Community Detection Clustering in Complex Networks using Gumbel Softmax

Visaj Nirav Shah (201801016)* and Vyom Saraf (201801062)†
Dhirubhai Ambani Institute of Information & Communication Technology,
Gandhinagar, Gujarat 382007, India
SC-435, Introduction to Complex Networks
Mentor: Prof. Mukesh Tiwari

In this project, we have mathematically analyzed two network datasets. Following which, we simulated their partitioning with a Deep Learning-based clustering algorithm using Gumbel Softmax. The cluster network analysis of both datasets to find the existence of a threshold value of number of edges between two clusters and probabilistic clustering in various forms has been done.

Python Code: Notebook

I. INTRODUCTION

Community Detection and Clustering continues to be an important problem in Networks Science. In community detection, we identify groups of nodes (clusters) which are highly connected among themselves, but are sparsely connected with members of other clusters. Such an analysis brings to light a deeper understanding of network data, large-scale patterns and structures not directly visible, and the nature of interactions [1].

Since we are working with a complex network, a Deep Learning-based clustering algorithm has an edge over a traditional algorithm since there is a lot of data. Also, Gumbel Softmax allows for a probablistic clustering using backpropagation. Clustering has several applications in real world, and in this project, we create a basic implementation of some on selected networks.

II. MATHEMATICS OF DATASETS

Before moving on to the research problem, we need to understand the networks/datasets we are working with. This section will highlight important mathematical aspects of the datasets, which will help us understand them better.

We are working with 2 datasets: Zachary's Karate Club Dataset [2] and Facebook Page-Page Networks Dataset for Artists' Pages [3]. For both the networks, the following properties hold:- Undirected, Unweighted, Node numbering is 0-indexed.

Note:- Apart from the mathematical aspects mentioned here, the Notebook also contains the following centrality measures and values:- Adjacency matrix, Degree of nodes, graph Laplacian, Betweenness, Closeness, and Eigenvalue centrality.

A. Dataset I: Zachary's Karate Club Dataset

The network represents the interactions between 34 members of a karate club. Each member is a node, and an edge represents that the nodes (connected by the edge) interacted outside the club activities.

	Quantity	Value
0	Nodes	34.000000
1	Edges	78.000000
2	Mean Degree	4.588235
3	Density	0.139037

FIG. 1: Brief mathematical description of Dataset I

B. Dataset II: Facebook Page-Page Networks Dataset for Artists' Pages

Famous personalities, brands, and companies run verified (blue tick) Facebook Pages. This is a complex network where the verified Facebook Pages of artists are the nodes, and mutual likes among the Pages are the edges. Hereon, the functions and codes get a lot more complex. We tried running them on the complete dataset, but our personal computers could not process complex computations on such a large dataset in a reasonable time. Hence, we found a way to truncate the dataset. We will select 2500 nodes with the highest degree. Only edges between these 2500 nodes will be considered. Some nodes will be lost out of these 2500 because some will only have edges with the nodes not considered in 2500. Finally, the Dataset II used for further analysis has 1370 nodes and 2192 edges.

*Electronic address: 201801016@daiict.ac.in
†Electronic address: 201801062@daiict.ac.in

	Quantity	Value
0	Nodes	50515.000000
1	Edges	819306.000000
2	Mean Degree	32.433851
3	Density	0.000642

FIG. 2: Brief mathematical description of Dataset II

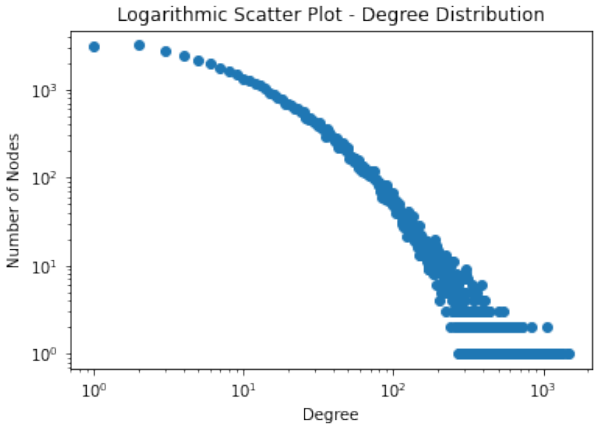

FIG. 3: Scatter plot for degree distribution of Dataset II

III. NORMAL CLUSTERING (AS PROPOSED IN [4])

[5] introduces Feature Selection and Extraction for Graph Neural Networks (FSEGNN). When running GNNs, not all features are considered. This is usually done to reduce the data and computational requirements. FSEGNN is a method to select the prominent features which have a greater impact on predicting the results. This algorithm is suited for labeled datasets only.

Our main paper, [4], builds on the works of [5], and extends the Gumbel Softmax approach for clustering to unlabeled datasets (unsupervised learning). Community Detection Clustering via Gumbel Softmax (CDCGS) [4] is the proposed approach for tackling this research problem.

Gumbel Softmax is a continuous distribution on the simplex which can approximate samples from a categorical distribution. The need for developing such a mechanism arises because stochastic neural networks cannot use discrete categorical variables since they are indifferentiable and cannot be used in backpropagation learning [6]. Mathematically,

$$z = one\_hot(\underset{i}{argmax}[g_i + log\pi_i]) \tag{1}$$

$$y_i = \frac{\frac{exp((log(\pi_i)+g_i)}{\tau}}{\sum_{j=1}^{k}\frac{exp((log(\pi_j)+g_j)}{\tau}} \tag{2}$$

z in (1) is a categorical random variable with stated class probabilities . Range of i is [1, k] where k = number of features selected. Since (1) is not differentiable, we approximate it with a Softmax function as shown in (2). serves as a way of controlling the value of the approximation. This is known as the temperature parameter in Deep Learning.

For implementing CDCGS,

$$G - Cluster(A) = Softmax(W_C^t A W_C) \tag{3}$$

$$Softmax(x_i) = \frac{exp(x_i)}{\sum_{j=1}^{m} exp(x_j)} \tag{4}$$

A_{nXn} is the Adjacency Matrix of G (undirected graph) with n nodes and W_C is the gumbel cluster weight matrix with dimensions nXk where k is the number of clusters desired. The RHS of (3) results in a matrix R_{kXk} which shows the strength of clustering. Finally, we implement the Softmax function as given in (4) for getting a discrete probability distribution.

For determining the allotted clusters, we turn to the gumbel cluster weight matrix, i.e. W_C. With n rows and k columns, each row in W_C denotes a node's probability of belonging to the k clusters. For example, W_C[i][j] denotes the probability of i^{th} node belonging to the j^{th} cluster (0-indexed). We find the maximum value in each row, i.e. max(W_C[i][j]) for each i (j from [0, k)). The corresponding cluster j is allotted to node i. FIG. 4 shows the allotment of nodes to clusters. It is clear,

$$\text{For a given i,} \sum_{j=0}^{k-1} W_C[i][j] = 1 \tag{5}$$

IV. EXTENDING THE WORK

A. Custom Threshold Clustering (Multiple Clustering)

In the previous section, the algorithm finds, for each node, the cluster for which the probability is maximum. This node is put in that cluster. This restricts a node to a single cluster. However, often, our requirements demand multiple clustering, i.e., a single node belonging to multiple clusters. There are many algorithms to achieve multiple clustering, and here, we are considering one particular case.

For multiple clustering, we consider a threshold probability that the user gives instead of considering

```
In [44]: # Size of each cluster

         df.groupby('cluster_num').size()

Out[44]: cluster_num
         0    132
         1    126
         2    142
         3    157
         4    122
         5    127
         6    143
         7    182
         8     98
         9    141
         dtype: int64
```

FIG. 4: Dataset II: Num. of nodes allotted to each cluster (clusters_number = 10)

the maximum probability. The node is allotted to all those clusters for which the probability is greater than or equal to the threshold. So, now, a node can belong to multiple clusters.

There is substantial practical utility for such clustering. Consider Amazon wants to divide its customer base based on the category of products they are likely to buy (customer segregation). Amazon sets a threshold probability, and customers are segregated accordingly. If the clusters show that a person X is more likely to buy 'Clothes' and 'Perfumes', Amazon can target advertisements and give recommendations accordingly. The original algorithm would not put X in both categories, and we lose such insights.

This feature can be further developed by allowing the user to set different threshold probabilities for different clusters.

```
In [63]: # Count of nodes assigned to more than 1 cluster

         count = 0
         for i in df['cluster_num']:
             if len(i) > 1:
                 count += 1

         print(str(count) + " nodes got assigned to more than 1 cluster.")

         14 nodes got assigned to more than 1 cluster.
```

FIG. 5: Dataset II: Num. of nodes = 14 assigned to more than 1 cluster

B. Determining Ideal Number of Clusters

We got the idea for implementing this part from the elbow method used in Machine Learning. In k-means clustering, the user needs to input k, the number of clusters. However, it is hard to determine the value of k for which the algorithm will yield an ideal clustering. So, we run the model for a range of k, and we select the value which creates a sharp up-tick or down-tick ("elbow") in the value of the variable under consideration. This is known as the elbow method.

Applying an analogous idea to networks, one measure of the effectiveness of clustering is modularity. While there are exceptions to this [7], here we assume that higher modularity denotes better partitioning. So, we run our clustering algorithm in a loop for a range of values (which can be changed by the user) of the number of clusters. For each iteration, we store the modularity value. Then, we plot a line graph to observe the behavior of modularity as the number of clusters changes. The ideal number of clusters is the one that yields the highest modularity.

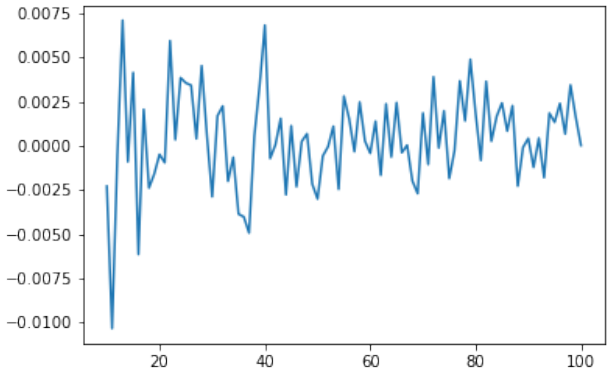

FIG. 6: Dataset II: Modularity (Y-Axis) vs. Num. of Clusters ([10, 101)) (X-Axis) plot

```
In [22]: print("Maximum modularity in this range = ", max(yAxis))
         print("Which is achieved for clusters_number = ", d[max(yAxis)])

         Maximum modularity in this range =  0.007106964942191916
         Which is achieved for clusters_number =  13
```

FIG. 7: Dataset II: Ideal clusters_number = 13

While this method is not full-proof, it definitely gives us a fair idea of how to proceed with partitioning a dataset. Users can understand the behavior of modularity in a single glance with the plot. Determining the ideal number of clusters is an important research problem in many fields, including Networks Science and Machine Learning.

C. Cluster Network Analysis

The idea behind this work is that when a social media user either reacts or even views another social media user's post, what is the potential impact of the same to the post in terms of the number of viewers and number of reactions. The reasoning behind the same is that when any social media influencer claims to have a significant impact on any social media post, this work will analyse their actual impact rather than the false impact analysed by their number of followers or close following. With reasoning in mind, the analysis has been done accordingly, which is explained as follows. Any social media user is part of a particular social circle or group, which can also be considered the close following of that user. We have assumed that all these closed followers usually belong to the same cluster in the entire social media network. The reasoning for the same is that there are many connections between these close following or social circles. As a result of the high number of edges between these nodes, we believe that it is safe to assume that all the members of these circles belong to the same cluster.

Now, since we have analysed that the close following belong to a cluster, we would like to propose a hypothesis that we can treat all these nodes present in a single cluster as one because of the high connectivity between them. Since, in terms of a social media user's close following, it is most likely that the close following would definitely react to any post made by one of the users belonging to this circle. Based on this hypothesis, we can further assume that any post reacted by any user in the circle will also be reacted by the entire social media circle. Therefore, rather than analysing the true impact of one user, it is better to analyse the effect of the entire social media circle on the post. This would help us in two ways as we would be able to analyse the true impact of a social media circle and the analysis of an individual user. Similarly, we would replace all individual user nodes with clusters, and instead of analysing the connections between two nodes, we would analyse the connections between two clusters.

This leads us to the final part of the analysis, in which we have formed an adjacency matrix of the clusters and kept in mind the number of edges between the clusters. After creating the adjacency matrix, which also contained the number of edges between any two clusters, we found the maximum number of edges between any two clusters of the entire network. After finding the maximum number of edges, we normalised the adjacency matrix by dividing the number of edges between two clusters by this maximum value. As a result, the value of the number of edges between any two nodes came between 0.0 and 1.0, making further analysis much easier for us.

This leads to the fundamental analysis part. We were trying to find a threshold value for the number of edges between any two clusters as it would provide us with the actual true impact of that cluster. It is important to note that there might exist a threshold value between 0.0 and 1.0 representing the number of edges between the clusters, which would be an actual massive difference in the number of edges below and above that value. So we experimentally tried to analyse the results and hence tried finding the number of edges remaining between the clusters below a threshold value. We started the analysis by taking 0.1 intervals for the threshold value between 0.0 and 1.0. This helped us find an approximate range between which the threshold value would be present. We are also checking the same, looking at the total number of edges left between clusters.It is important to note that we would have been able to find the threshold value up to 2 decimal accuracies correctly had the dataset been much smaller.

```
The maximum number of connections between any two clusters is :
101
```

Out[25]:

	Threshold Value	Edges Left in the Network
0	0.0	1940.0
1	0.1	1940.0
2	0.2	1920.0
3	0.3	1688.0
4	0.4	1268.0
5	0.5	821.0
6	0.6	488.0
7	0.7	101.0
8	0.8	101.0
9	0.9	101.0

FIG. 8: Dataset II: Experimental Analysis of Threshold Value

V. CONCLUSION AND FUTURE WORK

In this project, we started with a mathematical exploration of two networks. Then, we applied a Deep Learning-based clustering algorithm based on Gumbel Softmax approach to these networks. Extending that work, we were able to add some practical utilities that make probablistic clustering more meaningful. If formalized, our experimental threshold value analysis can have important ramifications in Social Media Analysis. We were limited by the computing powers of our personal computers in this project. But, with access to GPUs, more accurate results with better models and larger datasets are possible. The most important future goal for the project is to perfectly analyze and try and find the threshold value for the original Dataset II. Apart from that, another application is to try and automate the process of finding the threshold value, as right now, we are approaching it experimentally.

[1] M. Newman, Networks: An Introduction. Oxford: Oxford University Press, 2010.

[2] W.W. Zachary, "An Information Flow Model for Conflict and Fission in Small Groups", Journal of anthropological research, vol. 33, no. 4, 1977, 10.1086/jar.33.4.3629752.

[3] B. Rozemberczki, R. Davies, R. Sarkar, and C. Sutton, "Gemsec: Graph embedding with self clustering," in Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2019. ACM, 2019, pp. 65–72.

[4] D.B. Acharya and H. Zhang, "Community detection clustering via gumbel softmax", SN Computer Science, vol. 1, no. 5, Aug 2020. Available: https://doi.org/10.1007%2Fs42979-020-00264-2

[5] D.B. Acharya and H. Zhang, "Feature selection and extraction for graph neural networks", In: Proceedings of the 2020 ACM southeast conference, ACM SE '20, page 252–255, New York, NY, USA. Association for Computing Machinery; 2020.

[6] Jang E, Gu S, Poole B. "Categorical reparameterization with Gumbel-softmax", ICLR: Toulon; 2017.

[7] Kehagias, Athanasios Pitsoulis, Leonidas. (2012). Bad Communities with High Modularity. The European Physical Journal B. 86. 10.1140/epjb/e2013-40169-1.