1 Problem definition

In this handout, we will consider the problem of how people influence each other to do things. For example, if your friend signs up for Facebook, you might want to sign up too, depending on how you are feeling that day and which friend it is.

We may think of this in terms of "activating" nodes in a social graph. A node that is "activated" adopts some behavior, like signing up for Facebook or buying an iPhone. Once each node is activated, it has the opportunity to activate each of its neighbors with a certain probability. This probability may depend on the edge and is usually given as an edge weight.

Example: Your mom really loves you, so whenever you sign up for a service, she signs up for it too. We can say that the edge from you to your mom has weight 1, and that once you are activated, she is guaranteed to be activated as well.

Example: Your friend Mary doesn't put much stock in your opinions, so the edge from you to Mary only has weight 0.1. If you sign up for Facebook, she has a 1 in 10 chance of signing up for Facebook because of you. However, Mary does think highly of your mom's opinions, and the edge from your mom to Mary has weight 0.8. So if your mom gets activated, then your mom will activate Mary with an 80% probability. And if both you and your mom get activated, then Mary gets two chances to be activated – one activation succeeds with probability 0.1, and the second activation succeeds with probability 0.8.

1.0.1 Influence maximization

How many nodes eventually get activated depends on which nodes were activated to begin with. (It's better to send an alpha version of your software to a tech journalist than a random guy on the street.) The **influence maximization** problem asks you to choose a "starting set" S of k nodes to activate that maximizes the expected number of nodes $f(S)$ that will eventually get activated in the network. These nodes are considered the most "influential" nodes in the graph.

1.0.2 Deterministic influence maximization

To solve this problem, we consider a version of the influence maximization problem where all the edge weights are guaranteed to be 0 or 1. (That is, we already know exactly what set of nodes an active node will activate.) We will see that we can use the solution to the deterministic influence maximization problem to solve the real influence maximization problem.

2 Influence maximization is NP-hard

As it turns out, even the deterministic influence maximization problem is NP-hard.

To prove it is NP-hard, we must find an example of an NP-hard problem that can be reduced to influence maximization. This would show that if we had a solution to the influence maximization problem, we could easily translate that into a solution to a problem that is known to be hard, which means influence maximization is a hard problem.

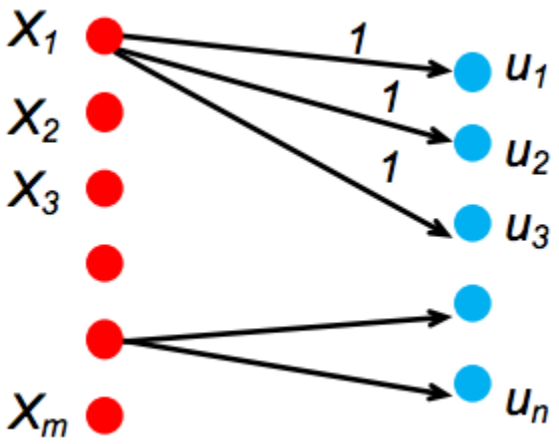
The NP-hard problem we will use is called Set Cover.

Set Cover: Given a universe of elements U and m subsets $X_1, \ldots, X_m$ of U, and a parameter k:

- Return YES if there exists a collection of k subsets whose union is U.
- Return NO otherwise.

To perform the reduction, we pretend we have a solution to the influence maximization problem. Then we start with an arbitrary instance of the Set Cover problem, and show that our solution can be used to solve the problem.

Reduction: Suppose we have an instance of the Set Cover problem, where the sets are $X_1, \ldots, X_m$, and each set X_i is composed of elements $u_{i1}, u_{i2}, \ldots, u_{in_i}$. We can then create a bipartite graph, where the sets are on the left and the items are on the right, and there is an edge from X_i to u_j of weight 1 if u_j is contained in set X_i. We then aim to maximize influence on this bipartite graph.



Observe that there exists a size k set cover in the Set Cover problem if and only if there exists a set S of size k that influences at least $k + n$ nodes in the corresponding influence maximization problem. Note that the optimal way to choose S would always involve picking nodes from the left side of the graph, since if we picked nodes from the right side of the graph, choosing its corresponding left hand neighbor instead could only increase the number of nodes influenced. And if you choose k left hand nodes and influence $k + n$ nodes, those left hand nodes correspond to a collection of sets that contain all the items u.

Therefore, by applying influence maximization to the graph with parameter k, we can solve the Set Cover problem with parameter k, and influence maximization is NP-hard.

3 Greedy approximation algorithm

We've discovered that it's hard to solve the influence maximization problem (even in the deterministic case). However, we can find an approximation algorithm for the influence maximization problem, that produces a set S that influences close to the optimal number of nodes.

(Let's switch back to discussing the probabilistic version of the influence maximization problem, since we've spent long enough discussing the deterministic version.)

Let $f(S)$ be the expected number of nodes influenced by set S, and $f(OPT)$ be the expected number of nodes influenced by the optimal set of nodes of size k. Then there is a greedy algorithm that finds a set S where

$$f(S) \geq (1 - 1/e)f(OPT)$$

Note that S does not have to be an optimal solution, but our theorem guarantees that the performance of S is close to the performance of the optimal solution, in expectation.

3.1 Algorithm

Build up our set S one node at a time, each time adding the node that increases $f(S)$ the most. Note that this doesn't mean we should always add the most well-connected nodes – it's more important to add nodes whose neighborhoods haven't been touched by other activations yet.

Notation: Let S_i denote the set S at iteration i (where $S_i = \emptyset$).

3.2 f is submodular

We will later show that our method works for all monotone, submodular functions f. However, first we should show that f is monotone and submodular, and to do that we should first define what those terms mean.

Definition (Monotone): If S is a subset of T, then $f(S) \leq f(T)$ and $f(\emptyset) = 0$. (This just means that if you start off with influencing a set of nodes T, then you'll end up influencing at least as many people as if you had influenced a subset of those nodes.)

Definition (Submodular): If S is a subset of T, then for any node u,

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

This means that adding a node to a set has less impact ("marginal gain") than adding the same node to a smaller subset of that set.

3.3 Proof that f is submodular

3.3.1 f can be written as a combination of simpler functions

Observe that since $f(S)$ is an expected value (i.e. the expected number of nodes influenced by set S), we can find it by simulating the activation process a bunch of times, counting the number of nodes activated, and taking the average of the results. In order to simulate the activation process, we can flip one (unfair) coin for each edge of the graph, and color the edge red if the coin comes up heads. After doing this, the nodes that are activated are precisely the ones reachable via red edges from the starting set S. (Note that we can flip all the coins prior to knowing what S is, if we want to.)

Let $i \in I$ be a combination of coin flips, and $f_i(S)$ be the number of nodes that are activated by S given the outcome of those coin flips. Then $f(S) = \frac{1}{|I|} \sum_{i \in I} f_i(S)$.

3.3.2 Each function f_i is submodular

Observe that each function $f_i(S)$ is submodular. This is because $f_i(S)$ can be expressed as $|\cup_{v \in S} X_v^i|$, where X_v^i is the set of nodes that are reachable from node v along the red paths from coin-flip combination i, and adding another set to this union (i.e. adding another element to S) will give you a smaller marginal gain than if you had added the same set to a subset of that union.

3.3.3 f is submodular

Because f is a positive linear combination of submodular functions, f is submodular.[1]

3.4 Proof that submodular functions work

Now we show that because f is monotone and submodular, after our greedy procedure we will have $f(S) \geq (1 - 1/e) f(OPT)$.

3.4.1 Lemma: If $B = \{b_1, \ldots, b_k\}$, then $f(A \cup B) - f(A) \leq \sum_{j=1}^{k} [f(A \cup \{b_j\}) - f(A)]$

Let B_i be the set containing the first i elements of B, i.e. $\{b_1, \ldots, b_i\}$ (and $B_0 = \emptyset$). Now we have k such sets, $B_1, B_2, \ldots, B_k$.

Now we can write $f(A \cup B) - f(A)$ as a telescoping sum, i.e.

$$(f(A \cup B_1) - f(A)) + (f(A \cup B_2) - f(A \cup B_1)) + \cdots + (f(A \cup B_k) - f(A \cup B_{k-1}))$$

[1] See https://en.wikipedia.org/wiki/Submodular_set_function#Properties

Observe that because f is submodular, $f(A\cup B_{i-1}\cup\{b_i\})-f(A\cup B_{i-1})\leq f(A\cup\{b_i\})-f(A)$. Therefore, we have

$$\begin{aligned} f(A\cup B)-f(A) &= \sum_{i=1}^{k}[f(A\cup B_i)-f(A\cup B_{i-1})] \\ &= \sum_{i=1}^{k}[f(A\cup B_{i-1}\cup\{b_i\})-f(A\cup B_{i-1})] \\ &\leq \sum_{i=1}^{k}[f(A\cup\{b_i\}]-f(A)] \end{aligned}$$

proving the lemma.

3.4.2 $\delta_{i+1}\geq(1/k)(f(OPT)-f(S_i)))$

Define δ_i to be the marginal gain at step i, i.e. $\delta_i=f(S_i)-f(S_{i-1})$. Then we want to get a lower bound on δ_i, so we can sum up all the δ's and get a lower bound on how good our greedy approach is. We adopt an analysis strategy where we replace elements of the optimal solution with elements of the greedy solution and see how that affects the marginal gain.

Suppose the optimal solution OPT is $\{t_1,t_2,\ldots,t_k\}$. Then

$$\begin{aligned} f(OPT) &\leq f(S_i\cup OPT) && \text{(monotonicity)} \\ &= f(S_i\cup OPT)-f(S_i)+f(S_i) && \text{(adding and subtracting terms)} \\ &\leq \sum_{j=1}^{k}[f(S_i\cup\{t_j\})-f(S_i)]+f(S_i) && \text{(by previous lemma)} \\ &\leq \sum_{j=1}^{k}[f(S_{i+1})-f(S_i)]+f(S_i) \end{aligned}$$

because S_{i+1} is produced by choosing the element that maximizes the marginal gain. So we have $f(OPT)\leq\sum_{j=1}^{k}\delta_{i+1}+f(S_i)=f(S_i)+k\delta_{i+1}$, and

$$\delta_{i+1}\geq\frac{1}{k}[f(OPT)-f(S_i)]$$

3.4.3 $f(S_{i+1}) \geq (1-1/k)f(S_i) + (1/k)f(OPT)$

We have

$$\begin{aligned} f(S_{i+1}) &= f(S_i) + \delta_{i+1} \\ &\geq f(S_i) + \frac{1}{k}[f(OPT) - f(S_i)] \\ &\geq \left(1 - \frac{1}{k}\right) f(S_i) + \frac{1}{k} f(OPT) \end{aligned}$$

3.4.4 $f(S_i) \geq [1-(1-1/k)^i]f(OPT)$ for all i

We proceed by induction.

Base case: For $i = 0$, $f(S_0) = f(\emptyset) = 0$, and the right hand side is also 0.

Inductive step: Assume the statement is true for S_i, and prove it is true for S_{i+1}. At $i+1$ we have

$$\begin{aligned} f(S_{i+1}) &\geq \left(1 - \frac{1}{k}\right) f(S_i) + \frac{1}{k} f(OPT) \\ &\geq \left(1 - \frac{1}{k}\right) \left(1 - \left(1 - \frac{1}{k}\right)^i\right) f(OPT) + \frac{1}{k} f(OPT) \qquad \text{by the induction hypothesis} \\ &= \left[1 - \left(1 - \frac{1}{k}\right)^{i+1}\right] f(OPT) \end{aligned}$$

which proves the lemma.

3.4.5 $f(S_k) \geq (1-1/e)f(OPT)$

From the previous lemma, we get that $f(S) = f(S_k) \geq [1-(1-1/k)^k]f(OPT)$.

Now we can apply the inequality $1 + x \leq e^x$. We have $x = -1/k$, so $(1-1/k)^k \leq (e^{-1/k})^k = 1/e$. Substituting this in, we get that $f(S) \geq (1-1/e)f(OPT)$.

4 Conclusion

Therefore, we have found a lower bound on the efficacy of our greedy approach. The greedy approach works pretty well in practice compared to other sensible methods of influence maximization.