络，建模如下：将每个成员视作一结点，两成员有社交关系则对应的两个结点间存在一条边，由上述规则构造 $G =< U, V >$。其中 $U = \{u_1, u_2, \cdots u_n\}, n = 34$ 为结点集，V 为边集。同时有邻接矩阵 $A_{n\times n}$，其中

$$a_{ij} = \begin{cases} 1 & < u_i, u_j > \in V \\ 0 & otherwise \end{cases} \tag{1}$$

我们定义结点的邻居结点：

$$N(u_i) = \{u_j | A_{ij} = 1; j = 1, 2, \cdots, n; i \neq j\} \tag{2}$$

则两结点相似性定义为其邻居节点交集的基数比上二者邻居节点并集的基数。即公共邻居节点所占总邻居节点的比例。公式如下：

$$Similar(u_i, u_j) = \frac{|N(u_i) \cap N(u_j)|}{|N(u_i) \cup N(u_j)|} \tag{3}$$

2.2 层次聚类算法

对于图 1 所示的社交网络结构，采用如下的层次聚类算法对其进行聚类分析。

Algorithm 1 层次聚类算法

Output: 层次聚类结果。

1: 将所有节点看做一个单独的簇
2: 遍历 similar 矩阵，得到相似度最高的两个簇
3: 选取最高相似度，检测高于阈值则输出聚类结果
4: 将相似度最高的两个簇合并，并更新 similar 矩阵：将合并的两个簇中的最小 similar 值更新为新的簇的值，记录合并的簇
5: 聚类结果稳定后停止，计算模块度量值 Q，输出 Q 值最大时的聚类结果

2.3 模块度评价指标 Q

2004 年 Newman 等人[2]提出了模块度 Q，之后广泛应用于衡量社区的划分质量。其基于这样的考虑：划分的两个社区间边应尽量少；对于如图 1 所示的无向无权的同质网络，Q 值可按如下公式计算：

$$Q_i == \frac{1}{2|U|} \sum_{ij} (a_{ij} - \frac{k_i k_j}{2|U|}) \delta(C_i, C_j) \tag{4}$$

其中：k_i 为结点 u_i 的度；C_i 为结点 u_i 的类别号，二值函数 δ 刻画两个类别号是否相等，若 $C_i = C_j$，则 $\delta(C_i, C_j) = 1$，反之则为零。

3 具体实现

3.1 数据读取

首先读取数据：

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

graph = ig.Graph.Read_GML('karate.gml')  # Karate Club
G = nx.read_gml('karate.gml', label='id')
M = graph.vcount()  # m=34
N = graph.ecount()  # n=78
edgelist = graph.get_edgelist()  # 边的list
neighbors = graph.neighborhood()  # 与各个点相连的其他点的集合
k = graph.degree()  # 各个点的度
Q_list = []
```

求解邻居结点与邻接矩阵：

```
neighbors=[]
for i in range(34):
    neighbors.append(g.neighbors(vertex=i))

A = np.zeros((m, m), dtype=np.int)
    for i in range(m):
        for j in range(m):
            if i is j:
                break
            # 由邻节点构建邻接矩阵
            if j in neighbors[i]:
                A[i][j] = 1
                A[j][i] = 1
```

3.2 求解相似度矩阵

```
similar=np.zeros((34,34))
for i in range(34):
    for j in range(34):
        p=q=[]
        for k in range(34):
            if(i in neighbors[k] and j in neighbors[k]):     # AND
                p.append(k)
            if(i in neighbors[k] or j in neighbors[k]):      # OR
                q.append(k)
        if(i!=j):
            similar[j][i]=similar[i][j]=len(p)/len(q)
```

3.3 定义层次聚类函数

```
def cluster(sim):
    m = len(sim)
    max_s=x=y=0
    # 选出最大相似度
    for i in range(m):
        for j in range(m):
            if(similar[i][j] > max_s):
                x = i
                y = j
                max_s = sim[i][j]
    # 全链算法更新相似度矩阵
    for i in range(m):
        similar[x][i] = min(similar[x][i], similar[y][i])
        similar[y][i] = min(similar[x][i], similar[y][i])
    return x, y
```

3.4 定义 Q 值计算函数

```
def compute_Q(f, adjacency):
    Q = 0
    for i in range(M):
```

```
        for j in range(M):
            if f[i] == f[j]:
                Q += (adjacency[i][j]-(degrees[i]*degrees[j]/(2*M)))
                    /(2*M)
    Q_list.append(Q)
    return Q
```

3.5 聚类

```
while(i < 100): #进行100次聚类
    x, y = cluster(similar)
    similar[x][y]=0
    if clusters[x] != -1 and clusters[y] != -1:
    # x,y都已经分类，则将x，y的聚类合并成为一个
        temp = clusters[y]
        for j in range(M):
            if clusters[j] == temp:
                clusters[j] = clusters[x]
    if clusters[x] == -1 or clusters[y] == -1:
    # 两个簇中至少有一个未分类
        if clusters[x] == -1 and clusters[y] == -1:      # 两个簇都未分
            类
            flag += 1
            # 新建一个分类保存此簇
            clusters[x] = clusters[y] = flag
        elif clusters[x] == -1:
        # x未分类y已分类，将x并到y上去
            clusters[x] = clusters[y]
        elif clusters[y] == -1:
        # x已分类y未分类，将y并到x上去
            clusters[y] = clusters[x]
    Q = compute_Q(clusters , A)
    if(Q > q_max):
    # 输出Q值最大的情况的分类
        outputData(clusters , edgelist)
```

```
            q_max = Q
        i += 1
```

4 实验结果

Q 值变化曲线如下：

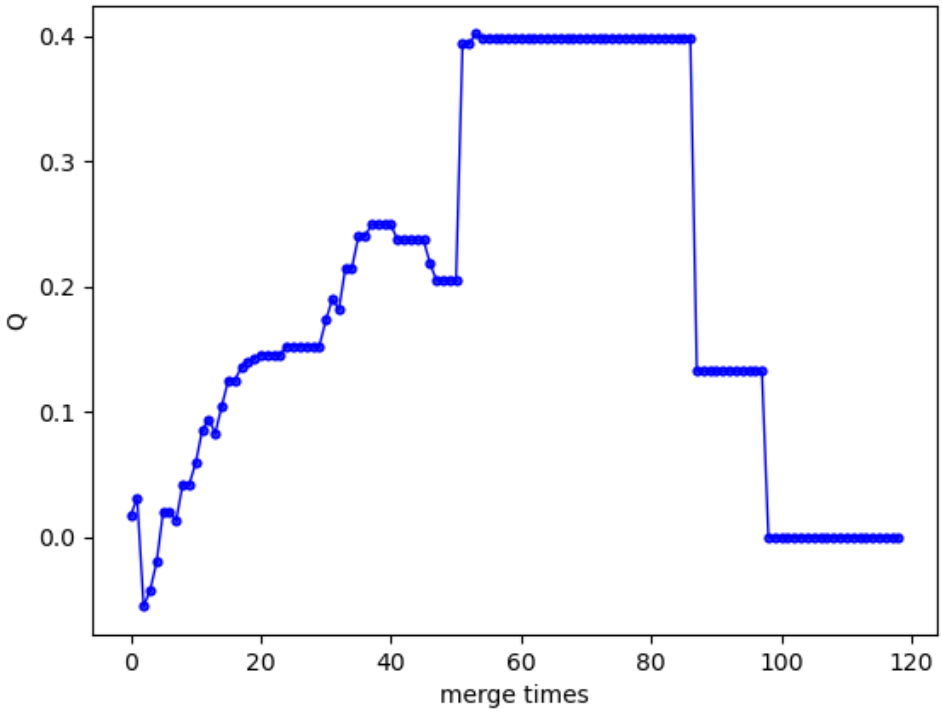

图 2: Q 值变化曲线

最终聚类结果如下：

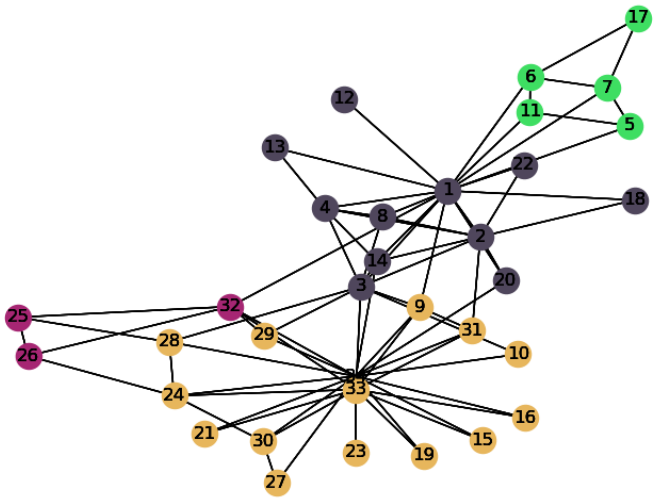

图 3: 聚类结果

5 结语与心得

之前做过图卷积的工作，于是最开始想用 pyg 库做，后来发现图神经网络与数据挖掘中的复杂网络完全不一样……只有虚心求学才能行稳致远。

参考文献

[1] ZACHARY W W. An Information Flow Model for Conflict and Fission in Small Groups[J/OL]. Journal of Anthropological Research, 1977, 33(4): 452-473. http://www.jstor.org/stable/3629752.

[2] NEWMAN M E, GIRVAN M. Finding and evaluating community structure in networks[J]. Physical review E, 2004, 69(2): 026113.