图的社区检测：标签传播（LPA）

覃雄派

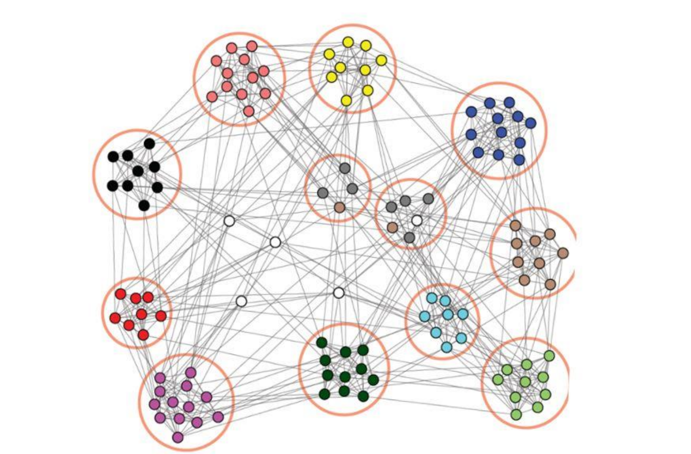
图的社区检测：标签传播（LPA）

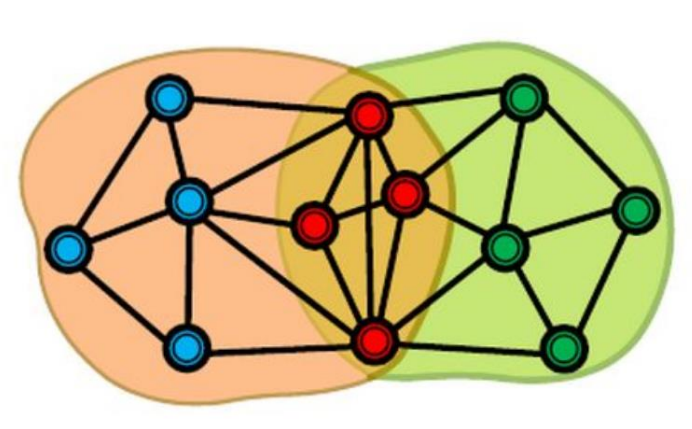
提纲

- 社区发现问题分类
- 非重叠社区发现的两大类算法
- 标签传播算法的思想
- 标签传播算法详解
  - 优点
  - 缺点
- 标签传播算法实践

图的社区检测：标签传播（LPA）

- 社区发现问题分为两大类
  - 非重叠社区发现和重叠社区发现
  - 非重叠社区发现问题描述的是
    - 一个网络中，每个节点均只能属于同一个社区，这意味这社区和社区之间是没有交集的





图的社区检测：标签传播（LPA）

- 在非重叠社区发现算法中，有不同种类的解法：
  - 1、基于模块度的社区发现算法
    - 基本思想是通过定义模块度（Modularity）来衡量一个社区的划分是不是相对比较好的结果，从而将社区发现问题转化为最大化模块度的问题进行求解
  - 2、基于标签传播的社区发现算法
    - 基本思想是通过标记节点的标签信息来更新未标记节点的标签信息，在整个网络中进行传播，直至收敛
    - 其中最具代表性的就是标签传播算法(Label Propagation Algorithm, LPA)

图的社区检测：标签传播（LPA）

- 标签传播算法(Label Propagation Algorithm, LPA)思想

Near linear time algorithm to detect community structures in large-scale networks

[1]Usha Nandini Raghavan, [2]Réka Albert and [1]Soundar Kumara
[1]*Department of Industrial Engineering, The Pennsylvania State University, University Park, Pennsylvania, 16802, USA*
[2]*Department of Physics, The Pennsylvania State University, University Park, Pennsylvania, 16802, USA*

Community detection and analysis is an important methodology for understanding the organization of various real-world networks and has applications in problems as diverse as consensus formation in social communities or the identification of functional modules in biochemical networks. Currently used algorithms that identify the community structures in large-scale real-world networks require *a priori* information such as the number and sizes of communities or are computationally expensive. In this paper we investigate a simple label propagation algorithm that uses the network structure alone as its guide and requires neither optimization of a pre-defined objective function nor prior information about the communities. In our algorithm every node is initialized with a unique label and at every step each node adopts the label that most of its neighbors currently have. In this iterative process densely connected groups of nodes form a consensus on a unique label to form communities. We validate the algorithm by applying it to networks whose community structures are known. We also demonstrate that the algorithm takes an almost linear time and hence it is computationally less expensive than what was possible so far.

PACS numbers: 89.75.Fb, 89.75.Hc, 87.23.Ge, 02.10.Ox

图的社区检测：标签传播（LPA）

- 标签传播算法(Label Propagation Algorithm, LPA)思想
  - LPA基本思想是节点的标签(community)依赖其邻居节点的标签信息，影响程度由节点相似度决定，并通过传播迭代更新达到稳定
  - 起初每个节点拥有独立的标签，那么网络中有n不同标签
  - 每次迭代中对于每个节点将其标签
    - 更改为其邻接点中出现次数最多的标签
    - 如果这样的标签有多个，则随机选择一个
  - 通过迭代，直到每个节点的标签与其邻接点中出现次数最多的标签相同，则达到稳定状态，算法结束

图的社区检测：标签传播（LPA）

- 标签传播算法(LPA)步骤

1. Initialize the labels at all nodes in the network. For a given node x, $C_x(0) = x$.
2. Set $t = 1$.
3. Arrange the nodes in the network in a random order and set it to X.
4. For each $x \in X$ chosen in that specific order, let $C_x(t) = f(C_{x_{i1}}(t), ..., C_{x_{im}}(t), C_{x_{i(m+1)}}(t-1), ..., C_{x_{ik}}(t-1))$. f here returns the label occurring with the highest frequency among neighbors and ties are broken uniformly randomly.
5. If every node has a label that the maximum number of their neighbors have, then stop the algorithm. Else, set $t = t + 1$ and go to (3).

  – 初始化阶段：每个节点都会初始化自己作为社区标签

图的社区检测：标签传播（LPA）

- 标签传播算法(LPA)步骤

1. Initialize the labels at all nodes in the network. For a given node x, $C_x(0) = x$.

2. Set $t = 1$.

3. Arrange the nodes in the network in a random order and set it to X.

4. For each $x \in X$ chosen in that specific order, let $C_x(t) = f(C_{x_{i1}}(t), ..., C_{x_{im}}(t), C_{x_{i(m+1)}}(t-1), ..., C_{x_{ik}}(t-1))$. f here returns the label occurring with the highest frequency among neighbors and ties are broken uniformly randomly.

5. If every node has a label that the maximum number of their neighbors have, then stop the algorithm. Else, set $t = t + 1$ and go to (3).

  – 传播阶段：遍历网络中所有的节点，找到当前节点的所有邻居节点
    - 获取所有邻居节点的社区标签，并找到权重最大(投票思想)的社区标签（对于有权图，就是所有同社区标签的edge weight之和；对于无权图，edge weight看做是1，就是出现次数做多的社区标签），将其为更新自己的社区标签

图的社区检测：标签传播（LPA）

- 标签传播算法(LPA)步骤

1. Initialize the labels at all nodes in the network. For a given node x, $C_x(0) = x$.
2. Set $t = 1$.
3. Arrange the nodes in the network in a random order and set it to X.
4. For each $x \in X$ chosen in that specific order, let $C_x(t) = f(C_{x_{i1}}(t), ..., C_{x_{im}}(t), C_{x_{i(m+1)}}(t-1), ..., C_{x_{ik}}(t-1))$. f here returns the label occurring with the highest frequency among neighbors and ties are broken uniformly randomly.
5. If every node has a label that the maximum number of their neighbors have, then stop the algorithm. Else, set $t = t + 1$ and go to (3).

  – 收敛判定阶段：遍历网络中所有的节点，找到当前节点的所有邻居节点
    - 获取所有邻居节点的社区标签，并找到权重最大的社区标签，判定是否是自己的社区标签，如果均判定通过，则算法结束
    - 为防止震荡情况出现，应设置一个最大迭代次数；达到最大迭代次数后也退出

图的社区检测：标签传播（LPA）

图的社区检测：标签传播（LPA）

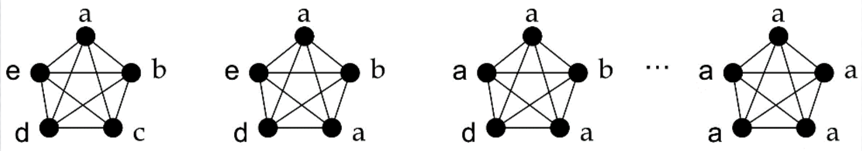
- **标签传播方式**
  - 标签传播方式分为两种：同步更新、异步更新
  - 1、同步更新：在第 t 次迭代中，每个节点依赖的都是邻居节点上一次迭代 t-1 时的社区标签
  - 2、异步更新：在第 t 次迭代中，每个节点依赖的是当前邻居节点的社区标签，若邻居节点进行了更新，则依赖的是 t 时的社区标签，若未进行更新，则依赖的是 t-1 时的社区标签

图的社区检测：标签传播（LPA）

- 主要的优点
  - 1、算法逻辑简单，时间复杂度低，接近线性复杂度，在超大规模网络下会有优异的性能，适合做baseline
  - 2、无须定义优化函数，无须事先指定社区个数，算法会利用自身的网络结构来指导标签传播

图的社区检测：标签传播（LPA）

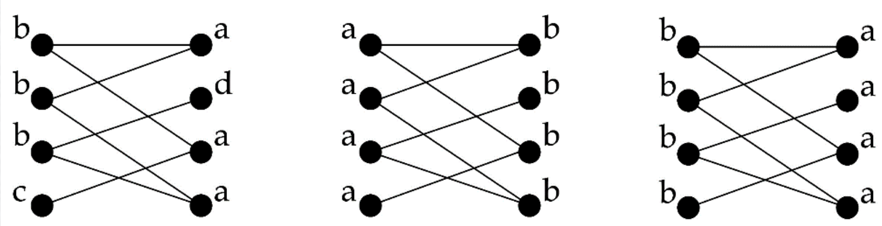
- 主要的缺点
  - 1、雪崩效应：社区结果不稳定，随机性强
    - 由于当邻居节点的社区标签权重相同时，会随机取一个；导致传播初期一个小的错误被不断放大，最终没有得到合适的结果
    - 尤其是异步更新时，更新顺序的不同也会导致最终社区划分结果不同



- 初始化阶段，每个节点都以自己作为社区标签。比如a的社区就是a，c的社区就是c
- 当进入传播阶段，节点c的邻居节点共4个：a，b，e，d。而社区标签也是4个：a，b，e，d，假设随机取了一个a
  - 如果是异步更新，此时b，d，e三个节点的邻居节点中社区标签均存在2个a，所以他们都会立马更新成a
- 如果c当时随机选择的是b，那么d,e就会更新成b，从而导致b社区标签占优，而最终的社区划分也就成b了

图的社区检测：标签传播（LPA）

- 主要的缺点有
  - 2、震荡效应：社区结果来回震荡，不收敛
  - 当传播方式处于同步更新的时候，尤其对于二分图或子图存在二分图的结构而言，极易发生



- 在同步更新的时候，每个节点依赖的都是上一轮迭代的社区标签
- 当二分图左边都是a，右边都是b时，a社区的节点此时邻居节点都是b，b社区的节点此时邻居节点都是a，根据更新规则，此时a社区的节点将全部更新为b，b社区的节点将全部更新为a
- 此时算法无法收敛，使得整个网络处于震荡中

图的社区检测：标签传播（LPA）

图的社区检测：标签传播（LPA）

- 标签传播（LPA）算法实践

名称	修改日期	类型	大小
02lpa.py	2021/8/5 12:26	Python File	1 KB

图的社区检测：标签传播（LPA）

- 标签传播（LPA）算法实践
  - 运行算法，打印信息

```python
#https://www.programmersought.com/article/19431905738/

import networkx as nx

G=nx.karate_club_graph()

from networkx.algorithms import community
def label_propagation_community(G):
    communities_generator = list(community.label_propagation_communities(G))
    m = []
    for i in communities_generator:
        m.append(list(i))
    return m

g=label_propagation_community(G)
print(g)

map_all ={}
community_num = 0;
for list_one in g:
    for list_item in list_one:
        map_all[list_item] = community_num
    community_num = community_num+1

print( map_all)
```

社区划分

[[32, 33, 2, 8, 9, 14, 15, 18, 20, 22, 23, 26, 27, 28, 29, 30], [16, 5, 6], [0, 1, 3, 4, 7, 10, 11, 12, 13, 17, 19, 21, 24, 25, 31]]

社区标签

{32: 0, 33: 0, 2: 0, 8: 0, 9: 0, 14: 0, 15: 0, 18: 0, 20: 0, 22: 0, 23: 0, 26: 0, 27: 0, 28: 0, 29: 0, 30: 0, 16: 1, 5: 1, 6: 1, 0: 2, 1: 2, 3: 2, 4: 2, 7: 2, 10: 2, 11: 2, 12: 2, 13: 2, 17: 2, 19: 2, 21: 2, 24: 2, 25: 2, 31: 2}

图的社区检测：标签传播（LPA）

- 标签传播（LPA）算法实践
  - 结果可视化（代码）

```python
import matplotlib.pyplot as plt
import matplotlib.cm as cm

partition =map_all
# draw the graph
pos = nx.spring_layout(G)
# color the nodes according to their partition
cmap = cm.get_cmap('viridis', max(partition.values()) + 1)
nx.draw_networkx_nodes(G, pos, partition.keys(), node_size=160,
                       cmap=cmap, node_color=list(partition.values()))
nx.draw_networkx_edges(G, pos, alpha=0.5)
plt.show()
```

图的社区检测：标签传播（LPA）

- 标签传播（LPA）算法实践
  - 结果可视化（效果）

```python
import matplotlib.pyplot as plt
import matplotlib.cm as cm

partition =map_all
# draw the graph
pos = nx.spring_layout(G)
# color the nodes according to their partiti
cmap = cm.get_cmap('viridis', max(partition.
nx.draw_networkx_nodes(G, pos, partition.key
                       cmap=cmap, node_color
nx.draw_networkx_edges(G, pos, alpha=0.5)
plt.show()
```