Scalable Influence Estimation in Continuous-Time Diffusion Networks

We are surrounded by social and information share networks, over which diffusions of information, events, virus, takes place constantly. We can often observe that after some influential users adopt certain new product or idea, they actively influence the behaviors of their friends, which in turn makes more friends of friends adopt the product through word-of-mouth.

The specific questions we seek to address in this **NIPS 2013 paper** is to accurately estimate the number follow-ups which can be triggered by the influential users, and then how we can identify such influential users, to whom we will give promotions, in order to trigger the largest expected number of follow-ups **as soon as possible** ? These questions are interesting for that advertisers who want to have an efficient and effective campaign for their new products. But, they are also very challenging in that these questions are **time sensitive** . Intuitively, 1000 buyers in a day will be much better than 1000 buyers in a year. Furthermore, the solution must be scalable enough to deal with large real-world networks.

About ConTinEst

ConTinEst is a scalable randomized algorithm for **influence estimation** in **continuous-time** diffusion networks. It can estimate the influence of every node in a network with $|\mathcal{V}|$ nodes and $|\mathcal{E}|$ edges to an accuracy of $n = \epsilon$ using $O(1/\epsilon^2)$ randomizations and $\tilde{O}(n|\mathcal{E}| + n|\mathcal{V}|)$ computations. When used as a subroutine in a greedy influence maximization approach, ConTinEst is guaranteed to find a set of C nodes with the influence of at least $(1 - 1/e)OPT - 2C\epsilon$, where OPT is the optimum value. Experiments on both synthetic and real-world data show that ConTinEst can easily scale up to networks of millions of nodes while significantly improves over previous state-of-the-arts in terms of the accuracy of the estimated influence and the quality of the selected nodes in maximizing the influence.

Papers

Scalable Influence Estimation in Continuous-Time Diffusion Networks (**Outstanding Paper Award** , full oral presentation). Nan Du, Le Song, Manuel Gomez Rodriguez, and Hongyuan Zha. Neural Information Processing Systems (**NIPS**). Dec. 5 - Dec. 10, 2013, Lake Tahoe, Nevada, USA. [PAPER] [SLIDE][POSTER][Bibtex]

Download

Terms of agreement. We appreciate your interest in our research. The C++ implementation is provided and maintained for our research projects conducted in the School of Computational Science and Engineering at Georgia Institute of Technology. All rights of the source code implementations are reserved. Comments, bugs, and suggestions, if any, can be directed to **Nan Du** (dunan AT gatech.edu). Redistributions and use of them in source or binary forms, with or without modification, are permitted only to academic purposes. If you use this code, or any part of it, please cite our paper as

@inproceedings{DuSonGomZha13, title={Scalable Influence Estimation in Continuous-Time Diffusion Networks}, author={Nan Du and Le Song and Manuel Gomez-Rodriguez and Hongyuan Zha}, booktitle={Advances in Neural Information Processing Systems (NIPS)}, year={2013} }

Download ConTinEst !

Documentation

Input network format. We include a sample network (1,024 nodes) in the implementation. The input file "std_weibull_DAG_core-1024-1-network.txt" consists of two blocks separated by a blank line. The node ID starts from 0 to 1,023. In the 1st block, each line has the format

node ID, node Name

In the 2nd block, each line represents a directed edge from node ID1 to node ID2 with the following format.

node ID1, node ID2, shape, scale

Because we assume the most general form for the pairwise transmission function, for the simplicity of the demo code, we take the Weibull Distribution as the transmission function with the two parameters **shape** and **scale**, respectively. When $shape = 1.0$, it is reduced to the exponential distribution; When $shape = 2.0$, it is equivalent to the Rayleigh Distribution. A valid network format can be the following.

0,0
1,1
2,2

0,1,0.0959956,0.0888947
0,2,3.95348,9.42268
1,2,1.61835,8.38794

The implementation consists of the following five classes

- **Graph** encapsulates basic graph operations based on the adjacent-list representation.
- **ConTinEst** encapsulates the implementation for influence estimation and maximization.
- **Simulation** encapsulates the implementation for naive sampling method.
- **SimpleRNG** implements several basic density functions.

- **TestModules** implements basic unit tests.

Class Graph

Graph has the following properties.

- **N** : number of nodes.
- **nodes** : array of nodes, each of which stores a set of parents and children, respectively.
- **edge_weight** : stores the sampled transmission time for each edge.
- **edge_parameter** : stores the parameters of the transmission function for each edge.
- **RNG** : generates samples from the Weibull distribution.
- **filename** : the input graph file name.

Graph has the following methods.

Graph(string g_filename, unsigned numNodes)

The constructor function where

- **g_filename** : the name of the input graph file.
- **numNodes** : number of nodes in the graph.

void LoadWeibullFormatNetwork(string splitter, bool reverse);

Read network structure from the input graph file.

- **splitter** : the delimiter to separate each line. Default is comma.
- **reverse** : indicate whether the directed edge is revered or not.

void SampleEdgeWeightWbl();

Draw samples from the pairwise transmission functions for each edge.

void PrintWblNetwork();

An utility function to print the loaded network.

pair<unsigned, unsigned> MaximumOutDegree();

An utility function to find the node with the maximum out-degree. Return a pair of (node ID, out-degree)

Class ConTinEst

ConTinEst has the following properties.

- **m_num_samples** : number of sampled sets of pairwise transmission times.
- **m_num_rankings** : number of sampled sets of random labels per node.
- **m_TableList** : stores the least-label list for each node given a particular set of transmission times and a specific set of random labels.

- **m_keys** : stores the random labels for each node given a particular set of transmission times for each edge.
- **m_RNG** : generates samples from the Weibull distribution.
- ***m_G_inverse** : pointer to a preloaded graph object.

ConTinEst has the following methods.

ConTinEst(Graph *G, unsigned num_samples, unsigned num_rankings);

The constructor function where

- ***G** : pointer to an initiated graph object.
- **num_samples** : number of sampled sets of pairwise transmission times, Default 10000.
- **num_rankings** : number of sampled sets of random labels per node, Default 5.

void LeastElementListsSet(float *d, const pair<float, unsigned> *key_node_pairs, vector<pair<float, unsigned> > *lists);

Generate least-label lists for each node given a sampled set of transmission times, and a sampled set of random labels.

- ***d** : a set of random labels
- **key_node_pairs** : a mapping from label value to node ID.
- ***lists** : store least-label lists for each node.

void GetLeastElementLists();

Generate all least-label lists for each node based on the required number of sampled sets of pairwise transmission times (m_num_samples), and the number of sampled sets of labels (m_num_rankings).

float EstimateNeighborhood(const set<unsigned>& sources, float T);

Estimate the influence of a given set of sources by time T.

- **sources** : a given set of source nodes.
- **T** : the observation window T, Default 10.

set<unsigned> LZGreedy(double T, unsigned K);

Maximize the influence using the greedy algorithm.

- **T** : the observation window T, Default 10.
- **K** : the size of the source set.

vector<set<unsigned> > Optimize(const vector<double>& setT, const vector<unsigned>& setK);

Return the set of selected sources with different sizes at different time.

- **setT** : a sequence of time window.
- **setK** : a sequence of numbers of the selected sources.

Compile on Linux/MacOS

To compile the sources, simply type the following to get the executable file with the name *execute*

tar -zxvf ConTinEst.tgz

cd ConTinEst

make

Run the demo on Linux/MacOS

To run the demo code within the directory *ConTinEst*, simply type

./execute

The demo code first loads the network file and calculate the least-label lists for each node.

std_weibull_DAG_core-1024-1-network.txt
Get all least-label lists : 10000 sets of transmission times; 5 sets of random labels; done

Then, it finds the node with the maximum out-degree.

node 0 has the largest out-degree 24

Next, it compares the estimated influence at different time given by ConTinEst with that given by the naive simulation for the node with the largest out-degree.

Estimate Influence T = 1 done
Simulation Check T = 1 with 10000 samples
ConTinEst : 7.70149
Simulation : 7.6682

Estimate Influence T = 2 done
Simulation Check T = 2 with 10000 samples
ConTinEst : 14.0139
Simulation : 13.8913

Estimate Influence T = 3 done
Simulation Check T = 3 with 10000 samples
ConTinEst : 24.2149
Simulation : 24.1826

Estimate Influence T = 4 done
Simulation Check T = 4 with 10000 samples
ConTinEst : 38.9468
Simulation : 39.0016

Estimate Influence T = 5 done
Simulation Check T = 5 with 10000 samples
ConTinEst : 57.3021

Simulation : 57.9743

Estimate Influence T = 6 done
Simulation Check T = 6 with 10000 samples
ConTinEst : 79.7948
Simulation : 79.656

Estimate Influence T = 7 done
Simulation Check T = 7 with 10000 samples
ConTinEst : 104.704
Simulation : 104.389

Estimate Influence T = 8 done
Simulation Check T = 8 with 10000 samples
ConTinEst : 133.896
Simulation : 133.148

Estimate Influence T = 9 done
Simulation Check T = 9 with 10000 samples
ConTinEst : 163.586
Simulation : 163.121

Estimate Influence T = 10 done
Simulation Check T = 10 with 10000 samples
ConTinEst : 192.411
Simulation : 192.212

Finally, it selects the 10 sources that can achieve the largest expected number of infected nodes by time 10.

Influence Maximization by selecting 10 nodes with T = 10 done
selected sources : 0;1;2;4;10;16;17;524;890;929;