社团挖掘算法审视

1 概述

社团挖掘将图的全体节点划分成一个个子集，每个子集称为一个社团，注意，社团挖掘问题一般只考虑点之间是否连通，而不考虑连接强度，边权默认是0-1的。设e_c表示社团c中的边的总数，K_c表示社团c中每个节点度之和，m表示整个网络中边的总，n表示整个网络中顶点的总数，γ是一个超参数，一个划分的模块度定义为：

$$Q = \frac{1}{2m}\sum_c \left(e_c - \gamma\frac{K_c^2}{2m}\right) \tag{1}$$

其中，

$$e_c = \sum_{i\in c, j\in c} A_{ij}, \quad k_i = \sum_j A_{ij}, \quad K_c = \sum_{i\in c} k_i, \quad 2m = \sum_{i,j} A_{ij} \tag{2}$$

A_{ij}表示点i和点j之间的边权(0-1)，k_i表示点i的度(权值之和)，c_i表示点i的社团，

$$\delta(c_i, c_j) = \begin{cases} 1, & \text{if } c_i == c_j \\ 0, & \text{else} \end{cases}, \quad s_{ic} = \begin{cases} 1, & \text{if } i \in c \\ 0, & \text{else} \end{cases}$$

Q值最初由 [1]提出，当时其形式如下：

$$Q = \sum_c (f_{cc} - f_c^2) \tag{3}$$

其中，f_{cc}表示社团内部边数占全部边数的比例，

$$f_{cc} = \frac{e_c}{\sum_{i,j} A_{ij}} = \frac{e_c}{2m}$$

$$f_c = \frac{\sum_{i\in c, j} A_{ij}}{\sum_{i,j} A_{ij}} = \frac{\sum_{i\in c} k_i}{2m} = \frac{K_c}{2m}$$

Q值越大表明社团划分质量越高，但是它的最优求解是困难的。 [2]给出了一个基于Q值的贪心算法-Fast Newman算法，这个算法初始把每个顶点当做一个单独的社团，每次计算每个社团pair合并导致的Q值增加量ΔQ，然后把使得ΔQ最大的合并对应的两个社团进行真正的合并。如果两个社团之间没有边相连，那么合并他们的$\Delta Q = 0$，所以只考虑有边的社团pair(最差时间复杂度为$O(m)$)。令$g_{c_1c_2}$表示社团c_1和社团c_2之间的边数，也就是：

$$g_{c_1c_2} = \sum_{i\in c_1, j\in c_2} A_{ij}$$

那么：

$$K_c = \sum_{i\in c}\sum_j A_{ij} = \sum_{i\in c}\sum_{c'}\sum_{j\in c'} A_{ij} = \sum_{c'}\sum_{i\in c}\sum_{j\in c'} A_{ij} = \sum_{c'} g_{cc'}$$

合并两个社团：$c_1, c_2 \to c$，对应的ΔQ为：

$$\begin{aligned}
\Delta Q &= \frac{1}{2m}\left(e_c - \frac{K_c^2}{2m} - e_{c_1} + \frac{K_{c_1}^2}{2m} - e_{c_2} + \frac{K_{c_2}^2}{2m}\right) \\
&= \frac{1}{2m}\left(g_{c_1c_2} - \frac{K_c^2 - K_{c_1}^2 - K_{c_2}^2}{2m}\right) && g\text{的定义} \\
&= \frac{1}{2m}\left(g_{c_1c_2} - \frac{K_{c_1}K_{c_2}}{2m}\right) && K_c = K_{c_1} + K_{c_2} \\
&= \frac{g_{c_1c_2}}{2m} - \frac{(\sum_{c'} g_{c_1c'}) * (\sum_{c'} g_{c_2c'})}{2m * 2m} && K_c = K_{c_1} + K_{c_2}
\end{aligned}$$

所以，初始矩阵$[g_{c_1,c_2}]$等于矩阵A(注意，初始对角线为1且这是对称矩阵)，每次合并c_1, c_2后就删除对应的两个行c_1, c_2和两个列c_1, c_2，并加入一个新列c，并且每个元素如下：

$$g_{cc'} = g_{c_1c'} + g_{c_2c'}$$

这一步的最差时间复杂度为$O(n)$。

Q也与下面常见的形式是等价的：

$$Q = \frac{1}{2m}\sum_{i,j}\left(A_{ij} - \gamma\frac{k_ik_j}{2m}\right)\delta(c_i, c_j) \tag{4}$$

二者的等价推导如下：

$$\begin{aligned}
&\frac{1}{2m}\sum_{i,j}\left(A_{ij} - \gamma\frac{k_ik_j}{2m}\right)\delta(c_i, c_j) \\
&= \frac{1}{2m}\left(\sum_i\sum_j A_{ij}\delta(c_i, c_j) - \sum_i\sum_j \gamma\frac{k_ik_j}{2m}\delta(c_i, c_j)\right) \\
&= \frac{1}{2m}\left(\sum_c\sum_{i\in c}\sum_{j\in c} A_{ij} - \sum_c\sum_{i\in c}\sum_{j\in c}\gamma\frac{k_ik_j}{2m}\right) && \text{通过在逐}c\text{累加去掉}\delta \\
&= \frac{1}{2m}\left(\sum_c e_c - \frac{\gamma}{2m}\sum_c\sum_{i\in c} k_i\sum_{j\in c} k_j\right) && e_c\text{的定义} \\
&= \frac{1}{2m}\left(\sum_c e_c - \frac{\gamma}{2m}\sum_c\sum_{i\in c} k_iK_c\right) && K_c\text{的定义} \\
&= \frac{1}{2m}\left(\sum_c e_c - \frac{\gamma}{2m}\sum_c K_c^2\right) \\
&= \frac{1}{2m}\sum_c\left(e_c - \frac{\gamma}{2m}K_c^2\right)
\end{aligned}$$

Q的定义是社团挖掘中具有开山意义的工作，所以我们一定要理解其定义的原理。我们看看图的随机化，给定一个图G，configuration model把每条边砍断成两半，每半叫stub，每个stub随机地和另外任意一个stub(除了他自己)连接起来，如此我们得到G的随机化的图。注意，这种随机图可能存在一条边的起点和终点都是一个点(self-loops)，而且可能两点之前存在多条边(multi-edges)。在这个随机图中点i有k_i的stub，整个图的stub总数为：

$$\sum_i k_i = 2m$$

$I_{ik} = 1$表示顶点i的第k个stub与顶点j的某个stub被随机连在一起，否则$I_{ik} = 0$。因为顶点i的第k个stub可以与等概率地与任意其余$2m-1$个stub连起来，而点j有k_j个stub，所以：

$$p(I_{ik} = 1) = E[I_{ik}] = \frac{k_j}{2m-1}$$

进而节点i与节点j之间边的期望总数为：

$$\sum_k E[I_{ik}] = \sum_k \frac{k_j}{2m-1} = \frac{k_i k_j}{2m-1}$$

对于一个很大的网络(m很大)，有

$$\frac{k_i k_j}{2m-1} \approx \frac{k_i k_j}{2m}$$

另外，对于一个很大的随机网络，self-loops和multi-edges会极难发生，所以我们我们说下式表示的是点i和点j之间存在边的概率：

$$\frac{k_i k_j}{2m}$$

而实际上点i和点j之间存在边的概率是A_{ij}，于是实际边数减去期望的边数为：

$$A_{ij} - \frac{k_i k_j}{2m}$$

到这里我们基本明白了模块度的定义了。

那么时有时无的γ是干什么的呢？Q的定义中用到了图的随机化，在这个过程中每个点可以等概率地与其他任何一个点连接起来，而实际中很多点(在完全随机情况下)只可能与一小部分其他点存在连接起来的可能性。如果这两个社团之间只有一条边$g_{c_1c_2}=1$，那么合并这两个社团(得到c)Q值增加为：

$$\begin{aligned}\Delta Q &= \frac{1}{2m}\left(g_{c_1c_2} - \frac{K_{c_1}K_{c_2}}{2m}\right) \\ &= \frac{1}{2m}\left(1 - \frac{K_{c_1}K_{c_2}}{2m}\right)\end{aligned}$$

如果m较大，同时c_1和c_2很小(意味着K_{c_1}和K_{c_2}也很小)，那么$\frac{K_{c_1}K_{c_2}}{m}$也会小于1，此时合并他们会导致Q值增大，于是便会将两个小社团进行合并。也就是在大网络中，小社团之间哪怕只有一条边相连，那么他们将会被合并，这便是由 [3]指出的resolution limit问题(无法发现小社团)。所以我们加入$\gamma > 1$的参数来缓解这个问题。这里还有不错的一个关于Q值的介绍。

我们再看看他的矩阵形式：

$$\begin{aligned}
&\frac{1}{2m}\sum_{i,j}\left(A_{ij} - \frac{k_i k_j}{2m}\right)\delta(c_i, c_j) \\
&= \frac{1}{2m}\sum_{i,j}\left(A_{ij} - \frac{k_i k_j}{2m}\right)\sum_c s_{ic}s_{jc} \\
&= \frac{1}{2m}\sum_{i,j}\sum_c\left(A_{ij} - \frac{k_i k_j}{2m}\right)s_{ic}s_{jc} \\
&= \frac{1}{2m}\sum_{i,j}\sum_c B_{ij}s_{ic}s_{jc} && \text{令} B_{ij} = A_{ij} - \tfrac{k_i k_j}{2m} \\
&= \frac{1}{2m}\sum_c\sum_i s_{ic}\sum_j B_{ij}s_{jc} \\
&= \frac{1}{2m}\sum_c\sum_i s_{ic}[BS]_{ic} && \text{令} S_{jc} = s_{jc} \\
&= \frac{1}{2m}\sum_c\sum_i S^T_{ci}[BS]_{ic} \\
&= \frac{1}{2m}\sum_c [S^T BS]_{cc} \\
&= \frac{1}{2m} tr(S^T BS)
\end{aligned}$$

2 Louvain算法与Leiden算法

Louvain算法 [4]把图G中每个顶点初始化为一个社团。然后执行多轮pass直图的全部顶点都成为一个社团，每个pass分成下面的两个phase：

- phase1. 然后扫描所有顶点。计算把该顶点迁入其邻居社团(与当前顶点有边)所带来的Q值增长。将本顶点迁入对应Q值增长最大且为正的邻居社团。
- phase2. 把phase1 形成的每个社团看成一个点，计算这些新点之间的边权，得到一个图G'，然后$G \leftarrow G'$，goto phase1。

这需要指出，尽管一开始每个顶点一个社团，但是随着我们合并的进行，很多社团的点会变多。比如一开始我们发现把$c_1 = \{v\}, c_2 = \{w\}$这两个社团合并起来Q值增长最大，那么我们得到一个新社团$c_3 = \{v, w\}$，接着我们可能发现应该合并$c_4 = \{u\}$与c_3，注意这个时候c_3里面可不止一个顶点了，于是我们将这个合并称为把点u迁入社团c_3。

我们这里(再次重复地)定义几个符号(注意，我们这里假定是加权有向图)：图表示为$G(V, E)$，$V(G)$表示图G的顶点集，C代表某个社团，另外，

$$N(v) = \{C \mid \exists w \in C \text{ 使得 } A_{vw} > 0\}, \quad m = \frac{1}{2}\sum_{vw} A_{vw}, \quad k_v = \sum_{w} A_{vw}$$

$$e_C = \sum_{u\in C, w\in C} A_{uw}, \quad k_v = \sum_{u} A_{uv}, \quad K_C = \sum_{u}\sum_{w\in C} A_{uw}, \quad S_{v\to C} = \sum_{w\in C} A_{vw}$$

因为：

$$\begin{aligned} Q &= \frac{1}{2m}\sum_c \left(e_c - \frac{K_c^2}{2m}\right) \\ &= \sum_c \left(\frac{e_c}{2m} - \left(\frac{K_c}{2m}\right)^2\right) \end{aligned}$$

所以我们可以得到把点v(也是社团$\{v\}$)迁入社团C时Q值增长为(注意当把点v迁入社团C中，则那些以v为起点，以任意C中点为终点的边将加入新社团)：

$$\begin{aligned} \Delta Q_{v\to C} &= \frac{e_C + \sum_{w\in C} A_{vw}}{2m} - \left(\frac{K_C + k_v}{2m}\right)^2 \\ &\quad - \left[0 - \left(\frac{k_v}{2m}\right)^2 + \frac{e_C}{2m} - \left(\frac{K_C}{2m}\right)^2\right] \qquad \text{注意} A_{vv} = 0 \\ &= \left[\frac{e_C + \sum_{w\in C} A_{vw}}{2m} - \left(\frac{K_C + k_v}{2m}\right)^2\right] - \left[\frac{e_C}{2m} - \left(\frac{K_C}{2m}\right)^2 - \left(\frac{k_v}{2m}\right)^2\right] \end{aligned} \tag{5}$$

两个社团c_1, C_2之间的边权重为：

$$A_{C_1C_2} = \sum_{u\in C_1, v\in C_2} A_{uv} \tag{6}$$

Louvain算法详细的过程见algorithm1。还有一个点需要特别指出，我们在phase1中会一遍又一遍地遍历图的顶点们看看是否有迁入可以执行，直到没有任何迁入可以执行，才进入phase2。为什么我们要多次遍历顶点呢？这是因为图在不断地变动，每个顶点的邻居社团们在不停地变。比如第一次遍历点u时发现没有邻居社团可以迁入，但是随着其他点加入其邻居社团，可能第二次遍历u时就会发现有某个邻居社团可以迁入。再比如，第一次遍历点u时发现迁入邻居社团C_1可以使Q值增长最大，但是第二次遍历发现迁入C_2可以使Q值增长更大(第一次与第二次不一样的原因是网络在不断地变化)，于是就会把u从C_1迁入C_2(图1)，这也正是Leiden算法的arguement，因为这会导致社团内部不连通。

Algorithm 1: Louvain Algorithm

初始化得到$G(V,E)$，每个顶点是一个社团，每个社团中只有一个顶点。
while $|V(G)|>1$ **do**
  $need \leftarrow false$。
  repeat
    $need \leftarrow false$。
    for $v \in V(G)$ **do**
      for $C \in N(v)$ **do**
        根据式(5)计算$\Delta Q_{v\to C}$。
      end
      if $\max_{C\in N(v)} \Delta Q_{v\to C} > 0$ **then**
        把v迁入$\arg\max_{C\in N(v)} \Delta Q_{v\to C}$。
        $need \leftarrow true$。
      end
    end
  until *need*;
  把上一步得到每个社团当做一个顶点，根据式(6)计算顶点间的边权，得到一副新图G'。
  $G \leftarrow G'$。
end

Louvain算法的结果跟节点的遍历顺序有关，实验发现最终的Q值其实与节点遍历顺序基本无关，不过计算时间确实收到这个顺序的影响。

Leiden算法 [5]提出Louvain算法会发现很差的解，甚至会出现社团内部不连通的情况。出现这种情况的原因是把一个社团内部的桥节点移走了。如图(2)中，本来红色的点与绿色的点们组成一个社团，后来如果把红色的点移走，而绿色社团保持不动的话，那么绿色社团便是一个内部不连通的社团。那么我们就要问了既然红色的点不该与绿色的点隶属一个社团那么我们怎么会傻傻地把它们放到一个社团然后又分开呢？这是因为Louvain算法是一个启发式算法，没有任何最优性的保证，所以完全可能出现一开始我们做了错误的分配然后进行更正的情况。只要实验中发现这种现象，那么Leiden算法论文的动机便是合理的。

出现图(2)的情况，显然应该对绿色的社团进行分裂({1,2,3,4,5}形成一个社团，{6,7,8,9,10}形成一个社团)，所以Leiden在Louvain算法的phase1和phase2之间加入一个分裂的phase(叫做refine操作)。分裂是逐个社团独立进行的，每个社团内部每个点被初始化为一个单点的社团，然后合并，合并的时候一个点v必须与其phase1移入的社团C是well connected才可能进行内部合并，这样我们就把那些桥节点剔除了，他们将自成一个社团，在下一个pass的phase1中与别的社团合并。well connected就是保证一个点与本社团至少γ%的点之间有边：

$$E(v, C-v) > \gamma(||C|| - 1)$$

$||C||$表示社团C中的点个数，$E(v,C-v)$表示点v与C之间的边数。而且在refine中，Leiden不像Louvain算法那样采用贪心原则仅仅把Q增长最大的点进行合并，而是将Q值增加量归一化为概率，以这个概率大小随机地与某个社团进行合并，这可以增加算法的探索能力。refine操作的伪代码见图(3)。

Louvain算法在每个pass的phase1中会反复遍历全部的点，直到没有Q值增加。而Leiden算法对此进行了改进，每次先将全部节点入队，然后依次出队节点，如果某节点v需要与某社团C合并，那么把v的邻居中不属于社团C且还没入队的点入队，因为只有这些点的Q值增量需要计算，直到队列为

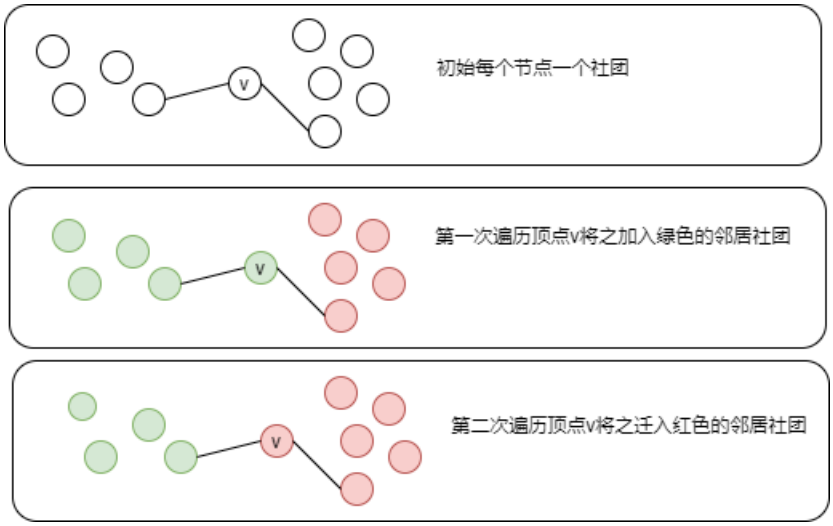

Figure 1: 顶点在社团间迁移的示意图

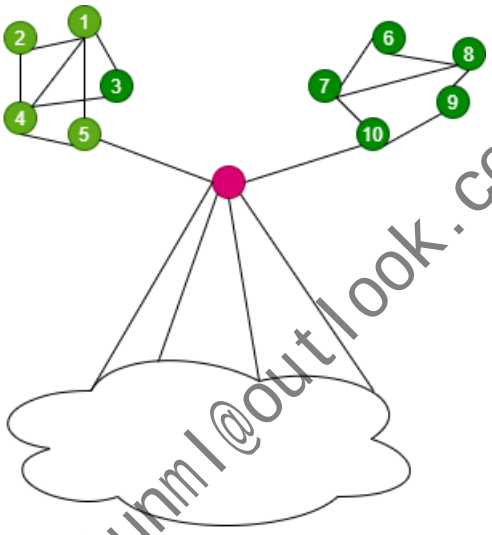

Figure 2: 桥节点移走的示意图

空，这样减少需要遍历的点的数量(这样所有节点先被访问一遍，然后就只有哪些Q值变化的点才会被访问)。

3 迭代优化下的社团挖掘

社团挖掘是一种图划分，与Graph Partitioning要做的事关系紧密。Graph Partitioning要解决的问题为：N和E表示一幅图的顶点集和边集，W_i表示每个定点的权重，W_{ij}表示边(i,j)的权重。Graph Partitioning旨在对N做一个划分，使得每个划分基本平衡(社团挖掘中不应该追求平衡)：

$$\begin{aligned} \min_{C_1,C_2,\ldots,C_k} \quad & \sum_{i,j} W(i,j)(1-\delta(c_i,c_j)) \\ \text{s.t} \quad & C_1 \cup C_2 \cup \cdots \cup C_k = N \\ & C_p \cap C_q = \emptyset \quad \forall p,q \\ & \sum_{i \in C_p} W_i \approx \sum_{i \in C_q} W_i \quad \forall p,q \end{aligned}$$

我们知道ratio-cut [6]和normalized-cut [7]是两个著名的谱聚类算法，而谱聚类是图划分的一类有效的算法，他们通过特征分解为每个节点找到一个embedding，然后基于embedding做k-means实现图的划分。那么啥叫平衡呢？也就是每个划分内顶点权重之和不要差别过大，比如 [8]通过最小化下面的目

P初始时是单点构成的社团集，S是上一步(phase1)得到的一个社团　　出现一个所谓well connected的概念，其意思就是一个点与本社团百分之多少的点之间有边，这个百分比通过超参数指定。

function MERGENODESSUBSET(Graph G, Partition $\mathcal{P}$, Subset S)
  $R = \{v \mid v \in S, E(v, S-v) \geq \gamma \|v\| \cdot (\|S\| - \|v\|)\}$ ▷ Consider only nodes that are well connected within subset S
  for $v \in R$ **do** 只考虑那些与S内部那些连通良好的点 ▷ Visit nodes (in random order)
    if v in singleton community **then** ▷ Consider only nodes that have not yet been merged
      $\mathcal{T} \leftarrow \{C \mid C \in \mathcal{P}, C \subseteq S, E(C, S-C) \geq \gamma \|C\| \cdot (\|S\| - \|C\|)\}$ ▷ Consider only well-connected communities
      $\Pr(C' = C) \sim \begin{cases} \exp\left(\frac{1}{\theta} \Delta \mathcal{H}_{\mathcal{P}}(v \mapsto C)\right) & \text{if } \Delta \mathcal{H}_{\mathcal{P}}(v \mapsto C) \geq 0 \\ 0 & \text{otherwise} \end{cases}$ for $C \in \mathcal{T}$ ▷ Choose random community C'
      $v \mapsto C'$ 只考虑社团S内部的子社团，这些子社团在S中不能是隔离的!!!! 然后计算吧S外部点移入这些子社团的Q值增益，把这个增益换算成移入概率。 ▷ Move node v to community C'
    end if
  end for 特别注意：
  return $\mathcal{P}$ 1. 每次得到一个社团划分后，做refine时是在这个社团划分的每个社团中看那些点适合合并成子社团，如此便实现了对社团的子划分。
end function 2. 合并时是根据Q增益大小对应的概率做随机合并，是一种探索。

也只考虑与内部那些连通良好的社团做合并

Figure 3: refine操作伪代码

标函数来得到一个平衡的二划分：

$$\frac{cut(C_1, C_2)}{weight(C_1)} + \frac{cut(C_1, C_2)}{weight(C_2)}$$

其中，

$$cut(C_1, C_2) = \sum_{j \in C_1} \sum_{j \in C_2} W_{ij}, weight(C) = \sum_{i \in C} W_i$$

这个问题最后变成求下面的广义特征值问题：

$$(\boldsymbol{D} - \boldsymbol{W})\boldsymbol{x} = \lambda \boldsymbol{W}_g \boldsymbol{x}$$

其中

$$\boldsymbol{D} = diag(\sum_i W_{ij}), \boldsymbol{W}_g = diag(W_i)$$

求其第二小，第三小，...,第$l+1$小特征值对应的特征向量，由此便形成node的长度为l的embedding，接着基于这个embedding做k-means。Ratio-cut中所有点的权重都是1，Normalized-cut中顶点的权重是其度(也就是k_i，见式(2))。 多划分时normalized-cut的loss为：

$$\sum_p \frac{cut(C_p, \overline{C}_p)}{vol(C_p)} = \sum_p \frac{\sum_{i \in C_k, j \in \overline{C}_p} W_{ij}}{\sum_{i \in C_p} k_i}$$

其中$\overline{C}_p$是C_p的补集。

我们这里介绍下GAP [9]，它使用deep learning来实现图划分的算法，GAP主要是提出了一个可微的目标函数。deep learning才是做embedding的专业户，而且是generalizable 呢！ GAP使用GNN(graph neural networks) 来得到节点的embedding，然后喂入一个MLP 得到每个节点i属于每个划分C_p的概率y_{ip}。GAP重新定义了cut：

$$\begin{aligned} cut(C_p, \overline{C}_p) &= \sum_i \sum_j y_{ip}(1 - y_{jp}) W_{ij} \\ &= \sum_i \sum_j [\boldsymbol{Y}[:, p](\boldsymbol{1} - \boldsymbol{Y}[:, p])^T]_{ij} W_{ij} \\ &= \sum_i \sum_j [\boldsymbol{Y}[:, p](\boldsymbol{1} - \boldsymbol{Y}[:, p])^T \odot W]_{ij} \end{aligned}$$

其中$\boldsymbol{1}$是全1矩阵，$[\boldsymbol{Y}]_{ip} = y_{ip}$，$\odot$表示element-wise multiplication。可以看出$y_{ip}(1 - y_{jp})$表示两个点之间的边被砍掉的期望损失(两点间有边，$W_{ij} > 0$才会计算对应的项，砍掉后i归入C_p，j归入$\overline{C}_p$)。这确实可以作为划分的loss！看起来很合理很自然。GAP还重新定义了$vol(C_p)$：

$$\begin{aligned} vol(C_p) &= \sum_i y_{ip} k_i \\ &= [\boldsymbol{Y}^T \boldsymbol{d}]_p \end{aligned}$$

其中$\boldsymbol{d}$是一个向量$\boldsymbol{d}_i = k_i$。这样normalized-cut的loss可以被写为：

$$
\begin{aligned}
&\sum_p \frac{cut(C_p, \overline{C}_p)}{vol(C_p)} \\
&= \sum_p \frac{\sum_i \sum_j y_{ip}(1-y_{jp})W_{ij}}{\sum_{i'} y_{i'p} k'_i} \\
&= \sum_p \sum_i \sum_j \frac{y_{ip}(1-y_{jp})W_{ij}}{\sum_{i'} y_{i'p} k'_i} \\
&= \sum_i \sum_j \sum_p \frac{y_{ip}(1-y_{jp})W_{ij}}{\sum_{i'} y_{i'p} k'_i} \\
&= \sum_i \sum_j \sum_p \frac{y_{ip}}{\sum_{i'} y_{i'p} k'_i}(1-y_{jp})W_{ij} \\
&= \sum_i \sum_j \left(\sum_p \frac{y_{ip}}{\sum_{i'} y_{i'p} k'_i}(1-y_{jp}) \right) W_{ij} \\
&= \sum_i \sum_j \left(\sum_p \frac{y_{ip}}{[\boldsymbol{Y}^T \boldsymbol{d}]_p}(1-y_{jp}) \right) W_{ij} \\
&= \sum_i \sum_j \left(\sum_p [\boldsymbol{Y} \oslash (\boldsymbol{Y}^T \boldsymbol{d})]_{ip}(1-y_{jp}) \right) W_{ij} \\
&= \sum_i \sum_j \left(\sum_p [\boldsymbol{Y} \oslash [\boldsymbol{Y}^T \boldsymbol{d}]]_{ip} [\boldsymbol{1} - \boldsymbol{Y}]^T_{pj} \right) W_{ij} \\
&= \sum_i \sum_j [\boldsymbol{Y} \oslash (\boldsymbol{Y}^T \boldsymbol{d})(\boldsymbol{1} - \boldsymbol{Y})^T]_{ij} W_{ij} \\
&= \sum_i \sum_j [\boldsymbol{Y} \oslash (\boldsymbol{Y}^T \boldsymbol{d})(\boldsymbol{1} - \boldsymbol{Y})^T \odot W]_{ij}
\end{aligned}
$$

对于$\boldsymbol{A} \in R^{M \times N}, \boldsymbol{b} \in R^N, [\boldsymbol{A} \oslash \boldsymbol{b}]_{ij} = \frac{\boldsymbol{A}_{ij}}{\boldsymbol{b}_j}$。

GAP告诉我们，离散问题可以通过对变量做放松从而转变为可微问题。我们套用这个思路可以把社团挖掘中的Q可微化：

$$
\begin{aligned}
Q &= \frac{1}{2m} \sum_c \left(e_c - \gamma \frac{K_c^2}{2m} \right) = \frac{1}{2m} \sum_c \left(\sum_{ij} y_{ic} y_{jc} W_{ij} - \frac{\gamma}{2m} (\sum_i y_{ic} k_i)^2 \right) \\
&= \frac{1}{2m} \left(\sum_c \sum_{ij} y_{ic} y_{jc} W_{ij} - \frac{\gamma}{2m} \sum_c (\sum_i y_{ic} k_i)^2 \right) \\
&= \frac{1}{2m} \left(\sum_{ij} \sum_c y_{ic} y_{jc} W_{ij} - \frac{\gamma}{2m} \sum_c (\sum_i y_{ic} k_i)^2 \right) \\
&= \frac{1}{2m} \left(\sum_{ij} [\boldsymbol{Y}\boldsymbol{Y}^T \odot W]_{ij} - \frac{\gamma}{2m} \sum_c (\boldsymbol{d}^T \boldsymbol{Y}[:, c])^2 \right) \\
&= \frac{1}{2m} \left(\sum_{ij} [\boldsymbol{Y}\boldsymbol{Y}^T \odot W]_{ij} - \frac{\gamma}{2m} \sum_c \boldsymbol{d}^T \boldsymbol{Y}[:, c] \boldsymbol{d}^T \boldsymbol{Y}[:, c] \right) && \text{两个实数的乘法} \\
&= \frac{1}{2m} \left(\sum_{ij} [\boldsymbol{Y}\boldsymbol{Y}^T \odot W]_{ij} - \frac{\gamma}{2m} \sum_c \boldsymbol{d}^T \boldsymbol{Y}[:, c] (\boldsymbol{Y}[:, c])^T \boldsymbol{d} \right) && \text{内积交换律} \\
&= \frac{1}{2m} \left(\sum_{ij} [\boldsymbol{Y}\boldsymbol{Y}^T \odot W]_{ij} - \frac{\gamma}{2m} \sum_c \boldsymbol{d}^T \left(\boldsymbol{Y}[:, c] (\boldsymbol{Y}[:, c])^T \right) \boldsymbol{d} \right) && \text{矩阵乘法结合律} (AB)C = A(BC) \\
&= \frac{1}{2m} \left(\sum_{ij} [\boldsymbol{Y}\boldsymbol{Y}^T \odot W]_{ij} - \frac{\gamma}{2m} \boldsymbol{d}^T \sum_c \left(\boldsymbol{Y}[:, c] (\boldsymbol{Y}[:, c])^T \right) \boldsymbol{d} \right) && \boldsymbol{x}^T \boldsymbol{A} \boldsymbol{x} + \boldsymbol{x}^T \boldsymbol{B} \boldsymbol{x} = \boldsymbol{x}^T (\boldsymbol{A} + \boldsymbol{B}) \boldsymbol{x} \\
&= \frac{1}{2m} \left(\sum_{ij} [\boldsymbol{Y}\boldsymbol{Y}^T \odot W]_{ij} - \frac{\gamma}{2m} \boldsymbol{d}^T \left(\boldsymbol{Y}\boldsymbol{Y}^T \right) \boldsymbol{d} \right)
\end{aligned}
\tag{7}
$$

我们这里强调一下，如果$y_{ic} = f_\theta(x_i)$，其中x_i表示节点i的embedding，f_θ代表参数为θ的神经网络，那么根据式(7)Q相对于θ是可微的，于是我们便可以用梯度下降来更新θ，使得Q值尽量小。当然我们这样做有如下几个损失：

- 我们得社团数(类别数)不能太多，那会导致$\boldsymbol{Y}$的列数过多。
- 必须提前取定社团数(类别数)，不能学习(有可能可以，不过目前路径不明)。
- 我们得到的社团(类别)结构只能是一个平面结构，无法将社团组织成一个层次结构(有可能可以，不过目前路径不明)

另外，我们得到是软分配，可能会太软了(点属于许多个类，但是每个隶属概率又特别小)，甚至会出现有些点均匀地属于每个类这种极差的解。我们可以找几个具有明显社团结构的数据集，每个点i属于每个社团的概率从分布$N(y_i, \sigma^2)$中随机生成并归一化，其中y_i是点i的真实社团或者使用Lovain算法得到的社团，σ的值就是我们分配的方差，越小越接近原来的硬分配，我们可以看看不同σ下Q的大小，如果Q随着σ单调递减且斜率大，那么这个方法是不可行的，就必须约束分配的方差。我们可以猜测Q绝对不会随着σ递增，否则Q的定义就是错误的，或者我们不该选用正态分布吧。另外如果下式的loss可行：

$$\frac{1}{2m}\left(\sum_{ij}[\boldsymbol{Y}\boldsymbol{Y}^T \odot W]_{ij} - \frac{\gamma}{2m}\boldsymbol{d}^T\left(\boldsymbol{Y}\boldsymbol{Y}^T\right)\boldsymbol{d}\right) + \sum_i ||\boldsymbol{Y}[i,:]||_1$$

也就是使得$\boldsymbol{Y}[i,:]$尽量稀疏，不要太过放松，那么更好，可问题是神经网络能优化L_1吗？大约能。另外这样对输出值而非参数做范数惩罚行得通吗？需要进一步研究下这些问题。

References

[1] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69(2), Feb 2004.

[2] M. E. J. Newman. Fast algorithm for detecting community structure in networks. *Physical Review E*, 69(6), Jun 2004.

[3] S. Fortunato and M. Barthelemy. Resolution limit in community detection. *Proceedings of the National Academy of Sciences*, 104(1):36–41, Dec 2006.

[4] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10):P10008, 2008.

[5] Vincent A Traag, Ludo Waltman, and Nees Jan Van Eck. From louvain to leiden: guaranteeing well-connected communities. *Scientific reports*, 9(1):1–12, 2019.

[6] Lars Hagen and Andrew B Kahng. New spectral methods for ratio cut partitioning and clustering. *IEEE transactions on computer-aided design of integrated circuits and systems*, 11(9):1074–1085, 1992.

[7] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000.

[8] Inderjit S Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 269–274, 2001.

[9] Azade Nazi, Will Hang, Anna Goldie, Sujith Ravi, and Azalia Mirhoseini. Gap: Generalizable approximate graph partitioning framework. *arXiv preprint arXiv:1903.00614*, 2019.