Project of IMP Implementation

Yuxing Hu 11510225

1 Preliminaries

1.1 Introduction

Influence maximization is the problem of finding a small subset of nodes (seed nodes) in a social network that could maximize the spread of influence.The influence spread is the expected number of nodes that are influenced by the nodes in the seed set in a cascade manner. And I will deeply study this problem into two parts. The first part is about to find a influence value on specific seeds into a network graph, the second part is to find the best spread seed in given network graph.

1.2 Software and Library

This project is written in Python using editor Sublime Text with python interpreter 2.7.13. The libraries being used includes getopt, random, sys, time, numpy, etc.

1.3 Algorithm

The Algorithm being used in this project including climbing greedy algorithm, the CELF method and a better version I updated by the original one, and degree discounting algorithm.

2 Methodology

2.1 Problem

The problem consisting two parts, in the first part we are given a social network is modeled as a directed graph $G = (V, E)$ with nodes in V modeling the individual in the network and each edge $u, v \in E$ is

associated with a weight w u, $v \in [0,1]$ which indicates the probability that u influences v, and we need to find a seed set S that maximizes $\sigma(S)$, subject to $S = k$.

The second problem, we are given a specific number of seeds and try to find the best combination of them to have a best spread, using the model we created in the first part of the project.

2.2 Model and Algorithm

There are two basic diffusion models which are used for evaluation of seeds selected according to our IMP algorithms: Independent Cascade(IC) and Linear Threshold(LT)
Three algorithms are implemented in this source codes, including General Greedy, original CELF, improved CELF, and DegreeDiscount algorithm.

2.3 Important implements

Network Reader: this method is used to read the network file which including the number of edges and points.

Seed reader: this method is used to read the seed file which including the original seed in the first part of project.

ic_model: this method implements how the Independent Cascade works, the pseudo code is as Algorithm 1 shows.

lt_model: this method is an implement of Linear Threshold function, the pseudo code us as Algorithm 2 shows.

Graph: this class construct a graph, including the inner direction lines and outer direction lines. Also store the weight of all the connected direct edges.

hill_greedy: this method is a basic implementation of greedy algorithm, which is too slow an inefficient, which is not being used anymore.

Algorithm 1 Improved Estimator for IC Model

input: V_{seeds} := set of seeds
initialize $S = \emptyset$ and $R = 10000$
for $i = 1$ to R **do**
  $V_{next} := V_{seeds}$
  $V_{new} := \emptyset, V_{activated} := \emptyset$
  $cnt := 0$
  while $V_{next} \neq \emptyset$ **do**
    for each vertex $v \in V_{next}$ **do**
      for each vertex u with $\overrightarrow{vu}$ in E **do**
        try to activate u with probability $p_{\overrightarrow{vu}}$
        if u is activated **then**
          $V_{new} := V_{new} \cup \{u\}$
        end if
      end for
    end for
    $V_{activated} := V_{activated} \cup V_{new}, V_{next} := V_{new}$
  end while
  $cnt := cnt + |V_{activated}|$
end for
output: cnt/R

Algorithm 2 Improved Estimator for LT Model

input: V_{seeds} := set of seeds
initialize $S = \emptyset$ and $R = 10000$
for $i = 1$ to R **do**
  give a random threshold for all $v \in V$
  $V_{new} := V_{seeds}$
  $V_{activated} := \emptyset$
  $cnt := 0$
  while $V_{new} \neq \emptyset$ **do**
    $V_{next} := \{u | u$ is neighbor of vertex in V_{new} with $\overrightarrow{vu}$ in $E\}$
    for each vertex $v \in V_{next}$ **do**
      calcuate the sum of p and try to activate v
      if v is activated **then**
        $V_{new} := V_{new} \cup \{v\}$
      end if
    end for
    $V_{activated} := V_{activated} \cup V_{new}, V_{next} := V_{new}$
  end while
  $cnt := cnt + |V_{activated}|$
end for
output: cnt/R

spread_check_dig: a checking method which allow user to run the ISE program for 10000 times.

celf_improved: this method all user to run the celf method, the way to do that can be found under the celf algorithm, I made a couple of improvement including adding strategy of not able to find the second best point, in this case the method will create an numpy array to store

Algorithm 3 CELFGreedy(G, k)

initialize $S = \emptyset$
H = maxHeap of u w.r.t $\Delta_u(S)$ and $u \in V \setminus S$
for $u \in V$ **do**
    $\Delta_u(S)$ = evaluateSpread($S \cup \{u\}$)
    Update H
end for
$max = -\infty$
while $|S| < k$ **do**
    u = pop a vertex from H
    if $\Delta_u(S) > max$ **then**
        $\Delta_u(S)$ = evaluateSpread($S \cup \{u\}$)
        if $\Delta_u(S) > max$ **then**
            $max = \Delta_u(S)$
        end if
        push u into the heap H
    else
        $S = S \cup \{\text{u}\}$
        $max = -\infty$
    end if
end while
output: S

the best possible value to go. The pseudo code is in graph algorithm 3.

degree_dis: this method is used to calculate the acceptable point set to minus some point into the original one. The description of this method which also called Degree Discount algorithm can be found below.

Algorithm 4 DegreeDiscountIC(G, k)

initialize $S = \emptyset$
for each vertex v **do**
  compute its degree d_v
  $dd_v = d_v$
  initialize t_v to 0
end for
for $i = 1$ to k **do**
  select $u = argmax_v\{dd_v | v \in V \setminus S\}$
  $S = S \cup \{u\}$
  for each neighbor v of u and $v \in V \setminus S$ **do**
    $t_v = t_v + 1$
    $dd_v = d_v - 2t_v - (d_v - t_v)t_v p$
  end for
end for
output: S

3. Empirical verification

The empirical verification is not easy because we are only given limited test example to test and to find the best path to go. However, on the other hand the test become easy and obvious to find because of all that matters to the solution is the cost that we used.

3.1 Design

The IMP problem is a NP hard problem and under the IC model and LT model its computation is #P hard. Because we are only given a little network data so the design of the testing plan become much more harder. We should consider the difference of model we use and the key factor should be the model, strategy, size of data.

3.2 Data and structure

Data used in this project is a small network graph which has 62 nodes and 159 edges. Data structure we use in this project including dictionary, list, set and NumPy array.

3.3 Performance

Because our handed data is so small so I'm not able to organize a time based testing. However, it's possible to test the running time on a mathematical way. The complexity of vital algorithm are shown in the chart below.

GeneralGreedy	O(knRm)
CELF	O(knR)
DegreeDiscount	O(klogn + m)

3.4 Result

3.4.1 Model IC, seed size = 4

lifesavermac:AI_IMP lifesaver$ python IMP.py -i network.txt -k 4 -m IC -b 0 -t 60 -r 3

56 58 50 53

Time cost: 4.10196900368

In IC model spread: 27.1473

In LF model: 31.2845

3.4.2 Model LT, seed size = 4

lifesavermac:AI_IMP lifesaver$ python IMP.py -i network.txt -k 4 -m LT -b 0 -t 60 -r 3

56 48 58 53

Time cost: 11.3791959286

In IC model spread: 27.0017

In LF model spread: 32.7513

3.4.3 Model IC, seed size = 8

lifesavermac:AI_IMP lifesaver$ python IMP.py -i network.txt -k 8 -m IC -b 0 -t 60 -r 3

41 48 50 53 56 58 28 62

Time cost: 11.2946920395

In IC model spread: 36.8522

In LF model spread: 46.033

3.4.4 Model LT, seed size = 8

lifesavermac:AI_IMP lifesaver$ python IMP.py -i network.txt -k 8 -m LT -b 0 -t 60 -r 3

60 48 50 53 56 58 28 62

Time cost: 27.2931079865

In IC model spread: 38.85

In LF model spread: 49.5593

3.5 Analysis

As we can see from the results, the CELF improved algorithm shows a better performance than the original greedy algorithm which is totally timely unacceptable. The degree discount algorithm use a little bit time and the majority of time is being used by calculating the spread value in the graph caused by calculating the spread influence.

However, if we decline the number of point that after being discount the result will be much more unacceptable. The further improvement should be on this point, on make a wiser choice of choosing the point.

References

[1]Chen, W., Wang, C., & Wang, Y. (2010, July). Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 1029-1038). ACM.

[2]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.

[3]Chen, W., Collins, A., Cummings, R., Ke, T., Liu, Z., Rincon, D., ... & Yuan, Y. (2011, April). Influence maximization in social networks when negative opinions may emerge and propagate. In *Proceedings of the 2011 SIAM International Conference on Data Mining* (pp. 379-390). Society for Industrial and Applied Mathematics.

[4]Tang, Y., Xiao, X., & Shi, Y. (2014, June). Influence maximization: Near-optimal time complexity meets practical efficiency. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data* (pp. 75-86). ACM.