CS 6604: Data Mining Large Networks and Time-series

B. Aditya Prakash
Lecture #11: Finding Sources in Epidemics
Virus Propagation

- Susceptible-Infected (SI) Model

Diseases over contact networks

CDC data: Visualization of the first 35 tuberculosis (TB) patients and their 1039 contacts

[AJPH 2007]
Culprits: Problem definition

2-d grid

Q: Who started it?
Culprits: Problem definition

2-d grid

Q: Who started it?
Who are the good effectors [Lappas+, 2010]

- **Input:** a single snapshot of the network and the activation state of nodes

- How do you evaluate a good set of initiators?

Input data

Possible final state

\[ S = \{A, B\} \]

Cost(S) = 2
Various Problem Definitions

- Cost(S) can be the **difference** between the observed activations and the expected activations given S [Lappas+, 2010]

**Problem 1** (k-Effectors problem). Given a social network graph \( G = (V, E, p) \) and an activation vector \( a \), find a set \( X \) of active nodes (effectors), of cardinality at most \( k \) such that

\[
C(X) = \sum_{v \in V} |a(v) - \alpha(v, X)|
\]

(2)

is minimized.
The **k-Effector** problem in arbitrary graphs is **NP-complete**.

The **k-Effector** problem in arbitrary graphs is **NP-hard to approximate**.

The **k-Effector** problem can be **solved optimally in polynomial time on trees**.
Alternative formulation [Shah and Zaman 2010]

- Best source is the one which maximizes data likelihood (assumed the SI model)

\[ \hat{v} = \text{arg max}_{v \in G_N} P(G_N | v^* = v) \]

The infected subgraph
\[ \hat{v} = \arg \max_{v \in G_N} P(G_N | v^* = v) \]

Propagation Ripples

Original Graph

Infected Snapshot

Ripple R1

Ripple R2
Rumor Centrality

\[ \hat{v} = \arg \max_{v \in G_N} P(G_N | v^* = v) \]

- so

\[ P(G_N | v^* = v) = \sum_{\text{all ripples}} R_i \]

Very hard to compute.
#P-hard in general
Rumor Centrality

- But can be computed efficiently for k-regular trees

Ripple == valid Permutation
Rumor Centrality

How to find and compute the probability of each of efficiently?

Ripple == valid Permutation

1 2 4 5 3 6 7 : VALID

1 4 2 5 3 6 7 : NOT VALID
In k-regular trees

Each permutation is equally likely! (due to memoryless-ness)

\[ P(\sigma|v^* = v) = \frac{1}{k} \cdot \frac{1}{k + (k - 2)} \cdots \frac{1}{k + (N - 2)(k - 2)} \]

\[ \hat{v} = \arg \max_{v \in G_N} P(G_N|v^* = v) = \arg \max_{v \in G_N} R(v, G_N) \]
Number of possible trees

\[ R(v, G_N) = N! \prod_{u \in G_N} \frac{1}{T_u} \]
But

\[ R(v, G_N) = N! \prod_{u \in G_N} \frac{1}{T_u^v} \]

\[ T_u^v = N - T_u^v \]

Number of nodes in the subtree rooted at node \( u \) with \( v \) as the source

Number of nodes in the subtree rooted at node \( v \) with \( u \) as the source
**A message passing algorithm**

**Algorithm 1** Rumor Centrality Message-Passing Algorithm

1: Choose a root node \( v \in G_N \)
2: for \( u \) in \( G_N \) do
3: if \( u \) is a leaf then
4: \( t_{u \rightarrow \text{parent}(u)}^{up} = 1 \)
5: \( p_{u \rightarrow \text{parent}(u)}^{up} = 1 \)
6: else
7: if \( u \) is source \( v \) then
8: \[ r_{v \rightarrow \text{child}(v)}^{down} = \frac{N!}{N} \prod_{j \in \text{children}(v)} p_{j \rightarrow v}^{up} \]
9: else
10: \( t_{u \rightarrow \text{parent}(u)}^{up} = \sum_{j \in \text{children}(u)} t_{j \rightarrow u}^{up} + 1 \)
11: \( p_{u \rightarrow \text{parent}(u)}^{up} = t_{u \rightarrow \text{parent}(u)}^{up} \prod_{j \in \text{children}(u)} p_{j \rightarrow u}^{up} \)
12: \[ r_{u \rightarrow \text{child}(u)}^{down} = r_{\text{parent}(u) \rightarrow u}^{down} \frac{t_{u \rightarrow \text{parent}(u)}^{up}}{N - t_{u \rightarrow \text{parent}(u)}^{up}} \]
13: end if
14: end if
15: end for
How to extend for general graphs

- Unclear

- May extract a tree, say MST, BFS and then use this
Culprits: Problem definition

2-d grid

Q: Who started it?
Formulation

- How to find both the number and identity?
Culprits: Exoneration

(a) A chain
Culprits: Exoneration

(a) A chain

(b) A chain-star
Who are the culprits

- Two-part solution
  - use MDL for *number* of seeds
  - for a given number:
    - exoneration = centrality + penalty

- Running time = $O(k^*(E_I + E_F + V_I))$
  - linear! (in edges and nodes)

NetSleuth
Modeling using MDL

- Minimum Description Length Principle == Induction by compression
- Related to Bayesian approaches
- MDL = Model + Data
- Model
  - Scoring the seed-set

\[ \mathcal{L}(S) = \mathcal{L}_N(|S|) + \log \left( \begin{pmatrix} N \\ |S| \end{pmatrix} \right) \]

- Encoding integer \(|S|\)
- Number of possible \(|S|\)-sized sets
Modeling using MDL

- Data: Propagation Ripples

![Original Graph](image1)

![Infected Snapshot](image2)

![Ripple R1](image3)

![Ripple R2](image4)
Modeling using MDL

- **Ripple cost**

\[
\mathcal{L}(R \mid S) = \mathcal{L}_N(T) + \sum_{t}^{T} \mathcal{L}(\mathcal{F}^t)
\]

- **Total MDL cost**

\[
\mathcal{L}(G_I, S, R) = \mathcal{L}(S) + \mathcal{L}(R \mid S)
\]
How to optimize the score?

- Two-step process
  - Given $k$, quickly identify high-quality set
  - Given these nodes, optimize the ripple $R$
Optimizing the score

- High-quality $k$-seed-set
  - Exoneration

- Best single seed:
  - Smallest eigenvector of Laplacian sub-matrix
  - Analyze a Constrained SI epidemic

- Exonerate neighbors

- Repeat
Optimizing the score

- Optimizing $R$
  - Get the MLE ripple!

- Finally use MDL score to tell us the best set

- NetSleuth: Linear running time in nodes and edges
  \[ O(k^* (E_I + E_F + V_I)) \]
Experiments

- Evaluation functions:
  - MDL based
    \[ Q_{\text{MDL}} = \frac{\mathcal{L}(G_I, S, R)}{\mathcal{L}(G_I, S^*, R^*)} \]
  - Overlap based
    \[ Q_{\text{JD}} = \frac{\mathbb{E}[JD_S(V_I)]}{\mathbb{E}[JD_{S^*}(V_I)]} \]
    
    \((JD = \text{Jaccard distance})\)
Experiments: # of Seeds

One Seed

Two Seeds

Three Seeds
Experiments: Quality (MDL and JD)

One Seed

Two Seeds

Three Seeds

Ideal = 1

$$Q_{MDL} = \frac{\mathcal{L}(G_I, S, R)}{\mathcal{L}(G_I, S^*, R^*)}$$

$$Q_{JD} = \frac{\mathbb{E}[JD_{S}(\mathcal{V}_I)]}{\mathbb{E}[JD_{S^*}(\mathcal{V}_I)]}$$
Experiments: Quality (Jaccard Scores)

One Seed

Two Seeds

Three Seeds

Closer to diagonal, the better
Experiments: Scalability