本科生实验报告

（2017-2018 学年秋季学期）

一、 实验题目

图聚类

二、 算法解析

1. NCut

正则化割（Ncut）用于描述不同类之间的不相似度，如下分别为两类见的分离度以及多类间的分离度。

$$Ncut(A,B)=\frac{cut(A,B)}{assoc(A,V)}+\frac{cut(A,B)}{assoc(B,V)}$$

$$N\text{cut}_k=\frac{cut(A_1,V-A_1)}{assoc(A_1,V)}+\frac{cut(A_2,V-A_2)}{assoc(A_2,V)}+...+\frac{cut(A_k,V-A_k)}{assoc(A_k,V)}$$

其中，$cut(A,B)=\sum_{u\in A,u\in B}w(u,v),\ assoc(A,V)=\sum_{u\in A,t\in V}w(u,t)$

最小的 Ncut 值下对应的划分就是图 G 的最优划分，最小化 Ncut 值的优化问题可以转化为如下公式：

$$D^{-\frac{1}{2}}(D-W)D^{-\frac{1}{2}}x=\lambda x$$

当 k=2 时，采用上式的第二个最小的特征值对应的特征向量最优划分图 G ，得到对应图像的一个分割结果。当聚类数为 k（k>2）时，可以采用将 k-way 划分转化为 2-way 划分的方式。

算法流程：

（1）预处理

a. 构建方程：$D^{-\frac{1}{2}}(D-W)D^{-\frac{1}{2}}=\lambda x$，求前 k1 个最小的特征值对应的特征向量，合并为特征矩阵；

b. 特征矩阵得到 NxK 维的矩阵，使用 K-means 算法进行聚类，得到 k1 个簇（k1<=100）

（2）基于 Ncut 准则两两合并簇

对于剩余的 k1 个簇，进行两两合并，每次合并两个簇，每次合并簇的准则为使得下式最小，直至剩下 k 个簇，为最终的聚类结果。

$$N\text{cut}_k = \frac{cut(A_1, V-A_1)}{assoc(A_1, V)} + \frac{cut(A_2, V-A_2)}{assoc(A_2, V)} + \ldots + \frac{cut(A_k, V-A_k)}{assoc(A_k, V)}$$

其中在每次合并的时候，最小 Ncut 的计算是通过穷举来进行的，需要考虑 k(k-1)/2 次，但是由于 k 比较小，可以通过枚举实现。

2. Louvain

Louvain 算法是基于模块度的社区发现算法，其优化目标是最大化整个社区网络的模块度。

模块度是评估一个社区网络划分好坏的度量方法，它的物理含义是社区内节点的连边数与随机情况下的边数只差，它的取值范围是 [−1/2,1)，其定义如下：

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

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

算法流程：

(1) 将图中的每个节点看成一个独立的社区，社区的数目与节点个数相同；

(2) 对每个节点 i，依次尝试把节点 i 分配到其每个邻居节点所在的社区，计算分配前与分配后的模块度变化 ΔQ ΔQ，并记录 ΔQ ΔQ 最大的那个邻居节点，如果 max ΔQ>0 max ΔQ>0，则把节点 i 分配 ΔQ ΔQ 最大的那个邻居节点所在的社区，否则保持不变；

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

(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，将所有在同一个社区的节点压缩成一个新节点，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）直到整个图的模块度不再发生变化。

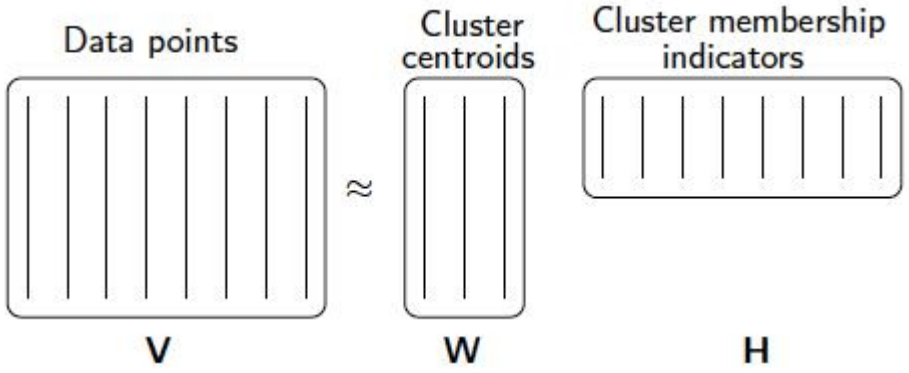
3. 非负矩阵分解（NMF）

非负矩阵分解，矩阵 V 被分解为矩阵 W 和矩阵 H。V=WH（W 权重矩阵、H 特征矩阵、V 原矩阵），通过计算从原矩阵提取权重和特征两个不同的矩阵出来。其中 W 和 H 中的所有元素都要大于 0。

$$V_{(F*N)} = W_{(F*K)} * H_{(K*N)}$$

其中原矩阵 V 中的 F 代表特征数，N 为样例；W 为特征矩阵，H 为系数矩阵。

NMF 常用来做矩阵降维，同时 NMF 还有聚类属性。通过 WH 来近似 V，即 $\min_{W,H} \| V - WH \|_F$,其中 H 矩阵给出 cluster indicator，对于矩阵 H 的第 i 列，最大的元素所对应的行即为样例 i 所属的类。如下图所示，矩阵 W 作为基矩阵，给出了聚类中心的情况，矩阵 H 则说明了每一个样例所属的类。



算法流程：

(1) 随机初始化矩阵 W,H

(2) 根据如下两个公式更新 W，H

(3) $W_{ik} = W_{ik} \cdot \frac{(VH^T)_{ik}}{(WHH^T)_{ik}}$

(4) $H_{kj} = H_{kj} \cdot \frac{(W^T V)_{kj}}{(W^T WH)_{kj}}$

(5) 计算 $\| V - WH \|_2$，直至不变，否则据需更新 W 和 H

(6) 得到的 H 矩阵为 K*N 的，根据每一列 H 值最大的行，确定每个样例的所属类别，k 为全部类的数量。

三、 代码实现

1. Ncut

（1）预处理得到 k2 个簇

```
W=A;
[N,~]=size(W);
k1=10;k2=50;
%计算对角矩阵
D=eye(N);
for i=1:N
    D(i,i) = sum(W(i,:));
end
%计算拉普拉斯矩阵
L=D-W;L = D^(-.5)*L*D^(-.5);
% %计算特征值特征向量
[eigVectors,V] = eigs(L,k1,'SA');
```

（2）计算 **Ncut** 值

```
pre_label=Label;
m_cl=unique(pre_label);%保存合并后还有簇的编号
kNum=length(unique(pre_label));%当前还有几个类
if(kNum==k)
    break;
end
ncut=zeros(N,3);
for i=1:kNum %对于每两个簇计算Ncut值
    for j=i+1:kNum
        Ncutk=0;
        pre_label(find(pre_label==m_cl(j)))=m_cl(i);
        tmp=m_cl(j);%当前被选出来的簇
        m_cl(j)=[];
        %计算Ncutk值
        for v=1:kNum-1
            in_v=find(pre_label==m_cl(v)); %在v里面的点
            assoc=0;
            for p=1:length(in_v)
                for q=in_v(p)+1:N
                    assoc=assoc+W(in_v(p),q);
                end
            end
```

（3）选择合并簇

```
ncut=sortrows(ncut,3); %升序排列;
[ib,jb,~]=ncut(1,:);
Label(find(Label==jb))=ib;%合并类，更改标签
```

2. Louvain

（1）主要变量的初始化

```
W=A;
N=size(W,1);%点数
m=sum(sum(W))/2;%边权重之和
IDX=1:N;
```

（2）计算ΔQ

```
%计算delQ
for k=1:length(in_c)
    for t=k+1:length(in_c)
        sumin=sumin+W(in_c(k),in_c(t));
    end
end
for k=1:length(in_i)
    for t=1:length(in_c)
        kiin=kiin+W(in_i(k),in_c(t));
    end
end
kiin=kiin*2;

for k=1:length(in_c)
    for t=1:N
        if(isempty(find(in_c==t)))
            tot=tot+W(in_c(k),t);
        end
    end
end
tot=tot+sumin;
```

（3）合并社区

```
        delQ=(kiin/(2*m))-(tot*ki/(2*m^2));
        if(delQ>Q_max)
            Q_max=delQ;
            bestj=j;
        end
    end
end
if(Q_max>0)%合并
    IDX(find(PRE_IDX==PRE_IDX(i)))=IDX(bestj);
    break;
end
```

3. NMF

```
[F_num,N_num]=size(V);%获取特征数以及样例数
V=double(V);
W=abs(rand(F_num,r));%随机初始化矩阵
H=abs(rand(r,N_num));
maxiter=100;        %最大迭代次数
lse=10000;%保存上一次迭代的损失
for i=1:maxiter
    H=H.*((W'*V)./((W'*W)*H));%更新W,H
    W=W.*((V*H')./(W*(H*H')));
    X=W*H;
    n_lse(i)=(norm(V-X)).^2;
    if n_lse(i)>=lse%停止条件
        break;
    end
    lse=n_lse(i);
end
[~,index]=max(H);
```

四、 效果分析

数据处理：

对于给定的数据集，主要采用了邻接矩阵的方式。对于每个点的特征向量，采用欧氏距离度量得到相似度矩阵，然后将相似度矩阵与邻接矩阵对应相乘，从而为矩阵加权。

```
W=zeros(N,N);
for i=1:N
    for j=1:N
        W(i,j)=norm(A(i,:)-A(j,:));
    end
end
A=A.*W;
```

以上描述的三种算法在一下四种数据集上的效果，如下表所示：

数据集	评测指标	NMF	Ncut	Louvain
cornell	ACC	0.4103	0.4513	0.3802
	NMI	0.1228	0.0397	0.1106
	PUR	0.5692	0.959	0.4328
texas	ACC	0.5775	0.4973	0.5423
	NMI	0.2037	0.0283	0.1503
	PUR	0.5989	0.8877	0.6033

washington	ACC	0.487	0.4506	0.3256
	NMI	0.1644	0.0855	0.1236
	PUR	0.5348	0.9203	0.5629
wisconsin	ACC	0.4415	0.4415	0.6683
	NMI	0.1153	0.0533	0.2036
	PUR	0.5208	0.959	0.4728

从上表可以看出，NMF与Louvain的效果较好，Ncut在PUR方面非常高，但是NMI非常低，基本上没有达到聚类的效果。

Ncut算法过程中，实际上包含两步的聚类。第一次是以ncut为准则的谱聚类，第二次是通过贪心将K-way转化为2-way来进行的。由于第一次已经被合并为社区的簇在接下啦不能再改变，第一次的预处理对第二次的合并分类产生的影响比较大。Ncut通过第二部的聚类主要提高了PUR，但是NMI没有提升。