Finding the Source of a Rumor

Networks out of Control

Matthias Grossglauser
Rumor Models

- Application: detecting source of an epidemic
  - Who leaked document X into the blogosphere?
  - Which bank started the financial crisis?
  - Who spread a rumor among your circle of friends?
  - Who is patient zero?

- Susceptible-Infected model of epidemics
  - Infected nodes infect susceptible neighbors = spreading the rumor
  - Infection remains forever = not forgetting
  - Network model: contacts, influence, dependence

- Reference:
SI Rumor Model
Goal: Find source among "○" given network $G(V,E)$
Model

- **Rumor spreading:**
  - When $u$ gets the rumor, then it takes time $\tau_{u,v}$ for $v$ to get infected
  - $\{\tau_{u,v}\}$ i.i.d.
  - $\tau_{u,v}$ exponential (i.e., memoryless)
  - $v$ infected by first neighbor whose “timer” expires

- **Observation of $N$ infected nodes**
  - Infected subgraph $G_N$
  - No prior knowledge (uniform prior):
    $$\hat{\nu} = \arg \max_v P(G_v |v)$$
  - Very hard to evaluate in general – look at special cases first: trees
What sequence of infections could have generated $G_4$, e.g., with source = 1?
Example and Permitted Permutation

Def: “rumor centrality” = \( R(\nu, G_N) \): # of permissible permutations

\( \nu = 1 \):
- \{1,2,3,4\}
- \{1,2,4,3\}
- \{1,3,2,4\}
- ...

\( \nu = 2 \):
- \{2,1,3,4\}
- \{2,1,4,3\}
- \{2,3,1,4\}
- \{2,3,4,1\}
- \{2,4,1,3\}
- \{2,4,3,1\}
- \{1,2,3,4\}
- ...

Rumor on Regular Tree is Markovian

- **Permitted permutation $\sigma$:**
  - Not forbidden by the topology
  - If $(u, v)$ is an edge, then the node closer to source has to be infected first

- **Infection process $\sigma_k$:**
  - $\sigma_k$: permutation order of the first $k$ nodes ($\sigma = \sigma_N$)
  - Property 1: Markov (because i.i.d. timers)
  - Property 2: Equiprobable transitions to possible next states (because constant degree)

$$P(\sigma|v) = \prod_{k=1}^{N-1} \frac{1}{dk-2(k-1)} \overset{\text{def}}{=} p(d, N)$$

Tree: $k$ nodes $\rightarrow$ $dk$ stubs; $k - 1$ internal edges are already saturated
ML Source Estimator for Regular Tree

- ML estimator:
  - $\hat{v} = \arg\max_v P(G_N|v)$
  - $= \arg\max_v \sum_\sigma P(\sigma|v)$
  - $= \arg\max_v R(v, G_N) \rho(d, N)$
  - $= \arg\max_v R(v, G_N)$

- In other words, key challenge is computing $R(v, G_N)$
Rumor Centrality: Recurrence

Def:
- $T(v, u)$: # nodes in subtree rooted at $u$ with source $v$
- $T(v, v) = N$
Rumor Centrality: Recurrence

- Recall: number of partitions
  - Set of size $n$: how many different partitions into $d$ sets of sizes $\{n_1, n_2, ..., n_d\}, \sum n_i = n$?
  - $n! = n_1! \times n_2! \times ... \times n_d!$

- Rumor centrality recursion:
  - $R(v, G_N) = \frac{(N-1)!}{\prod_{u \in ch(v)} T(v,u)!} \times \prod_{u \in ch(v)} R(u, G_N) = (N - 1)! \prod_{u \in ch(v)} \frac{R(u,T(v,u))}{T(v,u)!}$

To get a valid permutation: take a valid downstream permutation (among $R(u, T(v, u))$ permitted) in each subtree $T(v, u)$; interlace these in any order
Rumor Centrality: Recurrence (2)

- Expand to 2\(^{nd}\) level:
  - \(R(v, G_N) = (N - 1)! \prod_{u \in ch(v)} \frac{R(u, T(v, u))}{T(v, u)!} = \)
  - \(= (N - 1)! \prod_{u \in ch(v)} \frac{(T(v, u) - 1)!}{T(v, u)!} \times \prod_{w \in ch(u)} \frac{R(w, T(v, w))}{T(v, w)!} \)
  - \(= (N - 1)! \prod_{u \in ch(v)} \frac{1}{T(v, u)} \times \prod_{w \in ch(u)} \frac{R(w, T(v, w))}{T(v, w)!} \)

- Expand all the way to leaves:
  - \(R(v, G_N) = (N - 1)! \prod_{u \in G_N \setminus \{v\}} \frac{1}{T(v, u)} = \)
  - \(= N! \prod_{u \in G_N} \frac{1}{T(v, u)} \)
Rumor Centrality of Neighbors

\[ T(u, v) \]

\[ T(v, u) + T(u, v) = N \]
Rumor Centrality of Neighbors (2)

- Source = $u$
- Source = $v$

Recall:

$$R(v, G_N) = N! \prod_{u \in G_N} \frac{1}{T(v, u)}$$

$$R(u, G_N) T(u, v) = R(v, G_N) T(v, u)$$
Linear-Time Algorithm to Compute Centrality

- **Up-Phase**
  - Select a random reference node $v$
  - Start at leaves, proceed upwards towards $v$
  - For each $u$, compute $T(v, u)$ and $R(u, T(v, u))$ (downstream rumor centrality)

- **Down-Phase**
  - Compute full rumor centrality of every node
  - Uses source-conversion formula:

\[
R(u, G_N) = R(v, G_N) \frac{T(v, u)}{N - T(v, u)}
\]
Example: Distributed Source Finding

Up-message: (cumul. size, cumul. product)

3: (1,1)

4: (1,1)

5: (3+1+1=5, 1x3x5=15)

5: (2+1=3, 1x1x3=3)

2: (1,1)

1: (1,1)
Example: Distributed Source Finding (2)

- **1:** $1/105 \times 5/2$
- **2:** $1/105 \times 1/6$
- **3:** $1/42 \times 3/4$
- **4:** $1/42 \times 1/6$
- **5:** $1/56 \times 1/6$
- **6:** $1/56 \times 1/6$

Down-message: (rumor centrality w.r.t. source $u$)

$$\frac{N!}{630} \quad \frac{N!}{105} \quad \frac{N!}{42} \quad \frac{N!}{56} \quad \frac{N!}{336} \quad \frac{N!}{252} \quad \frac{N!}{336}$$
Example: Distributed Source Finding (3)

ML estimate = max \{\text{rumor centrality}\}
Detectability with Time: $d=2$ (Line)

- **Theorem:**
  - Probability of detection goes to zero

- **Intuition:**
  - Center of one-dimensional epidemic washes out too much
  - Too little information: only left and right boundary
  - Prob. of perfect symmetry goes to zero
Detectability with Time: $d > 2$ (Regular Tree)

- **Theorem:**
  - Probability of detection $> \alpha > 0$ for $t \to \infty$

- **Intuition:**
  - Exponential expansion of tree maintains enough detail about history of process
  - Much more information: exponential # of paths down tree
Conclusion: Rumor Source Identification

- **Result:**
  - SI epidemic model
  - ML estimator for source given set of infected nodes
  - Recursive expression for rumor centrality
  - Distributed, linear time algorithm to compute ML estimator
  - Regular trees:
    - \( d = 2 \): source becomes unidentifiable as epidemic spreads
    - \( d > 2 \): even for \( t \to \infty \), non-zero detection probability!

- Recent and exciting research area!