中山大学数据科学与计算机学院

移动信息工程专业-数据挖掘

本科生实验报告

（2018-2019 学年春季学期）

课程名称：数据挖掘

[redacted]			
[redacted]			

一、 实验题目

图聚类

二、 实验内容

1. 算法原理

（1）NCut（Normalized cut：正则化割）

该算法是基于图的算法，将样本看做无向图中的结点，将这些点用边连接起来。相似度较高的两个点之间的边的权重较高，相似度低的两个点之间的边的权重较低。通过找到权重最小，又能平衡切出子图大小的边，对所有样本组成的图进行切图。然后不同的子图代表不同的聚类结果。

该算法的流程：① 给定 G=（V， ε），得到（带权）邻接矩阵 W $\in R^{|V|x|V|}$

② 解方程组： $D^{-\frac{1}{2}}(D-W)D^{-\frac{1}{2}}x=\lambda x$ 得到前 n 小特征值对应的特征向量，构建矩阵 $F[y_1,y_2,\cdots,y_n]\in\mathbb{R}^{N\times n}$ $(y_i\in\mathbb{R}^{N\times 1})$，将该矩阵行向量作为每一个结点的特征向量，用 k-means 预处理得到 k’个簇。

③ 采用合并策略，每次选取两个簇合并成一个簇，直到最后剩下 k 个簇，选取规则是合并后能最小化：

$$Ncut_k=\frac{cut(A_1,V-A_1)}{assoc(A_1,V)}+\frac{cut(A_2,V-A_2)}{assoc(A_2,V)}+\cdots+\frac{cut(A_k,V-A_k)}{assoc(A_k,V)}$$

$$cut(A,B)=\sum_{u\in A,v\in B}w(u,v).\qquad asso(A,V)=\sum_{u\in A,t\in V}w(u,t)$$

该算法的特点：① 只需要数据之间的相似度矩阵，因此对于处理稀疏数据的聚类很有效； ② 在处理高维数据聚类时的复杂度比传统聚类算法好；③ 如果最终聚类的维度非常高，则谱聚类的运行速度和最后的聚类效果都不好； ④ 聚类效果依赖于相似矩阵，不同的相似矩阵得到的最终聚类效果可能很不同。

（2）Louvain

Louvain 是基于模块度（Modularity）的社区发现算法，其优化的目标是最大化整个图社区网络的模块度。

社区网络的模块度（Modularity）是评估一个社区网络划分好坏的度量方法，它的含义是社区内结点的连边数与随机情况下的边数之差。定义如下：

$$Q = \frac{1}{2m}\sum_{i,j}[A_{ij} - \frac{k_i k_j}{2m}]\delta(c_i, c_j)$$

$$\delta(u,v) = \begin{cases} 1 when\ u==v \\ 0\ else \end{cases}$$

其中：Aij 代表结点 i 和 j 之间的边权值（当图不带权时，边权值可以看成 1）。 ki 代表结点 i 的邻接边的边权和（当图不带权时，即为结点的度数）。m 为图中所有边的边权和。 ci 为结点 i 所在的社团编号。

模块度公式可以简化为：

$$Q = \frac{1}{2m}\sum_{i,j}[A_{ij} - \frac{k_i k_j}{2m}]\delta(c_i, c_j)$$

$$= \frac{1}{2m}[\sum_{i,j} A_{ij} - \frac{\sum_i k_i \sum_j k_j}{2m}]\delta(c_i, c_j)$$

$$= \frac{1}{2m}\sum_c[\Sigma in - \frac{(\Sigma tot)^2}{2m}]$$

其中：Σin 表示社区 c 内的边的权重和， Σtot 表示与社区 c 内的节点相连的边的权重和。

模块度增益的定义为：

$$\Delta Q = \left[\frac{\sum_{in}+k_{i,in}}{m} - \left(\frac{\sum_{tot}+k_i}{2m}\right)^2\right] - \left[\frac{\sum_{in}}{2m} - \left(\frac{\sum_{tot}}{2m}\right)^2 - \left(\frac{k_i}{2m}\right)^2\right],$$

其中：Σin 表示社区 c 内的边的权重和， Σtot 表示与社区 c 内的节点相连的边的权重和，ki 是与节点 i 相连的所有边的权重之和，ki,in 是节点 i 与社区 c 内部节点的边的权重之和，m 是整个网络中边的权重之和。

相对模块度增益定义为：

$$\Delta Q' = k_{i,in} - \frac{\sum_{tot} \times k_i}{m}$$

相对增益的值可能大于 1，不是真正的模块度增长值，但是它的正负表示了当前的操作是否增加了模块度。

Louvain 算法的思想：① 将图中的每个节点看成一个独立的社区，初始社区的数目和结点个数的相同；② 对于每个节点 i，尝试将其加入邻居 j 所在的社区，计算ΔQ ，选择一个能使ΔQ 最大 并且 $\Delta Q>0$ 的邻居进行合并，若增益非正，则节点 i 保持原有社区属性；③重复②，直到所有节点的所属社区不再变化；④ 对图进行压缩，将所有在同一社区的节点压缩成一

个新的节点，社区内节点之间的边的权重转化为新节点的环的权重，社区间的边权重转化为新节点间的边权重；⑤ 重复①直到 Q 不改变。

该算法的特点：① 与普通的基于模块度和模块度增益的算法相比，Louvain 速度很快，而且对于一些点多边少的图，进行聚类特别明显。

（3）NMF（非负矩阵分割）

NMF 定义：找到非负矩阵 W 与 H 使得 V=WH。NMF 的目标不是找到使得 V=WH 严格在成立的矩阵分解，而是使得 V 和 WH 尽可能接近。即找到两个非负矩阵 W 和 H，使得

$$V_{m\times n} \approx W_{m\times k} \times H_{k\times n} = \hat{V}_{m\times n}$$

为了能定量地比较矩阵 V 和$\hat{V}$的近似程度，定义基于欧几里得距离损失函数为：

$$\|A-B\|^2 = \sum_{i,j} (A_{i,j} - B_{i,j})^2$$

基于欧几里得距离的 NMF 的推导：

① 考虑无约束优化问题：

$$\min D_E(V \| WH) = \frac{1}{2}\|V - WH\|_2^2$$

② 利用梯度下降：

$$w_{ik} \leftarrow w_{ik} - \mu_{ik} \frac{\partial D_E(V\|WH)}{\partial w_{ik}} \qquad \frac{\partial D_E(V\|WH)}{\partial w_{ik}} = -\left[(V-WH)H^T\right]_{ik}$$

$$h_{kj} \leftarrow h_{kj} - \eta_{kj} \frac{\partial D_E(V\|WH)}{\partial h_{kj}} \qquad \frac{\partial D_E(V\|WH)}{\partial h_{kj}} = -\left[W^T(V-WH)\right]_{kj}$$

其中

③ 直接梯度下降，对于无约束的优化问题，无法保证结果是非负的。于是将梯度下降法变为乘法算法。

令 $$\mu_{ik} = \frac{w_{ik}}{\left[WHH^T\right]_{ik}}, \quad \eta_{kj} = \frac{h_{kj}}{\left[W^TWH\right]_{kj}}$$

梯度下降法变换为乘法算法：

$$w_{ik} \leftarrow w_{ik} \frac{\left[VH^T\right]_{ik}}{\left[WHH^T\right]_{ik}}$$

$$h_{kj} \leftarrow h_{kj} \frac{\left[W^TV\right]_{kj}}{\left[W^TWH\right]_{kj}}$$

2. 关键代码截图（带注释）

（1）NCut（Normalized cut：正则化割）

① 计算数据集中样本点之间的高斯距离，构建权重矩阵 W；

```
num = size(data,1); %样本数
%构建权重矩阵——高斯距离
data = data/max(max(abs(data))); %归一化
sigma = 7.15;
dis_fir = pdist(data); %data行与行之间的距离
dis_zero = squareform(dis_fir); % 对角线化为0，ij代表第xi和xj的距离 ||xi-xj||
dis_double = dis_zero.*dis_zero; %||xi-xj||^2
top = -dis_double/(2*sigma*sigma);
res = spfun(@exp,top);
S = full(res);
W = S;
```

② 将 W 的每一列元素加起来得到个数，将它们放在对角线上，其余地方为 0，构建归一化矩阵 D；

```
%计算归一化矩阵D
D = full(sparse(1:num,1:num,sum(W)))
```

③ 拉普拉斯矩阵 L： L＝D－W，归一化拉普拉斯矩阵： L＝D^（-0.5）* L * D^(0.5)

```
% L=D-W, 归一化L=D^(-1/2) * L * D^(-1/2) = I-D^(-1/2) * W * D^(-1/2)
L =eye(num)-(D^(-1/2)*W*D^(-1/2));
```

④ 计算归一化后 L 矩阵的 K 个最小特征值以及对应到的特征向量（将 K 个特征向量竖着并排放在一起，形成一个特征矩阵 Q）；

```
%找特征值特征向量T并排序，找前K个
K = 50;
%'sm' 绝对值最小特征值
[T,~] = eigs(L, K, 'SM');
%对特征向量求K-means
cluster = kmeans(T,K);
```

⑥ 对特征矩阵 Q 作 Kmeans 聚类，得到一个向量 C（分别对应 W 中每一行所代表的样本点所属的类别，即聚类结果）。

```
c = kmeans(T,K);
```

⑦ 采用合并策略，每次选取两个出合并成一个簇，直到最后剩下 K 个簇。

```
k=5;
%初始ncut
ncut = countNcut(0,0,cluster,W,num);
while 1
    difNum = unique(cluster,'rows');
    nc = size(difNum,1);
    if nc==k
        break;
    end
    a = rand(2,1);%生成两行一列 0~1之间的随机小数
    b = a*nc+1; %将随机小数映射到1~nc之间，包括nc
    c = floor(b); %取b的整数部分
    newcut = countNcut(difNum(c(1)),difNum(c(2)),cluster,W,num);
    if newcut < ncut
        r1 = find(cluster==difNum(c(1)));
        cluster(r1)=difNum(c(2));
        ncut = newcut;
    end
end
```

```
function ncut = countNcut(n,m,cluster,W,num)
    if n~=0 && m~=0
        r1 = find(cluster==n);
        cluster(r1)=m;
    end
    ncut = 0;
    difNum = unique(cluster,'rows');
    nc = size(difNum,1);
    rowSum = sum(W,2);
    for i=1:nc
        row = find(cluster==difNum(i)); %找出哪些点处于该簇
        nr = size(row,1);
        cut = 0;
        assoc = 0;
        for j=1:nr
            for k=1:num
                lib = ismember(k,row);
                if lib==0
                    cut = cut+ rowSum(row(j));
                end
            end
            assoc = assoc+sum(W(row(j)),2);
        end
        ncut = ncut + cut/assoc;
    end
end
```

（2）Louvain

① 根据邻接矩阵计算整个网络中边的权重之和；

```
%根据邻接矩阵计算边数
m = 0; %边数
for i=1:n
    for j=i+1:n
        if dataA(i,j)==1
            m = m+1;
        end
    end
end
```

② 将图中的每个节点看成一个独立的社区；

```
%记录每个结点的簇
cluster = zeros(n,1);
%初始化每个结点为一个簇
for i=1:n
    cluster(i) = i;
end
```

③ 计算初始 ki：

```
weight = sum(dataA~=0,2);
```

④ 对于每个节点 i：

《1》 计算 ki 、 $\sum in$、$\sum tot$

```
%对于每一个结点 计算出ki,in,tot
for i=1:num
    %计算ki
    row = find(cluster==difNum(i)); %找出哪些点处于该簇
    nr = size(row,1);
    for j=1:nr
        %把属于同一个簇的点的ki加起来就是该簇的ki
        ki(i) = ki(i)+ weight(row(j));
        if nr > 1
            for k=j+1:nr
                %把起点和终点都属于该簇的边的相加
                in(i) = in(i) + dataA(row(j),row(k));
            end
        end
    end
end
tot = ki;
```

《2》计算 ki_in

```
for i=1:num
    r1 = find(cluster==difNum(i));
    cutQ = zeros(1,num);
    %计算ki_in
     for j=i+1:num
        r2 = find(cluster==difNum(j));
        for k=1:size(r1,1)
            for p=1:size(r2,1)
                %找出r1中的点与r2中的点相连的边
                if dataA(r1(k),r2(p))==1
                    ki_in(i,j) = ki_in(i,j)+1;
                end
            end
        end
        ki_in(j,i) = ki_in(i,j);
     end
```

《3》尝试将节点加入邻居所在的社区，计算与所有邻居的ΔQ

```
for h =1:num
    if h~=i
        %相对增益
        cutQ(h) = ki_in(i) - tot(h)*ki(i)/m;
        %绝对增益
        %cutQ(h) = ((in(h)+ki_in(i))/m - ((tot(h)+ki(i))/(2*m))^2)
        %-(in(h)/(2*m)-(tot(h)/(2*m))^2-(ki(i)/(2*m))^2);
    end
end
```

《4》 选择一个能使ΔQ 最大 并且 ΔQ>0 的邻居进行合并，若增益非正，则节点 i 保持原有社区属性；

```
[big,pos] = max(cutQ(h));
if big > 0
    cluster(r1) = difNum(pos);
    update = 1;
end
```

《5》 当所有节点的所属社区不再变化时，跳出循环，得到最后结果。

```
%如果结点所属的簇不再改变
if update==0
    break;
end
```

（3）NMF（非负矩阵分割）

① 随机产生非负矩阵 W 与 H；

```
%随机产生非负矩阵W与H
W = abs(rand(n,r));
H = abs(rand(r,m));
```

② 迭代 iterate 次，更新 H 和 W：

```
for it = 1:iterate
    %更新H
    H = H.*(W'*V)./(W'*W*H+eps);
    %更新W
    W = W.*(V*H')./(W*H*H'+eps);
end
```

③ 对于 W 的每一行找出权重最大的一列即为该样本的簇号。

```
WS = size(W,1);
c = zeros(WS,1);
for i=1:WS
    [big,pos] = max(W(i));
    c(i) = pos;
end
```

三、 实验结果及分析

（1）NCut

① Cornell

聚类准确率ACC为0.54359
标准化信息NMI为0.28731
聚类纯度PUR为0.60513

② Texas

聚类准确率ACC为0.46524
标准化信息NMI为0.29726
聚类纯度PUR为0.50267

③ Washington

聚类准确率ACC为0.46957
标准化信息NMI为0.22415
聚类纯度PUR为0.54783

④ Wisconsin

聚类准确率ACC为0.57358
标准化信息NMI为0.33404
聚类纯度PUR为0.63774

（2）Louvain

① Cornell

聚类准确率ACC为0.44103
标准化信息NMI为0.088635
聚类纯度PUR为0.93333

② Texas

聚类准确率ACC为0.55615
标准化信息NMI为0.039844
聚类纯度PUR为0.96791

③ Washington

聚类准确率ACC为0.46087
标准化信息NMI为0.077789
聚类纯度PUR为0.93043

④ Wisconsin

聚类准确率ACC为0.4566
标准化信息NMI为0.061278
聚类纯度PUR为0.9434

（3）NMF

① Cornell

聚类准确率ACC为0.47179
标准化信息NMI为0.16355
聚类纯度PUR为0.49231

② Texas

聚类准确率ACC为0.53476
标准化信息NMI为0.22146
聚类纯度PUR为0.65241

③ Washington

聚类准确率ACC为0.46087
标准化信息NMI为0.32914
聚类纯度PUR为0.52174

④ Wisconsin

聚类准确率ACC为0.5283
标准化信息NMI为0.32693
聚类纯度PUR为0.63019