Université Paris Dauphine-PSL

Département MIDO

Maximizing the Spread of Influence through a Social Network

Implementation of an approach

Haithem BEN DRISSI

MASTER MODO

Realized at the workshop: Démarches, modèles et procédures d'aide à la decision.

Maximizing the Spread of Influence through a Social Network

Haithem BEN DRISSI
Université Paris Dauphine-PSL

Abstract

In this work we present the state of art in the subject of Maximizing the Spread of Influence through a Social Network, the greedy algorithm to select the most influencer elements and the cascade model to spread the information. Finally, we introduce the implementation of an approach based on the greedy algorithm and cascade model. We proceed with two layers, the first will evaluate the potential influence of each individual or element in the network and select the best k individuals, and in the second layer we will use the cascade model to spread the information starting from the result given by the greedy algorithm. We obtain finally after doing the simulation on an example of social network represented by an oriented graph the list of influenced individuals.

1 Introduction

As part of my MODO master's degree at Paris Dauphine-PSL University, in the DA approaches, models and procedures workshop, I am required to do the state of art in the subject of Maximizing the Spread of Influence through a Social Network and implement an approach to do the simulation of it. A bibliographic study on max influence topic, the greedy algorithm, and the Cascade model is presented in this work.

A social network—a diagram of the relationships and interactions among a set of individuals—plays a fundamental role as a medium for spreading information, ideas, and influence among its members. An idea or innovation will emerge—for example, college students using cell phones, the introduction of new drugs in medicine, or the rise of a political movement in an unstable society—and it can die quickly or have a major impact. impact on the public [1].

Suppose we have data on a social network that can estimate the degree to which individuals influence each other, and we want to market a new product that we hope will be adopted by the majority of the network. The premise of viral marketing is that by first targeting some "influencer" members of the network - for example, by giving them free product samples - we can trigger an influence cascade through which friends will recommend products Give it to other friends and eventually many will try it. But how should we choose a few key people with whom we start the process?

The optimization problem of selecting the most influential nodes is NP-hard. The generality of the models we consider lies between that of the polynomial-time solvable model of [2] and the very general model of [3], where the optimization problem cannot even be approximated to within a non-trivial factor.

In this work, we solve the problem of Maximizing the Spread of Influence through a Social Network using the greedy algorithm to obtain k vertices to activate initially that maximizes the influence after cascade model execution to spread the information in a social network.

The first part of this work would relate to the description of the problem, the state of art, and the explanation of the proposed algorithm. The second will focus on the implementation of the latter.

2 Diffusion Models

We refer to each individual node as either being active (an innovation adopter) or inactive when discussing operational models for the diffusion of an idea or innovation through a social network G, represented as a directed graph. Each node’s tendency to become active increases monotonically as more of its neighbors become active.

Therefore, from the perspective of an initially inactive node v, the process will essentially look like this: as time passes, more and more of v's neighbors will become active; eventually, this may cause v to become active, and v's decision may in turn cause other nodes to which v is connected to make decisions.

Granovetter and Schelling were among the first to propose models that capture such a process; their approach was based on the use of node-specific thresholds [4, 5]. Many models of this flavor have since been investigated (see e.g. [6, 7, 8, 9, 10, 11, 12, 13, 14]).

2.1 Linear Threshold Model

The following Linear Threshold Model lies at the core of most subsequent generalizations. In this model, a node v is influenced X by each neighbor w according to a weight $b_{v,w}$ such that $\sum_{w \text{ neighbor of v}} b_{v,w} \leq 1$. Following that, the process' dynamics go as follows. In order, for a node to become active, a threshold must be reached by a weighted percentage of its neighbors. Each node v selects its threshold uniformly at random from the range [0, 1]. The diffusion process develops deterministically in discrete steps given a random selection of thresholds and an initial set of active nodes A_0 (with all other nodes inactive). In step t, all active nodes from step $t + 1$ remain active, and we activate any node v for which the sum of its active neighbors' weights is at least θ_v :

$$\sum_{w \text{ neighbor of v}} b_{v,w} \geq \theta_v \qquad (1)$$

Thus, the thresholds θ_v intuitively represent the different latent tendencies of nodes to adopt the innovation when their neighbors do [1].

2.2 Independent Cascade Model

Another type of diffusion models are the dynamic cascade models for diffusion processes, which are based on work on interacting particle systems [15, 16] from probability theory. The Independent Cascade Model, which Goldenberg, Libai, and Muller recently studied in the context of marketing [17, 18], is the conceptually simplest model of this kind.

Again, the process begins with an initial set of active nodes A_0 and proceeds in distinct steps in accordance with the subsequent randomized rule. Node v is given a single opportunity to activate each of its inactive neighbors w when it first becomes active in step t. If it is successful, it does so with a probability $p_{v,w}$ — a parameter of the system — independent of previous history. The efforts are sequenced in any order (if w has several freshly activated neighbors). If v is successful, w will become active in step t + 1; nevertheless, regardless of whether v is successful, it is not possible for it to try again in following rounds. Up until there are no more activations left possible, the process repeats.

The Linear Threshold and Independent Cascade Models are two of the most basic and widely studied diffusion models.

3 Algorithms for Influence Maximization.

Now that we have the models mentioned above, we can formally state the optimization problem of selecting a good initial selection of nodes to target. There is an initial group of active nodes A_0 that initiates the diffusion process in both the Linear Threshold and Independent Cascade Models (as well as the generalizations that follow). Given that A is this initial active set A_0, we define the effect of a set of nodes A, abbreviated (A), as the anticipated number of active nodes at the conclusion of the procedure. Finding a k-node set with maximum influence for a parameter k is the goal of the influence maximization problem. (When dealing with algorithms for this problem, we will say that the chosen set A of k initial active nodes has been targeted for activation by the algorithm.).

Finding the optimal for influence maximization for the models we study is NP-hard. In both the Linear Threshold and Independent Cascade models, we find that the best solution for influence maximization can be efficiently estimated to within a factor of (1 − 1/e − ε), where e is the base of the natural logarithm and is any positive real number. This is a performance guarantee that is marginally higher than 63 percent. The analysis framework required to obtain a provable performance guarantee and the somewhat unexpected fact that hill-climbing is always within a factor of at least 63 percent of optimal for this problem make up the main content of this result. The algorithm that achieves this performance guarantee is a natural greedy hill-climbing strategy related to the approach considered in [3].

In the following we present the model used in our implementation approach as well as the algorithm used to select the set of vertices to activate initially in order to have maximum influence.

4 Cascade model

Consider a point in the cascade process where node v has just started to become active and attempts to activate its

neighbor w, with a probability of success of $p_{v,w}$. We can think of this random event's outcome as being decided by tossing a biased coin ($p_{v,w}$). From the perspective of the process, it is obvious that it makes no difference whether the coin was flipped immediately after v became active or at the very beginning of the entire process and is only now coming to light. Following along with this logic, we can actually assume that for each pair of neighbors (v, w) in the graph, a coin of bias $p_{v,w}$ is flipped at the start of the process (independently of the coins for all other pairs of neighbors), and the result is stored so that it can be later verified in the event that v is activated while w is still inactive.

The procedure can be seen as follows once all the coins have been flipped beforehand. The edges in G for which a successful activation is predicted by the coin flip are deemed to be live, while the other edges are deemed to be blocked. It is simple to figure out the complete set of active nodes at the conclusion of the cascade process if the coin flip results are fixed and then a set A is first activated.

A node x ends up active if and only if there is a path from some node in A to x consisting entirely of live edges.

Consider of the probability space where each sample point indicates one potential outcome for every edge coin flip.

Let X stand for one sample point in this space. X(A) is the total number of nodes activated by the procedure where A is the set initially targeted, and X is the set of results from all coin flips on edges. In reality, X(A) is a deterministic quantity because we have predetermined a choice for X, and there is a straightforward way to describe its value as follows. The set of all nodes that can be reached from v on a path made up only of live edges is denoted by the notation R(v, X).

$\sigma_X(A)$ is the number of nodes that can be reached on live-edge paths from any node in A, and so it is equal to the cardinality of the union $\cup_{v \in A} R(v, X)$.

The influence maximization problem is NP-hard for the Independent Cascade model.

By considering the operational models for the diffusion of an idea or an innovation through a social network G, represented by a directed graph, we will say that each individual node is either active (adopting the innovation) or inactive [1].

The Cascade model algorithm is presented as follow:

Algorithm 1 Cascade Model

Input:

- G = (V, E)
- Subset of vertices S ⊆ V: the initially activated vertices (influencers)
- All arcs are initially "unflipped"

Output: Set of influenced vertices

1: **While** there is an active vertex v ∈ S and an unflipped (v, w) arc **do**
2: **foreach** arc (v, w) **do**
3: Toss heads or tails with probability p;
4: **if** "head" :
5: arc(v,w) becomes active;
6: node w becomes active;
7 **else :**
8: arc(v,w) becomes inactive;
9: **end if.**
10: **end.**
11: **end.**

- Example:

Iteration1:

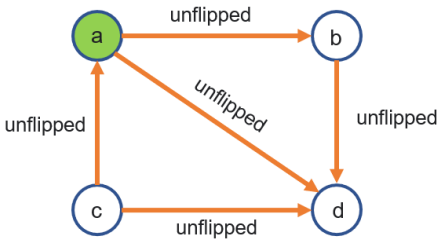

Figure 1: Cascade model - Iteration 1

G = (V, E)

- ✓ Nodes ∈ V \ S, are inactive.
- ✓ The node a ∈ S, is therefore initially active.
- ✓ All arcs are unflipped.

Iteration 2:

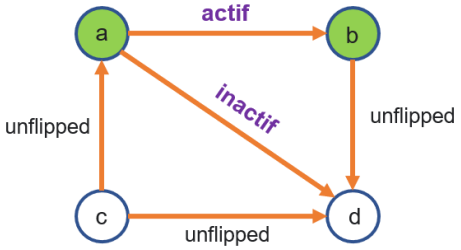

Figure 2: Cascade model - Iteration 2

- ✓ The unflipped arcs exiting through active nodes are (a,b) and (a,d).
- ✓ With a probability p, we draw heads or tails for these arcs.
- ✓ In this iteration we obtained:
  head for (a, b) => arc active => node b activated
  tails for (a, d) => arc inactive.

Iteration 3:

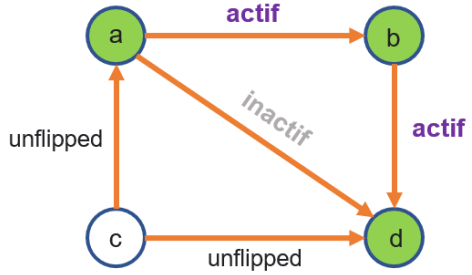

Figure 1: Cascade model - Iteration 3

- ✓ The unflipped arc exiting through an active node is (b, d).
- ✓ With a probability p, we draw heads or tails for this arc.
- ✓ In this iteration we obtained head for this arc => active arc => active node.
- ✓ No outgoing arc through an active node is unflipped.
- ✓ Starting with node 'a' as active, after iterating, we made nodes 'b' and 'd' active.

5. Maximum influence problem

In this part we present the algorithm to solve the maximum influence problem. The result after the execution of this algorithm are the k vertices to activate initially before the cascade model application and allowing to maximize the influenced vertices at the end of the application.

Let:

- S: the set of vertices to initially activate (Seed vertices)
- X(S): the set of vertices activated from S (random) $f_{inf}(S) = E(X(S))$

Having in input a graph G = (V, E), a probability p, and a number k ∈ ℕ representing the number of vertices initially activated. We expect an output of a Subset S ⊆ V of cardinality k that maximizes $f_{inf}(S)$.

The maximum influence problem is NP-hard, close to the Maximum coverage problem.

In Maximum coverage problems, we have in input Collection of sets $A_1, A_2, \dots, A_n$ of a universe U called the ground set, and a number K ∈ ℕ representing the number of sets to choose. In output, we should get the Choice of K indices that maximizes the cover function

$$f_{\text{cov}}(K) = |\bigcup_{i \epsilon K} A_i| \qquad (2)$$

To solve our initial problem the paper [1] proposes a greedy algorithm used in the resolution of the Maximum coverage problem.

Theorem: The Greedy algorithm returns a set S to $1 - (1 - \frac{1}{k})^k$ fractions of the maximum possible.

It is a greedy algorithm; we will build step by step the set S at each iteration.

The influence function is mathematical expectation, represents the average influence because we cannot calculate the function f_{inf} in polynomial time and this function will therefore be approximate.

The greedy algorithm guarantees a performance of $1 - (1 - \frac{1}{k})^k$ of the maximum possible.

The greedy algorithm is presented as follow:

Algorithm 1 Greedy algorithm

Input:

- G = (V, E)
- Probability p
- k ∈ ℕ: number of vertices to generate

Output: Subset S ⊆ V of cardinality k

1: **S := Ø**
2: **for** j from 1 to k **do:**
3: #maximize the increase in influence
4: Choose vertex v* s.t:
v* = $\underset{v \in V}{\operatorname{argmax}}\{f_{\inf}(S \cup \{v\}) - f_{\inf}(S)\}$
5: S := S ∪ {v*}
6: **end.**
7 **return S**
8: **end.**

To calculate $f_{\inf}(S) = \mathrm{E}(\mathrm{X}(\mathrm{S}))$, it is necessary to launch a simulation of arc activation over several executions, which will allow each time to have a set H and a number $|f_{inf}(S \cup \{v_i\}) - f_{inf}(S)|$. Finally take the average of this number over the whole simulation for each vertex.

The intuition behind using the greedy algorithm

The intuition behind the use of the greedy algorithm used to solve the maximum coverage problem in the maximum influence problem:
If consider H ⊆ E: Set of active arcs, in this case:

Max Influence Problem = Max Coverage Problem

We can thus reduce the influence problem on the graph H to a coverage problem defined by:

- Ground set: V
- For each vertex v ∈ V:
  $A_{\mathrm{v,\,H}}$: set of vertices influenced from v via a graph H fixed according to the probability p.

On the graph H we apply the greedy algorithm to determine the best set S maximizing the influence.
We will therefore have the coverage function to be maximized for each H:

$$f_{\mathrm{H}}(S) = |\bigcup_{v \in S} A_{\mathrm{v,\,H}}| \quad (3)$$

Finally, the influence function is the weighted sum over all possible graphs given by the probability p:

$$f_{\inf}(\mathrm{S}) = E_{\mathrm{H}}(f_{\inf}(\mathrm{S})) = \sum_{h \subseteq H}(p_{\mathrm{h}}\, f_{\mathrm{h}}) \quad (4)$$

With: p_h : Probability of activating h.

After presenting the model responsible for information propagation and the algorithm giving the best k elements to active initially in order to influence a maximum number of vertices. In the following section we present the implementation of the approach composed of the cascade model and the greedy algorithm. Thereafter we present the results of simulations on some graphs.

6 Implementation of the approach based on greedy algorithm and cascade model

Our experiments are executed on Google colab, on 0,90GB memory and one CPU of 2.30GHZ under the Linux operating system and using the programming language Python.

6.1 Implementation of the greedy algorithm

We start with the implementation of the greedy algorithm, by implementing some functions necessary for the execution of the algorithm. Here is the list of prepared functions:

- ***graph_arcs(graph):*** Takes in input the graph composed of vertices that represent the individuals and the arcs representing the connection between these individuals. This function returns the set of arcs in the graph.
- ***active_arcs(arcs):*** This function returns a list H of all the possible sets of active arcs in the graph according to a probability p_u. Every list in H is a possible scenario of arcs activation
- ***find_all_paths(graph, start, end, path=[]):*** this function returns the possible paths from a given graph, starting from one vertex to reach another vertex.
- ***active_graph(H):*** This function returns the active graph generated from active arcs. The H in the input is a list of active arcs. The output is the actual propagation graph composed of active arcs.

- ***vertices_influence(propagation_graph):*** This function returns the influence of each vertex. The output is a list of tuples composed of a vertex and the number of vertices it can influence.
- ***biggest_infl_to_S(vertices_infl, S):*** this function add to S the most influencer vertex and return the set S.
- ***S_infl(S, vertices_infl):*** This function returns the number of vertices in the graph influenced by the vertices in S.
- ***S_infl_U_v(S, v, vertices_infl):*** This function returns the number of vertices in the graph influenced by the vertices in S union a vertex v influence.
- ***greedy_alg(k):*** Taking in input the number k of vertices we want to activate initially; this function returns the k vertices maximizing the influence as list of seed vertices with its influence result.

6.2 Implementation of the cascade model

In this part, using the initial graph, we select the arcs coming out of the vertices in S which are initialized as "unflipped", and according to a probability p we decide to activate or not each of the arcs. Here is the list of prepared functions:

- ***graph_of_selected_vertices(graph, S):*** This function returns the graph containing only arcs and vertices composed using the returned list S by the greedy algorithm.
- ***cascade_model(graph, S):*** In this function we execute the cascade model on the graph starting by activating the vertices in the list S. The function returns the list of active vertices after executing the cascade model.

6.3 Combination of Greedy algorithm and Cascade model

Each combination of arcs activation generates a propagation graph for the information to pass. By applying the greedy algorithm on each combination, we obtain the k vertices which will have a maximum influence on the execution of the cascade model. On all combinations of activation, we calculate the average or the total sum of influence of each vertex. The k vertices having the average, or the highest score will be selected to be activated initially for the execution of the cascade model. The execution of the cascade model will return all the vertices activated after the propagation of the information.

The whole implementation of this approach for is presented in this GitHub repository: https://github.com/haith-gi/Maximizing-the-Spread-of-Influence-through-a-Social-Network

6.4 Results of some simulation examples

In this part we present the results of some executions of the proposed approach. below is a table containing the number of vertices of each graph, the number of vertices activated by the greedy algorithm, and the final result which corresponds to the total number of activated vertices after the execution of the cascade model starting from the k vertices proposed by the greedy algorithm.

Number of vertices in the graph	Number of vertices: k	Number of final activated vertices
6	2	5
6	2	4
7	2	7
7	2	5
7	2	6
8	2	2
8	2	7

The few executions of the example graphs show that we could have cases where the number of vertices influenced by the vertices initially activated is equal to zero if no arc will be activated in the cascade propagation model since their activation is controlled by a probability of 0.5. It is also obvious that adding a vertex to the graph with links with other vertices there is a remarkable increase in execution time, and this is normal because we are executing a greedy algorithm that is building step by step the set S at each iteration.

7 Conclusion

In this work, we presented the problem of Maximizing the Spread of Influence through a Social Network using the greedy algorithm to obtain k vertices to activate initially that maximizes the influence after cascade model execution to spread the information in a social network.

The first part of this work was related to the description of the problem, the state of art, and the explanation of the proposed algorithm. We also presented an implementation of the approach.

References

[1] David Kempe, Jon Kleinberg, Eva Tardos. Maximizing the Spread of Influence through a Social Network. SIGKDD '03 Washington, DC, USA, 2003.

[2] M. Richardson, P. Domingos. Mining Knowledge-Sharing Sites for Viral Marketing. Eighth Intl. Conf. on Knowledge Discovery and Data Mining, 2002.

[3] P. Domingos, M. Richardson. Mining the Network Value of Customers. Seventh International Conference on Knowledge Discovery and Data Mining, 2001.

[4] M. Granovetter. Threshold models of collective behavior. American Journal of Sociology 83(6):1420-1443, 1978.

[5] T. Schelling. Micromotives and Macrobehavior. Norton, 1978.

[6] E. Berger. Dynamic Monopolies of Constant Size. Journal of Combinatorial Theory Series B 83(2001), 191-200.

[7] M. Macy. Chains of Cooperation: Threshold Effects in Collective Action. American Sociological Review 56(1991).

[8] M. Macy, R. Willer. From Factors to Actors: Computational Sociology and Agent-Based Modeling. Ann. Rev. Soc. 2002.

[9] S. Morris. Contagion. Review of Economic Studies 67(2000).

[10] D. Peleg. Local Majority Voting, Small Coalitions, and Controlling Monopolies in Graphs: A Review. 3rd Colloq. on Structural Information and Communication, 1996.

[11] T. Valente. Network Models of the Diffusion of Innovations. Hampton Press, 1995.

[12] D. Watts. A Simple Model of Global Cascades in Random Networks. Proc. Natl. Acad. Sci. 99(2002), 5766-71.

[13] H. Peyton Young. The Diffusion of Innovations in Social Networks. Santa Fe Institute Working Paper 02-04-018(2002).

[14] H. Peyton Young. Individual Strategy and Social Structure: An Evolutionary Theory of Institutions. Princeton, 1998.

[15] R. Durrett. Lecture Notes on Particle Systems and Percolation. Wadsworth Publishing, 1988.

[16] T.M. Liggett. Interacting Particle Systems. Springer, 1985.

[17] J. Goldenberg, B. Libai, E. Muller. Talk of the Network: A Complex Systems Look at the Underlying Process of Word-of-Mouth. Marketing Letters 12:3(2001), 211-223.

[18] J. Goldenberg, B. Libai, E. Muller. Using Complex Systems Analysis to Advance Marketing Theory Development. Academy of Marketing Science Review 2001.