CS 490 - Diffusion Source Identification in Networks

Daniel Pearson

Abstract

When an outbreak of a disease occurs, it is often necessary to identify the source of the outbreak. This problem can be modeled on a graph by assuming the disease started at one node and spread over time to other nodes. After observing some number of the infected nodes, the goal is to predict the source node. This problem is also of use in identifying sources of malicious information on the internet and in identifying the origin of viruses. The ultimate goal of the research we were working on is to figure out the optimal way to choose which infected nodes to observe. In order to gain insight into how to solve this problem, we looked at strategies for solving the problem given a set of observed infected nodes. One solution to this problem was proposed in Back to the Past: Source Identification in Diffusion Networks from Partially Observed Cascades (Farajtabar et al., 2015). This paper proposes a two-stage process where first network parameters are learned by observing independent cascades on the underlying. Next these parameters are used to find the most likely source node given the current observed infected nodes. For this project, I explored the model used to solve this problem. Using the algorithm presented in the paper above, I created a simulation that imitated stage two of the process where the source node is identified given previously estimated network parameters. This is a complex estimation and optimization problem and involved techniques such as Monte Carlo integration, importance sampling, and solving multiple non-convex optimization problems.

1 Introduction

The problem of interest is how to identify the source of diffusion given incomplete knowledge about how the information has spread. Our research is working toward strategies for placing k observers in the network to enable source identification. In the process, we have read papers about algo-
rithms for identifying the source of diffusion given the k observers are already placed. One of the most important papers we read was called “Back to the Past: Source Identification in Diffusion Networks from Partially Observed Cascades”.

This paper uses a continuous time model. We assume some source node begins the infection. Each infected node spreads to the next node independently over each edge where the random transmission time, $\tau$, is distributed according to the function $f_{ji}(\tau; \alpha_{ji})$. The first neighbor that transmits the infection to a node is the true parent of that node. We assume that we have observed some set of nodes, $\mathcal{O}$, and another set of nodes, $\mathcal{H}$, that are unobserved. We assume the source node $s$ is in the set $\mathcal{H}$. A cascade, $t := (t_1, ..., t_n)$, is an n-dimensional vector where $n$ is the number of total nodes and each $t_i$ is the time that node $i$ became infected. Our goal is to identify $s$ given $\{t_j\}_{j \in \mathcal{O}}$.

2 Problem Solution

This section describes the algorithm developed in the paper Back to the Past: Source Identification in Diffusion Networks from Partially Observed Cascades. In a previous paper, the following equations were derived for the likelihood of a complete cascade:

$$p(t|s) = \prod_{i \in \mathcal{O} \cup \mathcal{H}} p(t_i|\{t_j\}_{j \in \pi_i})$$  \hspace{1cm} (1)

where $p(t_i|\{t_j\}_{j \in \pi_i}) = \prod_{j \in \pi_i, S(t_i - t_j; \alpha_{ji}) \sum_{l \in \pi_i} H(t_i - t_l; \alpha_{ji})}$ and $H_{ji}(\tau; \alpha_{ji}) = \frac{1}{\alpha_{ji}}$ and $S_{ji}(\tau; \alpha_{ji}) = e^{-\alpha_{ji} \tau}$.

This requires the entire cascade to compute, but will ultimately be very helpful in solving our problem. The actual problem we want to solve is the multi dimensional integral below.

$$p(\{t_i\}_{i \in \mathcal{O}}|s) = \int p(\{t_i\}_{i \in \mathcal{O} \cup \mathcal{H}}|s) \prod_{i \in \mathcal{H}} dt_i$$  \hspace{1cm} (2)

By doing a complex form of importance sampling that introduces an auxiliary distribution and with a clever choice of proposal distribution and auxiliary distribution, we can show that

$$p(\{t_i\}_{i \in \mathcal{O}}|s) \approx \frac{1}{L} \sum_{l=1}^{L} \prod_{i \in \mathcal{O}} p(t_i|\{t_j\}_{j \in \pi_i \setminus \mathcal{O}}, \{t_j\}_{j \in \pi_i \cap \mathcal{O}}) \prod_{i \in \mathcal{M}} \frac{p(t_i|\{t_j\}_{j \in \pi_i \setminus \mathcal{O}}, \{t_j\}_{j \in \pi_i \cap \mathcal{O}})}{\hat{p}(t_i|\{t_j\}_{j \in \pi_i \setminus \mathcal{O}}, \{\eta_j\}_{j \in \pi_i \cap \mathcal{O}})} = \phi_L(s)$$
where M is the set of hidden nodes with at least one observed parent. The final goal is then to use this to find

$$\arg\max_{s \in H} \max_{t_s} \phi_L(t_s)$$

3 My Implementation

For my implementation, I used a small network with 64 nodes. I assumed the node with index 50 is the actual source node. I also assume that we have already run some type of parameter estimation algorithm so $\alpha_{ji}$ is known for each edge in the graph. The values of $\alpha_{ji}$ are chosen at the start uniformly from 5-10 and we use the exponential distribution for $f_{ji}(\tau; \alpha_{ji})$ where $\alpha_{ji}$ is the mean of the exponential distribution. My implementation then chooses 7 nodes at random that are not the actual source node to be the observed nodes.

It then simulates an actual cascade. This is done by sampling values of $\tau$ independently for each edge and then using the shortest path property to compute $t_i$ for each node $i$. The observed node times are the ones that will be used in the future to determine the source node. Next, my simulation samples L cascades for each of the 64 nodes in the graph. It uses these values in the appropriate equations to approximate $\phi_L(t_s)$. Next all of the changepoints are enumerated. Using the changepoints calculated, we then recalculate this value at each of the changepoints. Instead of optimising within each continuous segment, the algorithm settles with the endpoint that gives the maximum $\phi_L(t_s)$. This means our solution is not exact, but this helps speed up the algorithm and many times, the endpoints are close enough that this doesn’t make a significant difference.

When you run the algorithm on the graph, the output you should look for is the matrix totalresults. totalresults(i) shows how likely node i is to be the source node. The index that gives the max value of totalresults is the best guess at the source node. Since 50 is the source node, we hope that totalresults(50) is given one of the higher values.

4 Performance

This turned out to be a very difficult algorithm to implement. Many of the difficulties are related to the run time of the algorithm. The original paper suggested an L of 150, but this would have taken over an hour to run on my computer with my current run times. I shrunk L to 30 where I can run
1 independent cascade in around 7 minutes. I also had to keep the number of independent cascades low so that the runtime would be reasonable. The default is at 1. Some other issues with run time were figuring out how to recalculate $\phi_L(t_s)$ at each changepoint and then again at points within the interval. The paper gives a description of how this can be done efficiently, but it is not clear to me if I implemented this faster method or if I misunderstood and just implemented it the slow way. Perhaps there would be additional speedups if I knew the proper technique to do this. Another issue was the large number of 0 probability events. Each of the L terms summed to equal $\phi_L(t_s)$ which is a product of many other terms. If any of those terms are 0, then the whole product is 0, so it is very easy to end up with 0s heavily influencing results especially when L is lower than 10. This can be avoided by increasing L or evaluating our function at more points but both of these will further slow down the algorithm.

Another issue is that it is very difficult to tell if the algorithm is working properly because it has such a low success rate. I tested it by seeing how often the actual source node appears in the top 10 most likely nodes which happens much more likely than chance. I also checked to see if adding more independent cascades increases the likelihood of getting the correct source node guess which it does. These two pieces of evidence indicate that, at a minimum, it seems like the algorithm converges to the correct results.

For future work, I would look to figuring out how to best optimize the calculation of $\phi_L(t_s)$. The current calculation is barely better than brute force, but I think there is a significant amount of repetition in the calculation that could be removed.

5 References