Market Segmentation using Attributed Graph Community Detection

Overview: Market segmentation divides a broad target market into subsets of consumers or businesses that have or are perceived to have common needs, interests, and priorities. These segments help firms or businesses focus on their target groups effectively and allocate resources efficiently. Traditional segmentation methods are solely based on attribute data such as demographics (age, sex, ethnicity, education, etc.) and psychographic profiles (lifestyle, personality, motives, etc.). However, social networks have recently become important for marketing. Depending on the nature of the market, social relations can even become vital in forming segments. Such social relations combined with demographic properties can be used to find more relevant subsets of consumers or businesses (i.e., *communities*).

In this project, we aim to find such market segments given social network data. These social relations can be captured in a graph framework where nodes represent customers/users and edges represent some social relationship. The properties belonging to each customer/user can be treated as node attributes. Hence, market segmentation becomes the problem of community detection over attributed graphs, where the communities are formed based on graph structure as well as attribute similarities.

Influence Propagation: It has been discovered that by targeting the influential users/groups, desirable marketing goals can be achieved. Therefore, one way to evaluate the quality of the market segments (communities) is to influence an entity in each segment and measure how fast the influence propagates over the entire network. The faster that influence propagates through the entire network, the more likely an advertising campaign, for example, will be successful.

Goal: To implement a community detection algorithm for attributed graphs, find the relevant market segments, and evaluate the obtained segments via influence propagation.

Datasets

You are provided with two small datasets: `fb_caltech_small_edgelist.txt` and `fb_caltech_small_attrlist.csv`. This dataset contains a facebook network of a US university (given as an edgelist) with each node corresponding to a user profile having the following attributes: student/faculty status, gender, major, second major, dorm, and year information. For the similarity convenience, these attribute values have been converted into asymmetric binary variables. The original dataset, which we downsampled to 324 users, can be found here.

Project Requirements

You are expected to implement an algorithm from the paper “Community detection based on structural and attribute similarities.”

Project Details:

1. Implement the SAC-1 method described in the section-IV part-A in the publication. For the implementation you just need to focus on section III and section IV part-A.
2. Use either Python or R for your implementation. Your file should be called either `sac1.py` or `sac1.R`. Your code should take one argument as input, the alpha value, and produce a file called `communities.txt` as output. This should contain one community per line with vertex id's separated by commas, e.g.,
```
1,4,8,9,12,17,85
2,7,25,66,97,45
```
3. You should use the igraph package, which is available in both R and Python. In the paper, modularity is used as structural similarity metric. You may choose to use igraph's built-in modularity function for this purpose.
4. Use cosine similarity to measure attribute similarity.
5. Limit the algorithm convergence to have a maximum of 15 iterations (this is an additional parameter that is not mentioned in the publication, but can be used in your implementation if it takes more than 15 iterations before convergence).
6. Produce separate outputs for different values of α = 0, 0.5, and 1. Save these files in the directory under the names `communities_0.txt`, `communities_5.txt`, and `communities_1.txt`, respectively.

Cluster Evaluation: You will be provided with code to evaluate, via influence propagation, the quality of the communities you obtain. The code will measure and compare the time steps taken by the influence propagation algorithm to achieve maximum propagation and compare this with kmeans clustering. To run this code, simply call the evaluation script,

```
$ Rscript evaluation.R
```

This script will assumes you have a file named `communities.txt`.

Submission Instructions:

Submit a single zipped folder containing the following items:

- Your code, which should be well-commented. The main file that will be run should be called `sac1.py` or `sac1.R`.
- Three .txt files for the output communities from running your code on α = 0, 0.5, and 1.
- Three plots generated from the results from running evaluation.R on the three communities files. Call these `communities_0.pdf`, `communities_5.pdf`,and `communities_1.pdf`.
- The original data folder.