社会网络中信息传播优化

马茂原, 任翌玮, 侯许凡

目录

摘要

本文分析了在社交网路中信息传播的优化问题. 利用了独立级联模型, 建立了从节点到节点的单向传播流程, 并从社交网络有向图的结构出发, 对该网络进行了聚类分析和出度分析, 并基于出度分析提出了一种贪心算法, 使在图结构比较稀疏的情况下达到多项式复杂度. 具有很好的求解效率. 并给出了该问题的一个最优解.

1 问题分析

1.1 问题重述

该问题中给出了一个社会网络网络模型的有向图 (图 1), 共包含 1377 个节点和 2279 条边. 问题中, 假设传播概率 $p = 20\%$, 总共传播次数为 8 次. 要求选择初始的 10 个节点使经过 8 次传播后能够传播的点最多.

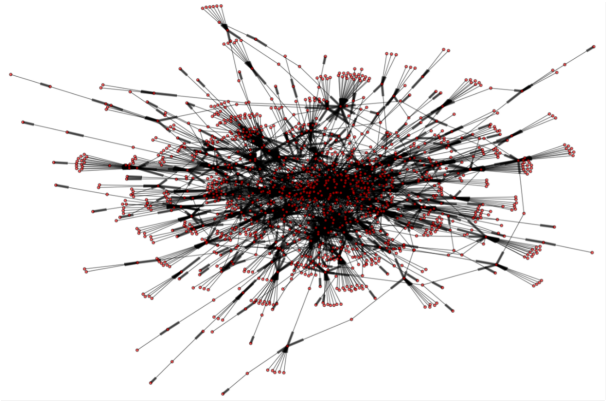


图 1: 社交网络有向图

1.2 模型与分析

1.2.1 模型

本问题采用了独立级联模型, 一个节点代表一个用户, 一条有向边代表了信息的流向. 整个信息传播流程如下:

1. 在 $t = 0$ 时刻, 信息从某些节点开始第一次传播. 这些初始节点被认为是处于激活状态, 构成初始激活集合 S_0

2. 在 $t = 1$ 时刻, 集合 S_0 中的节点 $v \in S_0$ 可以将信息以概率 p 传播给它们未被激活的邻居节点 $u \notin S_0$(邻居节点即有边与之相连的节点, 信息传播方向与边的指向一致). 如果传播成功, 即 u 被激活, 则 u 将被加入 $t = 1$ 时刻的激活集合 S_1. 集合 S_1 包含 S_0 中的所有节点, 以及在 $t = 1$ 时刻被激活的所有节点.

3. 在 $t = t_0$ 时刻, 集合 S_{t_0-1} 中的节点 $v \in S_{t_0-1}$ 可以将信息以概率 p 传播给它们未被激活的邻居节点 $u \notin S_{t_0-1}$. 如果传播成功, 即 u 被激活, 则 u 将被加入 t_0 时刻的激活集合 S_{t_0}. 集合 S_{t_0} 包含 S_{t_0-1} 中的所有节点, 以及在 t_0 时刻被激活的所有节点.

1.2.2 分析

由于本问题是多个初始点同时进行传播多次, 而各个点之间传播的相互影响不可忽视. 因此, 多个初始点选取需要考虑其传播时的相互影响, 尽可能的使他们传播时不重复激活某一点.

2 模型求解

2.1 数据处理

由于原数据中, 该图节点为 10 位节点号存储. 为求解方便以及显示方便起见, 对节点编号作处理: 分别将这些节点用 1 1377 的编号将原数据替换, 同时将对应的边数据也进行替换, 得到新的数据.

2.2 有向图分析

2.2.1 聚类分析

为更好的选取初始点, 我们对原数据进行聚类分析. 如果聚类结果较好, 可以从这些不同的类中选取初始点, 使之相互之间干扰减少, 从而使传播时更少的激活同一个节点. 然而以节点之间边的跳数作为长度衡量聚类, 在 10 类聚类下, 聚类效果并不好 (图 2). 发现由于此图中心比较集中, 使绝大多数点聚在一类中, 难以分类.

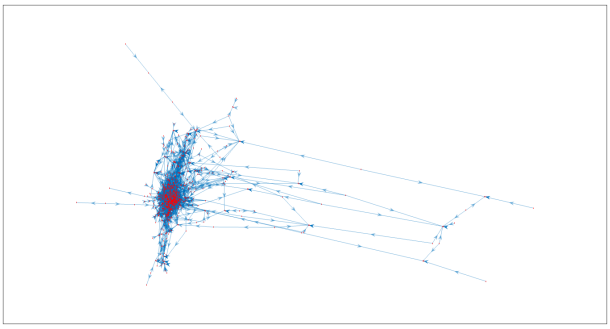


图 2: 聚类分析结果图

2.2.2 度数分析

在聚类效果不好的情况下, 基于出度更大的点在一次传播中可能传播的点更多这一想法, 我们统计了该图中所有节点的出度情况 (图 3), 发现该图大多数点的出度小于等于 2, 连接非常稀疏.

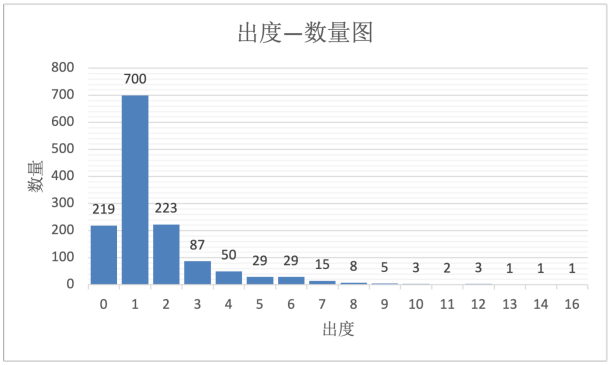


图 3: 出度分析图

2.3 求解过程

2.3.1 求解算法

通过对图的分析, 我们得知这是一个非常稀疏的图结构. 我们考虑使用选点配合模拟传播的方法进行求解. 为了减少算法的复杂度, 我们选取了 235 个度数最大的点作为初始点数组 S, 每次选取初始点只会从该集合中选取. 该算法 (**Algorithm** 1) 具体过程为:

Algorithm 1 寻找最优初始点

```
input: 有向图 G, 初始点数组 S, 点的总个数 n, 总传播次数 t, 总计选取初始点个数 m
GlobalRecord[len(S), m+1] := zeros(len(S), m+1) //初始化全局记录数组
for i = 1:len(S) do
  Record[m] := zeros(m) //初始化每次选取第一个初始点的记录数组
  active[n]:=zeros(n) //初始化激活状态数组
  active[i] = 1 //激活初始选点
  for epoch = 1:t do
    spead(active, G, S[i]) //进行一次传播, 并更新 active
  end for
  S1 := S\S[i]
  //选取其余的 t-1 个配合点, 并将之最优化
  for times = 2:m do
    for j = 1:len(S1) do
      active1 := active
      for epoch = 1:t do
        spead(active1, G, S1[j])
      end for
      if j == 1 then
        max = sum(active1) - sum(active)
        maxi = j
        MaxActive := active1
      else
        if max < sum(active1) - sum(active) then
          max = sum(active1) - sum(active)
          maxi = j
          MaxActive := active1
        end if
      end if
    end for
    active = MaxActive //合并该次选点最大激活点图的激活状态
    Record[times] = maxi
  end forGlobalRecord[i, :] = [Record, sum(active)]
end for
output: 二维数组 GlobalRecord, 其中记录第一个初始点, 和第一初始点选取后其余最优配合点
以及该第一初始点选取下的最大激活点数
```

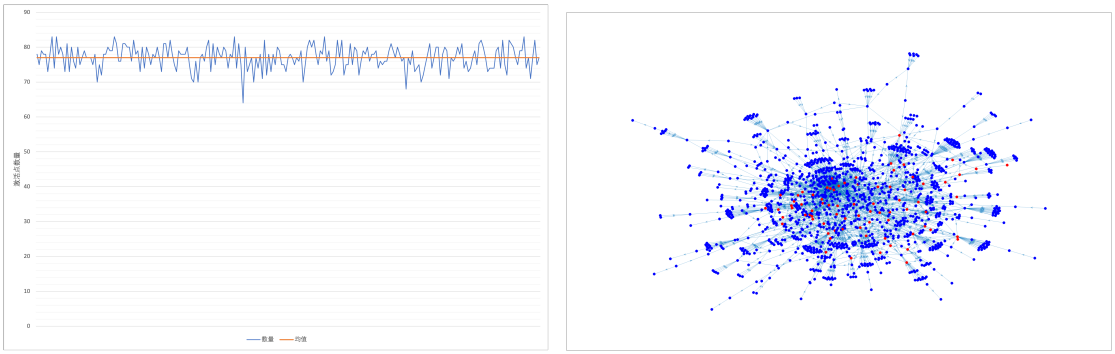
2.3.2 求解结果

利用 MATLAB 求解, 得到 235 组数据, 最大传播点数为 83, 平均传播点数为 77(取整). 在优传播组合其中一组为 (表 1)

表 1: 最优初始点组合

新编号	32	145	749	821	1246
原始编号	1667516751	1657301617	1644352745	1518610444	1650458000
新编号	1079	116	701	620	1071
原始编号	1652969205	1629926057	1676267572	1642825604	1687284412

最终结果图如图 (图 4) 所示. 图 4.(b)中, 红点代表已经激活的节点.



(a) 传播数量统计图　　(b) 最终传播点图

图 4: 最终结果图

3 算法分析

用于求解该问题的算法在每次选点时都会选取局部最优的点, 可以认为是贪心算法的一种. 假设选取初始点数组长度为 S, 总点数为 n, 总共传播次数为 t, 共选取初始点 m 个, 以每次传播为单次操作, 则该算法 (**Algorithm** 1) 算法时间复杂度为

$$o(S \times \frac{S!}{(S-m)!} \times t)$$

而若考虑到传播的操作分解, 记每个节点平均邻接节点数为 $\bar{x}$, 则总算法时间复杂度为

$$o(S \times \frac{S!}{(S-m)!} \times t \times n \times \bar{x})$$

可见, 若该有向图是稀疏图, 并且选取初始点不是太多, 可以认为该算法的时间复杂度是 $o(n)$ 级的. 而该算法的空间复杂度为该有向图的存储空间和多个所需要的辅助数组, 是 $o(n^2)$ 级的.

4 附录

MATLAB 求解

```
clc;clear;
filename = 'edges.xlsx';
M = readmatrix(filename);
x = M(:, 1);
y = M(:, 2);
n = 1377;

G = digraph(x,y);
```

```
A = adjacency(G);
number=147;
start_sum=zeros(1,n);
s=[];
outdegree = sum(A,2)';
[B,I] = maxk(outdegree,number); %选择出度最多的numbers个点,
% B为出度数量vec, I为对应点
global_match = zeros(length(I), 10+1); %二维数组, dim1是中心点,
% dim2中10是best_match的十个点, 1是激活点个数
p = 0.2;
max_epoch = 8;

for index = 1:length(I) %将I中每个点都先视为中心点,
    % 相当于len(I)次选择中心点
    start = I(index);
    global_match(index, 1) = start;
    active = zeros(1, n);
    active(start) = 1;
    for epoch = 1:max_epoch
        active = spead(active, G, p);
    end
    new_I = I;
    new_I(ismember(new_I, start)) = []; %从new_I中删去start
    best_match = zeros(1, 10);
    best_match(1) = start;
    for times = 2:10 %选取剩余9个配合点
        if times > 2
            new_I(ismember(new_I, new_I(maxi))) = [];
            %从new_I中删去maxi
        end
        diff = zeros(length(new_I));
        %用于存储第times次选点后的active点个数差值
        for i = 1:length(new_I)
            new_active = active;
            new_active(new_I(i)) = 1;
            for epoch = 1:max_epoch
                new_active = spead(new_active, G, p);
            end
            diff(i) = sum(new_active) - sum(active);
            if i == 1
                maxi = 1;
                max = diff(1);
                max_active = new_active;
            else
                if max<diff(i)
                    maxi = i;
```

```
                        max = diff(i);
                        max_active = new_active;
                    end
                end
            end
            active = max_active; %合并最大active传播图
            best_match(times) = new_I(maxi);
        end
        global_match(index, :) = [best_match, sum(active)];
    end

    global_match
    plot(G,'layout','force','NodeColor', ...
    ['active', 0*active', 1-active'], 'MarkerSize', 5);

    function active = spead(active, G, p)
        active_index = find(active);
        for u = active_index
            neighbors = successors(G,u);
            for v = neighbors
                if active(v) == 0
                    r = rand;
                    if r < p
                        active(v) = 1;
                    end
                end
            end
        end
    end
```