实验内容

1、导入 karate.gml 中的空手道网络数据；

2、根据网络结构特征给出节点相似性度量指标；

3、采用层次聚类过程对网络数据进行聚类；

4、计算模块性指标 Q 值，当 Q 值最大时输出聚类结果；

5、采用 Cytoscape 工具，可视化聚类结果

分析及设计

1. 选择相似性度量

观察数据发现，给定数据是一个无向无权图，所以不能通过欧式距离来衡量距离。备选的方案包括余弦相似度，Jaccard系数和最短跳数。对于本实验，Jaccard和余弦相似度非常接近，所以我后面尝试使用了余弦相似度和最短跳数。

2. 实现层次聚类算法

1. 根据1中定义的度量可以计算出一个相似度/距离矩阵，利用这个矩阵可以进行层次聚类
2. 层次聚类采用递归实现，使用一个列表存每个点所属的类。合并类的时候，将两个类标号归到二者中较小的标号上。
3. 初始状态下，令每个点都是一个簇。
4. 每次选择在原图中有边，而在当前状态下还没连上边的两个节点。利用参数函数(Single/MAX/组平均)计算这两个点所在簇之间的相似度，取相似度最大的两个簇进行合并，并更新列表。同时记录相关数据，便于后续绘图。
5. 根据更新后的结果计算模块化度量Q，如果大于目前的最优解，则更新最优解。
6. 重复4，5步直到合并为1类。

详细实现

1. 计算余弦相似度

```
def calCosSimilarity(g):
    # 计算余弦相似度，返回相似度矩阵
    data = []
    for i in range(34):
        tmp = []
        sa = set(g.neighbors(i))
        for j in range(34):
            sb = set(g.neighbors(j))
            # 用两个点的邻居节点的交集/两个集合长度乘积的平方根
            tmp.append(len(sa & sb)/math.sqrt(len(sa)*len(sb)))
        data.append(tmp)
    return data
    # return 34- np.array(g.shortest_paths()) # 最短跳数，因为后面都是按相似度编程，所以用34-最短条数，等价于相似度
```

2. 实现层次聚类

```
        def solve(kind,maxQ):
            skind = set(kind)
            real = 0
            null =0
            all = [0 for i in range(len(kind))]

            ############################ 计算模块化度量Q ####################
            for i in range(g.vcount()):
                for j in range(g.vcount()):
                    if i in g.neighbors(j) and kind[i] == kind[j]:
                            real += 1 # real每个簇内边的个数
            for i in range(g.vcount()):
                all[kind[i]] += g.degree(i)# all[i]表示标号为i的簇中节点的度的和
            for i in all:
                null += (i/(2*self.m))**2
            nowQ = (real/(2*self.m)) -null # 带入公式计算Q值
            num_kind = len(set(kind))
            print("Class: ", num_kind, "Q: ", nowQ)
            print(kind)
            self.kind[num_kind] = kind.copy() # 记录一些数据用于画图
            self.x.append(num_kind)
            self.y.append(nowQ)
            if len(skind) == 1: # 递归结束标志：当只剩下一个类的时候结束递归
                return maxQ
            if nowQ > maxQ:# 判断是否需要更新结果
                maxQ = nowQ
                self.res = kind.copy()
            maxx = -1
            tmpi,tmpj = -1,-1
            ################### 选择相似度最大的两个簇进行合并
######################
            for i in range(g.vcount()):
                for j in range(g.vcount()):
                    # 如果两个点已经在同一个簇，或者不直接相连，那么可以先跳过
                    if i==j or kind[i] == kind[j] or j not in
self.g.neighbors(i): continue
                    # 计算这两个簇之间的单链/全链/组平均相似度
                    tmp = fun(self.data,[x for x in range(g.vcount()) if kind[x]
== kind[i]]
```

```
                                ,[x for x in range(g.vcount()) if kind[x] ==
kind[j]])
                    if tmp > maxx:# 记录相似度最大的两个簇
                        maxx = tmp
                        tmpi = i
                        tmpj = j
            if tmpi == -1 or tmpj == -1:
                return maxQ
            if tmpi > tmpj:# 始终保持tmpi < tmpj 便于管理
                tp = tmpi
                tmpi = tmpj
                tmpj = tp
            old = kind[tmpj]
            for i in range(len(kind)):# 将两个簇的标号合并为较小的一个
                if kind[i] == old:
                    kind[i] = kind[tmpi]
            return max(solve(kind,maxQ),maxQ)
```

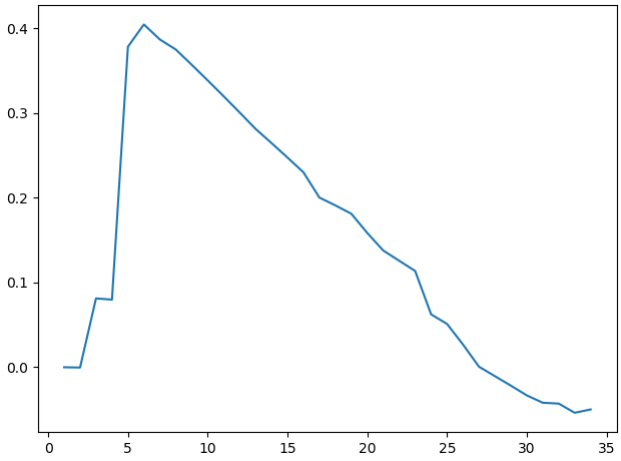
3. 实现单链/全链/组平均

```
def Singlle(data,l1,l2):# 带入公式即可，注意是在处理相似度
    # li代表第i个簇的全体节点的下表列表
    maxx = 0
    for i in l1:
        for j in l2:
            maxx = max(data[i][j],maxx)
    return maxx
def MAX(data,l1,l2):
    minn = 10000
    for i in l1:
        for j in l2:
            minn = min(data[i][j],minn)
    return minn
def AVG(data,l1,l2):
    sum = 0
    for i in l1:
        for j in l2:
            sum += data[i][j]
    return sum /(len(l1)*len(l2))
```

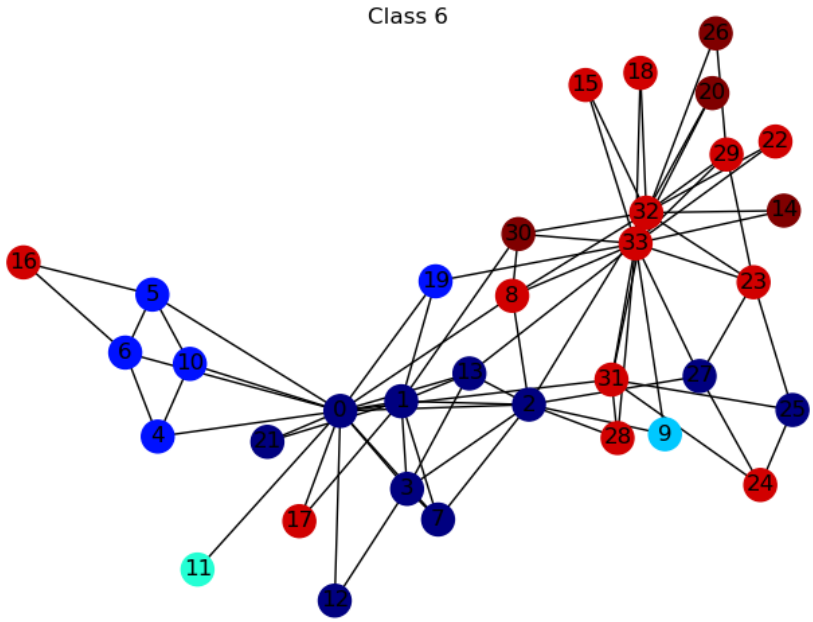
实验结果

1. 基于余弦相似度

1.1 全链

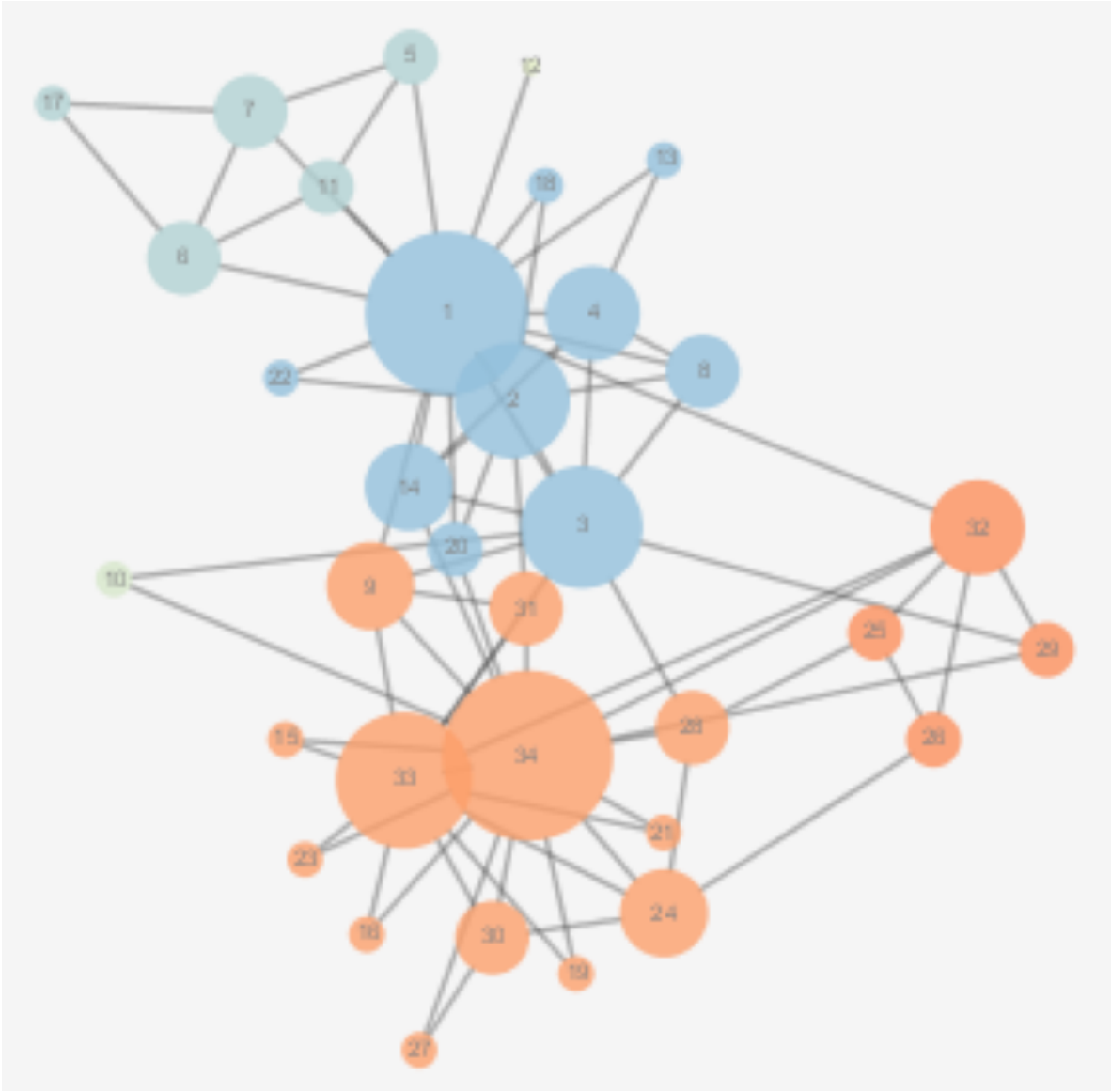


x轴为对应的簇数

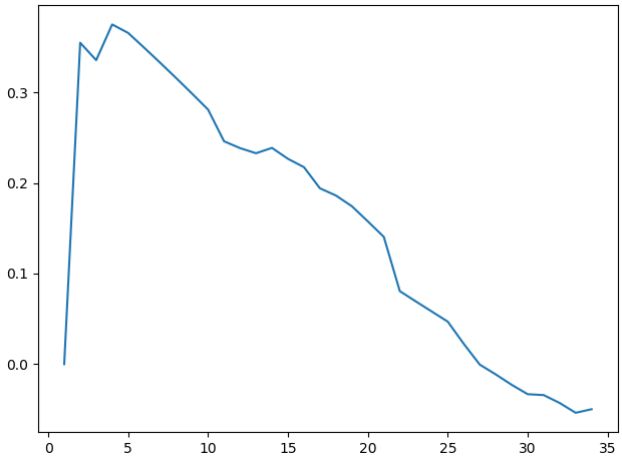


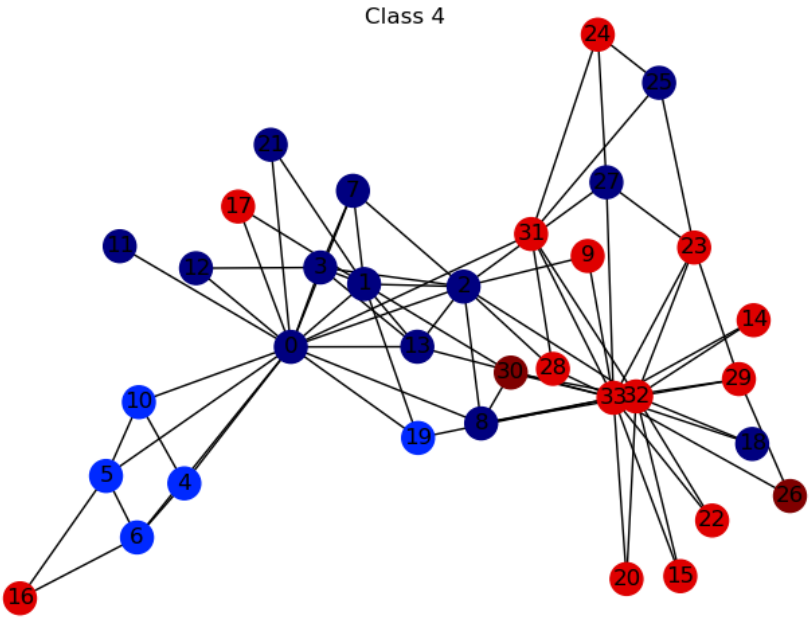
最优的Q为 0.40466798159105843，合并为6类

Cytoscape结果



1.2 组平均(Cytoscape结果见附件)

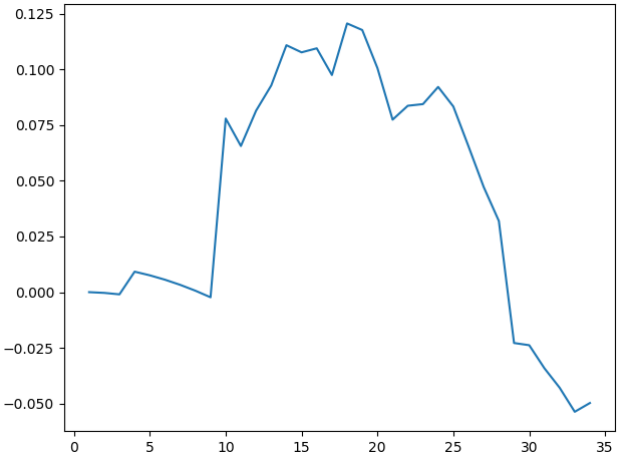




最好情况Q=0.37516436554898086被分成四类

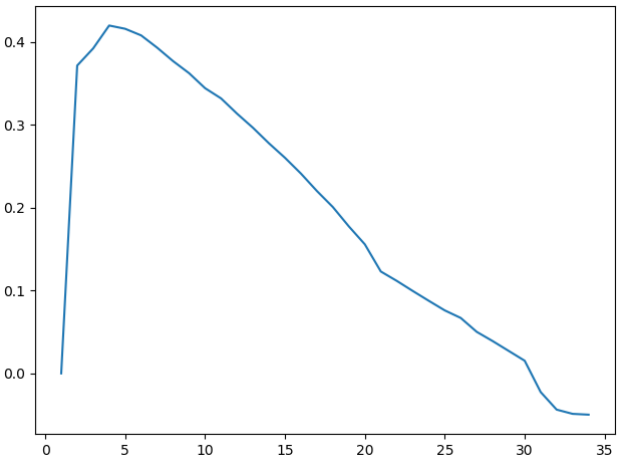
1.3 单链

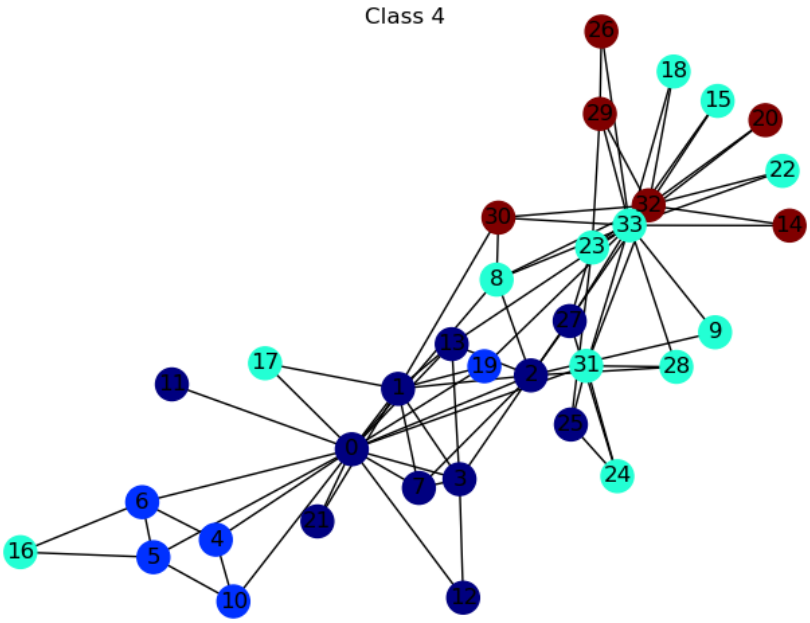
单链效果很差。从单链的特性考虑，可能是因为他更善于识别椭圆的簇，并且对噪声和离群点敏感



2. 基于最短跳数

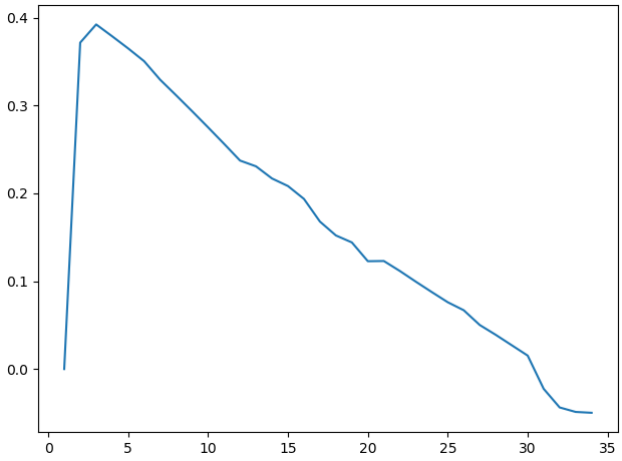
2.1 组平均(Cytoscape结果见附件)

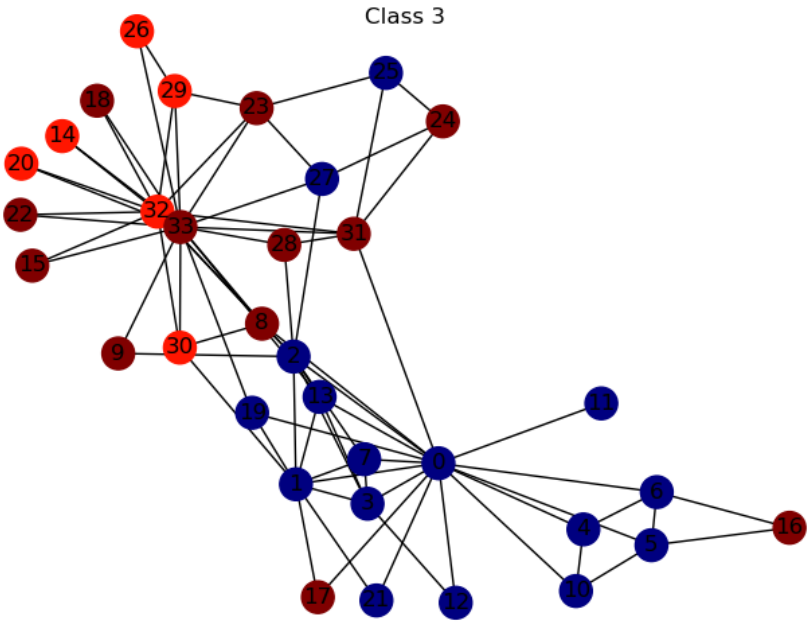




最好情况Q=0.41978961209730437被分成四类

2.2 全链(Cytoscape结果见附件)





最好情况Q=0.3921761998685076被分成三类

总结

1. 整体来看，基于最小跳数的度量比基于余弦相似度的度量具有更好的Q值，全链和组平均技术比单链技术具有更好的Q值。
2. 余弦相似度反应的是两个人朋友圈的交集的大小，最小跳数直观的反应两人间的距离。我认为都是有道理的度量，基于最小条数的度量结果更好可能与网络规模有关。另外对于大规模网络，计算最小跳数的复杂度也很高，我还是倾向于使用余弦相似度。
3. 我认为全链技术和组平均能更全面的衡量两个簇间的距离，所以比单链的性能好。

心得体会

1. 通过本次实验，实现了层次聚类算法。通过多种技术和度量标准，对比实验，将学习到的知识加以实验，令我有了直观的感受。
2. 对于模块化度量的值没有概念，不知道具体多少是比较正常的数值。以后还应该多多学习，多做实验，增加经验。
3. 学习的时候就感觉单链技术会导致以偏概全的情况，本次实验后发现，单链技术确实对某些问题性能较差，本次实验令我受益匪浅。

17069130005

崔恩博