Fun Statistics

Bryan Xu

1 Global Analysis

- **Density:** Lets us know how *dense* our network is. For a directed graph with n nodes, we will have $n(n-1)$ possible edges supposing we count $A \to B$ and $B \to A$ separately.
- **Degree:** Tells us how many edges are connected to the node. In a directed graph, we have to consider both the "in" degree and the "out" degree - i.e. followers and following. The former tells us how many profiles follow a specific profile, the latter tells us how many other profiles a specific profile follows. We suspect the average in degree to equal the average out degree as every edge connects two nodes in our graph.
  - ↪ We should expect that on average, profiles have as many followers as they are following. This means both the in-degree and out-degree should follow a *power law distribution*. This means that there are more profiles who follow / are followed by a small number of profiles than there are profiles who follow / are followed by a large number of profiles.
  - ↪ If this is the case, then we should have a *scale-free network*: the characteristics of our network are independent of the size of the network. Thus when the network grows, the underlying structure should remain the same.
  - ↪ For the power-law distribution (in contrast with a Normal distribution), the number of nodes with really high numbers of edges is much higher. This means that in networks we will often find a small number of very highly connected nodes. They have a number of connections that would not occur if the distribution would be normal. But generally well connected nodes are more common in a normal distribution.
- **Average Shortest Path Length:** Given two random profiles A and B, what is the minimum number of profiles we must visit to get from A to B? The average on Facebook is 4.74. The will be lower the number, the more connected your friends are on Instagram.

2 Local Analysis

- **Betweenness centrality:** If we follow a shortest path, we traverse through different nodes. The *betweenness centrality* of a certain node is the number of shortest paths that pass through that specific node. As this number scales with the total number of nodes in the network, betweenness centrality is often rescaled by dividing by the number of pairs for which the shortest path doesn't contain the node in question. This result is then further normalised without loss of precision. In other words, betweenness centrality identifies nodes that act as bridges in our network.

  ↪ Mathematically,

  $$g(v) = \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}},$$

  where σ_{st} is the total number of shortest paths from node s to node t and $\sigma_{st}(v)$ is the number of those paths that pass through v.

- **Closeness centrality:** For a specific node, this measure is calculated by taking the reciprocal of the sum of all shortest path lengths between this node and every other node. In other words, it tells us which nodes we should target to reach all the other nodes most quickly.

  ↪ Mathematically,

  $$C(x) = \frac{1}{\sum_y d(y,x)},$$

  where $d(y,x)$ is the distance between vertices x and y.

- **Degree centrality:** Because we are dealing with a directed graph, both in-degree centrality as well as out-degree centrality can be calculated. These centrality measures tell us the number of nodes a specific node is connected to (out-degree centrality) or how many nodes are connected to the specific node (in-degree centrality). Both measures are normalised by dividing by the maximal possible degree. Degree centrality thus tells us which nodes are able to influence the highest number of other nodes directly (one step connections).

  ↪ Mathematically,

  $$C_D(v) = \deg(v),$$

  for a graph $G := (V, E)$ with $|V|$ vertices and $|E|$ edges.

- **Eigenvector centrality:** This centrality measure extends degree centrality by not only taking into account the number of direct connections (degree), but also considering the degree of the connected nodes, the degree of the connected nodes of these connected nodes and so on.

↪ Suppose for example I only knew one person. My degree would be 1 and I wouldn't be considered an important node in the network. If Ryan knew a lot of people and my one friend is Ryan, I'll be able to have great influence in the network. Eigenvector centrality takes these kinds of connections into account.

↪ Mathematically,

$$x_v = \frac{1}{\lambda} \sum_{t \in M(v)} x_t = \frac{1}{\lambda} \sum_{t \in G} a_{v,t} x_t,$$

where $A = (a_{v,t})$ is the adjacency matrix and $M(v)$ is the set of neighbors of v.

◇ **Percolation centrality:** In essence, this form of centrality seeks to determine the importance of a single node in a complex node taking into account changing states. For example, we can have the situation of one account spreading a rumor on Instagram where we can assign each node as either knowing the rumor or not. How important would a specific vertex be in aiding the "percolation" of said rumor?

↪ Mathematically,

$$\mathrm{PC}^t(v) = \frac{1}{N-2} \sum_{s \neq v \neq r} \frac{\sigma_{sr}(v)}{\sigma_{sr}} \cdot \frac{x_s^t}{\sum [x_i^t] - x_v^t}.$$

3 Clustering Methods

Generally, there are two ways to cluster graph networks: aggolmerative methods and divisive methods. Agglomerative methods assign each node to its own cluster and iteratively merge these clusters, while divisive methods assign all nodes to the same cluster and then iteratively split the cluster into new clusters.

- **Louvain method:** This method maximises the modularity of the network. *Modularity* is the fraction of edges within the clusters minus the expected fraction if edges were distributed randomly. A higher modularity thus yields better clustering.

  ↪ First start by calculating the modularity:

  $$Q = \frac{1}{2m} \sum_{ij} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j),$$

  where A_{ij} represents the edge weight between nodes i and j, k_i and k_j are the sum of the weights of the edges, m is the sum of all of the edge weights, c_i and c_j are communities of the node, and δ is the Kronecker delta.

↪ First, each node in the network is assigned to its own community. Then for each node i, the change in modularity is calculated for removing i from its own community and moving it into the community of each neighbor j of i. This value is easily calculated by two steps: (1) removing i from its original community, and (2) inserting i to the community of j. The two equations are quite similar, and the equation for step (2) is:

$$\Delta Q = \left[\frac{\Sigma_{in} + 2k_{i,in}}{2m} - \left(\frac{\Sigma_{tot} + k_i}{2m} \right)^2 \right] - \left[\frac{\Sigma_{in}}{2m} - \left(\frac{\Sigma_{tot}}{2m} \right)^2 - \left(\frac{k_i}{2m} \right)^2 \right].$$

- **Girvan-Newton method:** Girvan-Newman is an iterative method and it is based on edge betweenness. Just as with node betweenness, we can calculate edge betweenness as the number of shortest paths between two nodes that traverse a specific edge. Girvan-Newman starts by calculating the edge betweenness for every edge (1). It then removes the edge with the highest edge betweenness (2). After this step, the betweenness of all edges affected by the removal is recalculated (3). Step (2) and (3) are repeated until no more changes happen. The result is a dendrogram (like a family tree). The deeper you go in the tree, the more clusters you will have. To determine the depth of the Girvan-Newman method, I calculated the modularity for the clustering at each depth. If the newly calculated modularity was higher than the previous one, we go one step deeper. This process is repeated until the modularity stops increasing.