Influence Maximization Problems(IMP)

Peijun Ruan 11610214

School of Computer Science and Engineering
Southern University of Science and Technology
Email: 11610214@mail.sustc.edu.cn

1. Preliminaries

1.1. Introduction

Influence maximization problem was raised to study the influence among members in social network.A piece of message could quickly become pervasive through the "word-of-mouth" propagation among the familiar people in the social network.This kind of phenomenon is very powerful in many applications like adoption of political standpoints[1] .This problem can be described as find a subset of nodes **S** in a social network **G** that could maximize the expected spread of influence $\sigma(S)$, and the spread of influence $\sigma(S)$ is the nodes that can be activated by **S** after a diffusion model ends. The most commonly used diffusion models are Independent Cascade model (**IC**) and Linear Threshold Model (**LT**). This problem has already been proved to be NP-hard, so the target of IMP is to find a approximately optimal solution(a node set **S**).

1.2. Software

This project is written in Python 3.6 using IDE Pycharm.The library used including sys,getopt,random,time,numpy and copy.

1.3. Algorithm

The algorithm implemented in this project is a improved CELF algorithm which was proposed in [2].

2. Methodology

This project contains two part, the first part is to calculate the spread of influence for a given node set and a network in two models.The second is to find a optimal node set for a given network.

2.1. Representation

In both two parts, the network is a directed graph $G = (V, E)$ where V notes the node set and E notes the edge set.Each edge (u, v) in E has a weight w represent the probability that u activates v and $w \in [0, 1]$.

So there are two main things we need to represent in algorithms.

- network G: a two-dimensional numpy array,store the edge and its weight in the graph.
- seed_set S: a list storing the result node set or the original given node set.

And in the implementation of algorithm, there are one important data that needs to store.

- mgset: a set store the index of nodes,which are the nodes activated by one node.

2.2. Functions

Here shows the main function I implement in this project.

For ISE.py

- get_input: get the input from Terminal and store the given data in data structures.
- build_graph: read the given file and build the corresponding graph.
- IC_model: the implementation of independent cascade model.
- LT_model: the implementation of linear threshold model.
- get_weight: calculate the total weight for incoming edges of given node.

For IMP.py

- get_input: get the input from Terminal and store the given data in data structures.

- build_graph: read the given file and build the corresponding graph.
- IC_model: the implementation of independent cascade model.
- LT_model: the implementation of linear threshold model.
- get_weight: calculate the total weight for incoming edges of given node.
- getInfluence: using specific model according to the Terminal input.
- Lv_CELF:the main algorithm to maintain the seeds for given network.

2.3. Model Design

From the introduction we know, to maximize the $\sigma(S)$,we need to find a seed set S that can activated as many nodes as possible, and we can use LT or IC model to determine how many nodes a given seed set can activate.

So we can first find the $\sigma(S)$ of each single node in the network, and each time pick the node with biggest $\sigma(S)$ as a seed.Then we can find a relatively nice seed set.Of course, we need to optimize the time complexity of the greedy algorithm because it runs not so fast as we want it to be.Then I choose to use the improved CELF algorithm[2] to replace the normal greedy algorithm.

2.4. Detail of Algorithm

Here shows the psudocode of the main algorithms.

Algorithm 1 IC_model

Input: $network\ G, seed\ set\ S$
Output: $activated\ set\ A$

$V_{new} = S$
$V_{ans} = \phi$
while $V_{new} \neq \phi$ **do**
    $V_{temp} = \phi$
    for each node i in V_{new} **do**
        for each not active neighbor j of i **do**
            try to activate j
            if j is activated **then**
                $V_{temp} = V_{temp} \cup j$
            end if
        end for
    end for
    $V_{new} = V_{temp}$
    $V_{ans} = V_{ans} \cup V_{temp}$
end while
return V_{ans}

Algorithm 2 LT_model

Input: $network\ G, seed\ set\ S$
Output: $activated\ set\ A$

$V_{new} = S$
$V_{ans} = \phi$
give a random threshold for all node in G
while $V_{new} \neq \phi$ **do**
    $V_{temp} = \phi$
    for each node i in V_{new} **do**
        for each not active neighbor j of i **do**
            calculate weight of j,try to activate j
            if j is activated **then**
                $V_{temp} = V_{temp} \cup j$
            end if
        end for
    end for
    $V_{new} = V_{temp}$
    $V_{ans} = V_{ans} \cup V_{temp}$
end while
return V_{ans}

Algorithm 3 Lv_CELF

Input: $network\ G, seed\ size\ K$
Output: $seed\ set\ S$

$V_{ans} = \phi$
$Q = \phi$
for each node u in G **do**
    u.mgset=getInfluence(G,u)
    u.mg=|u.mgset|
    u.flag=0
    add u to Q
end for
sort Q by mg in descending order.
while $|V_{ans}|$¡K and $|Q|$¿0 **do**
    u=Q[top]
    if u.flag == $|V_{ans}|$ **then**
        $V_{ans} = V_{ans} \cup u$
        Q=Q-u.mgset
    else
        u.mgset=getInfluence(G,V_{ans}+u)-getInfluence(G,V_{ans})
        u.mg=$|u.mgset|$
        u.flag=$|V_{ans}|$
        resort Q by mg in descending order.
    end if
end whilereturn V_{ans}

3. Empirical Verification

Since we have only one test network example, there is little I can do to for the empirical verification. Fortunately the result of test is obvious and easy to explain.

3.1. Design

To show the usability and improvement of this algorithm, I implement a general greedy algorithm for IMP to compare the result and running time.I choose two seed sizes and use both two model to get the result.

3.2. Data

Data used in this test is the small network given on the sakai which contains 62 nodes and 159 edges to compare the efficiency of two algorithm.And I did not use other data set from internet.

3.3. Test Environment

CPU:i5-7300HQ 2.50GHz
RAM:8.0GB
Number of Process: only one process

3.4. Hyperparameters

In ISE:
IC/LT model calculation times:10000

In IMP:
Total loop for result:500
IC/LT model calculation time for **mgset** of each node:1

3.5. Result

3.5.1. model=LT seed_size=4.

Lv_CELF:
seeds = 58 52 28 62
time cost = 0.0259s
spread of influence: IC 25.5052 LT 29.2557

normal greedy:
seeds = 56 8 58 32
time cost = 0.6244s
spread of influence: IC 21.0879 LT 23.4523

3.5.2. model=IC seed_size=4.

Lv_CELF:
seeds = 56 58 48 53
time cost = 0.0197s
spread of influence: IC 26.9789 LT 32.7696

normal greedy:
seeds = 56 57 2 54
time cost = 0.2575s
spread of influence: IC 16.0074 LT 17.5406

3.5.3. model=LT seed_size=8.

Lv_CELF:
seeds = 41 47 48 52 53 56 58 28
time cost = 0.0405s
spread of influence: IC 32.5524 LT 39.7248

normal greedy:
seeds = 4 60 48 52 23 58 28 30
time cost = 1.5897s
spread of influence: IC 30.047 LT 36.9852

3.5.4. model=IC seed_size=8.

Lv_CELF:
seeds = 32 45 28 48 52 56 58 60
time cost = 0.0337s
spread of influence: IC 31.7086 LT 38.9769

normal greedy:
seeds = 40 60 50 20 52 54 58 28
time cost = 0.9692s
spread of influence: IC 29.6898 LT 35.4652

3.6. Analysis

From the above result we can easily figure out that the improved CELF algorithm is much faster and more reliable than normal greedy algorithm. The time cost improvement even reaches 96.5%! Besides, the quality of Lv_CELF is also nice.

And we can find that as the seed size grow, the improvement of time cost gets more obvious,which shows the improved CELF algorithm can deal more complex and larger scale data set than normal greedy algorithm.

References

[1] Li Y, Fan J, Wang Y, et al. Influence Maximization on Social Graphs: A Survey[J]. IEEE Transactions on Knowledge & Data Engineering, 2018, PP(99):1-1.

[2] Lv, J., Guo, J., Yang, Z., Zhang, W., & Jocshi, A. (2014). Improved Algorithms OF CELF and CELF++ for Influence Maximization. Journal Of Engineering Science And Technology Review, 7(3), 32-38. doi: 10.25103/jestr.073.05

[3] Kempe, D. , Kleinberg, J. , & Tardos, va. (2003). [acm press the ninth acm sigkdd international conference - washington, d.c. (2003.08.24-2003.08.27)] proceedings of the ninth acm sigkdd international conference on knowledge discovery and data mining, - kdd ̈03 - maximizing the spread of influence through a social network. 137.

[4] Goyal, A., Lu, W., & Lakshmanan, L. V. (2011, March). Celf++: optimizing the greedy algorithm for influence maximization in social networks. In Proceedings of the 20th international conference companion on World wide web (pp. 47-48). ACM.