Ranking Mechanisms for Interaction Networks

Ramasuri Narayananam Vinayaka Pandit

IBM India Research Lab
Bangalore and New Delhi, India
Email ID: \{nramasuri, pvinayak\}@in.ibm.com

December 21, 2011
Agenda

1. **Viral Marketing: Basic Concepts**
2. Node Ranking Mechanisms for Viral Marketing
3. Edge Ranking Mechanisms for Viral Marketing
Social networks play a crucial role in the spread of information

**Viral Marketing**: This phenomenon exploits the social interactions among individuals to promote awareness for new products. Also known as *information diffusion* or *influence maximization* in social networks

Given Information: Social network of individuals and information about the extent individuals in the network influence each other

We want to market a new product that we hope will be adopted by a large fraction of the network.

A key issue in viral marketing is to select a set of *initial seeds* in the social network and give them free samples of the product to trigger cascade of influence over the network.
Models for Diffusion of Information

- Linear threshold model
- Independent cascade model
Linear Threshold Model

- Call a node active if it adopts the product/information
- Initially every node is inactive except the nodes in the initial target.
- Let us consider a node \( i \) and represent its neighbors by the set \( N(i) \)
- Node \( i \) is influenced by a neighbor node \( j \) according to a weight \( w_{ji} \). These weights are normalized in such a way that

\[
\sum_{j \in N(i)} w_{ji} \leq 1.
\]

- Further each node \( i \) chooses a threshold, say \( \theta_i \), uniformly at random from the interval \([0,1]\)
- This threshold represents the weighted fraction of node \( i \)'s neighbors that must become active in order for node \( i \) to become active
Given a random choice of thresholds and an initial set (call it $S$) of active nodes, the diffusion process propagates as follows:

- In time step $t$, all nodes that were active in step $(t-1)$ remain active.
- We activate every node $i$ for which the total weight of its active neighbors is at least $\theta_i$.
- If $A(i)$ is assumed to be the set of active neighbors of node $i$, then $i$ gets activated if
  \[
  \sum_{j \in A(i)} w_{ji} \geq \theta_i.
  \]
- This process stops when there is no new active node in a particular time interval.
Linear Threshold Model: An Example

$\theta = 0.64$
Linear Threshold Model: An Example
Linear Threshold Model: An Example

[Diagram of a network with nodes and edges.]

- Active Node

θ = 0.64

Weights: 0.41, 0.12, 0.25, 0.15, 0.07
Linear Threshold Model: An Example

0.41 + 0.25 > \( \theta = 0.64 \)

\( \theta \) Active Node
Linear Threshold Model: An Example

0.41 + 0.25 > \theta(= 0.64)

\[ \text{Active Node} \]
Independent Cascade Model

- Initially every node is inactive except the nodes in the initial target.
- The process unfolds in discrete steps according to the following randomized rule. When node $j$ first becomes active in step $t$, it is given a single chance to activate each currently inactive neighbor $i$; it succeeds with a probability $p(j, i)$.
- If $i$ has multiple newly activated neighbors, their attempts are sequenced in an arbitrary order.
- If $j$ succeeds, then $i$ will become active in step $t + 1$; but whether or not $j$ succeeds, it cannot make any further attempts to activate $i$ in subsequent rounds.
- This process runs until no more activations are possible.
Influence Maximization Problem

- **Objective Function:** We define the *influence* of a set of nodes $A$, denoted $\sigma(A)$, to be the expected number of active nodes at the end of the process.

- For economic reasons, we would like to limit the size of $A$.

- **Influence Maximization Problem:** For a given constant $k$, influence maximization problem seeks to find a set of nodes $A$ of cardinality $k$ that maximizes $\sigma(A)$. 
A Few Key Results

- **Lemma 1**: [Kempe, et al. (2003)] The influence maximization problem is NP-hard for the Linear Threshold Model.

- **Lemma 2**: [Kempe, et al. (2003)] The influence maximization problem is NP-hard for the Independent Cascade model.

**Submodular Function**: An arbitrary set function $f(.)$ that maps subsets of a ground set $U$ to real numbers is called submodular if

$$f(S \cup \{i\}) - f(S) \geq f(T \cup \{i\}) - f(T), \quad \forall S \subseteq T \subseteq U$$

- **Lemma 3**: [Kempe, et al. (2003)] For an arbitrary instance of the Linear Threshold Model, the resulting influence function $\sigma(.)$ is submodular.

- **Lemma 4**: [Kempe, et al. (2003)] For an arbitrary instance of the Independent Cascade Model, the resulting influence function $\sigma(.)$ is submodular.
Greedy Algorithm [Kempe, et al. (2003)]

1. Set $A \leftarrow \phi$.
2. for $i = 1$ to $k$ do
3. Choose a node $n_i \in N \setminus A$ maximizing $\sigma(A \cup \{n_i\})$
4. Set $A \leftarrow A \cup \{n_i\}$.
5. end for

**Theorem:** Let $S^*$ be the set that maximizes $\sigma(.)$ over all $k$-element sets and let $S$ be the set of $k$ nodes constructed by the greedy algorithm. Then $\sigma(S) \geq (1 - \frac{1}{e})\sigma(S^*)$; in other words, $S$ provides $(1 - \frac{1}{e})$-approximation.
A Node Ranking Mechanism (SPIN):
- Game theory based mechanism
- Running time is faster than that of the greedy asymptotically
- A drawback of the greedy algorithm is its running time is proportional to $k$ (i.e. the cardinality of initial seed set $S$)

An Edge Ranking Mechanism (SPINE):
- Greedy algorithm of KKT (2003) runs very slow in practice even in small size data sets
- Social networks of practical interest consist of millions of nodes and edges
- Graph sparsification is a data-reduction technique that retains only key edges revealing the backbone of information propagation over the network
Cooperative Game Theory

**Definition:** A cooperative game with transferable utility is defined as the pair \((N, v)\) where \(N = \{1, 2, \ldots, n\}\) is a set of players and \(v : 2^N \rightarrow \mathbb{R}\) is a characteristic function, with \(v(.) = 0\). We call such a game also as a game in coalition form, game in characteristic form, or coalitional game or TU game.

**Example:** There is a seller \(s\) and two buyers \(b_1\) and \(b_2\). The seller has a single unit to sell and his willingness to sell the item is 10. Similarly, the valuations for \(b_1\) and \(b_2\) are 15 and 20 respectively. The corresponding cooperative game is:

- \(N = \{s, b_1, b_2\}\)
- \(v(\{s\}) = 0\), \(v(\{b_1\}) = 0\), \(v(\{b_2\}) = 0\), \(v(\{b_1, b_2\}) = 0\)
- \(v(\{s, b_1\}) = 5\), \(v(\{s, b_2\}) = 10\), \(v(\{s, b_1, b_2\}) = 10\)
The Shapley’s Theorem

**Theorem:** There is exactly one mapping \( \phi : \mathbb{R}^{2N-1} \to \mathbb{R}^N \) that satisfies Symmetry, Linearity, and Carrier axioms. This function satisfies: \( \forall i \in N, \forall v \in \mathbb{R}^{2N-1}, \)

\[
\phi_i(v) = \sum_{C \subseteq N \setminus \{i\}} \frac{|C|!(n - |C| - 1)!}{n!} \left\{ v(C \cup \{i\}) - v(C) \right\}
\]

**Example:** Consider the following cooperative game: \( N = \{1, 2, 3\} \), \( v(1) = v(2) = v(3) = v(23) = 0, v(12) = v(13) = v(123) = 300. \) Then we have that

\[
\phi_1(v) = \frac{2}{6} v(1) + \frac{1}{6} (v(12) - v(2)) + \frac{1}{6} (v(13) - v(3)) + \frac{2}{6} (v(123) - v(23))
\]

It can be easily computed that \( \phi_1(v) = 200, \phi_2(v) = 50, \phi_3(v) = 50 \)
SPIN: A Node Ranking Mechanism

- It is a cooperative game theoretic framework for the influence maximization problem
- Measures the influential capabilities of the nodes as provided by the Shapley value
- Shapley value based discovery of Influential Nodes (SPIN):
  1. Ranking the nodes,
  2. Choosing the top-$k$ nodes from the ranking order.
- Advantages of SPIN:
  1. Quality of solution is same as that of popular benchmark approximation algorithms
  2. Works well for both sub-modular and non-submodular objective functions
  3. Running time is independent of the value of $k$
Let $\pi_j$ be the $j$-th permutation in $\hat{\Omega}$ and $R$ be repetitions.

Set $MC[i] \leftarrow 0$, for $i = 1, 2, \ldots, n$.

for $j = 1$ to $t$ do

Set $temp[i] \leftarrow 0$, for $i = 1, 2, \ldots, n$.

for $r = 1$ to $R$, do

assign random thresholds to nodes;

for $i = 1$ to $n$, do

$temp[i] \leftarrow temp[i] + v(S_i(\pi_j) \cup \{i\}) - v(S_i(\pi_j))$

for $i = 1$ to $n$, do

$MC[i] \leftarrow temp[i]/R$;

for $i = 1$ to $n$, do

compute $\Phi[i] \leftarrow \frac{MC[i]}{t}$

Sort nodes based on the average marginal contributions of the nodes
Initially all nodes are inactive.
Randomly assign a threshold to each node.
Fix a permutation $\pi$ and activate $\pi(1)$ to determine its influence.
Next consider $\pi(2)$. If $\pi(2)$ is already activated, then the influence of $\pi(2)$ is 0. Otherwise, activate $\pi(2)$ to determine its influence.
Continue up to $\pi(n)$.
Repeat the above process $R$ times (for example 10000 times) using the same $\pi$.
Repeat the above process $\forall \pi \in \hat{\Omega}$. 
Choosing Top-$k$ Nodes

1. Naive approach is to choose the first $k$ in the RankList[] as the top-$k$ nodes.
2. **Drawback**: Nodes may be clustered.
3. RankList[] = \{5, 4, 2, 7, 11, 15, 9, 13, 12, 10, 6, 14, 3, 1, 8\}.
4. Top 4 nodes, namely \{5, 4, 2, 7\}, are clustered.
5. Choose nodes:
   - rank order of the nodes
   - spread over the network
<table>
<thead>
<tr>
<th>$k$ value</th>
<th><strong>Greedy</strong> Algorithm</th>
<th><strong>Shapley Value</strong> Algorithm</th>
<th><strong>MDH based Algorithm</strong></th>
<th><strong>HCH</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td>7</td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>10</td>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>12</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td>5</td>
<td>13</td>
<td>13</td>
<td>10</td>
<td>8</td>
</tr>
<tr>
<td>6</td>
<td>14</td>
<td>14</td>
<td>13</td>
<td>8</td>
</tr>
<tr>
<td>7</td>
<td>15</td>
<td>15</td>
<td>13</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>15</td>
<td>15</td>
<td>13</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>15</td>
<td>15</td>
<td>13</td>
<td>10</td>
</tr>
<tr>
<td>10</td>
<td>15</td>
<td>15</td>
<td>13</td>
<td>11</td>
</tr>
<tr>
<td>11</td>
<td>15</td>
<td>15</td>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>12</td>
<td>15</td>
<td>15</td>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>13</td>
<td>15</td>
<td>15</td>
<td>14</td>
<td>14</td>
</tr>
<tr>
<td>14</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
</tr>
</tbody>
</table>
Running Time of SPIN

- Overall running time of SPIN is $O(t(n + m)R + n \log(n) + kn + kRm)$ where $t$ is a polynomial in $n$.

- For all practical graphs (or real world graphs), it is reasonable to assume that $n < m$. With this, the overall running time of the SPIN is $O(tmR)$ where $t$ is a polynomial in $n$. 
Experimental Results: Data Sets

- Random Graphs
  - Sparse Random Graphs
  - Scale-free Networks (Preferential Attachment Model)

- Real World Graphs
  - Co-authorship networks,
  - Networks about co-purchasing patterns,
  - Friendship networks, etc.
## Experimental Results: Data Sets

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Number of Nodes</th>
<th>Number of Edges</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sparse Random Graph</td>
<td>500</td>
<td>5000 (approx.)</td>
</tr>
<tr>
<td>Scale-free Graph</td>
<td>500</td>
<td>1250 (approx.)</td>
</tr>
<tr>
<td>Political Books</td>
<td>105</td>
<td>441</td>
</tr>
<tr>
<td>Jazz</td>
<td>198</td>
<td>2742</td>
</tr>
<tr>
<td>Celegans</td>
<td>306</td>
<td>2345</td>
</tr>
<tr>
<td>NIPS</td>
<td>1061</td>
<td>4160</td>
</tr>
<tr>
<td>NetScience</td>
<td>1589</td>
<td>2742</td>
</tr>
<tr>
<td>HEP</td>
<td>10748</td>
<td>52992</td>
</tr>
</tbody>
</table>
Experiments: Random Graphs

![Graph 1](image1.png)

![Graph 2](image2.png)

![Graph 3](image3.png)

![Graph 4](image4.png)
Experiments: Real World Graphs

![Graph 1](image1.png)

![Graph 2](image2.png)

![Graph 3](image3.png)

![Graph 4](image4.png)
Top-10 Nodes in Jazz Dataset
Top-10 Nodes in NIPS Co-Authorship Data Set

[Image of a network graph with nodes and links, indicating influential nodes with a special symbol]
**SPINE: An Edge Ranking Mechanism**

- **Given Information:** A social network of individuals and a log of past propagations (or a log of past actions performed by the nodes in the network)
- Assume that these actions have propagated in the network via the independent cascade model
- Maximum likelihood parameters of this model can be found for instance by using the EM algorithm
- Given the parameters, the sparsification problem stated as follows: it is required to preserve the set of $k$ links that maximize the likelihood of the observed data.
- Sparsifying a network with respect to a log of past actions can be seen as revealing the backbone of information propagation in the network
Estimating Influence Probabilities for IC Model

- Every trace generated by the independent cascade model is associated with a likelihood value.
- For an action $\alpha$, (i) $F^{+}_{\alpha}(v) =$ the set of nodes that positively influenced $v$, and (ii) $F^{-}_{\alpha}(v) =$ the set of nodes that definitely failed to influence $v$.
- Then the likelihood $L_{\alpha}(G)$ of the trace for action $\alpha$ can be written as

$$L_{\alpha}(G) = \prod_{v \in V} P^{+}_{\alpha}(v) P^{-}_{\alpha}(v)$$

where $P^{+}_{\alpha}(v) = 1$ if $F^{+}_{\alpha}(v) = \emptyset$ and

$$P^{+}_{\alpha}(v) = 1 - \prod_{u \in F^{+}_{\alpha}(v)} (1 - p(u,v))$$

otherwise;

$$P^{-}_{\alpha}(v) = \prod_{u \in F^{-}_{\alpha}(v)} (1 - p(u,v)).$$

- Then the total log-likelihood of the given traces of actions is given by:

$$\log L(G) = \sum_{a \in A} \log L_{\alpha}(G) = \sum_{a \in A} \sum_{v \in V} \left( \log P^{+}_{\alpha}(v) + \log P^{-}_{\alpha}(v) \right)$$
Need to estimate the influence probabilities $p(u, v)$ of the independent cascade model from a set of traces.

Consider a set of actions $A$. For each action $\alpha \in A$, we observe its propagation trace.

The probability values $p(u, v)$ that maximize the log-likelihood of the given traces can be computed using the following iterative formula:

$$p^{k+1}(u, v) = \frac{p^k(u, v)}{|A^+_v| + |A^-_v|} \sum_{\alpha \in A^+_v} \frac{1}{P^+_\alpha(v)}$$

where actions in the set $A^+_v = \{\alpha \in A | F^+_\alpha(v) \ni u\}$ have traces where $u$ positively influence $v$, and the actions in the set $A^-_v = \{\alpha \in A | F^-_\alpha(v) \ni u\}$ have traces where $u$ definitely failed to influence $v$. 

Ramasuri Narayanan, Vinayaka Pandit ()
ECL, CSA, IISc
December 21, 2011
**Sparsification**

**Sparsification Problem:** Given a network $G = (V, D)$ with probabilities $p(u, v)$ on the arcs, a set $A$ of action traces, and an integer $k$, find a sparse subnetwork $G_s = (V, D_s)$ of $G$ of size $|D_s| = k$, so that the log-likelihood function $\log L(G_s)$ is maximized.

- Sparsification problem is not solved by selecting the $k$ arcs $(u, v)$ in $D$ with the largest probability values $p(u, v)$
- For $k = 3$, the best sparse model $G_s = (V, D_s)$ is the one with $D_s = \{ (\Omega, u_1), (\Omega, u_2), (u_2, v) \}$ even though $p(u_2, v) < p(u_1, v)$.
- Note that the alternative option of $D_s = \{ (\Omega, u_1), (\Omega, u_2), (u_1, v) \}$ leads to zero likelihood.
For the sparse network $G_s = (V, D_s)$ to have finite log-likelihood, it is necessary that the traces of all actions $A$ are possible for its set of arcs $D_s$

That is, if node $v$ performs an action $\alpha$ in $A$, then $D_s$ must include an arc from at least one of the nodes $F^+_{\alpha}$ that possibly influence $v$

**Lemma:** Deciding whether Sparsification Problem has finite solution is NP-hard.
**Hint:** It is not difficult to obtain a reduction from the Hitting Set problem.

**Hitting Set Problem:** Given a collection of sets \( S = \{S_1, S_2, \ldots, S_m\} \) over a universe of \( n \) elements \( U = \{1, 2, \ldots, n\} \) (i.e. \( S_j \subseteq U \)), a hitting set for \( S \) is a set \( H \subseteq U \) that intersects all sets in \( S \).

**Theorem:** Approximating Sparsification Problem up to any multiplicative factor is NP-hard.
A Greedy Algorithm: SPINE

- SPINE produces a solution $D_s$ to the Sparsification Problem in $k$ steps, adding to $D_s$ one arc at each step.

- These $k$ steps are divided into two phases:
  - In the first phase, SPINE aims to identify a solution $D_0$ of finite log likelihood.
  - In the second phase, it greedily seeks a solution of maximum log likelihood.

- This two phase approach is due to the observation that Sparsification Problem is at least as difficult as identifying a solution of finite log likelihood.
SPINE: First Phase

For each node \( v \), we seek for a hitting set of collection

\[
C(v) = \{D_\alpha^+(v) \neq \emptyset, \alpha \in A\}
\]

Since hitting set is NP-hard, use the greedy approximation algorithm describes in Johnson (STOC 1973) as follows:

- Order the arcs \((u, v)\) by the number \( n(u, v) \) of actions for which \( u \) possibly influenced \( v \) where
  \[
  n(u, v) = |\{D_\alpha^+(v) \in C(v), (u, v) \in D_\alpha^+(v)\}|
  \]
- At each step, the arc \((u, v)\) with the maximum number \( n(u, v) \) is selected and all sets \( D_\alpha^+(v) \) that contain \((u, v)\) are ignored for the rest of this process
- The first phase ends when either the limit of \( k \) arcs is reached or selected arcs lead to a finite log likelihood
**SPINE: Second Phase**

- Let $G_0 = (V, D_0)$ be the associated sparse network at the end of First Phase.
- If $|D_0| < k$, then we still need to select $k - |D_0|$ arcs.
- Choose these $k - |D_0|$ arcs by selecting greedily at each step the arc that offers the largest increase in log-likelihood.

**Lemma:** Let $D_{opt}$ be a superset of $D_0$ that contains $k$ arcs and induces a subgraph $G_{opt} = (V, D_{opt})$ of $G$ with maximum log-likelihood. Also, let $D_{sp}$ by the set of arcs returned by SPINE and let $G_{sp} = (V, D_{sp})$ be the induced subgraph. That is, $D_{sp}$ is also superset of $D_0$ and it has $k$ arcs. Then, provided that $\log L(G_0)$ is finite, we have

$$\log L(G_{sp}) \geq \frac{1}{e} \log L(G_0) + (1 - \frac{1}{e}) \log L(D_{opt})$$
Experiments - SPINE for Influence Maximization

- Apply the SPINE on the network of YMEME-S (consists of 2573 nodes and 466284 edges) to identify two sparse networks $G_1$ and $G_2$ of $k_1 = 25688$ and $k_2 = 38899$ arcs respectively.
- Note that here $G_1$ is the smallest network with non-zero likelihood identified with SPINE and $G_2$ is the smallest network of maximum likelihood.
- Run the greedy algorithm of Kempe, et al. (KDD 2003) on each of $G$, $G_1$, and $G_2$ respectively.
Thank You