POLITECNICO
MILANO 1863

Project of Data Intelligence Application

Matteo Costantino (10494637)

Part 4 - Social Influence

Index

Project Description

1. Imagine three social networks with, respectively, 10^2, 10^3, and 10^4 nodes. Every edge can be characterised by 4 features and the probability of the edge is given by a linear combination of the features. Assign a cost of every node and define a reasonable budget for the social influencer — we suggest using 3 different values of the budget for the 3 social networks.

2. Apply, to the three scenarios, the greedy algorithm to solve approximately the influence maximization problem in the case in which the probabilities of the edges are known. In doing that, use Monte Carlo simulations and show how the value of the objective function varies for different values of the approximation bound.

3. Assume that the probabilities of the edges are not known. Apply: combinatorial bandit algorithms (TS- and UCB1-like) in the case the linear structure of the probabilities is not exploited (and therefore the probability of every edge is estimated by using only samples related to that edge), the UBC1-like algorithm exploiting the linear structure of the probabilities of the edge. Show how the expected regret varies as the number repetitions of the problem increases.

4. Modify the social networks to assure that every node has at most 15 neighbors. Apply a UCB1-like algorithm when edges cannot be observed and have to be estimated from data. Show how the expected regret varies as the number repetitions of the problem increases.

Graph construction

A social network is graph defined with this model:

- Every node is a user
- Each user can be of different states (susceptible, active, inactive)
- The arc of the graph are connection in the social network
- Each arc has his probability (influence)
- Time is discrete
- If a node a becomes active at time t, then at time t+1 it can activate neighbour b with probability p
- If node b does not activate at t+1, then node a cannot activate node b in future

A seed is an active node that tries to influence neighbours and start a cascade. A social influencer uses his budget to buy seeds that will influence the network. The goal of this project is to implement algorithms that maximize influence given a social network and a budget.

We create 6 different social networks:

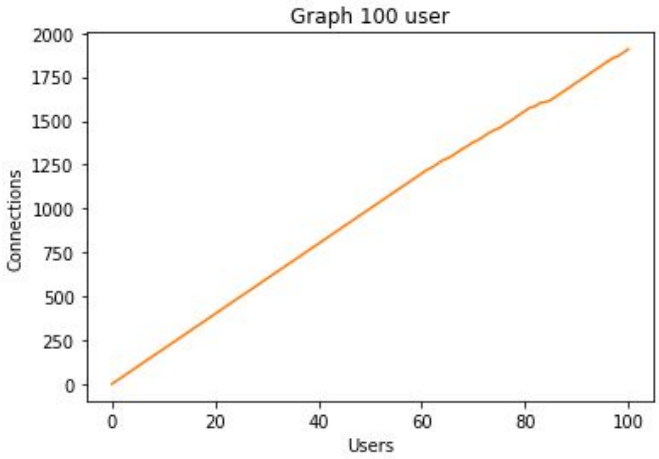
- 3 with respectively 100, 1000, 10000 users and no constraints (only realistic ones)
- 3 with respectively 100, 1000, 10000 users and max 15 neighbours

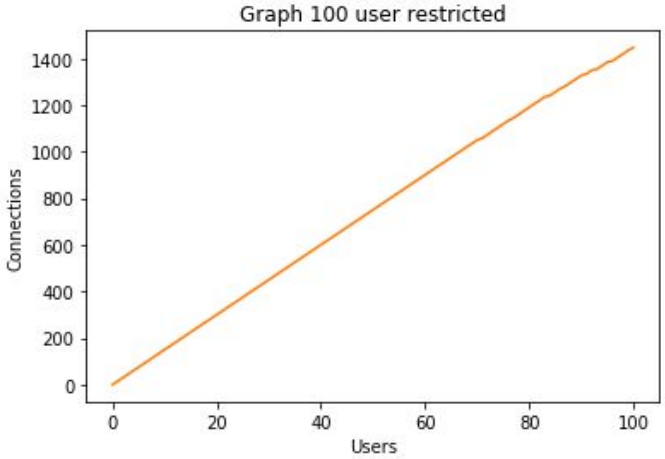
We know that each connection between users is reciprocate (if a is connected to b then b is connected to a), but the probabilities are different. Also the probabilities are linear combination of 4 features.

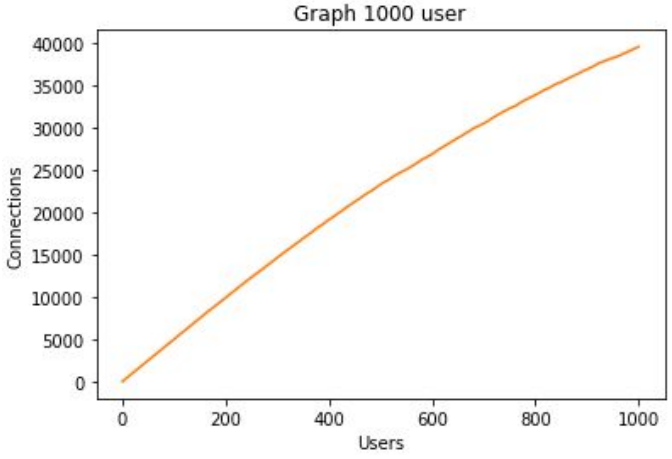
In our project:

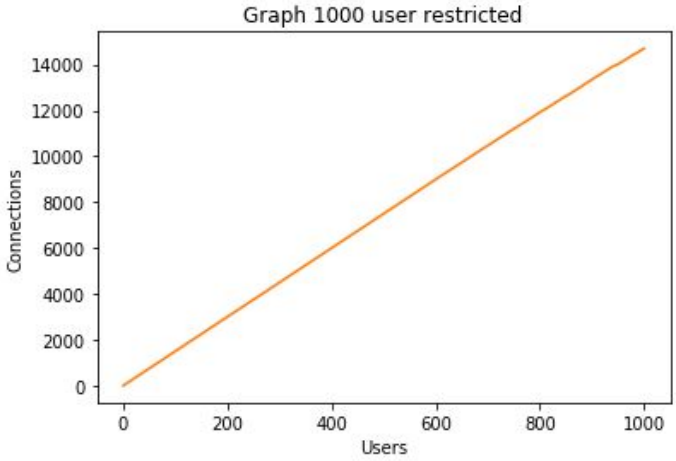
Features weights = [0.2, 0.55, 0.1, 0.15]

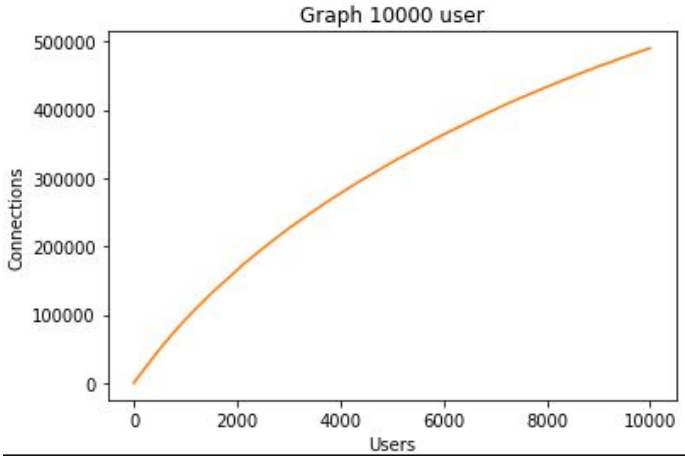
This are the generated graphs:

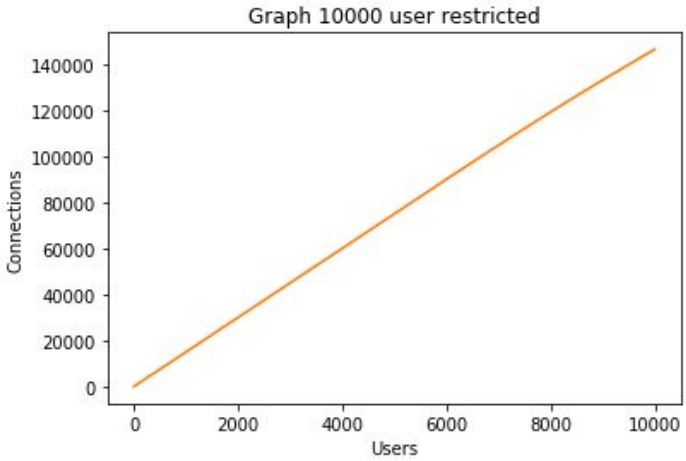












Greedy algorithm

We want to maximize the influence on the network. To do this we need to find the best subset of nodes (seeds) that fits in our budget and influences the maximum number of nodes.

To accomplish this task we use a greedy algorithm with Montecarlo approximations.

Montecarlo sampling

1. For every node *i* assign $z_i = 0$
2. Generate randomly a live-edge graph according to the probability of every edge
3. For every node that is active due to the generated live-edge graph, assign $z_i = z_i + 1$
4. Go to Step 1 unless *k* repetitions have been done
5. For every node return z_i / k

To find the number of repetition k, we can use this property of the algorithm:

With probability 1-d the estimated activation probability of every node is subject to an additive error of +-e*n when the number of repetitions is K = $O\left(\frac{1}{\epsilon^2}\log(|S|)\log\left(\frac{1}{\delta}\right)\right)$, where S is the set of seeds and n is the number of nodes.

Greedy algorithm

A simple greedy algorithm provides a constant approximation of 1 - 1/e ~= 0.63

The algorithms has this step:

- For every node that is not a seed, evaluate the marginal increase in the objective function if that node were a seed
- Select the node for which the marginal increase is maximised and add it to the set of seeds
- Repeat until you have some budget left

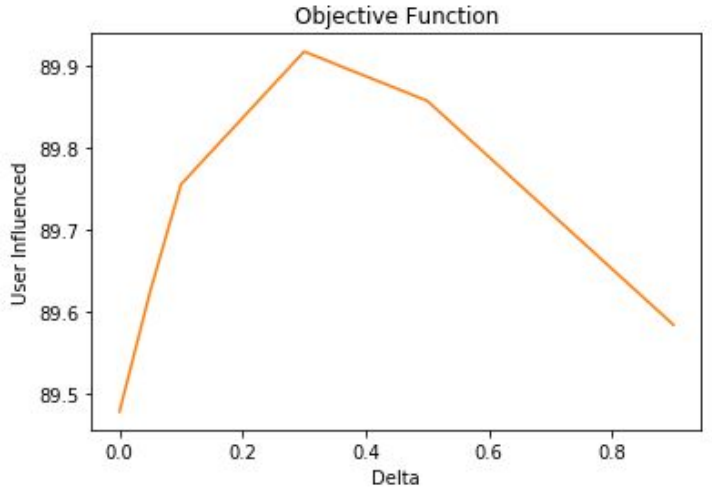
This algorithm approximation bound follows from the sub-modularity property

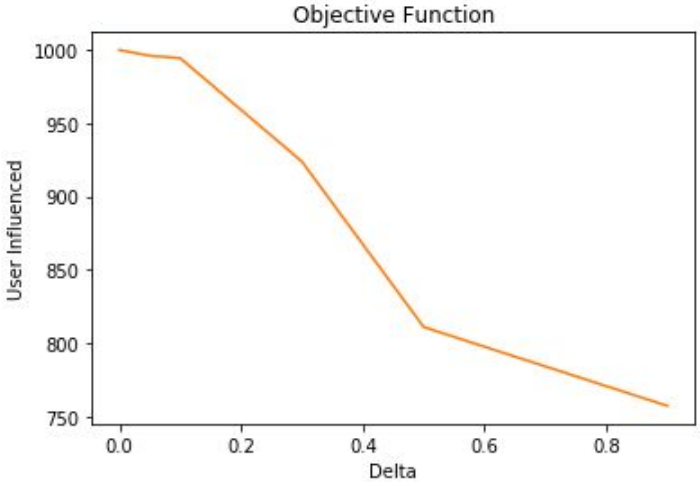
$$\forall X \subseteq Y, x \notin Y : f(X \cup \{x\}) - f(X) \geq f(Y \cup \{x\}) - f(Y)$$

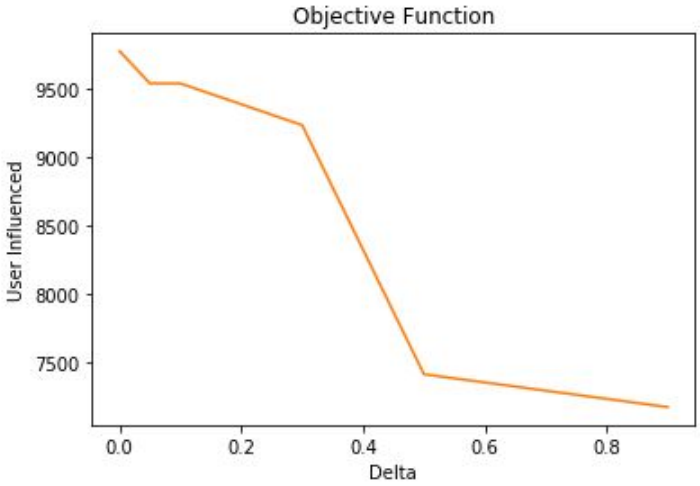
Interpretation: adding a seed to a set of seeds gives a marginal increase in the utility function larger than adding the same node to a strictly larger set of seeds

This algorithm works incrementally, but the process is not simulating that the nodes becomes seed incrementally. Once the algorithm has chosen the set of seeds, they become seeds together

This is how the objective function (number of users influenced) varies based on different values of approximation bound (delta) using the 3 base networks, the greedy algorithm and Montecarlo sampling:







Unknown edges probabilities

The greedy algorithm works really well when we can observe the probabilities of the connections between the nodes of the network. If these probabilities are unknown the algorithm cannot be used.

When this probabilities are not known the process that we have to use is the following:

- First we learn this probabilities
- Then we apply the greedy algorithm to get the results

There are 3 different situations that can occur when we are in this situations (3 learning scenarios):

- We can observe the activation of all the edges
- We can observe the activation of a small portion of edges
- We can observe only the activation of nodes

In the first case we are in a Standard Combinatorial MAB problem so we can solve it this way:

- Every edge is a Bernoulli variable whose expected value is unknown
- For every edge we can draw a sample from a Beta distribution (TS approach) or use an upper confidence bound (UCB1 approach)
- Once we have a value for every edge we apply the greed algorithm

In the second case we cannot observe the whole network (common with big graphs), so we focus on the part that can be observed and exploit this information to infer probabilities on the non-observable edges.

We assume that the edges probability are combination of features F (known)

- Parameters of the features are estimated with linear regression
- Given the values of the probabilities a UCB1-like term is added to obtain an upper bound
- The optimization is performed with greedy algorithm

In the final case we need to observe the evolution of the cascade, at any time of the cascade we take trace of the active nodes so that we can understand which node activates which. If more than one node can activate another node, we use maximum likelihood to decide the best one.

References

- Course slides and lectures