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]

一、 实验题目

Network Embedding（）

二、 实验内容

1. 算法原理

（1）MNMF

MNMF 模型保留了 network embedding 的微观结构（运用成对节点相似度）和细观结构（社区）。

对于微观结构，MNMF 将节点的一阶和二阶相似度结合起来使用矩阵分解来学习表示（representations）。

- 一阶相似度：由两个节点之间是否有边来衡量相似度
  邻接矩阵 $\mathbf{S}^{(1)} = \mathbf{A} \in \mathbb{R}^{n\times n}$.
- 二阶相似度：两个节点之间没有边并不代表它们不相似，如果两个节点有很多共同的邻居，那么它们也有相似性。
  $\mathbf{S}^{(2)} \in \mathbb{R}^{n\times n}$,有共同邻居的节点也有相似性
  $$S_{ij}^{(2)} = \frac{\mathcal{N}_i \mathcal{N}_j}{\|\mathcal{N}_i\|\|\mathcal{N}_j\|}$$
  N_i 是 $\mathbf{S}^{(1)}$ 的第 i 行
- 相似度：
  $\mathbf{S} = \mathbf{S}^{(1)} + 5\mathbf{S}^{(2)}$

对于微观结构，通过 modularity 来刻画社区结构的信息。

$$Q = \frac{1}{4e}\sum_{ij}(A_{ij} - \frac{k_i k_j}{2e})h_i h_j$$

k_i是节点i的度, $2e = \sum_i k_i = \sum_{ij} A_{ij}$. 当社区个数为2时， 如果i属于第一个社区，$h_i = 1$，否则$h_i = -1$。

当社区个数$k > 2$时， 社区指示矩阵: $\mathbf{H} \in \mathbb{R}^{n\times k}$, 令$\mathbf{B} \in \mathbb{R}^{n\times n}$,

$B_{ij} = A_{ij} - \frac{k_i k_j}{2e}$

$$Q = tr(\mathbf{H}^T\mathbf{B}\mathbf{H}), \quad s.t. \quad tr(\mathbf{H}^T\mathbf{H} = n)$$

然后利用网络中的节点表示（node representations）与社区结构之间的共识关系与辅助社区表示矩阵（auxiliary community representation matrix）进行联合优化。

MNMF 的目标函数是：

$$\min_{\mathbf{M},\mathbf{U},\mathbf{H},\mathbf{C}} \|\mathbf{S}-\mathbf{M}\mathbf{U}^T\|_F^2+\alpha\|\mathbf{H}-\mathbf{U}\mathbf{C}^T\|_F^2-\beta tr(\mathbf{H}^T\mathbf{B}\mathbf{H})$$
$$s.t., \mathbf{M}\geqslant 0, \mathbf{U}\geqslant 0, \mathbf{H}\geqslant 0, \mathbf{C}\geqslant 0, tr(\mathbf{H}^T\mathbf{H})=n,$$

α和β都是正的参数，把**S**和**H**为信息映射到**U**, $\mathbf{U}\in\mathbb{R}^{n\times m}$ 是network embedding 的结果，m是降维之后的维数。

将目标函数优化成 4 个子问题并迭代优化它们，直到目标函数小于某一阈值时停止。

$$\mathbf{M}\leftarrow\mathbf{M}\odot\frac{\mathbf{S}\mathbf{U}}{\mathbf{M}\mathbf{U}^T\mathbf{U}}.$$

$$\mathbf{U}\leftarrow\mathbf{U}\odot\frac{\mathbf{S}^T\mathbf{M}+\alpha\mathbf{H}\mathbf{C}}{\mathbf{U}(\mathbf{M}^T\mathbf{M}+\alpha\mathbf{C}^T\mathbf{C})}.$$

$$\mathbf{C}\leftarrow\mathbf{C}\odot\frac{\mathbf{H}^T\mathbf{U}}{\mathbf{C}\mathbf{U}^T\mathbf{U}}.$$

$$\mathbf{H}\leftarrow\mathbf{H}\odot\sqrt{\frac{-2\beta\mathbf{B}_1\mathbf{H}+\sqrt{\Delta}}{8\lambda\mathbf{H}\mathbf{H}^T\mathbf{H}}}, \qquad (13)$$

where $\Delta = 2\beta(\mathbf{B}_1\mathbf{H})\odot 2\beta(\mathbf{B}_1\mathbf{H}) + 16\lambda(\mathbf{H}\mathbf{H}^T\mathbf{H})\odot(2\beta\mathbf{A}\mathbf{H}+2\alpha\mathbf{U}\mathbf{C}^T+(4\lambda-2\alpha)\mathbf{H})$.

$$B1_{ij}=\frac{k_i k_j}{2e}$$

（2）LANE

LANE 可以平滑地将标签信息合并到属性 network embedding 中，同时保持其相关性。它可以模拟属性网络空间和标签信息空间的结点近似，并且将它们联合嵌入到统一的低维表示中。

LANE 的主要思想：归属网络中的每个节点都与特定标签相关联。LANE 通过属性 network embedding 和 label informed embedding 这两个模块共同嵌入属性网络和标签。首先，将网络结构和属性信息中的节点近似映射为两个潜在表示 U（G）和 U（A），然后通过提取它们的相关性将 U（A）合并到 U（G）中。其次，使用学习的联合邻近来平滑标签信息并将它们均匀地嵌入到另一个潜在表示 U(Y)中。最后将所有学习到的潜在表示投影到同一的嵌入表示 H。

LANE 利用的信息有三种: ① 连接的信息：$\mathbf{G}\in\mathbb{R}^{n\times n}$ 邻接矩阵，n 是节点数目。② 节点的属性信息：$\mathbf{A}\in\mathbb{R}^{n\times m}$，m 是属性维度。③ 节点的类标信息：$\mathbf{Y}\in\mathbb{R}^{n\times k}$，k 是类标个数。

- Attributed Network Embedding Module
  - 对于 G：

通过总结成对相似度和相应向量表示距离的乘积来衡量不一致程度。

$$\underset{\mathbf{U}^{(G)}}{\text{minimize}} \quad \frac{1}{2}\sum_{i,j=1}^{n} s_{ij}\left\|\frac{\mathbf{u}_i}{\sqrt{d_i}}-\frac{\mathbf{u}_j}{\sqrt{d_j}}\right\|_2^2.$$

$d_i = \sum_j s_{ij}$，这里的s_{ij}是$\mathbf{S} \in \mathbb{R}^{n\times n}$的元素，$n$是节点数目，$s_{ij} = \frac{g_i \cdot g_j}{||g_i|| ||g_j||}$，$\mathbf{G}$ 是邻接矩阵,g_i是$\mathbf{G}$的第i行，s_{ij}的计算就是在求两个向量的cosine similarity。

将上式写成矩阵形式：

$$\frac{1}{2}\sum_{i,j=1}^{n} s_{ij}||\frac{u_i}{\sqrt{d_i}} - \frac{u_j}{\sqrt{d_j}}||_2^2$$
$$= Tr(\mathbf{U}^{(G)^T}\mathbf{I}\mathbf{U}^{(G)}) - Tr(\mathbf{U}^{(G)^T}\mathbf{D}^{(G)^{-1/2}}\mathbf{S}\mathbf{D}^{(G)^{-1/2}}\mathbf{U}^{(G)})$$

ui,u$_j$ 是 d 维向量，即最后降维的结果，是$\mathbf{U}^{(G)} \in \mathbb{R}^{n\times d}$ 的行向量。D（G）是对角矩阵，对应 S 的行和。那么将最小化转化为最大化：

$$\max_{\mathbf{U}^{(G)}} \mathcal{J}_G = Tr(\mathbf{U}^{(G)^T}\mathcal{L}^{(G)}\mathbf{U}^{(G)})$$
$$s.t. \quad \mathbf{U}^{(G)^T}\mathbf{U}^{(G)} = \mathbf{I}$$

$$\mathcal{L}^{(G)} = \mathbf{D}^{(G)^{-1/2}}\mathbf{S}\mathbf{D}^{(G)^{-1/2}}$$

这样得到了 U（G）——基于连接的 embedding 结果。

➢ 对于 A：

基于节点属性的 embedding 结果 U（A）的获取方法和 U（G）相似。

$$\underset{\mathbf{U}^{(A)}}{\text{maximize}} \quad \mathcal{J}_A = \mathrm{Tr}(\mathbf{U}^{(A)^T}\mathcal{L}^{(A)}\mathbf{U}^{(A)})$$
$$\text{subject to} \quad \mathbf{U}^{(A)^T}\mathbf{U}^{(A)} = \mathbf{I}.$$

$$\mathcal{L}^{(A)} = \mathbf{D}^{(A)^{-\frac{1}{2}}}\mathbf{S}^{(A)}\mathbf{D}^{(A)^{-\frac{1}{2}}},$$

通过将 U(A)投影到 U(G)的空间中，并使用投影矩阵的方差作为相关性的度量，将 U(A)并入到 U（G）中,并最大化 $\rho_1 = Tr(\mathbf{U}^{(A)^T}\mathbf{U}^{(G)}\mathbf{U}^{(G)^T}\mathbf{U}^{(A)})$

◆ Label Informed Embedding Module

采用学习属性网络邻近度来平滑标签信息建模。

➢ 对于 Y：

$$\underset{\mathbf{U}^{(Y)}}{\text{maximize}} \quad \mathcal{J}_Y = \mathrm{Tr}\left(\mathbf{U}^{(Y)^T}(\mathcal{L}^{(YY)} + \mathbf{U}^{(G)}\mathbf{U}^{(G)^T})\mathbf{U}^{(Y)}\right)$$
$$\text{subject to} \quad \mathbf{U}^{(Y)^T}\mathbf{U}^{(Y)} = \mathbf{I}.$$

$$\mathcal{L}^{(YY)} = \mathbf{D}^{(Y)^{-\frac{1}{2}}}\mathbf{S}^{(YY)}\mathbf{D}^{(Y)^{-\frac{1}{2}}}$$

得到由三种信息产生的 embedding 结果 U（G），U（A），U（Y）之后，将他们共同映射到最后的 embedding 结果空间 H：

$$\rho_2 = Tr(\mathbf{U}^{(G)^T}\mathbf{H}\mathbf{H}^T\mathbf{U}^{(G)})$$

$$\rho_3 = Tr(\mathbf{U}^{(A)^T}\mathbf{H}\mathbf{H}^T\mathbf{U}^{(A)})$$

$$\rho_4 = Tr(\mathbf{U}^{(Y)^T}\mathbf{H}\mathbf{H}^T\mathbf{U}^{(Y)})$$

即最大化： $\max\limits_{\mathbf{U}^{(\cdot)},\mathbf{H}} \quad \mathcal{J}_{corr} = \rho_2 + \rho_3 + \rho_4$

◆ LANE 最后要优化的目标函数：

$$\max_{\mathbf{U}^{(\cdot)},\mathbf{H}} \mathcal{J} = (\mathcal{J}_G + \alpha_1 \mathcal{J}_A + \alpha_1 \rho_1) + \alpha_2 \mathcal{J}_Y + \mathcal{J}_{corr}$$

$$s.t. \quad \mathbf{U}^{(G)^T}\mathbf{U}^{(G)} = \mathbf{I}, \; \mathbf{U}^{(A)^T}\mathbf{U}^{(A)} = \mathbf{I},$$

$$\mathbf{U}^{(Y)^T}\mathbf{U}^{(Y)} = \mathbf{I}, \mathbf{H}^T\mathbf{H} = \mathbf{I}$$

a1 和 a2 是设置的正参数。

U（G）、U（A）、U（Y）、H 的更新公式：

$$(\mathcal{L}^{(G)} + \alpha_1 \mathbf{U}^{(A)}\mathbf{U}^{(A)^T} + \alpha_2 \mathbf{U}^{(Y)}\mathbf{U}^{(Y)^T} + \mathbf{H}\mathbf{H}^T)\mathbf{U}^{(G)} = \boldsymbol{\lambda}_1 \mathbf{U}^{(G)}$$

$$(\alpha_1 \mathcal{L}^{(A)} + \alpha_1 \mathbf{U}^{(G)}\mathbf{U}^{(G)^T} + \mathbf{H}\mathbf{H}^T)\mathbf{U}^{(A)} = \boldsymbol{\lambda}_2 \mathbf{U}^{(A)},$$

$$(\alpha_2 \mathcal{L}^{(YY)} + \alpha_2 \mathbf{U}^{(G)}\mathbf{U}^{(G)^T} + \mathbf{H}\mathbf{H}^T)\mathbf{U}^{(Y)} = \boldsymbol{\lambda}_3 \mathbf{U}^{(Y)},$$

$$(\mathbf{U}^{(G)}\mathbf{U}^{(G)^T} + \mathbf{U}^{(A)}\mathbf{U}^{(A)^T} + \mathbf{U}^{(Y)}\mathbf{U}^{(Y)^T})\mathbf{H} = \boldsymbol{\lambda}_4 \mathbf{H}.$$

2. 关键代码截图（带注释）

（1）MNMF

① 计算一阶相似度矩阵、二阶相似度矩阵和相似度矩阵

```
A = udata.A; %邻接矩阵
n = size(A,1); %样本数
s1 = A; %一阶相似度
s2 = zeros(n,n); %二阶相似度
for i=1:n
    for j=1:n
        s2(i,j) = sum(s1(i,:).*s1(j,:))/(sum(s1(i,:))*sum(s1(j,:)));
    end
end
S = s1+5*s2; %相似度矩阵
```

② 计算 K，ki 是节点 i 的度。

```
K = zeros(n,1); %节点i的度
for i=1:n
    K(i)=sum(A(i,:));
end
```

③ 计算 B 和 B1 ： $B1_{ij} = \frac{k_i k_j}{2e}$ ， $B_{ij} = A_{ij} - \frac{k_i k_j}{2e}$

```
B = zeros(n,n);
B1 = zeros(n,n);
for i=1:n
    for j=1:n
        B(i,j)=A(i,j)-K(i)*K(j)/sum(K);
        B1(i,j) = K(i)*K(j)/sum(K);
    end
end
```

④ 对 M、U、H、C 设置随机初始值，并且确定一些相关参数的值。

```
m = 5; %m是降维之后的维数
k = 5; %k是社区个数
M = rand(n,m); %初始化基矩阵
U = rand(n,m); %节点的初始化表示
H = rand(n,k); %初始化社区指标矩阵
C = rand(k,m); %社区的初始化表示
%参数
alpha =0.1;
beta =0.2;
lambda =1e9 ;
```

⑤ 根据更新公式更新 M、U、C、H。

```
I = eye(k);
X = U';
%更新M
M = M.*((S*U)./max(realmin,M*(U'*U)));
%更新U
X = X.*((M'*S+alpha*C'*H')./max(realmin,(M'*M+alpha*(C'*C))*X));
U = X';
%更新C
C = C.*((H'*U)./max(realmin,C*U'*U));
%更新H
B1H = B1*H;
HHH = H*(H'*H);
AH = A*H;
UC = U*C';
sqrtDeta = sqrt((2*beta*B1H).^2+16*lambda*HHH.*(2*beta*AH+2*alpha*UC+(4*lambda-2*alpha)*H));
H = H.*sqrt((-2*beta*B1H+sqrtDeta)./max(realmin,(8*lambda*HHH)));
```

⑥ 使用 kmeans 进行分类。

```
preY = kmeans(U,5);
```

（2）LANE

① 确定一些系数的值。

```
alpha2 = 36; %标签信息的权重
numiter = 5; %最大迭代次数
delta1 = 0.97; %用于构建测试表示H2的网络信息的权重
delta2 = 1.6; %用于构建测试表示H2的节点属性信息的权重

d = 100; %嵌入表示的维度
G = udata.A; %邻接矩阵
A = udata.F; %属性矩阵
n = size(G,1); %样本数
G(1:n+1:n^2) = 1; %让对角线为1
label = udata.label;
labelId = unique(label); % 类别
Y=[];
for i=1:length(labelId)
    Y = [Y,label==labelId(i)];
end
Y = Y*1;
```

② 使用 K 折交叉验证划分训练集与测试集。

```
Indices = crossvalind('Kfold',n,20); % K折交叉验证
Group1 = find(Indices <= 16); % 训练集
Group2 = find(Indices >= 17); % 测试集
%训练集
G1 = sparse(G(Group1,Group1)); %训练组中的节点网络
A1 = sparse(A(Group1,:)); %训练组中节点的节点属性
Y1 = sparse(Y(Group1,:)); %训练组中节点的标签
%测试集
A2 = sparse(A(Group2,:));%测试组中节点的节点属性
GC1 = sparse(G(Group1,:));%用于构建测试表示H2
GC2 = sparse(G(Group2,:));%用于构建测试表示H2
Y2 = sparse(Y(Group2,:));%测试集中节点的标签
```

③ 根据更新公式不断地迭代更新，最后得到测试集的 embedding 结果 H1。

《1》 计算 label 的归一化图拉普拉斯算子：

```
%计算inpx的归一化图拉普拉斯算子
function Lapx = norLap(Inpx)
    Inpx = Inpx';
    Inpx = bsxfun(@rdivide,Inpx,sum(Inpx.^2).^.5); %归一化
    Inpx(isnan(Inpx))=0;
    sx = Inpx'*Inpx;
    nx = length(sx);
    sx(1:nx+1:nx^2) = 1+10^-6;
    dxInv = spdiags(full(sum(sx,2)).^(-.5),0,nx,nx);
    Lapx = dxInv*sx*dxInv;
    Lapx = .5*(Lapx+Lapx');
end
```

```
LY = norLap(Label*Label');
```

《2》 更新 U（G）：

```
LG = norLap(Net); %归一化网络拉普拉斯算子
LA = norLap(Attri); %归一化节点属性拉普拉斯算子
UAUAT = zeros(n,n); %UA*UA^T
opts.disp = 0;
```

```
UYUYT = zeros(n,n);

HHT = H*H';
TotalLG1 = LG + alpha1*UAUAT + alpha2*UYUYT + HHT;
[UG,~] = eigs(.5*(TotalLG1+TotalLG1'),d,'LA',opts);
UGUGT = UG*UG';
```

《3》更新 U（A）：

```
TotalLA = alpha1*(LA+UGUGT) + HHT;
[UA,~] = eigs(.5*(TotalLA+TotalLA'),d,'LA',opts);
UAUAT = UA*UA';
```

《4》更新 U（Y）：

```
TotalLY = alpha2*(LY+UGUGT)+HHT;
[UY,~] = eigs(.5*(TotalLY+TotalLY'),d,'LA',opts);
UYUYT = UY*UY';
```

《5》更新 H：

```
TotalLH = UAUAT + UGUGT+UYUYT;
[H,~] = eigs(.5*(TotalLH+TotalLH'),d,'LA',opts);
```

④ 根据公式得到测试集的 embedding 结果。

$$\mathbf{H}_{test} = \mathbf{G}_2((\mathbf{H}_{train})^{\dagger}\mathbf{G}_1)^{\dagger} + \delta\mathbf{A}_{test}((\mathbf{H}_{train})^{\dagger}\mathbf{A}_{train})^{\dagger}$$

$(\cdot)^{\dagger}$表示伪逆,δ参数自己设置，G1：训练集的节点和所有节点的邻接矩阵，G2：测试集的节点和所有节点的邻接矩阵。

```
H2 = delta1*(GC2*pinv(pinv(H1)*GC1))+delta2*(A2*pinv(pinv(H1)*A1));
```

⑤ H1、H2 分别作为训练集和测试集，对应类别 Y1 和 Y2，用 KNN 分类算法进行分类。

```
c1 = kmeans(H1,5);
acc = classificationACC(c1,Y1);
```

三、 实验结果及分析

（1）MNMF

① cornell

```
alpha =0.1;
beta =0.2;
lambda =1e9 ;
```
聚类准确率ACC为0.50352

② texas

```
alpha =0.3;
beta =0.8;
lambda =1e9 ;
```
聚类准确率ACC为0.62132

③ Washington

```
alpha =0.3;
beta =0.6;
lambda =1e9 ;
```
聚类准确率ACC为0.54132

④ Wisconsin

```
alpha =0.3;
beta =0.6;
lambda =1e9 ;
```
聚类准确率ACC为0.62352

（2）LANE

① cornell

```
alpha1 = 0.2; %节点属性信息的权重
alpha2 = 0.6; %标签信息的权重
numiter = 5; %最大迭代次数
delta1 = 0.97; %用于构建测试表示H2的网络信息的权重
delta2 = 1.6; %用于构建测试表示H2的节点属性信息的权重
```
聚类准确率ACC为0.5

② texas

```
alpha1 = 0.4;
alpha2 = 0.8;
numiter = 5; %
delta1 = 0.97
delta2 = 1.6;
```
聚类准确率ACC为0.46154

③ Washington

```
alpha1 = 0.2;
alpha2 = 0.6;
numiter = 5; %
delta1 = 0.97;
delta2 = 1.6;
```
聚类准确率ACC为0.65217

④ Wisconsin

```
alpha1 = 0.3;
alpha2 = 0.5;
numiter = 10;
delta1 = 0.97
delta2 = 1.6;
```
聚类准确率ACC为0.63636