CS 498: Project Final Report

Adit Krishnan, UIN: 659913017, Net ID: aditk2 , Aravind Sankar, UIN: 671514664, Net ID: asankar3

December 7, 2016

1 Introduction

We address the task of discovering social circles in a user's ego network in this project. In addition to the baseline algorithms reported in the mid-term submission, we implement two algorithms for solving this problem. The first algorithm CESNA [5] is a state of the art technique for detecting overlapping communities in a social network. We provide a best-effort implementation and report the results that we obtain from our implementation. The second algorithm that we present here is our own model. We draw motivation from some ideas of existing techniques and our analysis of the datasets [3]. In the next section, we provide a very brief description of the main idea behind CESNA. After that, we describe our proposed model in detail. Finally, we present the results of the two algorithms on the facebook ego-network dataset.

2 Algorithm 1 - CESNA

This paper proposes a probabilistic model for generating network topology and node attributes. They assume that each node in the network has a community membership which determines the edges observed in the network. Each community also has a relevance score with respect to each attribute. The community memberships along with attribute specific relevance score for each community are assumed to generate the observed network and attributes independently through a simple additive function. They develop a gradient ascent algorithm to optimize their likelihood function.

3 Algorithm 2 - Our model

We first provide the main motivation behind the design of our model.

3.1 Motivation

We propose our model by drawing the basic idea of community memberships for each node from CESNA. However, we aim to simplify the network and attribute generation process and improve it using insights that we obtain from our dataset. First, we analyzed the ground truth communities in the given dataset to determine the edge density for each community. In Table 1, we show the mean value of density in each ego-network, along with the standard deviation and coefficient of variation (CV). We can clearly see that there is a significant variation in the density of the ground truth communities with many values of CV close to 1. Thus it is preferable to have a model which permits different communities to have variable contribution to the edge likelihood. We capture this explicitly in our model by allowing each community to have a different influence in edge formation

(lets call this community density, since dense communities have more influence on edge likelihood) unlike most other models.

Graph ID	Mean edge density	Standard Deviation	Coef. of Variation
0	0.282	0.315	1.117021277
107	0.327	0.216	0.660550459
348	0.516	0.21	0.406976744
414	0.631	0.267	0.423137876
686	0.295	0.23	0.779661017
698	0.616	0.312	0.506493506
1684	0.393	0.27	0.687022901
1912	0.493	0.294	0.596348884
3437	0.506	0.32	0.632411067
3980	0.46	0.403	0.876086957

Table 1: Mean and standard deviation of ground truth communities in the facebook dataset

The second insight is that node attributes in ego-networks tend to be very sparse and noisy. This means that explicitly modeling the generation of node attributes based on community memberships could lead to ill formed communities. Instead, we use the node attributes to capture the profile similarity between a pair of users, which is now used to guide the formation of edges in a network based on community memberships of each node. This also has an added benefit in that the complexity of the model is reduced greatly as we only model the likelihood of the observed edges in the network.

To summarize, we model the likelihood of observing the network edges based on community memberships of each node, community density and node attributes. We define a likelihood function to capture this idea and propose a parallel optimization algorithm to infer the parameters of our model. We find that our model is much faster than CESNA (our implementation) in practice (both models were implemented in Java). Next, we provide a formal description of our model.

3.2 Model formulation

Given an undirected graph $G(V, E)$ with binary node attributes X, the goal is to detect C communities that could be overlapping. For now, we assume that the number of communities C is given. We describe the procedure to automatically find the best value of C later. We assume that there are N nodes in the network G and each node has K attributes. We denote the network by G, and attributes by X. Each node has a binary attribute vector denoted by $\mathbf{X_u}$ of size K, where $\mathbf{X_u}(k) = 1$ indicates the presence of attribute k for node u in the network. In addition to these basic observed variables, we have node community memberships F and community density weights W. For community memberships F, we assume that each node u has a non-negative affiliation weight F_{uc} to community c. Each community has a non-negative weight W_c which represents the edge density of the community. First we describe how similarity of two users is modeled using their node attributes.

User similarity

We model profile similarity of a pair of users (u, v) using their node attributes. We use the cosine similarity to model this similarity as below :

$$sim_{uv} = \frac{\mathbf{X_u}.\mathbf{X_v}}{\|\mathbf{X_u}\|\|\mathbf{X_v}\|}$$

This profile similarity will be used in combination user community memberships to model the likelihood of observing the network. Next, we describe a generative process of observing the links in the network.

Link generation

In this section, we describe how network structure depends on the different factors outlined earlier and provide a mathematical formulation of the same. First, we describe the effect of node community memberships. Node community affiliations positively influence the likelihood that a pair of nodes is connected. The degree of influence depends on the communities that the pair of nodes belong to. Each community influences the link probability independently but proportional to the density of the community. Using these intuitions, we model this influence of community memberships as proportional to $F_{uc}F_{vc}W_c$ for a pair of users (u, v) w.r.t. community c. Next, the link probability between a pair of nodes also depends on their user profile similarity, which is modeled as proportional to sim_{uv}. We combine the two effects to model the probability that two nodes u, v belonging to a community c are connected as :

$$P_{uv}(c) = 1 - \exp(-\left\{\alpha F_{uc}.W_c.F_{vc} + (1-\alpha)\frac{sim_{uv}}{C}\right\})$$

Here, α is a hyper-parameter that controls the effect of the two factors in modeling the link probability (set to 0.5 by default). We assume that each community c connects nodes u, v independently with probability $P_{uv}(c)$. For u and v to be connected, it must be connected by at least one community which gives a link generation probability P_{uv} as :

$$\begin{aligned}
P_{uv} &= 1 - \prod_{c=1}^{C}(1 - P_{uv}(c)) \\
&= 1 - \prod_{c=1}^{C}\exp(-\left\{\alpha F_{uc}.W_c.F_{vc} + (1-\alpha)\frac{sim_{uv}}{C}\right\}) \\
&= 1 - \exp(-\sum_{c=1}^{C}\left\{\alpha F_{uc}.W_c.F_{vc} + (1-\alpha)\frac{sim_{uv}}{C}\right\}) \\
&= 1 - \exp(-(\alpha\mathbf{F_u^T}\mathbf{W}\mathbf{F_v} + (1-\alpha)sim_{uv}))
\end{aligned}$$

Here, $\mathbf{F_u}$ denotes the vector of community memberships (of size C) for node u. $\mathbf{W}$(density matrix) is a diagonal matrix of size $C \times C$. This defines the link generation probability in the network. We assume a Multi-Bernoulli process for link generation i.e. each link (u, v) is modeled by a Bernoulli distribution with parameter P_{uv}. This completes the basic formalism of our model. We can clearly observe that the different factors mentioned earlier are captured in modeling this link generation probability. In the following section, we describe how to infer communities by formulating a likelihood function that we optimize using gradient ascent.

3.3 Community Inference

In this section, we describe how we detect network communities by estimating the model parameters from the given data. We aim to infer the values of latent variables F and W based on the observed network and attributes. We note that though the attributes are also observed, we only model the likelihood of the network structure as the attributes are captured in the link generation probability. We define the likelihood of the observed network G as $P(G \mid F, W, X)$ given by :

$$P(G \mid F, W, X) = \prod_{u,v \in E} P_{uv} \prod_{u,v \notin E} (1 - P_{uv})$$

To find the optimal $\hat{F}$ and $\hat{W}$, we optimize the log-likelihood of the observed network as :

$$\begin{aligned}
\mathcal{L} &= \log P(G \mid F, W, X) \\
&= \sum_{u,v \in E} \log \Big(1 - \exp(-(\alpha \mathbf{F_u^T W F_v} + (1-\alpha) sim_{uv}))\Big) - \sum_{u,v \notin E} (\alpha \mathbf{F_u^T W F_v} + (1-\alpha) sim_{uv})) \\
&= \sum_{u,v \in E} \log \Big(1 - \exp(-\psi(u,v)\Big) - \sum_{u,v \notin E} \psi(u,v)
\end{aligned}$$

where $\psi(u,v) = \alpha \mathbf{F_u^T W F_v} + (1-\alpha) sim_{uv} = \alpha \sum_{c=1}^{C} F_{uc} W_c F_{vc} + (1-\alpha) sim_{uv}$

Thus, the optimization problem that we aim to solve is:

$$\hat{F}, \hat{W} = \underset{F,W \geq 0}{\arg\max}\, \mathcal{L}$$

To solve this optimization problem, we use gradient ascent. We can easily show that the above likelihood function is concave in both F and W. Since we impose non-negativity constraints on F and W, we compute gradient updates for each variable and then project onto to space of non-negative real numbers $[0, \infty)$.

Gradient update for F

In order to update the community memberships $F_u$of a node u, we fix all other parameters (the memberships of all other nodes and W) and then compute the gradient update for u. Thus, the sub-problem that we solve for a node u is given by:

$$\mathcal{L}(F_u) = \sum_{v \in N(u)} \log(1 - \exp(-\psi(u,v))) - \sum_{v \notin N(u)} \psi(u,v)$$

Here, $N(u)$ denotes the set of all neighbors of node u in the network G. We now derive the gradient update rule for each F_{uc}, which is given as :

$$\begin{aligned}
\frac{\partial \mathcal{L}}{F_{uc}} &= \sum_{v \in N(u)} \alpha F_{vc} W_c \frac{\exp(-\psi(u,v))}{1 - \exp(-\psi(u,v))} - \sum_{v \notin N(u)} \alpha F_{vc} W_c \\
&= \sum_{v \in N(u)} \alpha F_{vc} W_c \frac{\exp(-\psi(u,v))}{1 - \exp(-\psi(u,v))} - \alpha \left\{ \sum_{v \in V} F_{vc} W_c - F_{uc} W_c - \sum_{v \in N(u)} F_{vc} \right\}
\end{aligned}$$

We obtain the first equation by taking the gradient of $\mathcal{L}$ w.r.t F_{uc}. We can simplify the summation over all non-neighbors of u by re-writing it as function of just the neighbors of u, since the term $\sum_{v \in V} F_{vc}$ can be computed in advance. The update equation for F_{uc} now becomes :

$$F_{uc}^{n+1} = \max\left(0, F_{uc}^{n} + \eta_u \frac{\partial \mathcal{L}}{\partial F_{uc}^{n}}\right) \tag{1}$$

In this equation, η_u is the learning rate for the node u, which is estimated using backtracking line-search [1] for performing the gradient updates of node u. We note here that the gradient equation (1) here lends itself to easy parallelism. Specifically, we can perform the gradient updates for each node u in parallel, which will leads to great speedup.

Gradient update for W

The gradient update for W is derived in a very similar manner to F. We keep the community memberships of all nodes (F) fixed when we perform the gradient updates for W. Thus, we get the gradient w.r.t W_c as :

$$\frac{\partial \mathcal{L}}{W_c} = \sum_{u,v \in E} \alpha F_{uc} F_{vc} \frac{\exp(-\psi(u,v))}{1 - \exp(-\psi(u,v))} - \alpha \left\{ \sum_{u,v \notin E} \alpha F_{uc} F_{vc} \right\}$$

The gradient update for W_c is given by :

$$W_c^{n+1} = \max\left(0, W_c^{n} + \eta_c \frac{\partial \mathcal{L}}{\partial W_c^{n}}\right) \tag{2}$$

In this case also, we can capture gradients for each community is parallel. The learning rate η_c is again learned using line search for each c.

The gradient ascent algorithm is performed by alternatively updating F and W in each iteration. We stop when the relative change in likelihood is less than 0.001 % or if 1000 iterations has been reached. We provide a parallel implementation of both our algorithm and CESNA in Java.

3.4 Operational Details

Detecting community assignments

The gradient ascent algorithm will give us community memberships for each node u in the network. Based on these values we need to perform assignments of nodes to communities. To determine if node u belongs to community c, we require that F_{uc} is greater than a threshold δ_c for community c. We set δ_c so that node u belongs to community c if u is connected to other members of c with an edge probability greater than $1/N$ (similar to CESNA). In doing so, we ignore the contribution of attributes to the edge probability. We determine δ_c as :

$$P(u,v) \geq (1 - \exp(1 - \delta_c^2 W_c)) \geq \frac{1}{N}$$

Where u and v are two nodes lying in c. Upon solving this inequality, we obtain :

$$\delta_c = \sqrt{\frac{\log(\frac{N}{N-1})}{W_c}}$$

i.e,

$$F_{uc} \geq \sqrt{\frac{\log(\frac{N}{N-1})}{W_c}} \implies u \in c$$

Choosing the best number of communities

In order to automatically find the best number of communities C, we remove 10% of the edges from the network to create a hold-out set. Then, we vary C in a range from 5 to 15 and fit the remaining network using our model. We evaluate the likelihood of the hold-out set using our estimated model. The value of C that gives us maximum likelihood on the held-out set is chosen as the number of communities.

In our model, hyper-parameter α determines the trade-off between user profile similarity and node community memberships. It is set to 0.5 by default.

Initial community memberships

It is essential to meaningfully initialize community memberships to ensure faster convergence of the gradient ascent and to produce better results. We use a simple graph based heuristic to perform initial community membership assignments. For each node, $n \in G$, where G is the input ego network, a locality L_n is constructed with the node and it's immediate neighbors:

$$\text{In-Out edge ratio}(L_n) = \frac{\text{Edges within } L_n}{\text{Edges emerging from } L_n}$$

All nodes are sorted by the In-Out ratio defined above. The nodes with the best ratios and meeting a minimum neighborhood size requirement, are assigned with higher initialization of F_{uc} along with their immediate neighbors to some community c. This makes the algorithm more likely to ultimately place these localities within a single community.

Initial community density

The initial community density weights W must be representative of the local density of edges. As described above, most communities are initialized in a manner that they favour some strongly connected neighborhoods. The density values of these communities are initialized to the In-Out edge ratios obtained for the respective neighborhoods.

Line Search for Gradient Ascent

In functions that involve updates over several variables, it is necessary to choose an appropriate value of the learning rate η in the gradient update equations 1 and 2. We use the Armijo Goldstein rule to choose learning rates for our variables. Learning rates are picked at the granularity of each user's community memberships in F updates and at the granularity of each community for W updates, in every gradient ascent iteration.

Choose the largest possible η_c for community c such that:

$$[L(W_c^n + \eta_c \frac{\partial \mathcal{L}}{\partial W_c^n}) - L(W_c^n)] \geq \tau(||\frac{\partial \mathcal{L}}{\partial W_c^n}||)\eta_c$$

Note that F need not be explicitly mentioned as part of the likelihood function here since the variables are updated one at a time. Each is maintained constant when the other is being updated.

Similarly for F_u choose the largest possible η_u such that:

$$[L(F_u^n + \eta_c u \frac{\partial \mathcal{L}}{\partial F_u^n}) - L(F_u^n)] \geq \tau(||\frac{\partial \mathcal{L}}{\partial F_u^n}||)\eta_u$$

The Norm here represents L2 Norm of the update vector, since F_u is a vector of length equal to number of communities. τ, the control parameter for line search, is set empirically.

4 Experimental Evaluation

4.1 Setup

We run our model on the 10 ego networks present in the Facebook dataset [4]. The edge - attribute tradeoff parameter α is set to 0.5. Stopping condition is set to 0.1% for cross validation fits and 0.001% to fit the final model once K has been determined. The code is parallelized across users for F_u update computation and across communities for W_c updates. The number of threads is set to 8 and executed on an 8-core machine for optimal performance.

4.2 An important observation

We find that in most graphs, the changes to the density parameter W tend to rely heavily on the initializations, especially for larger graphs. In some cases, a few communities tend to dominate over iterations and reduce the quality of communities generated. Based on our empirical evaluation with the facebook dataset, the initial density values which are assigned based on the edge ratio, tend to perform well and capture the general structure of communities generated. Hence as a simplifying step, we have removed density updates in the final submission to reduce running times, and to perform well on graphs of all sizes. The W parameters thus remains set constant to the initial density values assigned based on the In-Out edge ratio as defined in section 3.4. The subsequent runtimes are recorded with this simplification.

4.3 Quality

We evaluate the quality of the communities using the evaluation metric proposed in CESNA given by :

$$\frac{1}{2|C^*|} \sum_{C_i^* \in C^*} \max_{C_j \in C} \delta(C_i^*, C_j) + \frac{1}{2|C|} \sum_{C_j \in C} \max_{C_i^* \in C^*} \delta(C_i^*, C_j)$$

Here, C denotes the predicted communities and C^* the original ground truth communities. $\delta(C_i^*, C_j)$ is a similarity measure between the communities C_i^* and C_j which we assume to be the Jaccard similarity. The performance of our model and CESNA based on this metric, are shown below :

The plot depicting the comparison of the two algorithms is shown below.

Graph	CESNA	Our Model
Graph 0	0.1782	0.1837
Graph 107	0.2423	0.25264
Graph 348	0.3952	0.40088
Graph 414	0.5203	0.5278
Graph 686	0.3577	0.35028
Graph 698	0.5349	0.576
Graph 1684	0.3191	0.36822
Graph 1912	0.2696	0.32478
Graph 3437	0.101	0.10666
Graph 3980	0.3375	0.32564

Table 2: Performance comparison of our model and CESNA on the Facebook dataset

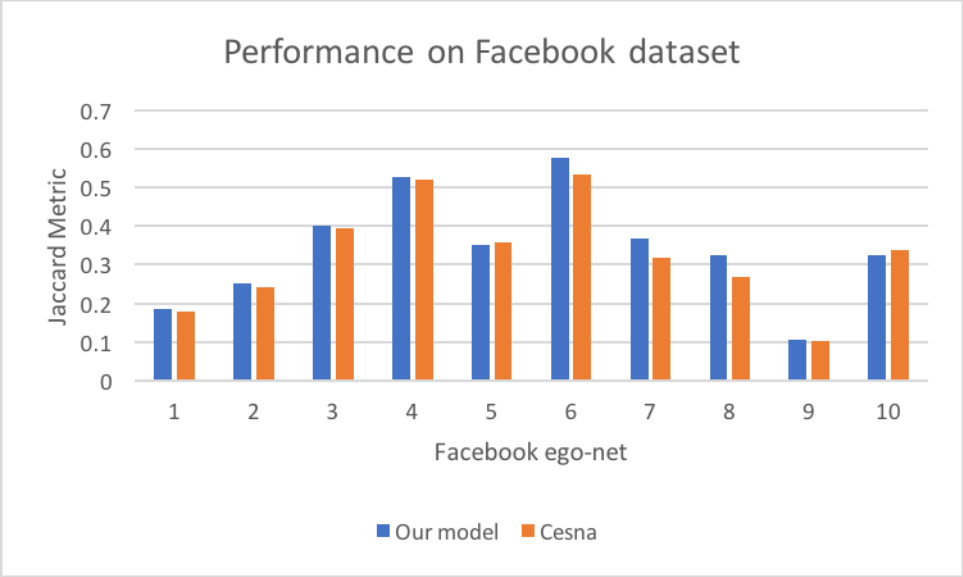

Figure 1: Histogram showing the performance of our model and CESNA on different graphs

References

[1] Boyd, Stephen, and Lieven Vandenberghe. Convex optimization. Cambridge university press, 2004.

[2] Leskovec, Jure. "Stanford network analysis package (snap)." URL http://snap. stanford. edu (2013).

[3] Leskovec, Jure, and Andrej Krevl. "SNAP Datasets: Stanford large network dataset collection, June 2014." URL: http://snap. stanford. edu/data (2014).

[4] McAuley, Julian J., and Jure Leskovec. "Learning to Discover Social Circles in Ego Networks." NIPS. Vol. 2012. 2012.

[5] Yang, Jaewon, Julian McAuley, and Jure Leskovec. "Community detection in networks with node attributes." 2013 IEEE 13th International Conference on Data Mining. IEEE, 2013.