Greedy Maximization Framework for Graph-based Influence Functions

Edith Cohen
Google Research
Tel Aviv University

HotWeb '16
Large Graphs

Model relations/interactions (edges) between entities (nodes)

- **Explicit**: Call detail, email exchanges, Web links, social networks (friend, follow, like), commercial transactions, video views, ...
- **Implicit**: Images, search queries, (edges are shared features or close embedding vectors)

Some nodes are more central than others
Diffusion in Networks

- Edges model *direct connections* between entities
- **Diffusion**: Contagion, information (news, opinions), ... can spread from *seed* nodes through edges to nodes multiple hops away

- **Influence**: A measure of the combined power/importance/coverage of a set of *seed nodes*. (according to the diffusion process)
Influence in Networks

**Applications:** $\text{Inf}(S) = \text{quality of a seed entities } S$ as: anchors, representatives, cluster centers, hubs in distribution networks, candidates for active learning of labels/properties, seeds for viral marketing

**Computational Problems:**

- **Influence queries:** Given seed set $S$, compute (approximate) $\text{Inf}(S)$
- **Influence maximization** $\arg\max_{S \mid |S|=s} \text{Inf}(S)$

Find a set of entities with maximum influence for its size. or With a “budget” $s$, who should we select?
Overview of contributions

- A unified model of graph-based influence functions: Includes functions proposed in previous work and extends to allow general submodular aggregations.

- A meta-algorithm for influence maximization: Modular design, near linear computation, statistical worst-case guarantees on approximation quality
Unified model: **Pairwise utility to influence**

- **Graph structure, diffusion process**
  \[ \rightarrow \text{pairwise utility } u_{ij} \text{ of node } i \text{ to node } j \]

- **The utility** of a seed set \( S \) to a node \( j \):
  \[ u_{Sj} = \text{aggregate}_{i \in S} u_{ij} \]
  e.g. aggregate = max

- **The influence** of seed set \( S \) is the sum over \( j \) of the utility of \( S \) to \( j \)
  \[ \text{Inf}(S) = \sum_j u_{Sj} = \sum_j \text{aggregate}_{i \in S} u_{ij} \]

**Centrality:** Influence of a single node \( \text{Inf}(i) = \sum_j u_{ij} \)
Aggregation functions

Utility of $S$ to $j$

\[ u_{Sj} = \text{aggregate } u_{ij} \]
\[ i \in S \]

- **Max**: value equal to that the utility of the most useful seed node $u_{Sj} = 5$
- **Sum**: The more the merrier
  \[ u_{Sj} = 5 + 4 + 2 \]
- **Top-2**: sum of top two seed utility values $u_{Sj} = 5 + 4$ (limited capacity)
  \[ + \text{diminishing return } u_{Sj} = 5 + \frac{1}{2} \cdot 4 \]

**Submodular top-$\ell$**: Up to top $\ell$ seeds contribute, non-increasing marginal contribution

When $|S| = 1$ (centrality): All “aggregate” are the same $u_{Sj}$

\[ S = \{ \text{\includegraphics{minions}}, \text{\includegraphics{minions}}, \text{\includegraphics{minions}} \} \]

\[ \Rightarrow \text{Influence function is submodular and monotone} \]
Pairwise utility from graph structure

Shorter paths, more paths, stronger edges on paths $\Rightarrow$ higher $u_{ij}$

Ways to define utility $u_{ij}$ from graph structure:

- **Reachability** $u_{ij} = 1 \iff i \sim j$ [Kempe Kleinberg Tardos 2003]++
- **Distance** $u_{ij} = \alpha(d_{ij})$ with decaying $\alpha$ [Bavelas 1948]++ [CK 2004] [Bloch Jackson 2007]++
  - Threshold: $u_{ij} = 1 \iff d_{ij} \leq T$ [Gomez Rodriguez et al ICML 11] [Du et al NIPS 13]++...
- ..... **Reverse-rank** $u_{ij} = \alpha(\pi_{ji})$ [Korn Muthu 01, Buchnik C 16]
- **Survival time** [C ‘16] (inspired by survival analysis)

+ randomized models to generate edge lengths/presence
Simplest Model: Reachability

Utility: $u_{ij} = I_{j \sim i}$

aggregate=max: $u_{Sj} = \max_{j \in S} u_{ij} = I_{\exists j \in S \text{ s.t. } j \sim i}$

$\text{Inf}(S) = \sum_{j} u_{Sj} = \left| \left\{ i \mid \exists j \in S, j \sim i \right\} \right|$ = #nodes reachable from at least one node in $S$. 

$\text{Inf}(\text{Yoda}) = 5$
Simplest Model: Reachability + max aggregation

\[ \inf_{\max} = 9 \]

Submodular and monotone!
Reachability + top-$\ell$ submodular aggregation

Utility: $u_{ij} = I_{j \sim i}$

$\text{Inf}(S) = \sum_j u_{sj}$

$u_{sj} = f(\#(j \in S \text{ s.t. } j \sim i))$

$f$ monotone concave
Randomized edge presence

Utility $u_{ij}$ should decrease with path length and increase with path multiplicity.

Independent Cascade (IC) model [KKT ‘03]:

Edge $e$ active with probability $p_e$ (independent)
Distance-based Influence

Utility: $u_{ij} = e^{-d_{ij}}$

Max aggregate

$u_{Sj} = e^{-d_{Sj}}$

$\text{Inf}(S) = \sum_{j} u_{Sj}$

$$\text{Inf} = 1 + \frac{5}{e} + \frac{1}{e^2} + \frac{1}{e^3} + \frac{1}{e^4}$$
Distance-based Influence

Utility: $u_{ij} = e^{-d_{ij}}$

Max aggregate

$u_{Sj} = e^{-d_{Sj}}$  \hspace{1cm} \text{Inf}(S) = \sum_j u_{Sj}$

$\text{Inf}(\text{network}) = \left(2 + \frac{7}{e} + \frac{2}{e^2} + \frac{1}{e^3} + \frac{1}{e^4}\right)$
Randomized edge lengths

Utility $u_{ij}$ should decrease with path length and increase with path multiplicity.

**Randomized:** Edge lengths $\sim \text{Exp}[w_e]$ drawn independently from Weibull/Exponential distribution

deterministic

= 

randomized

\[<<\]
Reverse-rank Influence
(special case) reverse NN

Utility: \( u_{ij} = I_{i=NN(j)} \quad u_{Sj} = I_{NN(j) \in S} \quad \text{Inf}(S) = \sum_j u_{Sj} \)

\[ \text{Inf}(\text{ }) = 2 \]
Reverse-rank Influence

$\pi_{ji}$: position of $i$ in increasing distances from $j$

Utility: $u_{ij} = \alpha(\pi_{ji})$

$\pi_{ji}$: rank of by $j$

with $u_{ij} = \frac{1}{\pi_{ji}}$:

$\text{Inf}(\text{ }) =$

$1 + \frac{1}{2} + 3 \cdot \frac{1}{3} +$

$+4 \cdot \frac{1}{4} + \frac{1}{5} + 2 \cdot \frac{1}{8}$
Survival time Influence

- Each edge has weight that corresponds to its “lifetime”.
- $t_{ij}$ maximum $t$ such that $j \sim i$ when using edges with lifetime $\geq t$ (connectivity survival time)

Utility: $u_{ij} = t_{ij}$

$$\text{Inf}(\text{边}) = 4 \cdot 5 + 4 \cdot 1$$
Overview of contributions

- A unified model of graph-based influence functions: Includes functions proposed in previous work and extends to allow general submodular aggregations.

- A meta-algorithm for influence maximization: Modular design, near linear computation, statistical worst-case guarantees on approximation quality
Influence maximization

Given $s$, find a set of seed nodes $S$ of size $s$ with maximum influence $\arg \max_{|S|=s} \text{Inf}(S)$

- NP hard, even to approximate [Feige ‘98]
- Influence function is submodular and monotone $\implies$ Greedy algorithm is polynomial with approximation ratio

$$\geq 1 - \left(1 - \frac{1}{s}\right)^s > 1 - \frac{1}{e} \text{ of opt} \ [NWF \ ‘78]$$

Greedy iteration:
- Select $i \not\in S$ with maximum $\text{Inf}(S \cup \{i\}) - \text{Inf}(S)$
- $S \leftarrow S \cup \{i\}$

Greedy sequence approximates the full size/quality tradeoff
But ...Exact greedy too slow - need near-linear algorithms
Meta-SKIM

Sketch Based Influence Maximization

- Computes an *approximate greedy sequence*. Key property: Approximate the *marginal influence* of nodes to identify approximate maximizer *at each iteration*.

- Scales by maintaining and updating weighted samples (sketches) of marginal influence sets.
Meta-SKIM influence maximization

SKIM:
- reachability + max [CDPW ICDM ‘14]
- General decay + distances + max [CDPW ’15]
- Threshold-decay + reverse-rank + max [BC Sigmetrics ’16]

Scales to networks with $10^8$ edges, actual approximation within few percent

Meta-SKIM: unifies and generalizes previous work
- General decay (with distance, reverse rank, ...)
- Reachability, distances, reverse ranks, survival threshold
- Submodular top-$\ell$ aggregations

Inherits scalability, statistical guarantees, computation bounds
Meta-SKIM

- Maintain **weighted samples** of “marginal influence sets” of nodes.
- **Repeat:**
  - Sample until estimates are accurate for “near-maximizers” of marginal influence.
  - Add the approximate maximizer to seed set.
  - Update residual problem

- **Approximation ratio guarantee:**
  \[ \geq 1 - \left(1 - \frac{1}{s}\right)^s - \epsilon \text{ times } \text{Opt} \]

- Near linear worst-case bound on computation!!
Meta-SKIM

- Maintain **weighted samples** of “marginal influence sets” of nodes.
- **Repeat:**
  - Sample until estimates are accurate for “near-maximizers” of marginal influence.
  - Add the approximate maximizer to seed set.
  - Update residual problem

**Modular:** Specified through **oracle access to utility matrix**

- Sampling uses **reverse sorted access oracle**: from $j$ returns nodes in order of non-increasing utility $u_{ij}$. **Implemented as graph search**
- Updates use **forward search oracle** from a new seed. Returns all nodes with updated marginal utility. **Also a graph search.**
Meta-SKIM

- Maintain **weighted samples** of “marginal influence sets” of nodes.
- **Repeat:**
  - Sample until estimates are accurate for “near-maximizers” of marginal influence.
  - Add the approximate maximizer to seed set.
  - Update residual problem

Randomization handled using multiple MC simulations and optimizing for the average over simulations
Influence vs. #seeds: full approx greedy sequence
IC Model: Reach+ max aggregation + randomization

[CWPW ICDM 2014] data sets from SNAP
Implementation by T. Pajor (available)
Influence vs. #seeds: full approx greedy sequence
Distance utility with harmonic or exponential decay, max aggregation + randomization

[CWPW 2015] data sets from SNAP
Implementation by T. Pajor
Influence vs. #seeds: full approx greedy sequence
Reverse Rank Utility, threshold decay (with different T)

[Buchnik C’ Sigmetrics 2016] Live Journal data set from SNAP
Implementation by E. Buchnik (available)
Summary of contributions

- Unified model of graph-based influence functions
  - Influence functions specified by
    - pairwise *utility* values (reach/distance/reverse-rank/survival; decay function; randomized generation)
    - Submodular aggregation function of seeds utility
  - Meta-SKIM Algorithm: Compute an approximate greedy maximizing sequence using near linear computation for all functions

Follow up:
- Applications (seed selection for active learning, ...)
- Modular implementation
Thank you !!