

Web 信息处理与应用实验二程序说明——金泽文 ( PB15111604 )

## 社区发现算法实现与比较

### 【实验内容】

实现 spectral clustering 等几个社区发现算法,并比较实验结果.

### 【实验环境】

编程语言: matlab

编程环境: Matlab2017b win10 Pro 64

运行环境: Matlab2017b win10 Pro 64 i5-6300hq

使用工具: matlab, Matlab-bgl, gephi

## 【实验步骤及方法】

### 1. 配置环境.

在 i.ustc.edu.cn 下载 matlab,并安装为独立版,以方便在无法连接校园网时使用 matlab.

### 2. 调研与学习 matlab.

通过各种渠道(如博客,谷歌,matlab 官方 help 文档(官方文档真的很清晰!!)等),学习 matlab 基础用法,了解相关特性.

### 3. 五个算法的实现:

#### a) alinkjaccard:

- i. alinkjaccard 函数中的距离为 jaccard 距离,使用 pdist 函数计算 jaccard 距离.
- ii. 通过 linkage 函数,average 规则,每次计算不同 cluster 中的平均距离.
- iii. 选择平均距离最小的两个 cluster,通过 cluster 函数进行 merge.
- iv. 开始时将每个节点都看作单独一类,循环,直到剩余社区数为 k.

#### b) girvannewman:

- i. GN 算法使用自顶向下的聚类方法.
- ii. 开始时将所有 node 看成一个 cluster,通过 bgl 工具箱的 betweenness centrality 函数计算每条边的边介数.
- iii. 每次找到边介数最大的边,删去,
- iv. 使用 components 函数得到连通片个数.
- v. 直到连通片个数等于 k.

c) rcut:

- i. 构造拉普拉斯矩阵  $L=D-A$
- ii. 调用 `eigs` 函数求  $L$  矩阵最小的  $k$  个特征值所对应的特征向量, 并由此得到  $n*k$  矩阵
- iii. 将这个矩阵看成是  $n$  个  $k$  维 node,调用 `kmeans` 聚类.

d) ncut:

与 rcut 类似

- i. 构造归一化的拉普拉斯矩阵  $L=D^{-0.5}(D-W)D^{-0.5}$
- ii. 调用 `eigs` 函数求  $L$  最小的  $k$  个特征值所对应的特征向量,并由此得到  $n*k$  矩阵.
- iii. 将这个矩阵看成  $n$  个  $k$  维 node,调用 `kmeans` 聚类.

e) modularity:

- i. 构造矩阵  $B=A-d*d' / 2m$ , $d$  为度数列向量, $m$  为总的边数.
- ii. 调用 `eigs` 函数求  $B$  最大的  $k$  个特征值所对应的特向量,并由此得到  $n*k$  矩阵
- iii. 将这个矩阵看成  $n$  个  $k$  维点,调用 `kmeans` 聚类.

4. 求解 Egonet 最佳  $k$  值:

根据这个式子 
$$\sum_{k=1}^K \left( e(C_c^k, S) - a(C_c^k, S)^2 \right)$$
,为了得到 Egonet 数

数据集的最佳社区数,所以编写了 `getBestK` 函数,来通过算出  $k$  在取值范围 2-30 之间的模块性  $Q$ ,来得到  $k$  的最佳值(也就是  $Q$  最大时的取值).

5. 实现 main.m 脚本,调用上述函数,得到结果.

6. 评价聚类效果:

对于给出 ground\_truth 的数据集,可以通过所给的 evaluation 得到 MNI 和 ACC.

7. 通过 gephi 可视化:

首先将得到的数据,转换成 csv 结尾的命名.并在 excel 中加入对应节点的 id 号.

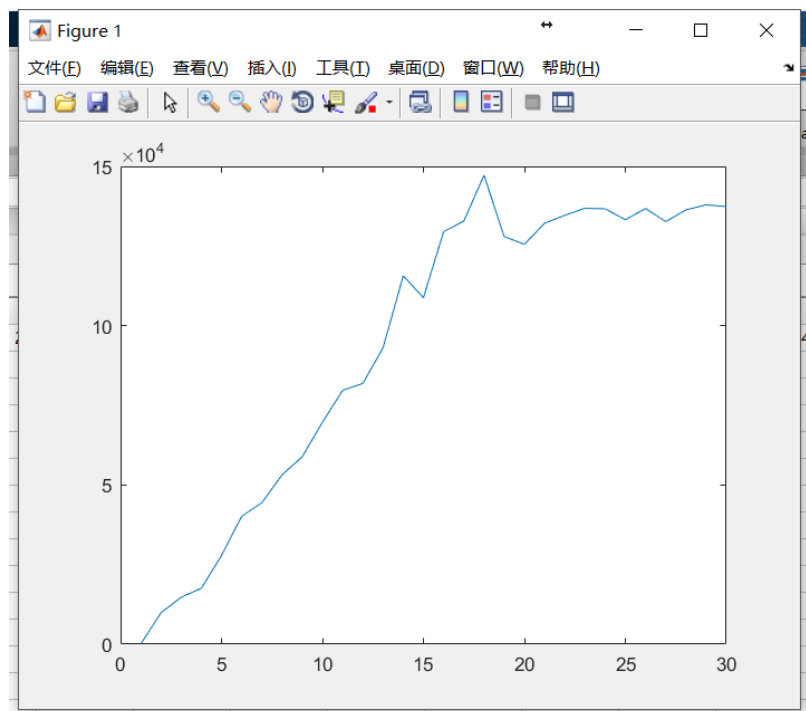
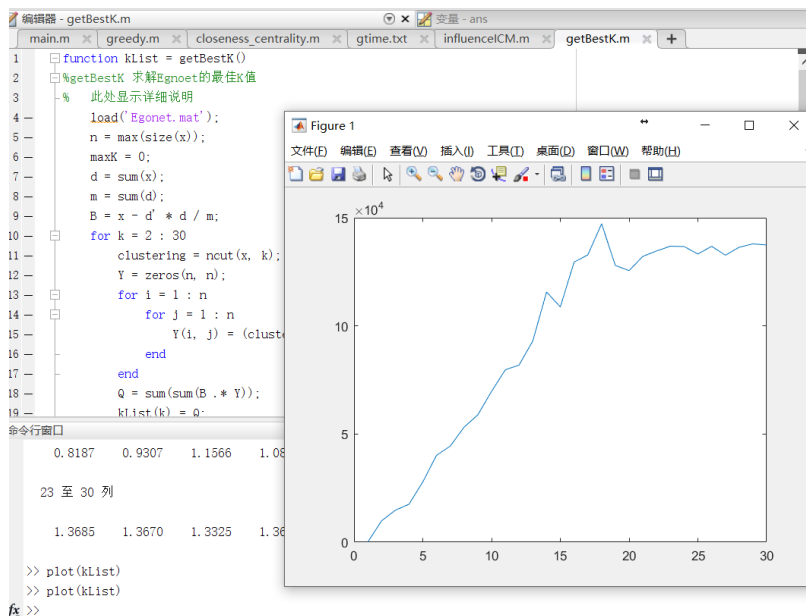
其次,从数据集.m 中导出边的信息.方法是通过  $[r,c]=\text{find}(A)$  得到稀疏矩阵的非零行列坐标.然后拼接出坐标向量,  $[x' ; y' ]'$ ,再通过 dlmwrite 函数将这个结果输出到文件中.最后将该文件转换成 csv 格式.

最后导入边和节点文件到 gephi 中,选择合适的颜色和布局,生成图片,并以 png 格式输出.

### 【实验结果说明及演示】

首先是对 Egonet 中 community 数量 k 的求解.

从 k=2:30 依次取值,算出对应的 modularity 的模块性 q,则最佳的 k,则是最大的 q 对应的 k,如下面两个图,得到最佳 k 值为 18 .所以 Egonet 的最佳社区数目为 18



























## 然后是 main.m 运行之后的结果

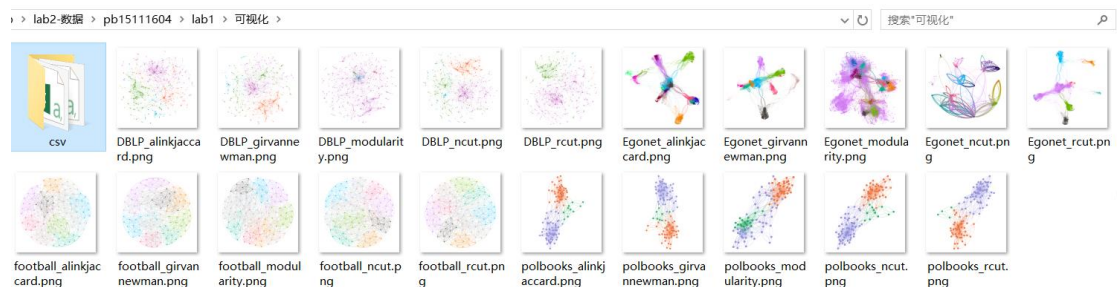
web > lab2-数据 > pb15111604 > lab1 >			
名称	修改日期	类型	大小
output	2018/1/19 20:51	文件夹	
可视化	2018/1/19 20:51	文件夹	
alinkjaccard.m	2018/1/18 21:47	MATLAB Code	1 KB
evaluation.m	2013/12/17 17:30	MATLAB Code	2 KB
getBestK.m	2018/1/19 21:47	MATLAB Code	1 KB
girvannewman.m	2018/1/18 14:41	MATLAB Code	1 KB
main.m	2018/1/19 19:24	MATLAB Code	3 KB
modularity.m	2018/1/17 20:56	MATLAB Code	1 KB
ncut.m	2018/1/17 20:43	MATLAB Code	1 KB
rcut.m	2018/1/17 20:39	MATLAB Code	1 KB
DBLP.mat	2013/12/17 17:45	MATLAB Data	45 KB
Egonet.mat	2015/12/29 23:13	MATLAB Data	238 KB
football.mat	2013/12/17 21:22	MATLAB Data	19 KB
polbooks.mat	2013/12/17 21:22	MATLAB Data	15 KB
football_gd.txt	2013/12/17 17:36	TXT 文件	1 KB
graph.txt	2017/12/10 21:26	TXT 文件	289 KB
polbooks_gd.txt	2013/12/17 15:59	TXT 文件	1 KB
Egonet_girvannewman.txt	2018/1/18 14:22	TXT 文件	10 KB
DBLP_girvannewman.txt	2018/1/18 10:28	TXT 文件	9 KB
Egonet_alinkjaccard.txt	2018/1/18 10:23	TXT 文件	11 KB
Egonet_modularity.txt	2018/1/18 10:23	TXT 文件	13 KB
Egonet_ncut.txt	2018/1/18 10:23	TXT 文件	15 KB
Egonet_rcut.txt	2018/1/18 10:23	TXT 文件	13 KB
DBLP_alinkjaccard.txt	2018/1/18 10:23	TXT 文件	6 KB
DBLP_modularity.txt	2018/1/18 10:23	TXT 文件	9 KB
DBLP_ncut.txt	2018/1/18 10:23	TXT 文件	9 KB
DBLP_rcut.txt	2018/1/18 10:23	TXT 文件	9 KB
football_alinkjaccard.txt	2018/1/18 10:23	TXT 文件	1 KB
football_girvannewman.txt	2018/1/18 10:23	TXT 文件	1 KB
football_modularity.txt	2018/1/18 10:23	TXT 文件	1 KB
football_ncut.txt	2018/1/18 10:23	TXT 文件	1 KB
football_rcut.txt	2018/1/18 10:23	TXT 文件	1 KB
polbooks_alinkjaccard.txt	2018/1/18 10:23	TXT 文件	1 KB
polbooks_girvannewman.txt	2018/1/18 10:23	TXT 文件	1 KB
polbooks_modularity.txt	2018/1/18 10:23	TXT 文件	1 KB
polbooks_ncut.txt	2018/1/18 10:23	TXT 文件	1 KB
polbooks_rcut.txt	2018/1/18 10:23	TXT 文件	1 KB

下面是可视化的输入文件.

eb > lab2-数据 > pb15111604 > lab1 > 可视化 > csv

名称	修改日期	类型	大小
 DBLP.csv	2018/1/17 22:20	Microsoft Excel ...	183 KB
 DBLP_alinkjaccard.csv	2018/1/19 20:43	Microsoft Excel ...	23 KB
 DBLP_girvannewman.csv	2018/1/19 20:43	Microsoft Excel ...	23 KB
 DBLP_modularity.csv	2018/1/19 20:43	Microsoft Excel ...	23 KB
 DBLP_ncut.csv	2018/1/19 20:42	Microsoft Excel ...	23 KB
 DBLP_rcut.csv	2018/1/19 20:42	Microsoft Excel ...	23 KB
 Egonet.csv	2018/1/17 22:21	Microsoft Excel ...	1,842 KB
 Egonet_alinkjaccard.csv	2018/1/19 19:37	Microsoft Excel ...	34 KB
 Egonet_girvannewman.csv	2018/1/19 19:36	Microsoft Excel ...	32 KB
 Egonet_modularity.csv	2018/1/19 19:36	Microsoft Excel ...	32 KB
 Egonet_ncut.csv	2018/1/19 19:36	Microsoft Excel ...	33 KB
 Egonet_rcut.csv	2018/1/19 19:40	Microsoft Excel ...	31 KB
 football.csv	2018/1/17 22:21	Microsoft Excel ...	9 KB
 football_alinkjaccard.csv	2018/1/19 19:34	Microsoft Excel ...	1 KB
 football_girvannewman.csv	2018/1/19 19:33	Microsoft Excel ...	1 KB
 football_modularity.csv	2018/1/19 19:33	Microsoft Excel ...	1 KB
 football_ncut.csv	2018/1/19 19:33	Microsoft Excel ...	1 KB
 football_rcut.csv	2018/1/19 19:32	Microsoft Excel ...	1 KB
 polbooks.csv	2018/1/17 22:21	Microsoft Excel ...	6 KB
 polbooks_alinkjaccard.csv	2018/1/19 19:31	Microsoft Excel ...	1 KB
 polbooks_girvannewman.csv	2018/1/19 19:31	Microsoft Excel ...	1 KB
 polbooks_modularity.csv	2018/1/19 19:31	Microsoft Excel ...	1 KB
 polbooks_ncut.csv	2018/1/19 19:31	Microsoft Excel ...	1 KB
 polbooks_rcut.csv	2018/1/19 19:30	Microsoft Excel ...	1 KB

下面是可视化的输出文件.

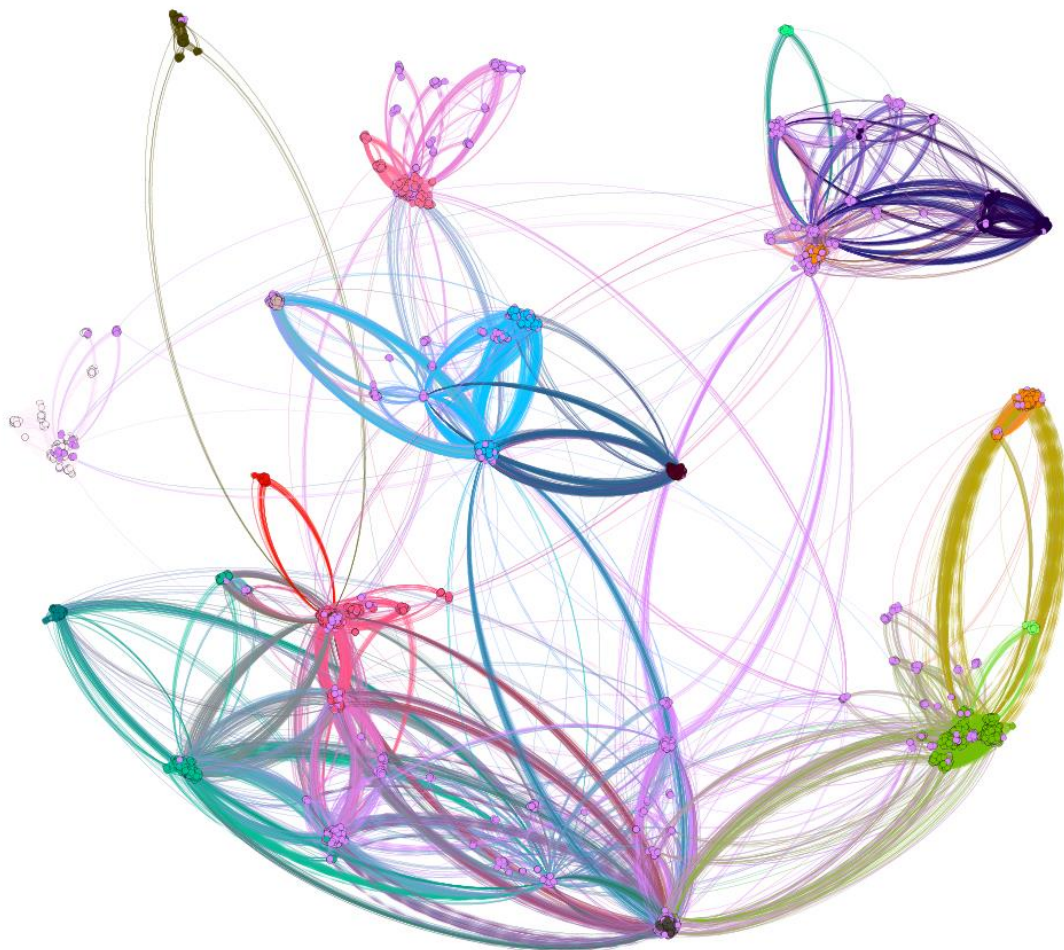


下面是运行 evaluation 之后得到的 polbooks 和 football 的 nmi 和 acc

数据集	算法	alinkjaccard	girvannewman	rcut	ncut	modularity
polbooks	nmi	0.431826852	0.436471844	0.507764996	0.425695195	0.251247237
	acc	0.771428571	0.79047619	0.80952381	0.761904762	0.6
football	nmi	0.263257046	0.260459018	0.263257046	0.263257046	0.25565791
	acc	0.147826087	0.147826087	0.156521739	0.139130435	0.139130435

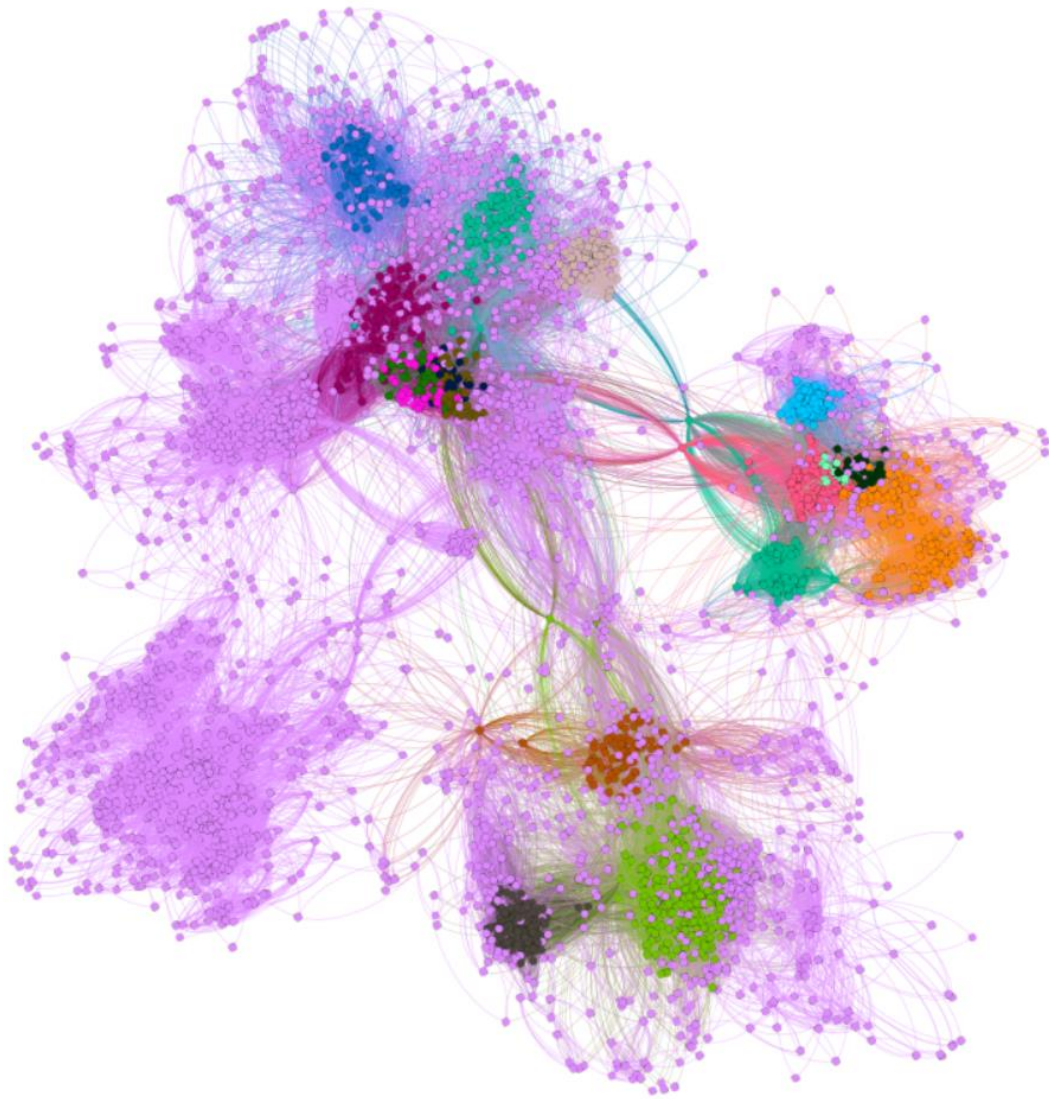
下面是可视化的部分效果图:

Egonet\_ncut:

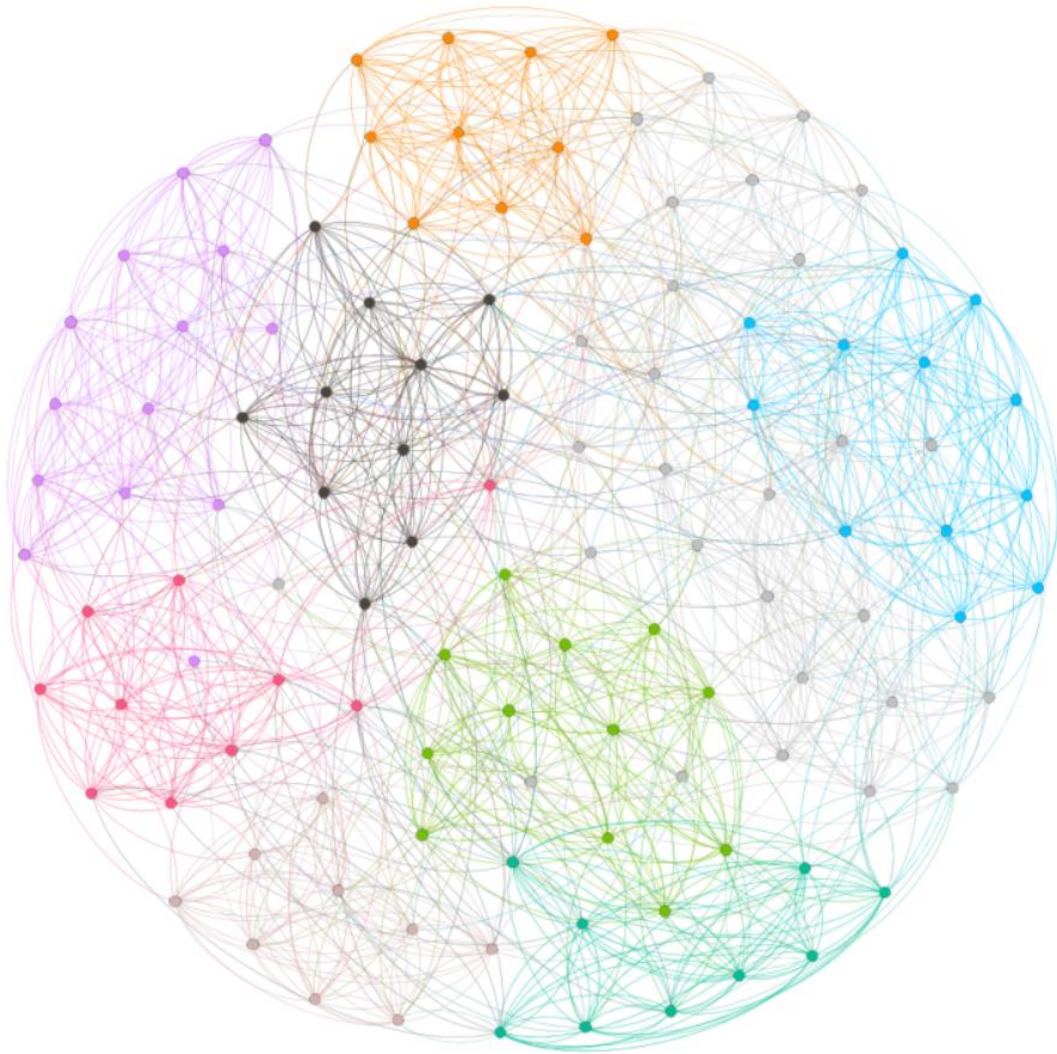




## Egonet\_modularity



football\_alinkjaccard:



# 影响力最大化算法

## 【实验内容】

利用 Degree Centrality、Closeness Centrality 等标准选取最具影响力的节点集合。将这些点作为种子节点，在独立级联模型(Independent Cascade Model)下进行模拟传播实验。比较不同方法选择种子节点的优劣。

## 【实验环境】

编程语言: matlab

编程环境: Matlab2017b win10 Pro 64

运行环境: Matlab2017b win10 Pro 64

使用工具: Matlab-bgl 库

## 【实验步骤及方法】

1. 首先从 graph.txt 得到邻接矩阵:

a) 这一步我是通过实现一个 getMatrix 函数,将 txt 转化出一个邻接矩阵.

2. 实现 ICM(独立级联模型) influenceICM:

a) 实现的时候要注意,每个节点只有一次激活其他节点的机会,必须把这个信息存储下来,以备下一轮迭代使用.

b) 每轮之后要更新节点,记录被激活节点数,

c) 激活条件是  $(G(i,j) > 0 \ \&\& \ \text{rand}() < G(i,j))$

3. 产生 seeds 的算法:

a) Random:直接生成随机数

















b) closeness centrality:调用 bgl 库的 all\_shortest\_paths 函数,得到最短路径矩阵,之后行和取倒数就是 closeness centrality.

c) degree centrality:直接 sum

d) greedy:只能枚举,时间复杂度很高,大部分时间都花在了这里.但是为了能够及时完成实验,所以我在 greedy 里面进行了 tradeoff:进行 5 次迭代.

## 【实验结果说明及演示】

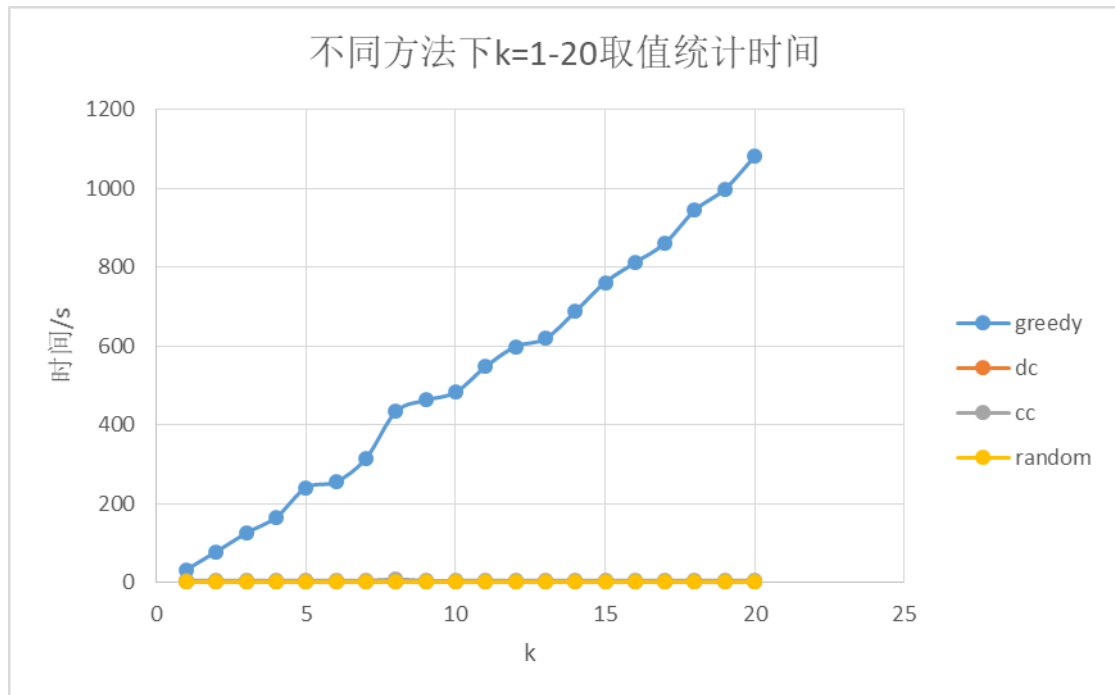
这是实验二的输出结果,红色为要求的结果,蓝色为对应的时间.

 closeness Centrality.txt	2018/1/19 22:39	TXT 文件	2 KB
 ctime.txt	2018/1/19 22:39	TXT 文件	1 KB
 degree Centrality.txt	2018/1/19 22:39	TXT 文件	2 KB
 dtime.txt	2018/1/19 22:39	TXT 文件	1 KB
 graph.txt	2018/1/19 22:39	TXT 文件	289 KB
 greedy.txt	2018/1/19 22:39	TXT 文件	2 KB
 gtime.txt	2018/1/19 22:39	TXT 文件	1 KB
 random.txt	2018/1/19 22:39	TXT 文件	2 KB
 rtime.txt	2018/1/19 22:39	TXT 文件	1 KB
 main.m	2018/1/19 22:26	MATLAB Code	3 KB
 greedy.m	2018/1/19 20:12	MATLAB Code	1 KB
 influenceCM.m	2018/1/19 20:09	MATLAB Code	2 KB
 getMatrix.m	2018/1/19 11:39	MATLAB Code	1 KB
 degree Centrality.m	2018/1/18 17:30	MATLAB Code	1 KB
 closeness Centrality.m	2018/1/18 17:28	MATLAB Code	1 KB
 random.m	2018/1/18 17:13	MATLAB Code	1 KB

得到不同方法的不同 k 值下的时间如表格所示:

K	1	2	3	4	5
greedy	32.86374	78.16028	125.4708	165.6249	240.69
Dc	1.489854	1.563401	1.57135	1.456035	1.634995
Cc	4.740274	4.99067	4.837012	5.004146	5.077318
random	1.499673	1.588912	1.532024	1.532783	1.527667
K	6	7	8	9	10
greedy	255.1885	314.7394	432.9169	462.3978	482.5892
Dc	1.501267	1.488281	1.9279	1.594863	1.470001
Cc	4.925484	4.755068	8.248774	4.740854	4.664398
random	1.520964	1.467066	2.341602	1.464589	1.450705
k	11	12	13	14	15
greedy	547.3584	598.3925	618.5405	687.2353	761.2699
Dc	1.535971	1.501109	1.506151	1.50536	1.611808
Cc	4.849777	4.861401	4.836915	4.836162	5.072834
random	1.620169	1.521067	1.533847	1.585249	1.598001
K	16	17	18	19	20
greedy	811.5662	860.7786	944.5612	997.6287	1080.963
Dc	1.535971	1.634995	1.57135	1.50536	1.563401
Cc	4.925484	5.004146	4.99067	4.740854	4.861401
random	1.60248	1.509032	1.602324	1.62144	1.632005

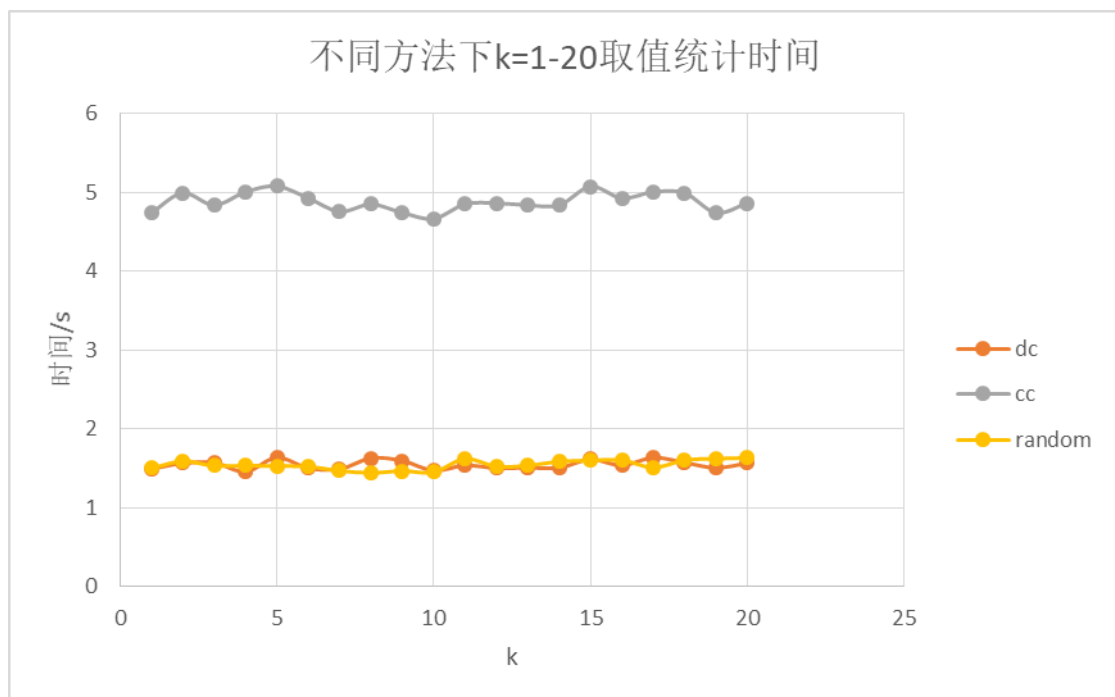
得到如下关系图:



可见 greedy 的时间复杂度是真的高.

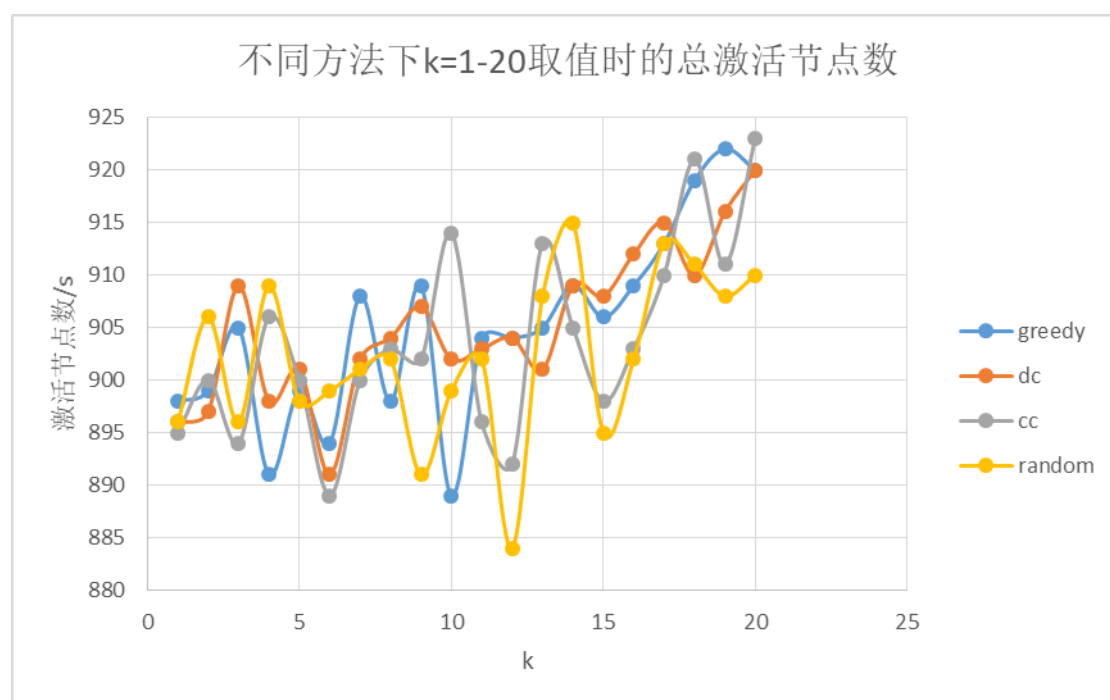
下面是除了 greedy 之外的比较.可见 closeness centrality 耗时也比其他两个高.

而其他两个耗时相似.



下面是不同方法下,不同 k 的被激活节点总数:

k	1	2	3	4	5
greedy	898	899	905	891	899
dc	896	897	909	898	901
cc	895	900	894	906	900
random	896	906	896	909	898
k	6	7	8	9	10
greedy	894	908	898	909	889
dc	891	902	904	907	902
cc	889	900	903	902	914
random	899	901	902	891	899
k	11	12	13	14	15
greedy	904	904	905	909	906
dc	903	904	901	909	908
cc	896	892	913	905	898
random	902	884	908	915	895
k	16	17	18	19	20
greedy	909	913	919	922	920
dc	912	915	910	916	920
cc	903	910	921	911	923
random	902	913	911	908	910



可能是因为我的 greedy 里面只枚举了 5 轮的原因,所以 greedy 的表现并没有十分突出.

当然也有可能因为这个数据集本身就是很快就会激活的个例.当然我更愿意相信是后者.

从这几张图中,可以看出,速度最快的是 random 和 dc,速度最慢的是 greedy.

而从激活综述来看,其实是不相上下的!

所以综合考虑时间和效果,没有太大的需求时,random 和 dc 是最值得考虑的!

特别是当节点数目庞大时!



## 【实验总结】

本次实验先是实现了 5 个聚类算法,最后又实现了影响力传播模型.

本次实验,可以说是这学期以来最头疼的实验了.

头疼的地方,一个是因为对整个社区发现算法的理解一开始是错误的,所以有很多地方,走了很多弯路,写错了很多东西,还好最后都改回来了.还有一个就是对实验要求的不理解.比如“把随机值作为阈值”,再比如“传播轮数要求 100 轮”,我认为阈值应该是一个一定时间内不变的量,而转移概率对每个边是不一样的,可以是个随机数.因为这个比较符合我们对日常影响力传播的认识.而传播轮数 100 轮我认为完全没有必要,传播 10 次可能就够了.另外想吐槽的是实验二中国年的个别点和其他店相连的太密集,这样的话似乎没有必要传播 100 轮,马上就会收敛.

通过本次实验呢,我还是学会了很多东西的:比如 matlab(第一次使用)(文档真的很全!),比如 gephi 这个好玩的东西,还了解了 csv 格式这么简单,知道了跑数据真的可以跑上 4-5 个小时,等等等等.

本次实验中,我还意识到了算法性能优化的重要性,知道了很多优化细节,比如 for 循环的优化等.如果不是这次两个实验每个都要跑一遍要差不多 4 个小时,我可能永远都不会这么清醒地认识到优化的重要性!

当然,最重要的还是自己动手解决问题的能力,再次得到提升!这是本次试验最大的收获所在了!

感谢助教!(我终于放假啦!!!!!!)