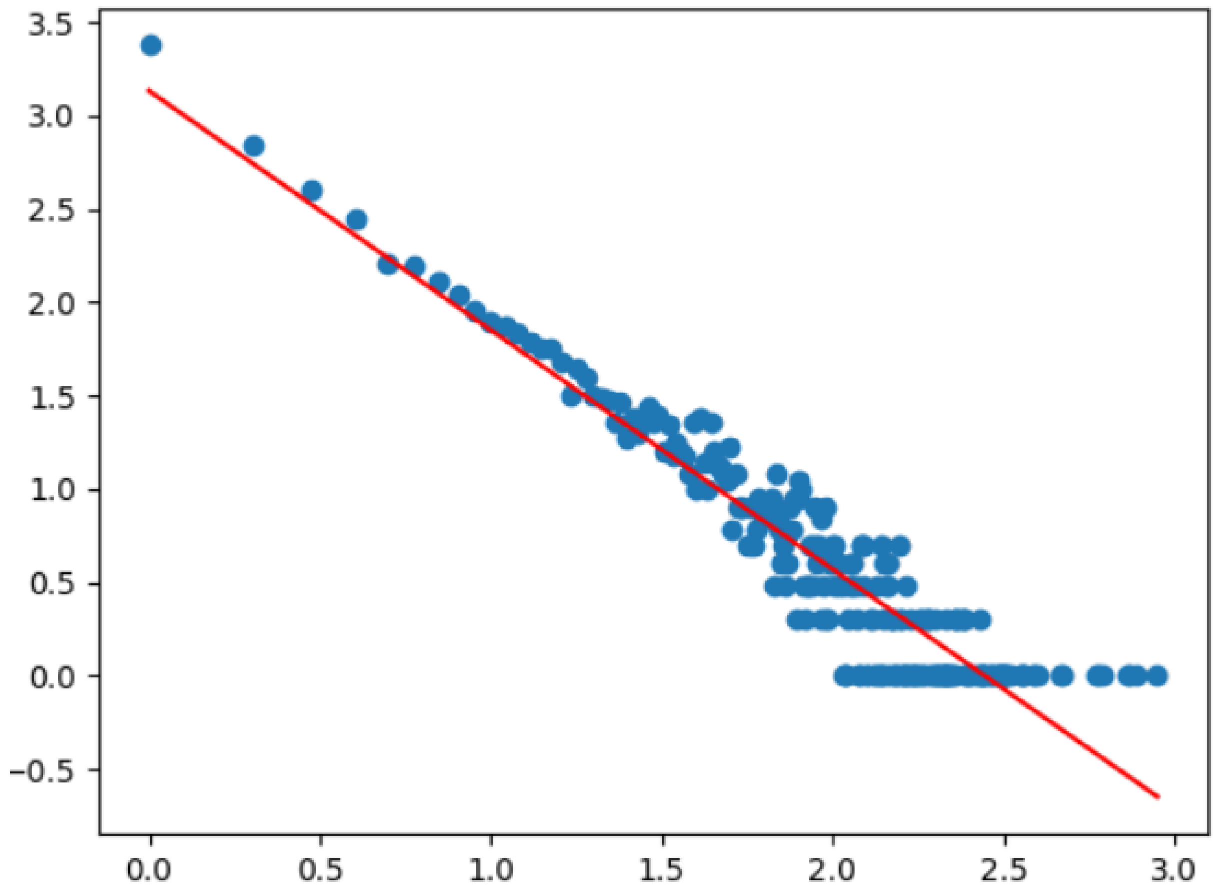
HW0

利用snap库函数分析复杂网络性质等，较简单,使用的网络为维基网络图，

直接给出图信息，

```
Python type TNGraph: Directed
  Nodes:                    7115
  Edges:                    103689
  Zero Deg Nodes:           0
  Zero InDeg Nodes:         4734
  Zero OutDeg Nodes:        1005
  NonZero In-Out Deg Nodes: 1376
  Unique directed edges:    103689
  Unique undirected edges:  100762
  Self Edges:               0
  BiDir Edges:              5854
  Closed triangles:         608389
  Open triangles:           12720413
  Frac. of closed triads:   0.045645
  Connected component size: 0.993113
  Strong conn. comp. size:  0.182713
  Approx. full diameter:    7
  90% effective diameter:  3.789745
```

绘图展示其度分布（对数坐标下），并做拟合，证明幂律分布公式的确可以拟合出图的度分布，



度分布对数坐标图

HW1

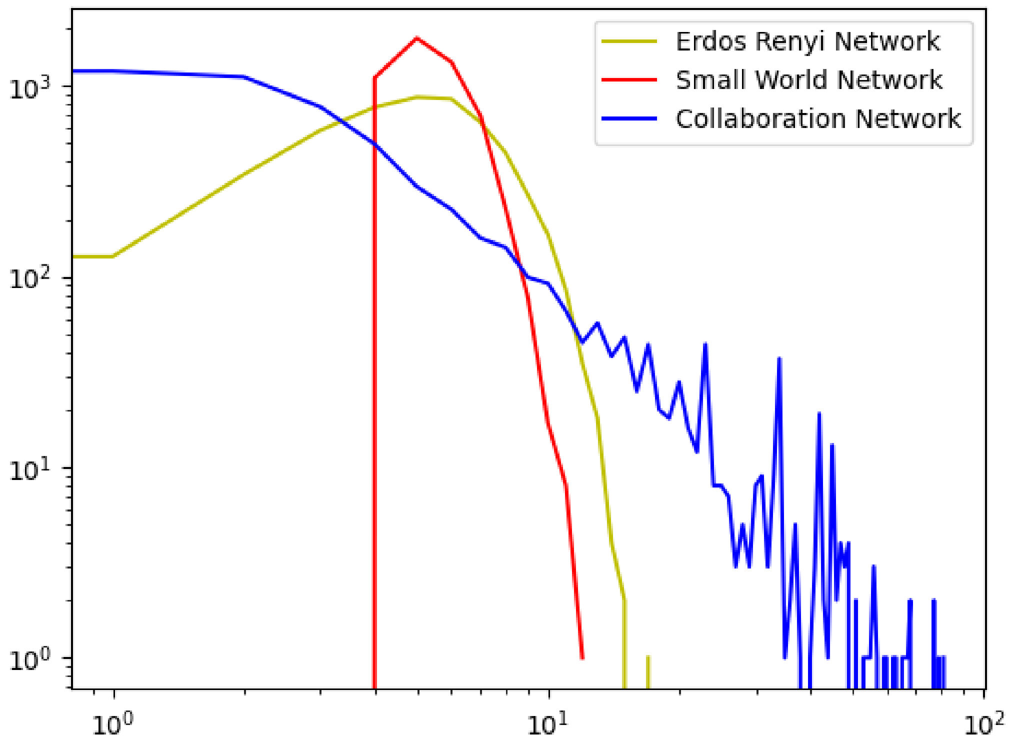
P1

网络特征，分别生成随机图、小世界模型图、并于真实图对比图度分布和聚类系数。

下面以小世界模型图的创建为例，

```
def getSmallGraph(n=5242,m=14484):
    #先构建初始图
    G = GenSmallWorld(5242,2,0)
    eg = set()
    cnt = 0
    #连边直到边数达到设定值
    for edge in G.Edges():
        eg.add((edge.GetSrcNId(), edge.GetDstNId()))
        cnt += 1
    while True:
        u = np.random.randint(n)
        v = np.random.randint(n)
        if u!=v and (u,v) not in eg and (v,u) not in eg:
            eg.add((u,v))
            G.AddEdge(u,v)
            cnt += 1
```

```
            if cnt==m:
                break
    return G

def Q1():
    RndG = GenRndGnm(PNGraph,5242,14484)
    SmallG = getSmallGraph(5242,14484)
    RealG = loadCollabNet('datasets\snap\ca-GrQc.txt\CA-GrQc.txt')
    x_erdosRenyi, y_erdosRenyi = getDataPointsToPlot(RndG)
    plt.loglog(x_erdosRenyi, y_erdosRenyi, color='y', label='Erdos Renyi Network')

    x_smallWorld, y_smallWorld = getDataPointsToPlot(SmallG)
    plt.loglog(x_smallWorld, y_smallWorld, color='r', label='Small World Network')

    x_collabNet, y_collabNet = getDataPointsToPlot(RealG)
    plt.loglog(x_collabNet, y_collabNet, color='b', label='Collaboration Network')

    plt.legend()
    plt.savefig('dump/degree_distribution.png')

    cf1 = RndG.GetClustCf()
    cf2 = SmallG.GetClustCf()
    cf3 = RealG.GetClustCf()
    print(cf1,cf2,cf3)
```



度分布对比图

图类型	聚类系数
随机图	0.0018
小世界模型	0.2834

图类型	聚类系数
真实图	0.5269

聚类系数对比表

采用的真实图为学术合作网络，对比可以发现真实图具有幂律分布、重尾分布、高聚类系数等特征，而随机图与真实图的特征不符合，小世界模型的某些特征在一定程度上较为类似真实图。

P2

用Rolx and ReFex 模型计算图中不同结点的结构特征，

Rolx采用如下几个结点特征，并采用余弦相似度定义结点之间的相似度，

1. the degree of v, i.e., $\deg(v)$;
2. the number of edges in the egonet of v, where egonet of v is defined as the subgraph of G induced by v and its neighborhood;
3. the number of edges that connect the egonet of v and the rest of the graph, i.e., the number of edges that enter or leave the egonet of v.

We use $\tilde{V}_u$ to represent the vector of the basic features of node u. For any pair of nodes u and v, we can use cosine similarity to measure how similar two nodes are according to their feature vectors x and y:

$$\text{Sim}(x, y) = \frac{x \cdot y}{||x||_2 \cdot ||y||_2} = \frac{\sum_i x_i y_i}{\sqrt{\sum_i x_i^2} \cdot \sqrt{\sum_i y_i^2}};$$

Also, when $||x||_2 = 0$ or $||y||_2 = 0$, we defined $\text{Sim}(x, y) = 0$.

ReFex算法在Rolx算法的基础傻瓜，采用如下公式递归地计算结点特征，

$$\tilde{V}_u^{(1)} = \left[\tilde{V}_u; \frac{1}{|N(u)|} \sum_{v \in N(u)} \tilde{V}_v; \sum_{v \in N(u)} \tilde{V}_v\right] \in \mathbb{R}^9,$$

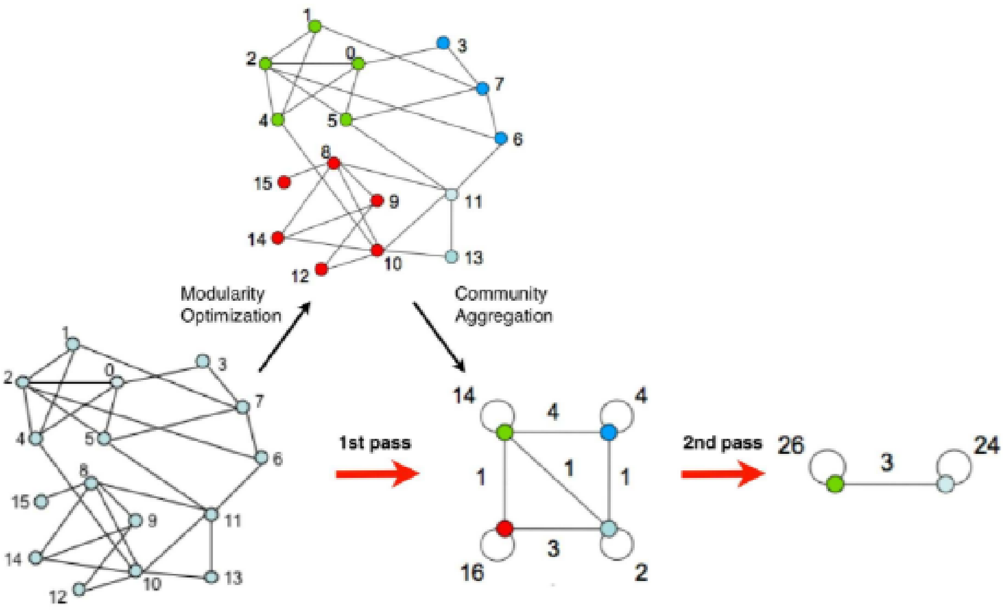
P3

社区发现算法，基于Louvain算法，采用如下公式定义社区模块度，好的社区发现算法应该带来高的总社区模块度，

$$Q = \frac{1}{2m} \sum_{1 \le i,j \le n} \left(\left[A_{ij} - \frac{d_i d_j}{2m} \right] \delta(c_i, c_j) \right)$$

对每一个结点，考虑将其加入一个相邻社区所带来的模块度增益，将其加入增益最大的社区中，

将同一个社区内的结点作为一个超节点，递归运行上述算法，直到发现的社区数量小于给定阈值。



Louvain算法运行示例图

算法基于贪心思想，关键是计算出将一个结点加入相邻社区的模块度增益，

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

符号定义示意图见下，

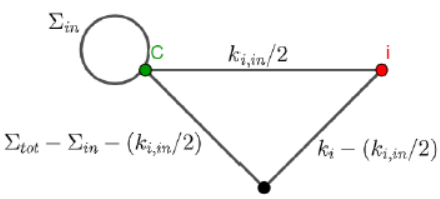


Figure 2: Before merging, i is an isolated node and C represents an existing community. The rest of the graph can be treated as a single node for this problem.

公式推导如下

$$
\begin{aligned}
Q &= \frac{1}{2m}\sum_{i,j}\delta(x_i,x_j)[A_{ij}-\frac{d_id_j}{2m}] \\
&= \frac{1}{2m}(\Sigma_{in}-\frac{1}{2m}\Sigma_{tot}^2)
\end{aligned}
$$

其中，Σ_{in}表示社区内部连边的数量和，Σ_{tot} 表示社区内所有结点的度数之和。

用上述化简后的公式，得到增益值

$$
\begin{aligned}
\Delta Q &= [\frac{\Sigma_{in}+k_{i,in}}{2m}-(\frac{\Sigma_{tot}+k_i}{2m})^2]-[\frac{\Sigma_{in}}{2m}-(\frac{\Sigma_{tot}}{2m})^2-(\frac{k_i}{2m})^2] \\
&= \frac{k_{i,in}}{2m}-\frac{\Sigma_{tot}+k_i}{2m^2}
\end{aligned}
$$

该算法基于贪心算法，但极可能陷入局部最小值，但后面可以揭示该方法和谱聚类等基于矩阵的算法的等价性。

P3

谱聚类，

先从二分类聚类开始，目标是将一张图划分为$S,\bar{S}$两个集合，定义，

$$
\begin{aligned}
&vol(S)=\sum_{i\in S}di \\
&cut(S)=cut(\bar{S})=\sum_{i\in S,j\notin S}w_{ij}
\end{aligned}
$$

目标是最小化以下函数，

$$
cut(S)(\frac{1}{vol(S)}+\frac{1}{vol(\bar{S})})
$$

$$
x_i=\begin{cases}\sqrt{\dfrac{\mathrm{vol}(\bar{S})}{\mathrm{vol}(S)}} & i\in S\\ -\sqrt{\dfrac{\mathrm{vol}(S)}{\mathrm{vol}(\bar{S})}} & i\in\bar{S}\end{cases}
$$

定义图的拉普拉斯矩阵$L=D-W$，证明拉普拉斯矩阵的一些性质，

(i) $L=\sum_{(i,j)\in E}(e_i-e_j)(e_i-e_j)^T$, where e_k is an n-dimensional column vector with a 1 at position k and 0's elsewhere. Note that we aren't summing over the entire adjacency matrix and only count each edge once.

(ii) $x^TLx=\sum_{(i,j)\in E}(x_i-x_j)^2$. *Hint: Apply the result from part (i).*

(iii) $x^TLx=c\cdot\mathrm{NCUT}(S)$ for some constant c (in terms of the problem parameters). *Hint: Rewrite the sum in terms of S and $\bar{S}$.*

(iv) $x^TDe=0$, where e is the vector of all ones.

(v) $x^TDx=2m$.

性质1，对E中的每一条边考虑，$(e_i-e_j)(e_i-e_j)^T$ 等价于令 $L_{i,i}=L_{j,i}=1$，$L_{i,j}=L_{j,i}=0$

性质2，类似性质1的推导，或者由1将L展，代入x计算，性质12展示了拉普拉斯矩阵的几何性质，

性质3，

$x^T Lx = 2\sum_{i\in S, j\notin S} x_i^T Lx_j = 2\sum_{i\in S, j\notin S}(\frac{vol(S)}{vol(\bar{S})} + \frac{vol(\bar{S})}{vol(S)} + 2) = 2\sum_{i\in S, j\notin S}\frac{(vol(S)+vol(\bar{S}))^2}{vol(S)vol(\bar{S})} = 4m\sum_{i\in S, j\notin S}\frac{1}{vol(S)} + \frac{1}{vol(\bar{S})} = 4mcut(S)(\frac{1}{vol(S)} + \frac{1}{vol(\bar{S})})$

性质4，代入验证可得

性质5，同样代入验证可得

因此问题转化为，

$$\begin{array}{ll} \underset{S\subset V,\, x\in\mathbb{R}^n}{\text{minimize}} & \dfrac{x^T Lx}{x^T Dx} \\ \text{subject to} & x^T De = 0,\ x^T Dx = 2m,\ x \text{ as in Equation} \end{array}$$

但该问题为NP问题，考虑问题的松弛解，

$$\begin{array}{ll} \underset{x\in\mathbb{R}^n}{\text{minimize}} & \dfrac{x^T Lx}{x^T Dx} \\ \text{subject to} & x^T De = 0,\ x^T Dx = 2m \end{array}$$

该问题可用谱分解的方法解决，进行变量替换，

$$\text{minimize } y^T \bar{L} y$$
$$\text{subject to } y^T D^{-1/2} e = 0, y^T y = 2m$$

令$y = \sum_{i>1} w_i q_i$, 其中q_i为对$\bar{L}$进行特征值分解得到的正交化特征向量，NOTE: $\bar{L}$ 为对称矩阵，且q_i不能取最小的特征值（因为限制 $y^T D^{-1/2} e = 0$ 要求y与q_1正交，问题转化为，

$$\text{mininize } \sum_{i>1} \lambda w_i^2$$
$$\text{subject to } \sum_{i>1} w_i^2 = 1$$

显然，该问题为求一个凸组合，最小值在顶点处取得，也即在第二小的特征值和特征向量取得，

该问题还有其他求解形式，考虑如下定义x, 也可以经过相似的松弛操作转化为类似的闭式解形式，

Given a partition $(A, \overline{A})$, define $\boldsymbol{x} \in \mathbb{R}^n$ such that

$$x_i = \begin{cases} \frac{1}{\text{vol}(A)} & \text{if } v_i \in A \\ -\frac{1}{\text{vol}(\overline{A})} & \text{if } v_i \in \overline{A} \end{cases}. \qquad (*)$$

One has

$$\boldsymbol{x}^\top \boldsymbol{L}\boldsymbol{x} = \text{cut}(A, \overline{A})(\text{vol}(A)^{-1} + \text{vol}(\overline{A})^{-1})^2,$$
$$\boldsymbol{x}^\top \boldsymbol{D}\boldsymbol{x} = \text{vol}(A)^{-1} + \text{vol}(\overline{A})^{-1}.$$

It follows that

$$\frac{\text{cut}(A, \overline{A})}{\text{vol}(A)} + \frac{\text{cut}(A, \overline{A})}{\text{vol}(\overline{A})} = \frac{\boldsymbol{x}^\top \boldsymbol{L}\boldsymbol{x}}{\boldsymbol{x}^\top \boldsymbol{D}\boldsymbol{x}}.$$

Thus, the normalized cut for two clusters is equivalent to

$$\min_{\boldsymbol{x}} \frac{\boldsymbol{x}^\top \boldsymbol{L}\boldsymbol{x}}{\boldsymbol{x}^\top \boldsymbol{D}\boldsymbol{x}} \quad \text{subject to} \quad \boldsymbol{x} \text{ is in the form of (*)}.$$

NOTE1:此处考虑优化的最小值函数称为NCUT定义，根据其他定义，也可以定义不同的x，经过类似地方法，得到类似的闭式解

NOTE2:上述考虑的问题为二分类谱聚类问题，对于多聚类问题，也应该定义不同的x ,此处暂略。

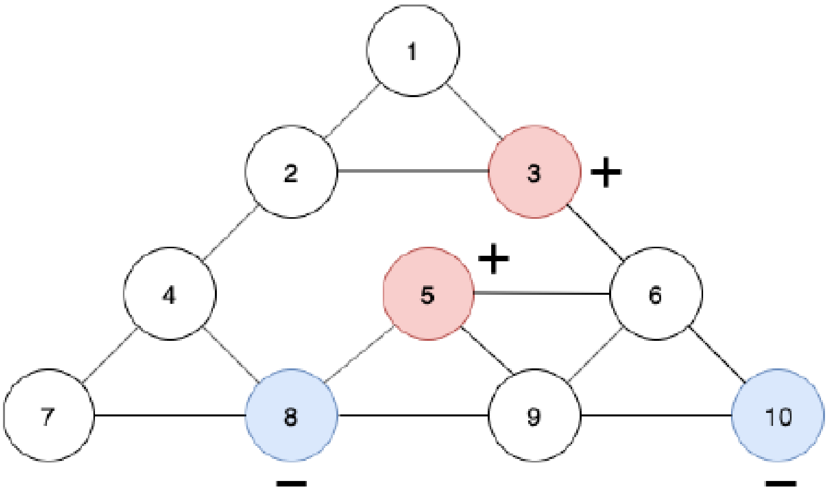
REF:https://www.cnblogs.com/pinard/p/6221564.html

HW2

陈乐偲

P1

1&2.计算联合概率，为所有最大团势函数的乘积。接着计算边缘概率，对其他变量求和即可，并可以证明消息传播算法得到了边缘概率。即证明了信念传播算法的正确性。



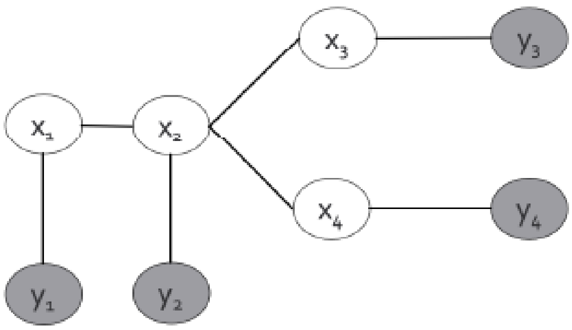
使用传播算法进行关系分类，直接迭代执行一遍，也可编程实现，略。

$$p(x_1, ..., x_n | y_1, y_2, ..., y_m) = \frac{1}{Z} \prod_{c \in C} \phi_c(x_c, y_c)$$

$$m_{ij}(x_j) = \sum_{x_i} \phi_i(x_i) \psi_{ij}(x_i, x_j) \prod_{k \in N_i \backslash j} m_{ki}(x_i),$$

条件随机场(CRF)的信念传播，

REF： http://helper.ipam.ucla.edu/publications/gss2013/gss2013_11344.pdf



1&2.计算联合概率，为所有最大团势函数的乘积。接着计算边缘概率，对其他变量求和即可，并可以证明消息传播算法得到了边缘概率。即证明了信念传播算法的正确性。

$$
\begin{aligned}
p(x1,x2,x3,x4,x5) &= \phi(x1,y1)\phi(x1,x2)\phi(x2,y2)\phi(x2,x3)\phi(x2,x4)\phi(x3,y3)\phi(x4,y4) \\
b(x1) &= \sum_{x2,x3,x4,x5} p(x1,x2,x3,x4,x5) \\
&= \phi(x1,y1)\sum_{x2} m_{21} \\
&= \phi(x1,y1)\sum_{x2}\phi(x2,y2)\sum_{x3} m_{32}\sum_{x4} m_{42} \\
&= \phi(x1,y1)\sum_{x2}\phi(x2,y2)\sum_{x3}\phi(x3,y3)\sum_{x4}\phi(x4,y4)
\end{aligned}
$$

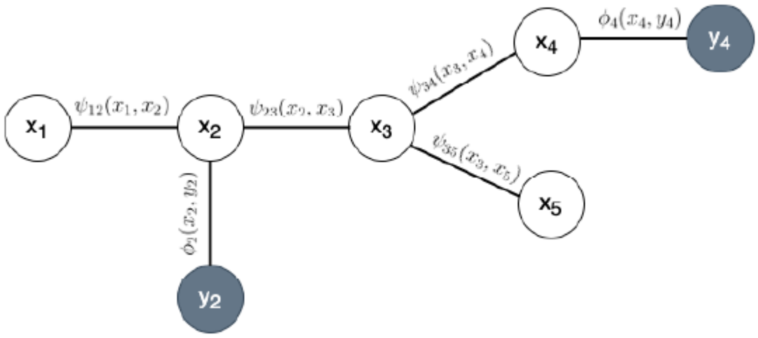
3. 

Figure 4: For problem (iii)

$\psi_{12}(x_1,x_2) = \psi_{34}(x_3,x_4) = \begin{bmatrix} 1 & 0.9 \\ 0.9 & 1 \end{bmatrix}$, $\psi_{23}(x_2,x_3) = \psi_{35}(x_3,x_5) = \begin{bmatrix} 0.1 & 1 \\ 1 & 0.1 \end{bmatrix}$, $\phi_2(x_2,y_2) = \phi_4(x_4,y_4) = \begin{bmatrix} 1 & 0.1 \\ 0.1 & 1 \end{bmatrix}$.

根据信念传播公式，如下执行消息传播，

$$
\begin{aligned}
m_{12} &= (1,1) \\
m_{22} &= (1,0.1) \\
m_{44} &= (0.1,1) \\
m_{53} &= (1,1) \\
m_{43} &= (1,1.09) \\
m_{32} &= (1.19,1.109) \\
m_{21} &= (1.2898,1.1819) \\
m_{23} &= (0.2,1.01) \\
m_{35} &= (1.1209,0.31009) \\
m_{34} &= (2.1881,2.18)
\end{aligned}
$$

消息传播全部结束之后，计算边缘概率，并归一化获得最终结果，

$$
\begin{aligned}
b(x1) &= (0.5218,0.4782) \\
b(x2) &= (0.91475,0.08525) \\
b(x3) &= (0.1528,0.8472) \\
b(x4) &= (0.0912,0.9088) \\
b(x5) &= (0.7833,0.2167)
\end{aligned}
$$

可见，信念传播也是一种传播方法，其中隐变量x2和隐变量x4受直接可观测变量y2，y4的影响最大。

P2

TransE算法，计算结点嵌入，

input Training set $S = \{(h, \ell, t)\}$, entities and rel. sets E and L, margin γ, embeddings dim. k.

initialize $\boldsymbol{\ell} \leftarrow \text{uniform}(-\frac{6}{\sqrt{k}}, \frac{6}{\sqrt{k}})$ for each $\ell \in L$

$\boldsymbol{\ell} \leftarrow \boldsymbol{\ell} / \|\boldsymbol{\ell}\|$ for each $\ell \in L$

$\mathbf{e} \leftarrow \text{uniform}(-\frac{6}{\sqrt{k}}, \frac{6}{\sqrt{k}})$ for each entity $e \in E$

loop

$\mathbf{e} \leftarrow \mathbf{e} / \|\mathbf{e}\|$ for each entity $e \in E$

$S_{batch} \leftarrow \text{sample}(S, b)$ // sample a minibatch of size b

$T_{batch} \leftarrow \emptyset$ // initialize the set of pairs of triplets

for $(h, \ell, t) \in S_{batch}$ **do**

$(h', \ell, t') \leftarrow \text{sample}(S'_{(h,\ell,t)})$ // sample a corrupted triplet

$T_{batch} \leftarrow T_{batch} \cup \{((h, \ell, t), (h', \ell, t'))\}$

end for

Update embeddings w.r.t. $\displaystyle\sum_{((h,\ell,t),(h',\ell,t')) \in T_{batch}} \nabla\left[\gamma + d(\boldsymbol{h} + \boldsymbol{\ell}, \boldsymbol{t}) - d(\boldsymbol{h'} + \boldsymbol{\ell}, \boldsymbol{t'})\right]_+$

end loop

1. $$\mathcal{L}_{\text{simple}} = \sum_{(h,\ell,t)\in S} d(\mathrm{h} + \ell, \mathrm{t}),$$

采用上述损失，当$l = (0, 0)$ 时，$h = t$, 损失达到最小值0，但不能提供任何有效嵌入信息，因此为无用嵌入。

2.

$$\mathcal{L}_{\text{no margin}} = \sum_{(h,\ell,t)\in S} \sum_{(h',\ell,t')\in S'_{(h,\ell,t)}} [d(\mathrm{h} + \ell, \mathrm{t}) - d(\mathrm{h}' + \ell, \mathrm{t}')]_+,$$

用上述损失改进，目标是最小化正样例距离和负样例距离的gap，但当所有正样例的距离和负样例距离都为0时，损失函数为0，此时仍为无用嵌入，

因此，需要采用带间隔的损失，

$$\sum_{((h,\ell,t),(h',\ell,t')) \in T_{batch}} \nabla\left[\gamma + d(\boldsymbol{h} + \boldsymbol{\ell}, \boldsymbol{t}) - d(\boldsymbol{h'} + \boldsymbol{\ell}, \boldsymbol{t'})\right]_+$$

3. 归一化的作用，

如果不归一化，会使得模长大的向量基本不变，而模长小的向量向模长大的向量方向靠拢，产生我们不想要的偏置。

4. TransE失败的情况

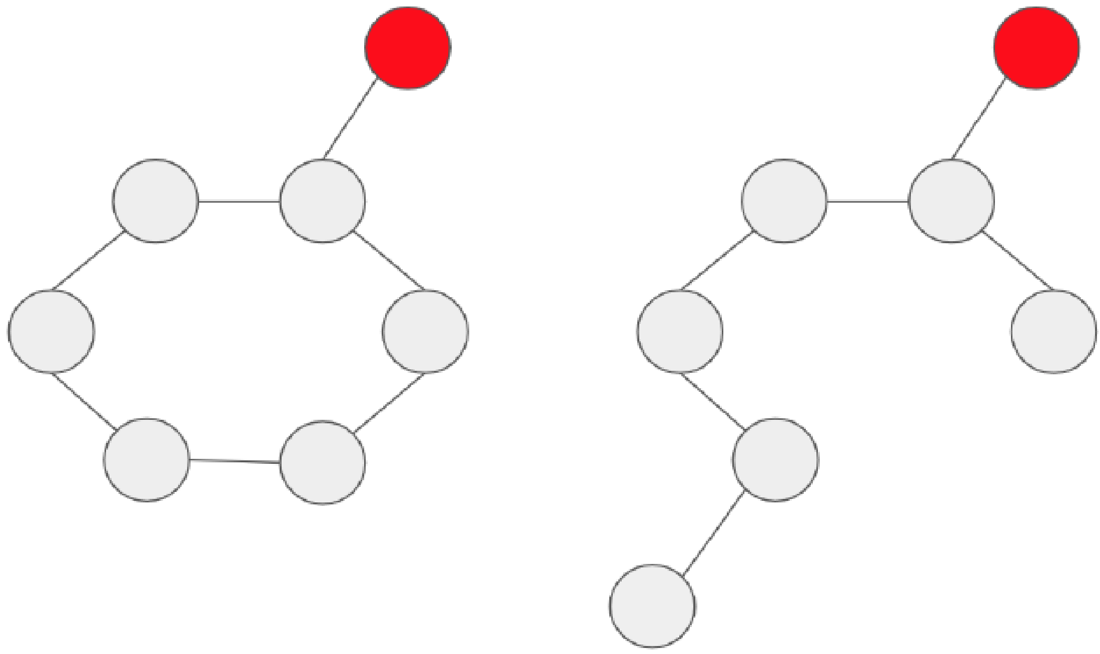
简单考虑一组1对n关系，每组关系相同，此时transE算法不能找到完美的嵌入使得所有正样例距离为0，

本质上，TransE的完美嵌入，相当于解一个线性方程组，但总可以轻易构建一个方程组使其无解。

P3

图神经网络，

1. GNN深度对模型判别能力的影响，



上述例子中两个红色结点具有完全相同的2跳邻居结构，所以2层及以下的GCN将为该节点构建相同的计算图，使得模型不能判别结点。

类似地，GCN可以用于判别环路，本质仍然是消息传递，和上面类似，略

NOTE：尽管GCN的能力受限于深度，但在现实社交网络中，根据六度空间理论，在六跳内一个结点大概率可以遍历整张图，所以并不需要采用深层GCN，很多时候浅层GCN的感受野已经可以满足需求。

2. GNN和随机游走的关系

一跳随机游走的转移公式可以写为， $H_{t+1} = D^{-1}AH_t$

增加自环的转移公式可以写为，$H_{t+1} = \frac{1}{2}D^{-1}(A+I)H_t$

也即，也可以把随机游走和基于随机游走的嵌入算法等，归结于GNN的消息传递的框架之下。

3.GNN的过平滑化证明，

考虑和上述一跳邻居随机游走相同定义的GNN， $H_{t+1} = D^{-1}AH_t$

令$L = D^{-1}A$, 可见L为随机矩阵，可以验证L的存在特征值为1，特征向量为$e = [1, 1...1]^T$ 的特征向量对，

即直接验证 $Le = e$,

且由反证法，可以证明1为L的最大特征值，下证：

若存在特征值λ大于1，且特征向量v不为$k[1, 1, \ldots 1]^T$ 的特征向量对，则存在$v_i = max|v_k|, j \neq max|v_k|$，

则$|(Lv)_i| < |v_i Le|_i = |v_i e|_i = |v_i|$, 与$\lambda > 1$的假设不符合， 矛盾，

也即L矩阵的谱半径为1，则迭代算法必收敛，且收敛到特征向量e,也即最终所有结点的标签都将相同，此即GNN的过平滑化。

NOTE：引入可学习参数、采用在多层GCN之间加入非线性激活函数等方法可减轻过平滑化的现象，但过平滑化仍然是GNN的重大问题。

4. 用GNN学习BFS，

将已经遍历的结点设为1，为遍历的结点设为0，每个时间戳，执行以下消息传递算法，

对于每个结点，如果状态为1，不变。如果状态为0，且存在一个邻居状态为1，将其状态设置为1.

每一个结点状态被设置为1的时间戳，便代表了其BFS序。如上定义的消息传递GNN，便可以学习到BFS序遍历。

NOTE:消息传递的性质天然适用于可达性查询等和BFS类似的图算法，因此也可以作为GNN的一种解释。

P4

训练GNN，比较三种GNN模型，GAT、GCN、SAGE

REF： https://colab.research.google.com/drive/1I8a0DfQ3fI7Njc62__mVXUlcAleUclnb?usp=sharing#scrollTo=ecJCNRmT2RsF

REF： https://colab.research.google.com/drive/14OvFnAXggxB8vM4e8vSURUp1TaKnovzX?usp=sharing

结点分类任务，GNN由两层卷积层组成

- GCN1(in_features, hidden_featrues)
- GCN2(hidden_features, num_classes)

```
import torch
from torch_geometric.datasets import Planetoid
import torch.nn.functional as F
from torch_geometric.nn import GCNConv,GATConv, SAGEConv

class GCNNet(torch.nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = SAGEConv(1433, 16)
        self.conv2 = SAGEConv(16, 7)

    def forward(self, data):
        x, edge_index = data.x, data.edge_index
        x = self.conv1(x, edge_index)
        x = F.relu(x)
        x = F.dropout(x, training=self.training)
        x = self.conv2(x, edge_index)
        return F.log_softmax(x, dim=1)

dataset = Planetoid(root='./datasets/Cora', name='Cora')
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
GCN = GCNNet().to(device)
data = dataset[0].to(device)
optimizer = torch.optim.Adam(GCN.parameters(), lr=0.01, weight_decay=5e-4)
GCN.train()
GCNaccs = []
for epoch in range(200):
    optimizer.zero_grad()
    out = GCN(data)
    loss = F.nll_loss(out[data.train_mask], data.y[data.train_mask])
    loss.backward()
    optimizer.step()
```

```
    _, pred = GCN(data).max(dim=1)
    correct = pred[data.test_mask].eq(data.y[data.test_mask]).sum()
    accuracy = correct / data.test_mask.sum()
    GCNaccs.append(accuracy.item())

print(GCNaccs)
GCN.eval()
_, pred = GCN(data).max(dim=1)
correct = pred[data.test_mask].eq(data.y[data.test_mask]).sum()
accuracy = correct / data.test_mask.sum()
print("accuracy", accuracy.item())

# acc 81.10%
```

图分类任务，ENZYMES数据集太难做（由于能力所限），准确率仅有30%左右，转而使用MUTAG数据集

GNN由几个结构组成，

- 多层GCN，计算结点嵌入
- 全局汇聚（sum/mean），计算图嵌入
- 线性层分类器，计算预测概率

```
import torch
from torch_geometric.datasets import TUDataset
from torch_geometric.data import DataLoader
from torch.nn import Linear
import torch.nn.functional as F
from torch_geometric.nn import GCNConv,GATConv,SAGEConv
from torch_geometric.nn import global_mean_pool
#dataset = TUDataset(root='datasets/ENZYMES', name='ENZYMES')
dataset = TUDataset(root='datasets/TUDataset', name='MUTAG')

#划分为数据集
torch.manual_seed(42)
dataset = dataset.shuffle()
train_dataset = dataset[:150]
test_dataset = dataset[150:]
print(len(train_dataset),len(test_dataset))

train_loader = DataLoader(train_dataset, batch_size=64, shuffle=True)
test_loader = DataLoader(test_dataset, batch_size=64, shuffle=False)

class GNN(torch.nn.Module):
    def __init__(self, hidden_channels):
        super().__init__()
        self.conv1 = SAGEConv(dataset.num_node_features, hidden_channels)
        self.conv2 = SAGEConv(hidden_channels, hidden_channels)
        self.conv3 = SAGEConv(hidden_channels, hidden_channels)
        self.lin = Linear(hidden_channels, dataset.num_classes)

    def forward(self, x, edge_index, batch):
        #结点嵌入
        x = self.conv1(x, edge_index)
        x = x.relu()
        x = self.conv2(x, edge_index)
        x = x.relu()
        x = self.conv3(x, edge_index)
        #全局汇聚
        x = global_mean_pool(x, batch)
        #分类
```

```
        #x = F.dropout(x, p=0.5, training=self.training)
        x = self.lin(x)
        return x

model = GNN(hidden_channels=64)
optimizer = torch.optim.Adam(model.parameters(), lr=0.01)
criterion = torch.nn.CrossEntropyLoss()

def train():
    model.train()
    for data in train_loader:
        optimizer.zero_grad()
        out = model(data.x, data.edge_index, data.batch)
        loss = criterion(out, data.y)
        loss.backward()
        optimizer.step()

def test(loader):
    model.eval()
    correct = 0
    for data in loader:
        out = model(data.x, data.edge_index, data.batch)
        pred = out.argmax(dim=1)
        correct += int((pred == data.y).sum())
    return correct / len(loader.dataset)

accs = []
for epoch in range(200):
    train()
    train_acc = test(train_loader)
    test_acc = test(test_loader)
    accs.append(test_acc)
    print(f'Epoch: {epoch:03d}, Train Acc: {train_acc:.4f}, Test Acc: {test_acc:.4f}')
print(accs)
```

cora是一个学术引用网络，每个结点表示一篇机器学习论文，结点特征为结点话题，边表示论文引用关系，数据集信息如下，

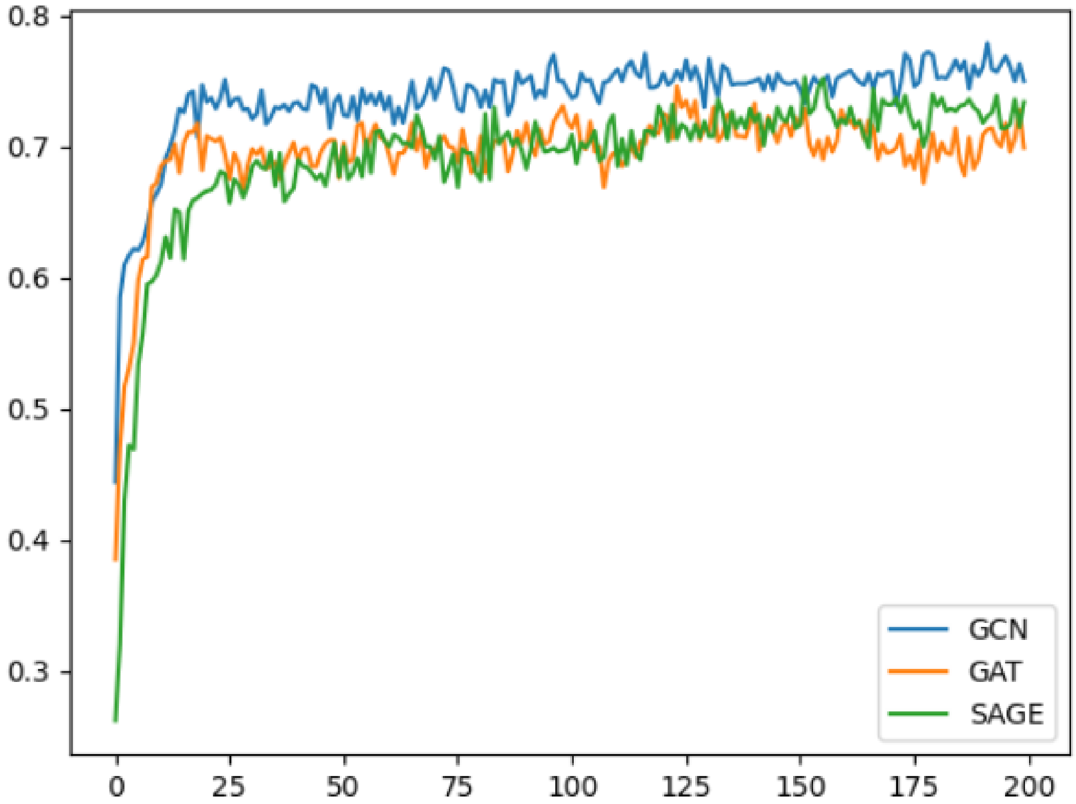
```
Dataset: Cora():
======================
Number of graphs: 1
Number of features: 1433
Number of classes: 7

Data(edge_index=[2, 10556], test_mask=[2708], train_mask=[2708], val_mask=[2708], x=[2708, 1433], y=
[2708])
===========================================================================================================
Number of nodes: 2708
Number of edges: 10556
Average node degree: 3.90
Number of training nodes: 140
Training node label rate: 0.05
Contains isolated nodes: False
Contains self-loops: False
Is undirected: True
```

MUTAG是一个化合物数据集，数据集中每张图表示一种化合物结构，目标是预测化合物是否为芳香族化合物，数据集信息如下，

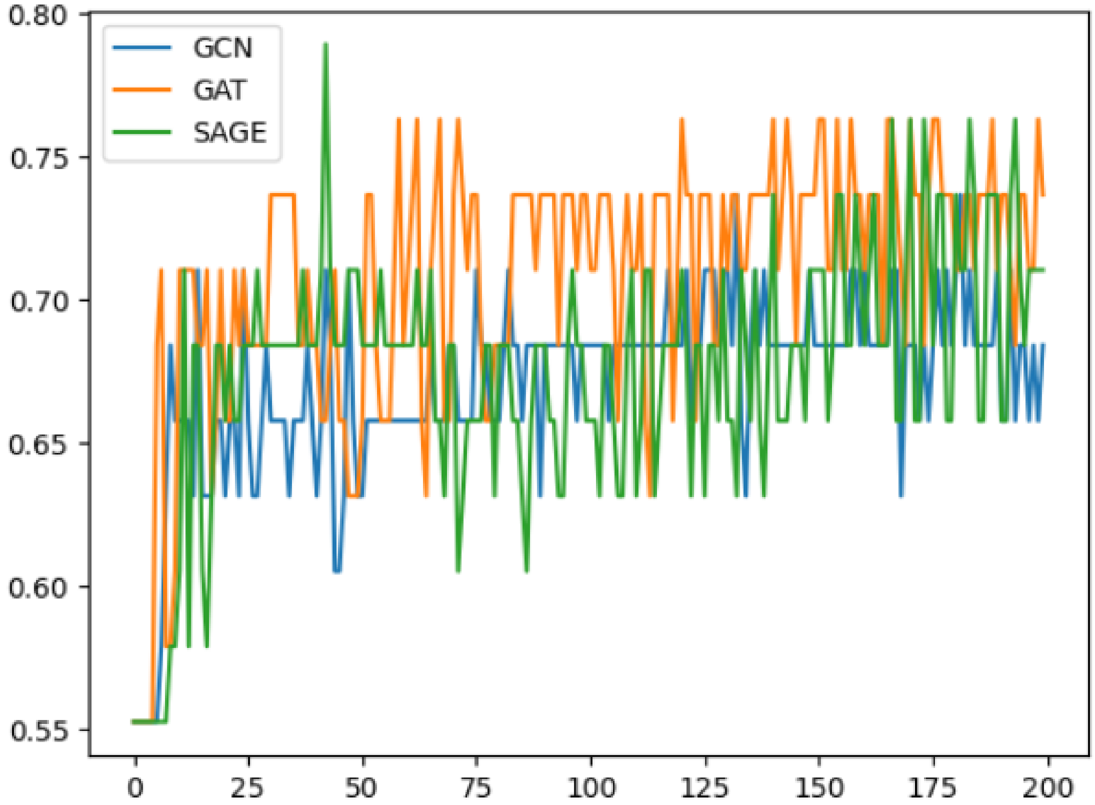
```
Dataset: MUTAG(188):
```

```
====================
Number of graphs: 188
Number of features: 7
Number of classes: 2

Data(edge_attr=[38, 4], edge_index=[2, 38], x=[17, 7], y=[1])
=============================================================
Number of nodes: 17
Number of edges: 38
Average node degree: 2.24
Contains isolated nodes: False
Contains self-loops: False
Is undirected: True
```

三种GNN对比如下，



结点分类任务不同GNN对比



图分类任务不同GNN对比

NOTE：

- 图分类任务比结点分类任务困难很多，训练时准确率的方差也很大。
- 在结点分类任务上，表现最好的模型是GCN，给出的解释是GCN可能更多地学习到了结点的不同角色
- 在图分类任务上，表现最好的模型是GAT,给出的解释是在化合物结构中，注意力机制具有很强烈的化学意义，比如由于某些化学基团的存在，不同种类原子对其连接的原子的重要性不尽相同，GAT天然具有该解释作用，也可以从一些GAT注意力的可视化中看出来。