Influence maximisation

Social and Technological Networks

Rik Sarkar

University of Edinburgh, 2016.

Maximise the spread of a cascade

- Viral marketing with restricted costs
- Suppose you have a budget of reaching k nodes
- Which k nodes should you convert to get as large a cascade as possible?

Classes of problems

- Class P of problems
  - Solutions can be *computed* in polynomial time
  - Algorithm of complexity O(poly(n))
  - E.g. sorting, spanning trees etc
- Class NP of problems
  - Solutions can be *checked* in polynomial time, but not necessarily computed
  - E.g. All problems in P, factorisation, satisfiability, set cover etc

Hard problems

- Computationally intractable
  - Those not (necessarily) in P
  - Requires more time, e.g. 2^n : trying out all possibilities
- Standing question in CS: is P = NP?
  - We don't know
- Important point:
  - Many problems are unmanageable
    - Require exponential time
    - Or high polynomial time, say: n^{10}
    - In large datasets even n^4 or n^3 can be unmanageable

Approximations

- When we have too much computation to handle, we have to compromise
- We give up a little bit of quality to do it in practical time
- Suppose the best possible (optimal) solution gives us a value of OPT
- Then we say an algorithm is a c-approximation
- If it gives a value of c*OPT

Examples

- Suppose you have k cameras to place in building how much of the floor area can your observation cover?
  - If the best possible coverage is A
  - A ¾ approximation algorithm will cover at least 3A/4

- Suppose in a network the maximum possible size of a cascade with k starting nodes is X
  - i.e a cascade starting with k nodes can reach X nodes
  - A ½-approximation algorithm that guarantees reaching X/2 nodes

Back to influence maximisation

- Models
- Linear contagion threshold model:
  - The model we have used: node activates to use A instead of B
  - Based on relative benefits of using A and B and how many friends use each

- Independent activation model:
  - If node u activates to use A, then u causes neighbor v to activate and use A with probability
    - $p_{u,v}$
    - That is, every edge has an associated probability of spreading influence (like the strength of the tie)

Hardness

- In both the models, finding the exact set of k initial nodes to maximize the influence cascade is NP-Hard
  - Intractable, unlikely that polynomial time algorithms exist unless P = NP

Approximation

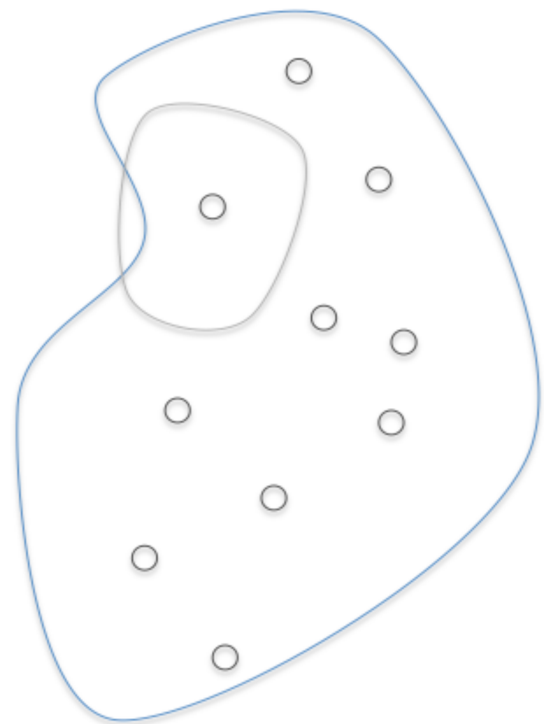
- OPT : The optimum result —the largest number of nodes reachable with a cascade starting with k nodes

- There is a polynomial time algorithm to select k nodes that guarantees the cascade will spread to $\left(1-\frac{1}{e}\right)\cdot OPT$ nodes

- To prove this, we will use a property called *submodularity*

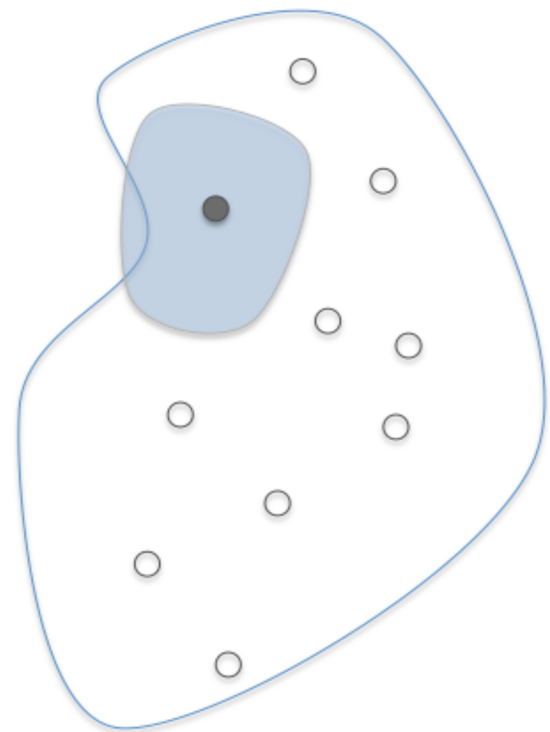
Example: Camera coverage

- Suppose you are placing sensors/cameras to monitor a region (eg. cameras, or chemical sensors etc)
- There are n possible camera locations
- Each camera can "see" a region
- A region that is in the view of one or more sensors is *covered*
- With a budget of k cameras, we want to cover the largest possible area
  - Function f: Area covered



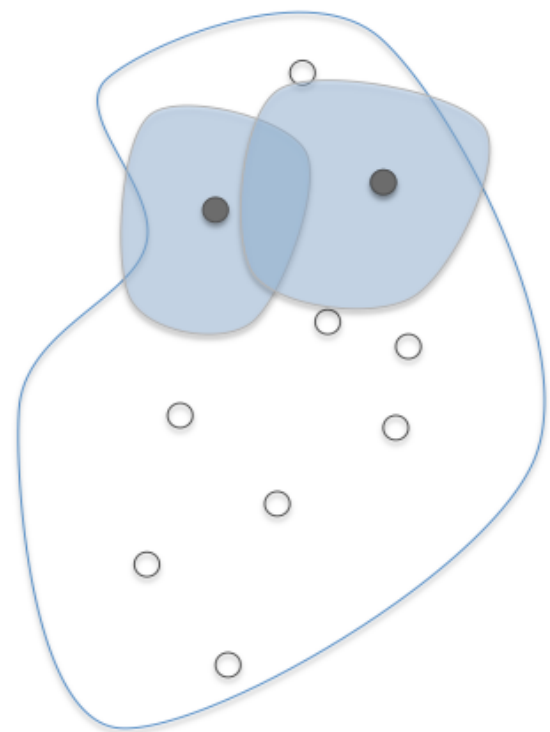
Marginal gains

- Observe:
- Marginal coverage depends on other sensors in the selection



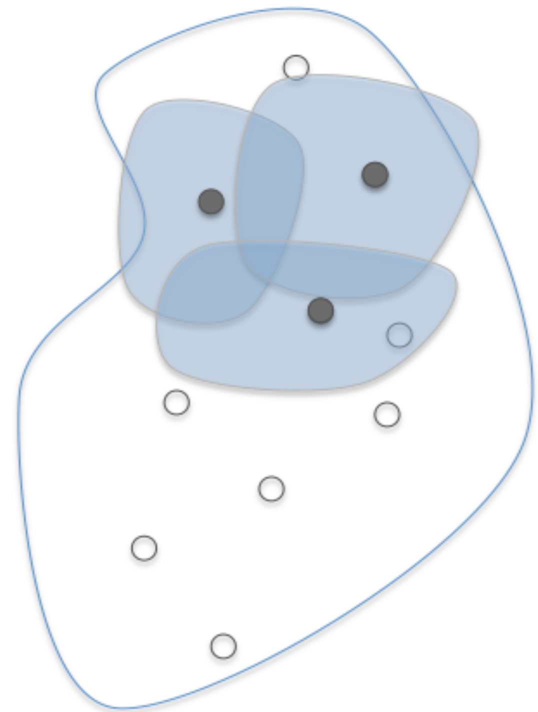
Marginal gains

- Observe:
- Marginal coverage depends on other sensors in the selection



Marginal gains

- Observe:
- Marginal coverage depends on other sensors in the selection
- More selected sensors means less marginal gain from each individual

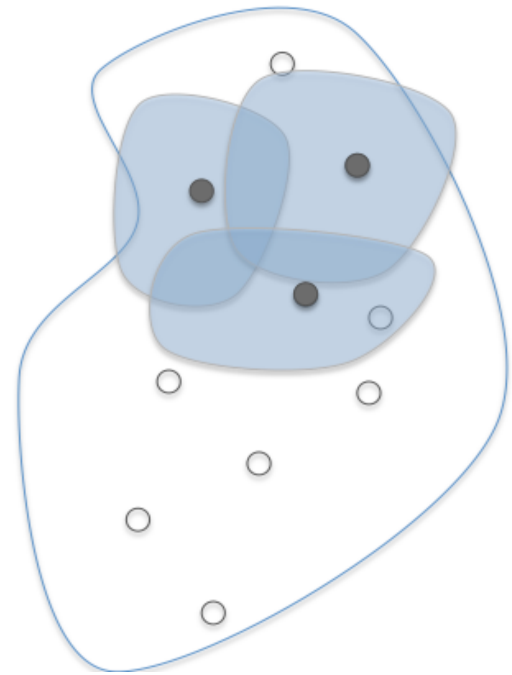


Submodular functions

- Suppose function f(x) represents the total benefit of selecting x
  - And f(S) the benefit of selecting set S
- Function f is submodular if:

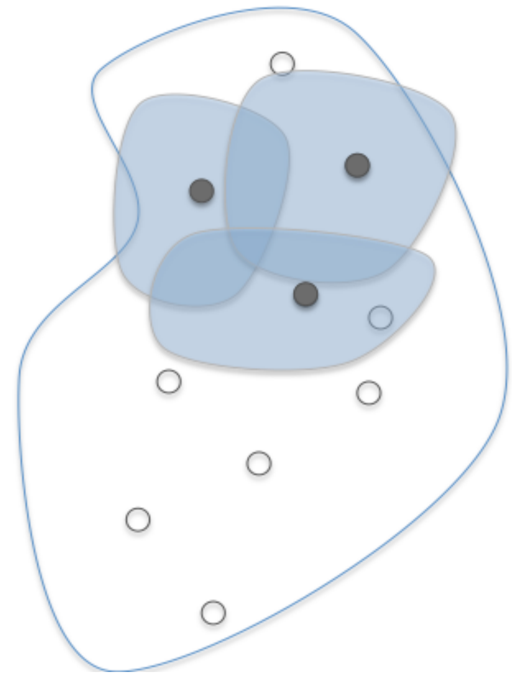
$$S \subseteq T \implies$$

$$f(S \cup \{x\}) - f(S) \geq f(T \cup \{x\}) - f(T)$$



Submodular functions

- Means diminishing returns
- A selection of x gives smaller benefits if many other elements have been selected

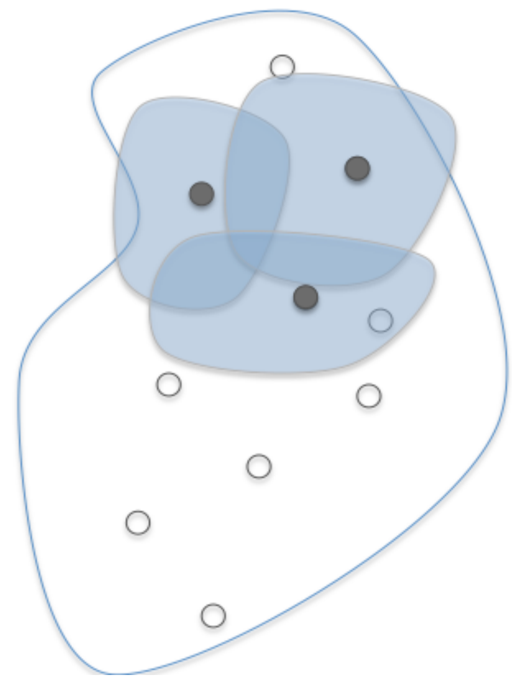


$$S \subseteq T \implies$$

$$f(S \cup \{x\}) - f(S) \geq f(T \cup \{x\}) - f(T)$$

Submodular functions

- Our Problem: select locations set of size k maximizes coverage
- NP-Hard



$$S \subseteq T \implies$$

$$f(S \cup \{x\}) - f(S) \geq f(T \cup \{x\}) - f(T)$$

Greedy Approximation algorithm

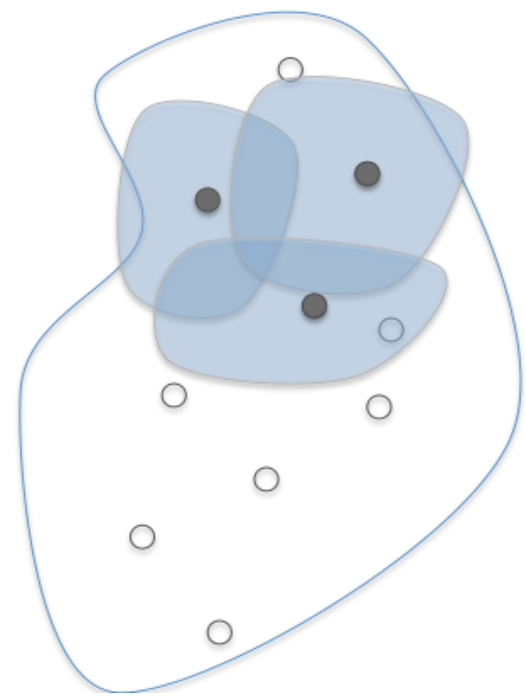
- Start with empty set S = $\varnothing$
- Repeat k times:
- Find v that gives maximum marginal gain:

$$f(S \cup \{v\}) - f(S)$$

- Add insert v into S

- Observation 1: Coverage function is submodular
- Observation 2: Coverage function is monotone:
- Adding more sensors always increases coverage

$$S \subseteq T \Rightarrow f(S) \leq f(T)$$



Theorem

- For monotone submodular functions, the greedy algorithm produces a $\left(1 - \frac{1}{e}\right)$ approximation

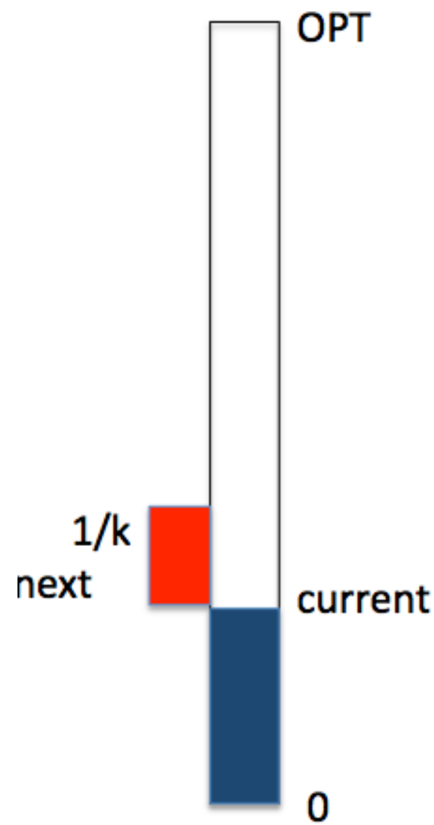
- That is, the value f(S) of the final set is at least

$$\left(1 - \frac{1}{e}\right) \cdot OPT$$

- (Note that this applies to maximisation problems, not to minimisation)

Proof

- Idea:
- OPT is the max possible
- On every step there is at least one element that covers 1/k of remaining:
- (OPT - current) * 1/k
- Greedy selects that element

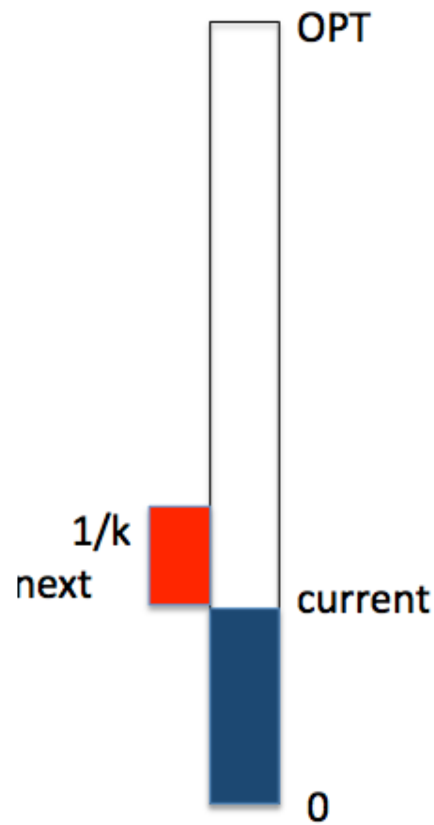


Proof

- Idea:
- At each step coverage remaining becomes

$$\left(1 - \frac{1}{k}\right)$$

- Of what was remaining after previous step



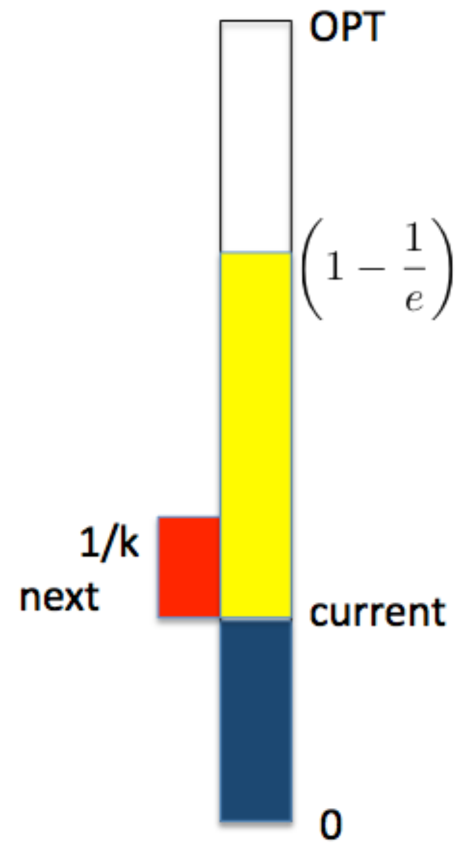
Proof

- After k steps, we have remaining coverage of OPT

$$\left(1 - \frac{1}{k}\right)^k \simeq \frac{1}{e}$$

- Fraction of OPT covered:

$$\left(1 - \frac{1}{e}\right)$$



- We have shown that monotone submodular maximization can be approximated using greedy selection

- To show that maximizing spread of cascading influence can be approximated:
  - We will show that the function is monotone and submodular

Cascades

- Cascade function f(S):
  - Given set S of initial adopters, f(S) is the number of final adopters
- We want to show: f(S) is submodular
- Idea: Given initial adopters S, let us consider the set H that will be the corresponding final adopters
  - H is "covered" by S

Cascade in independent activation model

- If node u activates to use A, then u causes neighbor v to activate and use A with probability
  - $p_{u,v}$
- Now suppose u has been activated
  - Neighbor v will be activated with prob. $p_{u,v}$
  - Neighbor w will be activated with prob. $p_{u,w}$ etc..
  - Instead of waiting for u to be activated before making the random choices, we can make the random choices beforehand
  - ie. if u is activated, then v will be activated, but w will not be activated… etc

Cascade in independent activation model

- We can make the random choices for u activation beforehand.
- Tells us which edges of u are “effective” when u is “on”
- Similarly for other nodes v, x, y ….
- We know exactly which nodes will be activated as a consequence of u being activated
- Exactly the same as “coverage” of a sensor/camera network
- Say, c(u) is the set of nodes covered by u.

- We know exactly which nodes will be activated as a consequence of u being activated
- Exactly the same as “coverage” of a sensor network
- Say, $c(u)$ is the set of nodes covered by u.
- $c(S)$ is the set of nodes covered by a set S
- $f(S) = |c(S)|$ is submodular

- Remember that we had made the probabilistic choices for each edge uv:
- That is, we made a set of choices representing the entire network
- Let us use x to represent this configuration
- We showed that given x, the function is submodular
- But what about other x?
  - Can we say that over all x we have submodularity?

- Now, we sum over all possible x, weighted by their probability.

- Non-negative linear combinations of submodular functions are submodular,
  - Therefore the sum of all x is submodular

- The approximation algorithm for submodular maximization is an approximation for the cascade in independent activation model with same factor

Linear threshold model

- Also submodular and monotone

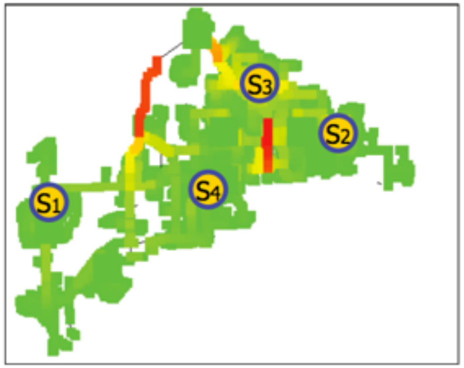
- Proof ommitted.

Applications of submodular optimization

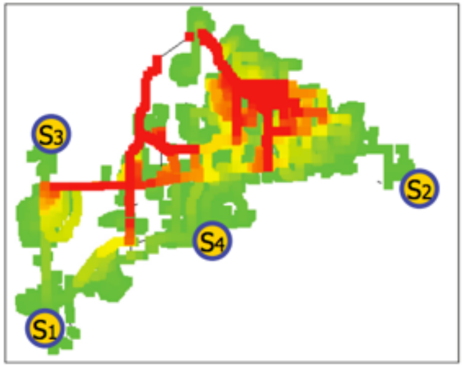
- Sensing the contagion
- Place sensors to detect the spread
- Find “representative elements”: Which blogs cover all topics?
- Machine learning
- Exemplar based clustering (eg: what are good seeds?)
- Image segmentation

Sensing the contagion

- Consider a different problem:
- A water distribution system may get contaminated
- We want to place sensors such that contamination is detected



(c) effective placement



(d) poor placement

Social sensing

- Which blogs should I read? Which twitter accounts should I follow?
  - Catch big breaking stories early
- Detect cascades
  - Detect large cascades
  - Detect them early…
  - With few sensors
- Can be seen as submodular optimization problem:
  - Maximize the “quality” of sensing

- Ref: Krause, Guestrin; Submodularity and its application in optimized information gathering, TIST 2011

Representative elements

- Take a set of Big data
- Most of these may be redundant and not so useful
- What are some useful "representative elements"?
  - Good enough sample to understand the dataset
  - Cluster representatives
  - Representative images
  - Few blogs that cover main areas…

Problem with submodular maximization

- Too expensive!
- Each iteration costs O(n): have to check each element to find the best
- Problem in large datasets
- Mapreduce style distributed computation can help
  - Split data into multiple computers
  - Compute and merge back results: Works for many types of problems

- Ref: Mirzasoleiman, Karbasi, Sarkar, Krause; Distributed submodular maximization: Finding representative elements in massive data. NIPS 2013.

Project

- Office hours:
- Every Wednesday 3:30 – 4:00.
- This Friday 11th Nov, 3pm – 5pm
- Simple tips about projects online on projects page
  - Take care in writing: What you write determines your grade