Data Analytics Assignment - 2

G Ramesh Babu
SR No. 20950

Community Detection

Abstract

This Report shows the analysis done on Community Detection methods as part of Data Analytics Assignment - 2. ***Complete Results are given in the text file***

1 Implementation Summary

1.1 Importing Data

1.1.1 Facebook Data

- The given text file consists of rows of edges. This is an undirected graph with weight one
- These edges are taken into a list and then checks were done to detect any duplicate edges
- Finally, this list is converted into an numpy array of size $(\mathbf{88234}, \mathbf{2})$ and returned
- We can see that there are a total of **4039** nodes and **88234** edges in the graph

1.1.2 Bitcoin Data

- The given csv file consists of 4 columns of which the first two columns give the edges and the 3rd column is weight of the edge and the 4th column is the time at which that data is saved
- Since we are considering the graph to be undirected, after taking the data into dataframe using pandas, only the first two columns are taken into the list and the duplicates are removed
- After removing the duplicates, we have more than **21492** edges. But it should be noted that the nodes are **5881** and the node IDs are not zero indexed. Inorder to avoid errors in the computation of Adjacency Matrix, using a dictionary, all the node IDs are mapped to $\mathbf{0} - \mathbf{5880}$ and hence the edges
- Finally, numpy array containing the edges is returned

1.2 Spectral Decomposition

- Firstly, the function **spectralDecomp_OneIter**() is implemented and then it is used for multiple partitions
- From the edges, Adjacency Matrix, Degree Matrix and Laplacian Matrix are computed and they are used to find the Fiedler vector (second smallest eigenvector) by solving the generalized eigenvalue problem $\mathbf{Lx} = \lambda \mathbf{Dx}$ using **scipy**.**linalg**.**eigh**(). NCut Algorithm is used to find the Fiedler vector
- After obtaining the Fiedler vector, checks are made to see if all its components have same sign and then the Fiedler vector is thresholded with **0** to obtain the communities of each node
- Based on these Community IDs, Graph-Partition Matrix is formed

- Adjacency Matrix, Graph Partition Matrix are arranged based on the sorted Fiedler vector and plotted.
- In order to form multiple Communities, the **spectralDecomposition**() function is used, in which **spectralDecomp_OneIter**() is called. Each time, based on the Fiedler vector returned by the function **spectralDecomp_OneIter**(), the given edges are divided into two Communities and **spectralDecomposition**() is called again on each of these Communities recursively. Finally these two graph partitions are stacked and returned
- The stopping criteria for this recursion is that the partitions are to be made only when the components of the Fiedler vector are not continuous

1.3 Louvain Algorithm

- In **louvain_one_iter**(), Firstly, each node is assigned itself as a Community and the Neighboring communities of every nodes are found
- For every node, neighboring nodes and hence their Communities are found and best community for this particular node is found in those Neighboring Communities
- For every node, QDemerge is found using the function **QDeMerge**() and it is used to find the best community for that node
- **BestComm**() function is used to find the best community for a node. For every neighboring Community, we find $\mathbf{\Delta Q}$ by adding QDemerge and QMerge by using **QDeMerge**() and **QMerge**() respectively. For any node, we find the Community with maximum improvement in Modularity if the node is assigned to it
- The stopping criteria for this Algorithm is to count the changes in community for every node and to stop when there are no such changes which happens when the Modularity doesnt change with any new Community assignment to any node

1.4 Plots and Results - Implementation

- The Graph Partition Matrix returned by either of the Algorithms is used to sort the Adjacency Matrix such that the nodes pertaining to the same Community stay together by using **createSortedAdjMat**() function
- Plotting is done using *plot*() function which takes the Fiedler vector, Adjcency Matrix and Graph Partition Matrix and plots the three of them for **spectralDecomp_OneIter**() and plots the Adjacency Matrix and Graph Partition Matrix for the rest of the two Algorithms
- After finding the final Communities using both the Algorithms, Modularity is calculated using **Modularity**() function which returns the Modularity using edges and Graph Partition Matrix

2 Plots

2.1 Facebook Data

2.1.1 Spectral Decomposition - One Iteration

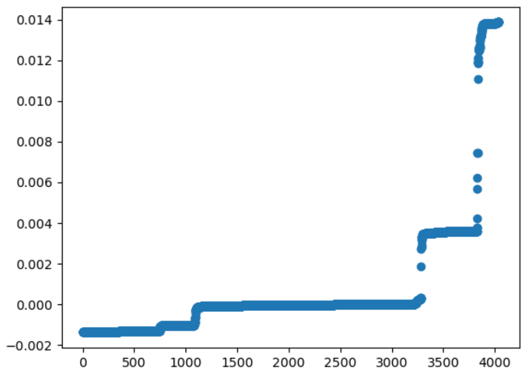

Figure 1: Fiedler Vector

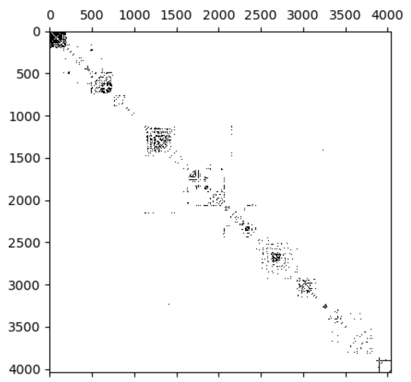

Figure 2: Adjacency Matrix

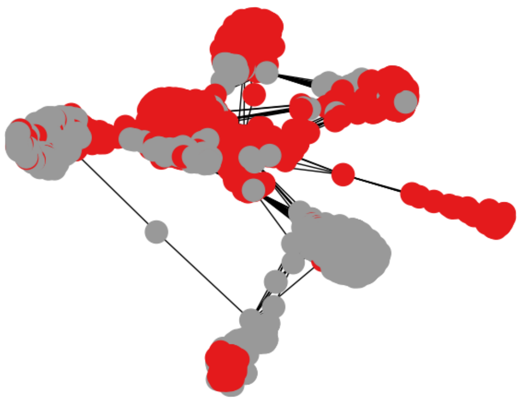

Figure 3: Graph Partition Matrix- 2 Communities

2.1.2 Spectral Decomposition

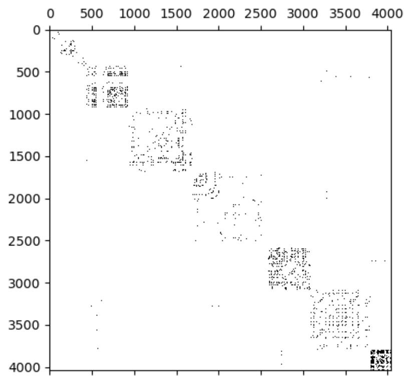

Figure 4: Adjacency Matrix

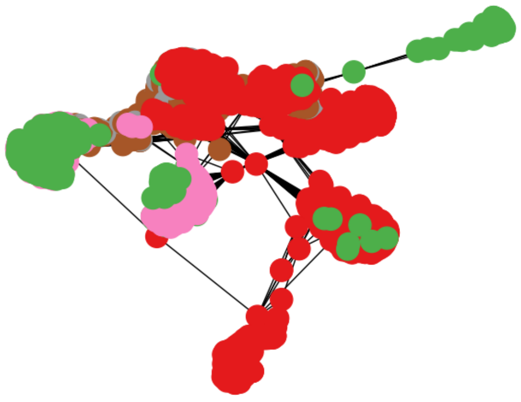

Figure 5: Graph Partition Matrix - 10 Communities

2.1.3 Louvain Algorithm

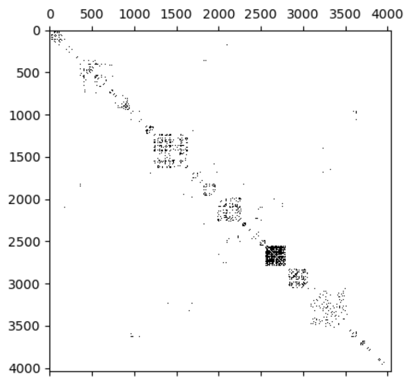

Figure 6: Adjacency Matrix

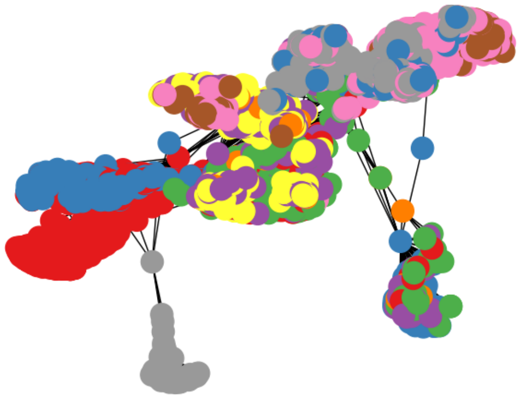
Figure 7: Graph Partition Matrix - 101 Communities

2.2 Bitcoin Data

2.2.1 Spectral Decomposition - One Iteration

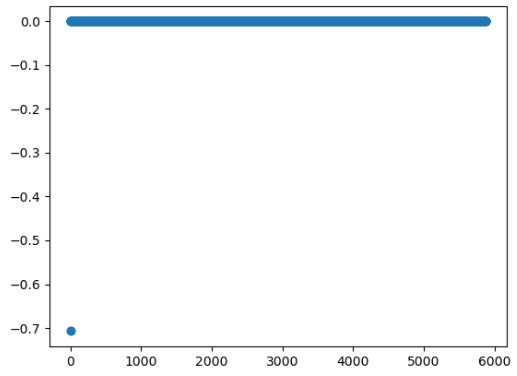

Figure 8: Fiedler Vector

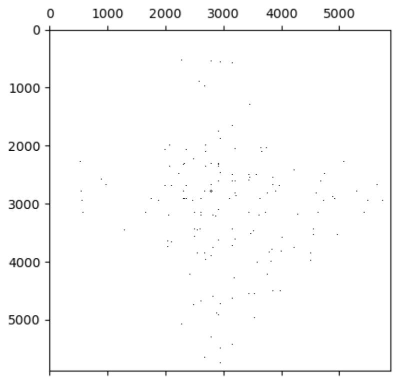

Figure 9: Adjacency Matrix

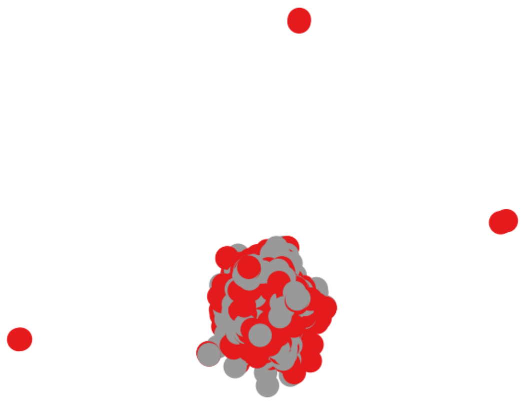

Figure 10: Graph Partition Matrix - 2 Communities

2.2.2 Spectral Decomposition

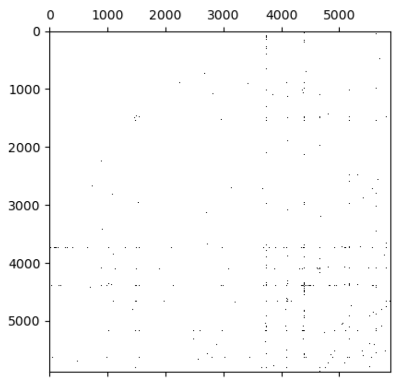

Figure 11: Adjacency Matrix

Figure 12: Graph Partition Matrix for $\beta = 1$, 169 Communities

2.2.3 Louvain Algorithm

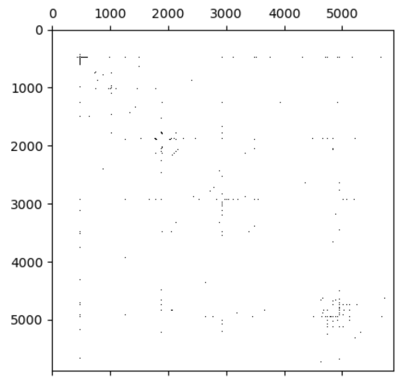

Figure 13: Adjacency Matrix

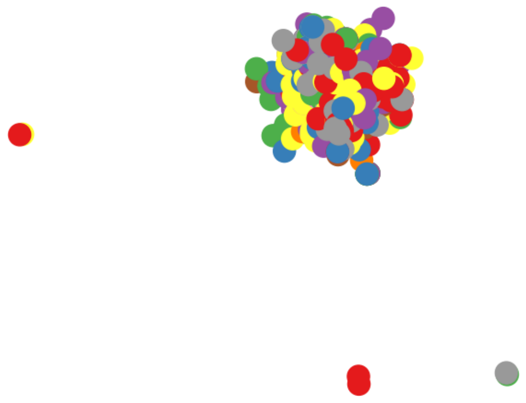

Figure 14: Graph Partition Matrix - 397 Communities

3 Communities formed by the Two Algorithms

- Communities formed by Spectral Decomposition on Facebook Data are **0, 1, 34, 58, 107, 472, 594, 1465, 1505, 1917 - 10** Communities
- Communities formed by Spectral Decomposition on Bitcoin Data are **0, 1, 2, 3, 4, 8, 10, 13, 14, 15, 16, 21, 26, 27, 42, 43, 221, 279, 576 - 19** Communities
- No. of Communities formed by the Louvain Algorithm on Facebook Data and Bitcoin Data are **101** and **397** respectively. They are given in the text file attached

4 Inferences

4.1 What would be the stopping criterion to determine the right set of communities using the spectral decomposition method?

- As per the NCut Algorithm, After finding the Fiedler vector, Partitions should be only made if the Fiedler vector is a stable vector. A stable vector should have abrupt jumps in its components, once they are all sorted
- While implementing the Spectral Decomposition to the given Data, it is observed that in some cases Fiedler vector can become unstable(do not have abrupt jumps in it) and in some cases the

sign of the Fiedler vector is not changing and since we are thresholding with zero, this could cause an issue

- If the Fiedler vector becomes unstable, we should not do the partition. In order to achieve this, the Fiedler vector is first sorted and the consecutive differences are computed and the maximum of consecutive differences is compared with the mean of the consecutive differences multiplied with a factor β
- Let **Max** be the Maximum value of the consecutive differences of the sorted Fiedler vector and **Mean** be the mean of the consecutive differences of the sorted Fiedler vector.
- Then if $\mathbf{Max} \leq \beta * \mathbf{Mean}$, partition is not done and otherwise partition is done. As β increases, the no. of Communities formed decreases. In this paper, experiments are shown by taking $\beta = \mathbf{200}$
- In the case when Fielder vector do not show any sign change, partition is not being done

4.2 How would you pick the best decomposition of nodes into communities?

- Some of the Metrics to pick the decomposition of nodes into communities are Modularity, Normalized Mutual Index
- Modularity measures the quality of a partition of nodes into communities based on the density of edges within communities compared to what would be expected by chance. A higher Modularity score indicates a better partition.
- Normalized Mutual Index measures the similarity between the true community structure (if available) and the detected communities
- Also, we can decide by viewing the communities and the edges between them visually

4.3 What was the running time of the Spectral decomposition algorithm versus the Louvain algorithm on the data sets you were given?

	No. of Communities	**Modularity**	**Elapsed time(secs)**
Facebook Data, Spectral Decomposition	**10**	**0.795**	**20.47**
Facebook Data, Louvain Algorithm	**101**	**0.815**	**2.97**
Bitcoin Data, Spectral Decomposition	**19**	**0.0384**	**30.8**
Bitcoin Data, Louvain Algorithm	**397**	**0.443**	**4.40**

Table 1: Performance of Algorithms on Facebook Data and Bitcoin Data

4.4 Which algorithm gave rise to better communities, why?

- It can be seen that Louvain Algorithm is better than Spectral Decomposition
- Modularity given by Louvain Algorithm is far better than Spectral Decomposition for Bitcoin Data
- Also, Only Phase-1 of Louvian Algorithm is implemented. But still it forms Communities with high Modularity in one-tenth of the time