ECE 232E Lecture 7 and Lecture 8 notes

Professor Vwani Roychowdhury

May 11, 2018

In this set of lecture notes, we will study the various properties of Erdos Renyi networks, community detection algorithms, and pagerank algorithm for undirected and directed networks.

1 Erdos Renyi network

In order to create an Erdos Renyi network, we start with a completely connected network and cut each edge in the completely connected network with probability $1 - p$. This is equivalent to saying that we keep each edge of a completely connected graph with probability p.

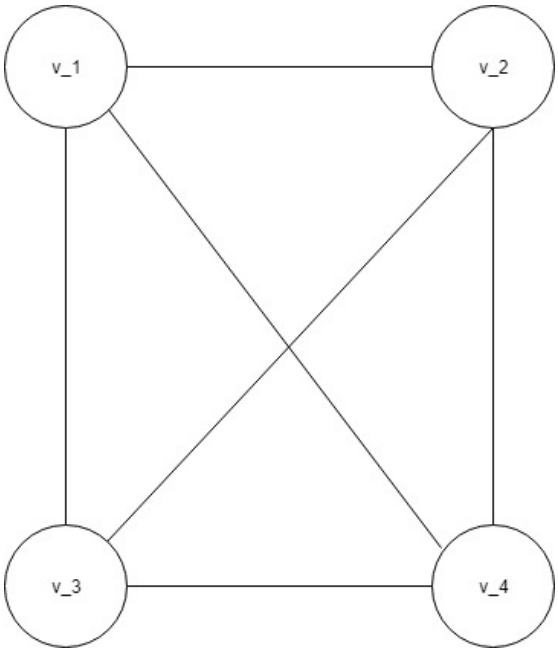

Figure 1: Completely connected network

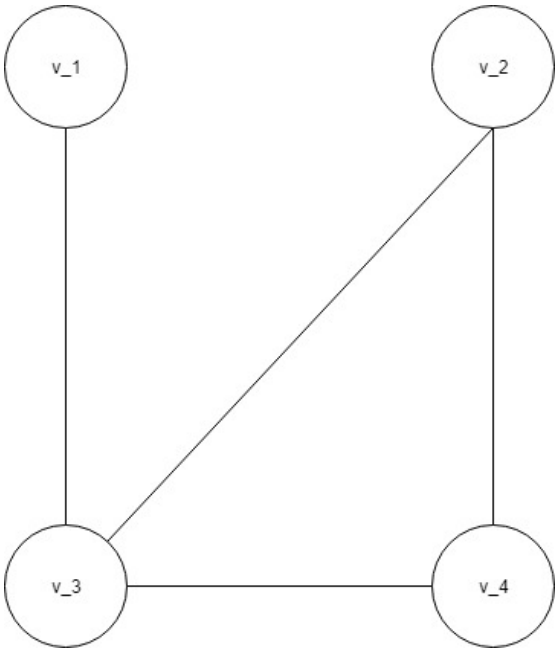

Figure 2: Erdos Renyi network obtained by cutting edges of completely connected network

1.1 Structural properties

Having defined the network creation process, we will now study some of the structural properties of Erdos Renyi networks. To be specific, we will explore

- Degree distribution
- Connectivity

1.1.1 Degree distribution

Let,

$$P_k = \mathbb{P}(\text{a randomly chosen node has degree } k)$$

Then, the expression for P_k is given by,

$$P_k = \binom{n}{k} p^k (1-p)^{n-k} \tag{1}$$

where n is the number of nodes in the network and p is the probability with which an edge is retained in the completely connected network. From equation 1, it can be inferred that Erdos Renyi networks have a binomial degree distribution. From the statistics of a binomial distribution, we have

$$\alpha = np \tag{2}$$

where α is the average degree of the Erdos Renyi network. For n large and α small, we have

$$P_k = \frac{e^{-\alpha}\alpha^k}{k!} \tag{3}$$

From equation 3, it can be inferred that the degree distribution of a large network with small average degree is poisson.

1.1.2 Connectivity

Connectivity of Erdos Renyi network depends on the value of np. Depending on the value of np, we can have

- Erdos Renyi network consists of very small size connected components (CC) where each such component has only $\ln n$ nodes. Thus the size of the largest CC as a fraction of the number of nodes, n, goes to zero as $n \to \infty$.
- Erdos Renyi network has a giant connected component (GCC), whose size is $O(\epsilon(p)n)$, and thus the GCC has a fractional size of $0 < \epsilon(p) \leq$ as $n \to \infty$.
- Erdos Renyi network is connected with probability 1 as $n \to \infty$.

For visualization purposes, the plot of the expected GCC size as a function of np is given in figure 3, as $n \to \infty$.

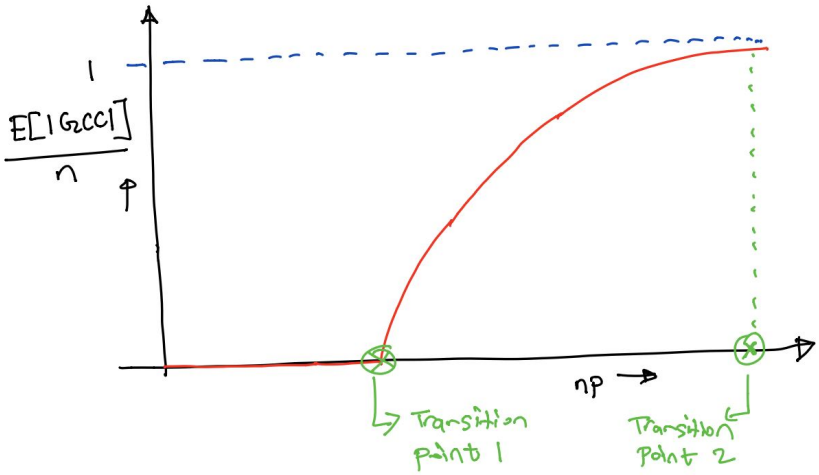

Figure 3: $\frac{\mathbb{E}(|GCC|)}{n}$ versus np plot

The value of np at which a giant connected component evolves in the Erdos Renyi network is defined as the Transition point 1. At Transition point 1,

$$np = 1 \tag{4}$$

$$\mathbb{E}(|GCC|) \propto \sqrt[2]{n}\ . \tag{5}$$

For any value of $np > 1$ the GCC has size $\epsilon(np)n$, where $< 0\epsilon(np) \leq n$. The value of np for which,

$$\lim_{n\to\infty} \mathbb{P}(|GCC| = n) = 1 \tag{6}$$

is defined as the Transition point 2. At Transition point 2,

$$np = c\ ln(n), \quad c > 1. \tag{7}$$

Now let's first show that the value of np given by equation 7 leads to a network with no disconnected single nodes. We have,

$$\mathbb{P}(i^{th} \text{ node is disconnected}) = (1-p)^{n-1}$$

For $x > 0$,

$$(1-x) < e^{-x}$$

Then,

$$(1-p)^{n-1} < e^{-p(n-1)}$$

If $p = c\frac{ln(n)}{n}$, then

$$\begin{aligned}(1-p)^{n-1} &< (e^{-ln(n)})^{c(1-\frac{1}{n})} \\ &= (\frac{1}{n})^{c(1-\frac{1}{n})}\end{aligned} \tag{8}$$

Then using the inequality given by equation 9 and the union bound, we have

$$\begin{aligned}\mathbb{P}(\text{any node is disconnected}) &\leq \sum_{i=1}^{n} \mathbb{P}(i^{th} \text{ node is disconnected}) \\ &\leq n(\frac{1}{n})^{c(1-\frac{1}{n})} \\ &= \frac{1}{n^{(c-1)}n^{-\frac{1}{n}}}\end{aligned} \tag{9}$$

Then taking the limit of equation 10, and using the fact that $c > 1$, we have

$$\lim_{n\to\infty} \mathbb{P}(\text{any node is disconnected}) = 0 \tag{10}$$

Therefore, from equation 11 it can inferred that the network does not have any disconnected single nodes when $p = c\frac{ln(n)}{n}$. Now let's show that the value of np given by equation 7 leads to a a connected network. First we note that a connected graph always has a spanning tree. The following theorem characterizes the number of spanning trees in a completely connected graph.

Cayley's theorem: A completely connected graph with k nodes has $k^{(k-2)}$ spanning trees.

Now we put an upper bound on the number of connected components of size k: (i) First we note that there are $\binom{n}{k}$ sets of k nodes; (ii) Next for each choice of k nodes, if we create a connected component (CC) by selecting a subset of edges from the completely connected subgraph, then it must have at least one spanning tree; (iii) By Cayley's theorem there are $k^{(k-2)}$ labeled spanning trees. Hence, the total number of CC for any choice of k nodes $\leq k^{(k-2)}$. This is because, for any given connected component, one can have multiple spanning trees. So the total number of spanning trees provides an upper bound on the number of connected components that one can form from a completely connected graph by choosing different sets of edges. So we get:

$$(\text{Number of distinct connected components of size} = k) \leq \binom{n}{k} k^{k-2}. \quad (11)$$

Now an upper bound on the probability of getting any given CC of size $= k$ can be computed from the following observations: (i) Pick any of its spanning trees. It must have all the $(k-1)$ edges in it, and the probability of this event is $(k-1)^p$; (ii) The probability of picking the rest of the edges in the CC (not included in the spanning tree) is ≤ 1; and (iii) All the edges to the $(n-k)$ nodes that are not in the CC must be cut; since there are $(n-k) * k$ such edges, the probability of this event is $p^{k(n-k)}$. Thus we get:

$$\mathbb{P}(\text{Creating a particular CC of size =k}) \leq p^{k-1}(1-p)^{(n-k)k} \quad (12)$$

Now combining the above two equations and using union bound we get:

$$\mathbb{P}(\text{there exists a connected component of size} = k) \leq \binom{n}{k} k^{k-2} p^{k-1} (1-p)^{(n-k)k} \quad (13)$$

Now plugging in $p = c\frac{ln(n)}{n}$ into equation (13) and then (i) using stirling's formula, and (ii) and optimizing for $2 \leq k \leq \frac{n}{2}$, we get:

$$\mathbb{P}(\text{there exists a connected component of size} = k \text{ (where } 2 \leq k \leq \frac{n}{2}) \leq n^{1-2c} \ .$$

Then, by union bound

$$\begin{aligned} \mathbb{P}(\text{there is any connected component of size } 2 \leq k \leq \frac{n}{2}) &\leq \frac{n}{2} n^{1-2c} \\ &= \frac{1}{2} n^{2(1-c)} \end{aligned} \quad (14)$$

Since $c > 1$, by taking the limit of equation (14), we get

$$\lim_{n\to\infty} \mathbb{P}(\text{there is any connected component of size } 2 \leq k \leq \frac{n}{2}) = 0 \quad (15)$$

From equation 15, it can be inferred that if $p = c\frac{ln(n)}{n}$, then $\mathbb{P}(|GCC| = n) = 1$ and hence the network is connected with probability 1.

2 Community structure of networks

Given a network G, we want to find a clustering of the nodes in the network such that the number of inter cluster edges is much smaller than the number of intra cluster edges. This is an intuitive definition for finding community structures in networks.

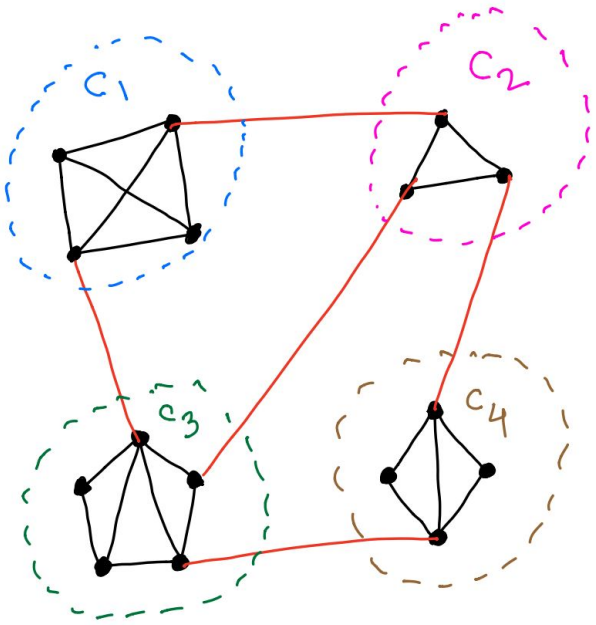

Figure 4: Community structure of a 16 node network

From figure 4, it can be observed that the number of inter cluster edges (colored in red) is much smaller than the number of intra cluster edges (colored in black). Having provided an intuitive definition for finding community structures in networks, now we will describe a mathematical framework for finding community structures in networks.

2.1 Mark Newman modularity index

The modularity of a cluster C_i is defined as the difference between the number of edges in C_i and the expected number of edges in C_i if the network was created randomly with the same degree distribution. Mathematically,

$$m_{C_i} = f_{C_i} - r_{C_i} \tag{16}$$

where m_{C_i} is the modularity score of cluster C_i, f_{C_i} the number of edges in C_i, and r_{C_i} is the expected number of edges in C_i if the network was created

randomly with the same degree distribution. The expression for f_{C_i} is given by

$$f_{C_i} = \sum_{i,j \in C_i} A_{ij} \tag{17}$$

where A is the node-node incidence matrix. Now we need to find an expression for r_{C_i}.

The way a random network with the same degree distribution is created can be described as follows: For each node with degree k_i we create k_i stubs. So we have $2m$ such stubs, where m is the number of edges in the network This follows from the identity that the sum of degrees of nodes in a graph equals twice the number of edges in it. Next, to create an edge in the random network, two stubs are picked randomly, and then these two stubs are connected to create an edge. This is repeated m times to create m edges and generate an instance of a random network that has the same degree sequence as the given network. For the ease of computations lets assume that this process is done with replacement.

Now, let's assume that nodes i and j with degrees k_i and k_j respectively belong to cluster C_i.

$$\mathbb{P}(\text{picking a stub belonging to the } i^{th} \text{ node}) = \frac{k_i}{2m} \ .$$

Similarly,

$$\mathbb{P}(\text{picking a stub belonging to the } j^{th} \text{ node}) = \frac{k_j}{2m}$$

Then, for m picks, the expected number of edges between node i and j under random stub matching is given by equation 18:

$$\mathbb{E}(\text{number of edges between node } i \text{ and } j) = 2m\frac{k_i k_j}{(2m)^2} \tag{18}$$

Therefore, the expression for r_{C_i} is given by,

$$r_{C_i} = \sum_{i,j \in C_i} \frac{k_i k_j}{2m} \tag{19}$$

Plugging the expressions for f_{C_i} and r_{C_i} into equation 15, we get

$$m_{C_i} = \sum_{i,j \in C_i} (A_{ij} - \frac{k_i k_j}{2m}) \tag{20}$$

Having derived an expression for the modularity of a cluster C_i, now we define the modularity index of a network partitioning, $P = \{C_1, \cdots, C_k\}$, (i.e. the network is separated into k disjoint clusters) denoted by $Q(P)$, as

$$\begin{aligned} Q(P) &= \sum_{C_i} \frac{1}{2m} m_{C_i} \\ &= \sum_{C_i} \frac{1}{2m} \sum_{i,j \in C_i} (A_{ij} - \frac{k_i k_j}{2m}) \end{aligned} \tag{21}$$

The goal of the network clustering algorithm is to find a partitioning of the network that has the highest modularity index. Mathematically,

$$\hat{P} = \underset{P}{argmax}\ Q(P) \tag{22}$$

where $\hat{P}$ is the network partitioning with the highest modularity index.

2.2 Greedy algorithm for modularity index maximization

The basic idea of this algorithm is to maximize the modularity index in a greedy manner.

1. Initially you start with all nodes as individual clusters.
2. Consider all pairs of clusters and compute the change in $Q(P)$ if the pair is merged into a single cluster. Merge a pair of clusters that leads to a maximum increase in $Q(P)$ after merging.
3. Repeat step 2 until no pair merges lead to an increase in $Q(P)$.

2.3 Randomized algorithm for modularity index maximization

In this algorithm, you do not merge the clusters deterministically. Rather, the merge is done probabilistically.

Let ΔQ_i be the change in the modularity if the i^{th} pair is merged. A merge is picked with probability $\propto e^{\frac{\Delta Q_i}{kT}}$, where KT is a constant (remiscent of the Boltzman's constant in statistical physiscs). This way of merging probabilistically, gives the following expression

$$\mathbb{P}(\text{selecting a merge with } \Delta Q_i \text{ change in } Q(P)) = \frac{e^{\frac{\Delta Q_i}{kT}}}{\sum_{j=1}^{|M|} e^{\frac{\Delta Q_j}{kT}}} \tag{23}$$

where $|M|$ is the number of possible merges. Thus each run of this algorithm would lead to a different clustering and also a different value of the modularity index. One can ten pick the partitioning with maximum final modulaity index. Another advantage of this method is that it can isolate the core nodes for each community: In every run of this algorithm, you get core nodes as belonging to the same community and the peripheral nodes are arbitrarily assigned to different communities.

2.4 Upper bound on modularity index

In this section, we will derive an upper bound on the modularity index $Q(P)$. Let's start with equation 20,

$$\begin{aligned} Q(P) &= \sum_{C_i} \frac{1}{2m} \sum_{i,j \in C_i} (A_{ij} - \frac{k_i k_j}{2m}) \\ &= \frac{1}{2m} \sum_{C_i} 2\Big(l_{C_i} - r_{C_i} \Big) \quad (l_{C_i} \text{ is the number of edges in } C_i) \\ &= \sum_{C_i} \left(\frac{l_{C_i}}{m} - \frac{r_{C_i}}{m} \right) \\ &\leq 1 \quad (\text{Since } \sum_{i=1}^{|P|} l_{C_i} \leq m) \end{aligned}$$

3 Pagerank algorithm for undirected and directed networks

In this section, we will describe the Pagerank algorithm for undirected and directed networks.

3.1 Pagerank algorithm for undirected networks

If the undirected network is connected, then we can obtain the pagerank score of node i by doing a random walk on the undirected network for a sufficiently large number of steps. From lecture 3 and 4 notes, we know that the steady state occupancy vector is given by equation 23

$$\pi(i) = \frac{k_i}{\sum_{j=1}^{|V|} k_j} = \frac{k_i}{2|E|}, \quad i = 1, 2, 3, \cdots, |V| \tag{24}$$

where k_i is the degree of the i^{th} node. Then,

$$\text{Pagerank score of node } i = \pi(i) \tag{25}$$

3.2 Pagerank algorithm for directed networks

To facilitate the derivation of the pagerank algorithm for directed networks, let's introduce some notation:

P_{ij} : Probability of going to node j given that you are at node i
P : Node transition matrix with entries P_{ij}
$k_{out}(i)$: Out-degree of node i
$k_{in}(i)$: In-degree of node i
A : Node-Node incidence matrix

Then with the above notation, we have the following expressions for Out-degree and In-degree

$$k_{out}(i) = \sum_{j=1}^{|V|} A_{ij}, \quad i = 1, 2, \cdots, |V| \tag{26}$$

$$k_{out}(i) = \sum_{j=1}^{|V|} A_{ji}, \quad i = 1, 2, \cdots, |V| \tag{27}$$

and the expression for the entries of the node transition matrix is given by

$$P_{ij} = \frac{1}{k_{out}(j)} A_{ij} \tag{28}$$

If we define,

$$\Sigma_{out} = diag(\frac{1}{k_{out}(1)}, \frac{1}{k_{out}(1)}, \cdots, \frac{1}{k_{out}(|V|)}) \tag{29}$$

then the expression for the node transition matrix P is given by,

$$P = (\Sigma_{out})^{-1} A \tag{30}$$

We can find the steady state node occupancy probability vector by solving the self consistency equation given by 30

$$\pi(i) = \sum_{j=1}^{|V|} P_{ji} \pi(j), \quad i = 1, 2, \cdots, |V| \tag{31}$$

Equation 30 can be expressed in matrix form,

$$\pi = P^T \pi \tag{32}$$

Once we have found the solution to the self consistency equation, then

$$\text{Pagerank score of node } i = \pi(i) \tag{33}$$

In a directed network, the existence of a solution to the self consistency equation is not guaranteed. A sufficient condition for the existence of a solution is that for any pair of nodes i and j there exists at least one directed path from i to j and at least one directed path from j to i. In other words, there exists a solution to the self consistency equation if the network is strongly connected. In the next section, we describe a trick to find the pagerank scores of nodes in networks that are not necessarily strongly connected.

3.3 Teleportation trick

A random walk with teleportation is described below:

At any node you randomly follow one of the outgoing edges with probability $1-\alpha$ and with probability α you go to a random node in the network.

Let's derive the self consistency equation for random walks with teleportation. The transition probabilities for random walks with teleportation is given by equation 33

$$P_{ij} = (1-\alpha)\frac{1}{k_{out}(i)}A_{ij} + \alpha\frac{1}{n} \tag{34}$$

Then, the self consistency equation is given by

$$\begin{aligned}\pi(i) &= (1-\alpha)\sum_{j=1}^{n}\frac{1}{k_{out}(j)}A_{ji}\pi(j) + \alpha\sum_{j=1}^{n}\frac{1}{n}\pi(j) \\ &= (1-\alpha)\sum_{j=1}^{n}\frac{1}{k_{out}(j)}A_{ji}\pi(j) + \frac{\alpha}{n}\end{aligned} \tag{35}$$

Solution to the self consistency equation 34 is the pagerank score of node i and the solution is guaranteed to exist. Pagerank score of a node can be thought of as the trust score of the node. In this teleportation trick α is a parameter that can be tuned. Google's pagerank algorithm uses $\alpha = 0.15$.

3.4 Variations of pagerank algorithm

In this part, we will analyze multiple variations to the pagerank algorithm.

3.4.1 Teleportation to a set of trusted nodes

A random walk with teleportation to a set of trusted nodes is described below:

At any node you randomly follow one of the outgoing edges with probability $1-\alpha$ and with probability α you go to a trusted node in the network.

Let's derive the self consistency equation for this variation. Let the set of trusted nodes be denoted by T. If $i \in T$, then

$$\begin{aligned}\pi(i) &= (1-\alpha)\sum_{j=1}^{n}\frac{1}{k_{out}(j)}A_{ji}\pi(j) + \alpha\sum_{j=1}^{n}\frac{1}{|T|}\pi(j) \\ &= (1-\alpha)\sum_{j=1}^{n}\frac{1}{k_{out}(j)}A_{ji}\pi(j) + \frac{\alpha}{|T|}\end{aligned} \tag{36}$$

If $i \notin T$, then

$$\pi(i) = (1-\alpha)\sum_{j=1}^{n}\frac{1}{k_{out}(j)}A_{ji}\pi(j) \tag{37}$$

From the above expressions, it can be seen that the nodes in T will get a boost in their pagerank scores. Also, if a node gets an in-link from a node with a high pagerank score, then it's own pagerank score is amplified. Therefore, sometimes webpages pays to get in-links from trusted webpages.

3.4.2 Personalized pagerank

Suppose we have a directed network of webpages, where each node is a url. We cluster the webpages belonging to a particular topic and denote this cluster as C. Then we can compute the personalized pagerank for this topic in the following manner:

At any node you randomly follow one of the outgoing edges with probability $1-\alpha$ and with probability α you go randomly to one of the nodes in C.

The self consistency equation for this variation is same as equations 35 and 36 with $|T|$ replaced by $|C|$. We can compute the relevance of the topic to a page i by defining the significance measure

$$\text{Significance measure} = \log\left(\frac{\pi^*(i)}{\pi(i)}\right) \tag{38}$$

where $\pi^*(i)$ is the personalized pagerank score of node i(computed using equation 35) and $\pi(i)$ is the pagerank score of node i (computed using equation 36).

3.4.3 Fixing the probability for end node of teleportation

In this variation, we fix the probability for the end node of the teleportation to be $\pi(i)$. With this variation, the self consistency equation is given by,

$$\begin{aligned}
\pi(i) &= (1-\alpha)\sum_{j=1}^{n}\frac{1}{k_{out}(j)}A_{ji}\pi(j) + \alpha\sum_{j=1}^{n}\pi(i)\pi(j) \\
&= (1-\alpha)\sum_{j=1}^{n}\frac{1}{k_{out}(j)}A_{ji}\pi(j) + \alpha\pi(i) \\
&= \sum_{j=1}^{n}\frac{1}{k_{out}(j)}A_{ji}\pi(j)
\end{aligned} \tag{39}$$

From the above derivation, we can observe that we have a consistency equation as if there is no teleportation. Therefore, this result cancels the effect of teleportation.