聚类技术---复杂网络社团检测

实验环境：python3.7

**1、实验内容**

复杂网络是描述复杂系统的有力工具，其中每个实体定义成一个节点，实体间的交互关系定义为边。复杂网络社团结构定义为内紧外松的拓扑结构，即一组节点的集合，集合内的节点交互紧密，与外界节点交互松散。复杂网络社团结构检测广泛的应用于信息推荐系统、致癌基因识别、数据挖掘等领域。

具体步骤如下：

1.1，导入网络数据；

1.2，根据网络结构特征给出节点相似性度量指标；

1.3，根据相似性指标利用数据计算出相似度权重矩阵S；

1.4，采用贪婪算法聚类；

1.5，利用matplotlib和networkx可视化结果。

**2、分析及设计**

步骤一、导入网络数据；

利用邻接矩阵A来存储网络，其中A\_{ij}表示第i个节点与第j个节点的是否有边相互链接，

1表示有，0表示没有

步骤二、根据网络结构特征给出节点相似性度量指标；

给定节点i, 其邻居节点定义为与该节点相链接的所有节点组成的集合，即

给定一对节点（i,j）,其相似性定义为这个两个节点

的公共邻居节点与邻居节点的并， 即：

,

其中表示集合中元素的个数。

步骤三、采用贪婪算法提取模块

随机选择一个未聚类的节点作为当前社团C，提取出社团C所有未聚类的邻居节

点。选择使得社团密度降低最小的那个节点v添加到社团C，更新当前社团

为。持续该过程知道当前社团的密度小于某个阈值。

步骤四、采用Cytoscape工具，可视化聚类结果。

**3、详细实现**

3.1、数据导入，利用networkx.read\_gml读取karate.gml

G = nx.read\_gml('karate/karate.gml',label='id')

num\_n = G.number\_of\_nodes()

num\_e = G.number\_of\_edges()

3.2、根据相似性度量指标计算相似度矩阵S，S[i,j]表示i节点和j节点的相似度，S应当是对称矩阵，且对角线元素全为1

S = np.zeros((num\_n+1,num\_n+1))

for i in G:

for j in G:

S[i,j] = len(set(G.adj[i])&set(G.adj[j]))/len(set(G.adj[i])|set(G.adj[j]))

3.3、#计算类的密度--平均密度

def ComputeDensityAVG(club):

if len(club)==1:

return 1

sum\_density = 0.0

times = 0

for i in club:

for j in club:

if(i==j): continue

sum\_density += S[i,j]

times += 1

return sum\_density/times

3.4、#计算类的密度--最小密度

def ComputeDensityMIN(club):

density = float('inf')

for i in club:

for j in club:

if(i==j): continue

density = min(S[i,j],density)

return density

3.5、利用贪婪算法提取模块，具体如下：

def FindClubs(threshold=0.25,compute\_density=ComputeDensityAVG):

clubs = []#存放所获得的社团

club = []#存放当前的社团

candidate = list(G.nodes)#存放剩余节点，初始为全部节点

while(len(candidate)>0):

if(club==[]):#当前社团没有节点，则任选一个节点放入

firstOne = candidate[random.randint(0,len(candidate)-1)]

club.append(firstOne)

candidate.remove(firstOne)

club\_density = compute\_density(club)#类密度

if(len(candidate)==0):

clubs.append(club)#如果没有节点了，将当前社团加入clubs

else:#当前社团有节点，利用贪心算法选择加入的节点

max\_density = 0#选择能使加入后的社团密度最大的点，初始化当前密度为0

best\_one = 0

for i in candidate:

new\_club = club+[i,]

max\_density,best\_one = (compute\_density(new\_club),i) if(compute\_density(new\_club)>max\_density) else (max\_density,best\_one)

if(max\_density>=threshold):#最优密度大于阈值，该节点加入社团

club.append(best\_one)

candidate.remove(best\_one)

club\_density = max\_density

if(len(candidate)==0):#如果没有节点了，将当前社团加入clubs

clubs.append(club)

else:#当前状况下，最优的密度也低于阈值，则放弃加入节点，当前社团加入clubs

clubs.append(club)

club = []

return clubs

3.6、获取随机颜色的函数

def GetRandomColor():

colorArr = ['1','2','3','4','5','6','7','8','9','A','B','C','D','E','F']

color = ""

for i in range(6):

color += colorArr[random.randint(0,14)]

return '#'+color

3.7、从键盘接受阈值并画图：

threshold = float(input("请输入阈值"))

position = nx.spring\_layout(G)

clubs = FindClubs(threshold,ComputeDensityAVG)

for club in clubs:

# nx.draw\_networkx\_nodes(G, pos=position, nodelist=club, node\_color=GetRandomColor(),label='threshold=0.25')

nx.draw(G, pos=position, nodelist=club, node\_color=GetRandomColor(), with\_labels=True)

plt.title("threshold=")

plt.show()

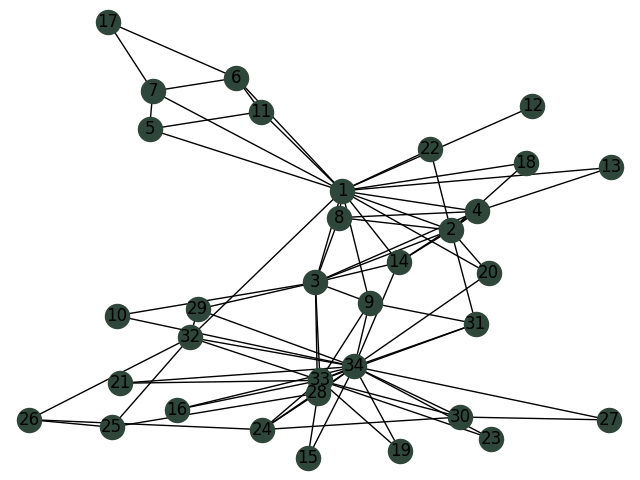
**4、实验结果**

获得的相似度矩阵S（部分值）：

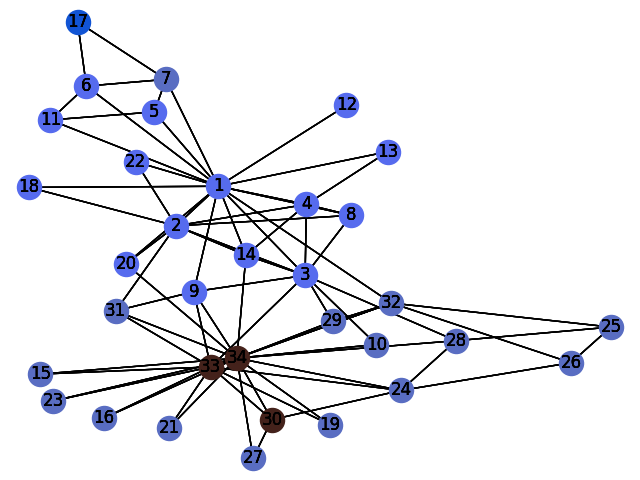


得到的分类结果：

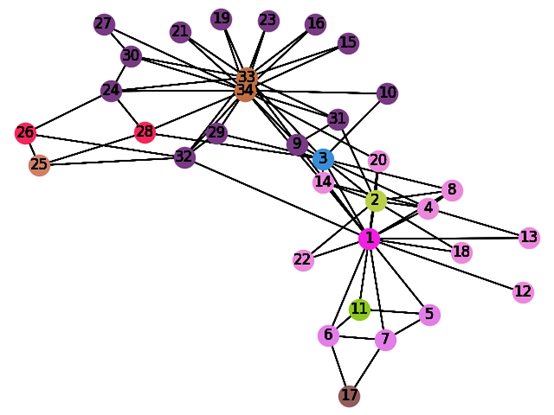
阈值0.01：



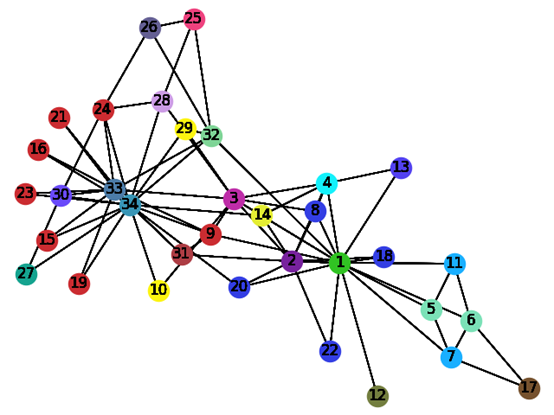
阈值0.25



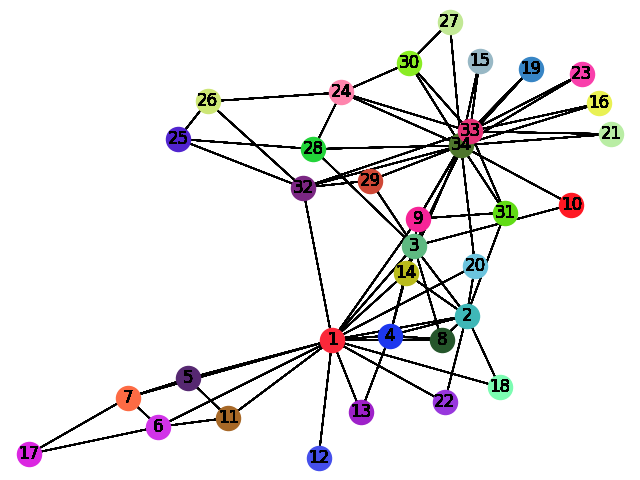
阈值0.40



阈值0.60



阈值1.01：



可以看到：

S矩阵是对称矩阵，且对角线元素全为1

社团的数量会随着阈值的增大而增多，阈值为0.01时只有一个社团，阈值为1.01时所有节点单独成团，符合预期，聚类基本正确。

**5、心得**

本次实验我学习了数据挖掘的聚类，了解了数据挖掘知识的应用领域与方法，学会了图和网络的很多操作包括图的读取、操作和绘画，明白了贪心算法的过程和实现。

在实验过程中也遇到过一些问题，在FindClubs（）中没有考虑到最后一个元素单独聚类的情形，导致聚类结果经常少一个元素，后来在debug的过程中发现了该错误，本次实验数据量不大，上手简单。

当然，本次实验只是最简单的聚类算法，还有密度聚类等算法，学无止境，以后我会尝试学习更好的算法不断地提升自己。