SF5 Interim Report Two

Part IIA - SF5: Networks, friendship, and disease

Edited By

Kyle McMillan
kcm40

University Of Cambridge

Classes

```
class Network(object):
    def __init__(self , num_nodes):
        self.adj = {i:set() for i in range (num_nodes)}

    def add_edge(self , i , j):
        self.adj[i].add(j)
        self.adj[j].add(i)

    def neighbors(self , i):
        return self.adj[i]

    def edge_list(self):
        return [(i,j)for i in self.adj for j in self.adj[i] if i < j]

    def num_edges(self):
        return sum([len(self.adj[i]) for i in self.adj])//2

    def find_component(self, i):
        c = set()
        stack = [i]
        while stack:
            node = stack.pop()
            c.add(node)
            stack.extend(self.neighbors(node) - c)
        return c

    def degree_distributions_individual(self):
        return np.array([len(self.adj[i]) for i in self.adj])

    def degree_distributions(self):
        degrees = [len(self.adj[i]) for i in self.adj]
        return pd.Series(degrees).value_counts().sort_index()

    def friends_degree_distribution(self):
        return np.array([np.mean([len(self.adj[j]) for j in self.adj[i]]) for i in self.adj])

class configuration_graph(Network):
    def __init__(self , num_nodes , degree_sequence):
        super().__init__(num_nodes)
        S = np.array([ i for i in range (num_nodes) for _ in range (degree_sequence[i])])
        S = np.random.permutation(S)
        if len (S) % 2:
            S = S[:-1]
        S = S.reshape(-1 ,2)
        for i , j in S:
            self.add_edge(i , j)

class poisson_configuration_graph(Network):
    def __init__(self , num_nodes , lambda_):
        super().__init__(num_nodes)
        S = np.random.poisson(lambda_ , num_nodes)
        S = np.array([ i for i in range (num_nodes) for _ in range (S[i])])
        S = np.random.permutation(S)
        if len (S) % 2:
            S = S[:-1]
        S = S.reshape(-1 ,2)
        for i , j in S:
            self.add_edge(i , j)

class geometric_configuration_graph(Network):
    def __init__(self , num_nodes , p):
        super().__init__(num_nodes)
        S = np.random.geometric(p , num_nodes) - 1
        S = np.array([ i for i in range (num_nodes) for _ in range (S[i])])
        S = np.random.permutation(S)
        if len (S) % 2:
            S = S[:-1]
        S = S.reshape(-1 ,2)
        for i , j in S:
            self.add_edge(i , j)

class popular_poisson_configuration_graph(Network):
def __init__(self , num_nodes , lambda_, popular_nodes_num):
    super().__init__(num_nodes)
    S = np.random.poisson(lambda_ , num_nodes)
    S = np.array([ i for i in range (num_nodes) for _ in range (S[i])])
    S = np.random.permutation(S)
    if len (S) % 2:
        S = S[:-1]
    S = S.reshape(-1 ,2)
    for i , j in S:
        self.add_edge(i , j)
```

```
popular_nodes = []
temp = self.adj.copy()
for i in range(popular_nodes_num):
    popular_nodes.append(max(temp, key=lambda x: len(temp[x])))
    temp.pop(max(temp, key=lambda x: len(temp[x])))
for i in popular_nodes:
    for j in popular_nodes:
        if i != j:
            self.add_edge(i, j)
```

Images

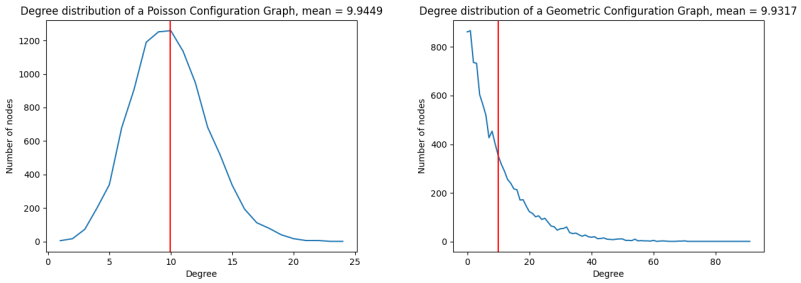

Figure 1: Degree Left: Poisson distribution, Right: Geometric distribution

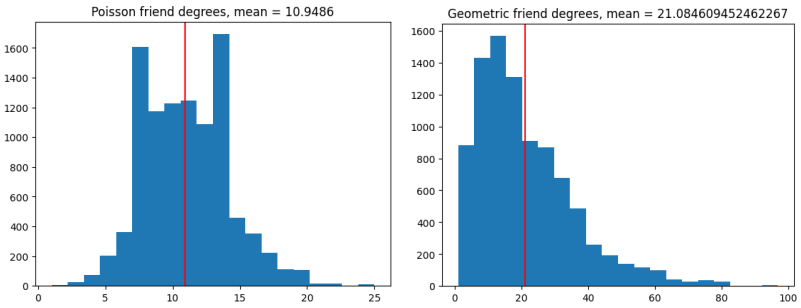

Figure 2: Friend Degree Left: Poisson distribution, Right: Geometric distribution

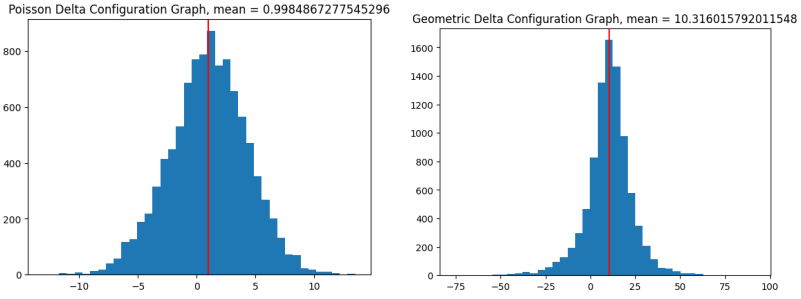

Figure 3: Delta Graph Left: Poisson distribution, Right: Geometric distribution

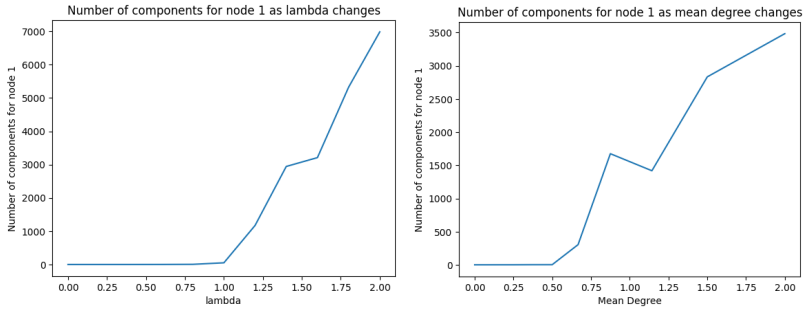

Figure 4: Components Left: Poisson distribution, Right: Geometric distribution

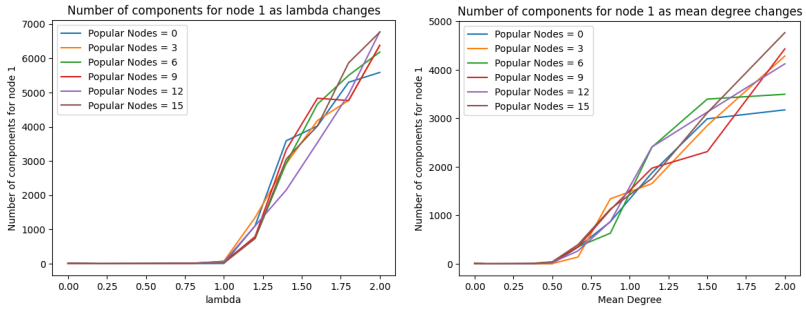

Figure 5: Popular Components Left: Poisson distribution, Right: Geometric distribution

1) For a configuration model network sample the degree distributions created using a poisson and geometric distribution with mean 10 and 10000 nodes.

Code:

```
degrees = [poisson_configuration_graph(10000, 10).degree_distributions()]
# degrees = [geometric_configuration_graph(10000, 1/11).degree_distributions()]
degrees = pd.DataFrame(degrees)
combined_degrees = degrees
values = combined_degrees.values.flatten()
#print(values)
coloumns = combined_degrees.columns
```

Figure 1 shows the degree distributions. This code generates a configuration model network with 10000 nodes and mean 10 degree distribution using a poisson and geometric distribution. The degree distributions are then plotted in the figure above. The poisson distribution is on the left and the geometric distribution is on the right. We can see that the means are roughly 10 for both distributions and both distributions follow the pmf.

2) Friend degree distribution

Code:

```
friend_degrees_poisson = []
for i in range(10000):
    neighbours = np.array(list(poisson_config_graph.neighbors(np.random.randint(0, 10000))))
    if len(neighbours) > 0:
        friend_degrees_poisson.append(len(poisson_config_graph.neighbors(np.random.choice(neighbours))))
        i-=1
```

Figure 2 shows the friend graph. From this can see the friend degree distribution is a lot larger for the geometric distribution than the poisson distribution but both are higher than the average degree distribution.

3) $\Delta_i = \kappa_i - k_i$

k_i - degree of node i κ_i - average degree of neighbours of node i

$$\kappa_i = \frac{1}{k_i}\sum_j A_{ij}k_j$$

In this histogram nodes of degree 0 are excluded as they would have a κ_i of infinity using the above equation. Code:

```
delta_friends = poisson_config_graph.friends_degree_distribution() - poisson_config_graph.degree_distributions_indiv
zero_index = np.where(poisson_config_graph.degree_distributions_individual() == 0)
delta_friends = np.delete(delta_friends, zero_index)
```

Figure 3 shows the delta graph. Results as expected. The Δ_i for the poisson distribution is centred around 1 and the geometric distribution is centred around 10. But both graphs similar to be a normal distribution.

4) Proving q_k - the probability that a randomly chosen neighbour k.

$$q_k = P(k_j = k | k_i -> k_j) \tag{1}$$

$$q_k = \frac{\text{Number of nodes with degree k that are neighbours of a node with degree i}}{\text{All degrees}} \tag{2}$$

$$q_k = \frac{kn_k}{\sum_{k_2} k_2 n_{k_2}} = \frac{kp_n n}{n\sum_{k_2} k_2 p_{k_2}} = \frac{kp_n}{\sum_{k_2} k_2 p_{k_2}} \tag{3}$$

$$E[\Delta_i] = \frac{1}{\mu}\sum_k k^2 p_k - \mu = \frac{\sigma^2}{\mu} \tag{4}$$

5) q_k and generating function

$$G'(z) = \sum_{k=1}^{\infty} k z^{k-1} p_k$$

$$Q(z) = \frac{\sum_k z^k p_k k}{\sum_{k_2} k_2 p_{k_2}} \tag{5}$$

$$Q(z) = \frac{zG'(z)}{G'(1)} \tag{6}$$

Poisson distribution:

$$G(z) = e^{\lambda(z-1)} \tag{7}$$

$$G'(z) = \lambda e^{\lambda(z-1)} \tag{8}$$

$$Q(z) = z e^{\lambda(z-1)} \tag{9}$$

This is zP(z) so is p_{k+1} - corresponds to Delta = 1 a lambda should be increased by 1. Geometric distribution:

$$G(z) = \frac{p}{1-(1-p)z} \tag{10}$$

$$G'(z) = \frac{p(1-p)}{(1-(1-p)z)^2} \tag{11}$$

$$Q(z) = \frac{zp^2}{(1-(1-p)z)^2} \tag{12}$$

This is $(k+2)p^2(1-p)^{k+2}$ - which is geometric distribution squared and added 2 to k.

6) Components

Code:

```
for i in range(11):
    num_components_one = [len(geometric_configuration_graph(10000, 1 - 2/3 * i/10).find_component(1)) for _ in range
    components.append(np.mean(num_components_one))
```

Figure 4 shows the components. The poisson graph starts increasing at 1 and the geometric graph starts increasing at 0.5. This makes sense as expected number of neighbours for the poisson distribution is lambda and so as long as a node has one edge it's component will increase as friends more popular. Same argument for geometric distribution but as the expected number of edges is $\frac{1-p}{p}$ this is only over 1 for p = 0.5.

7) Popular People

For this it is likely all popular people are in the same component. Then other people will be effectively be by themselves, with a few unpopular people or effectively small offshoots connected to the popular people. For both distributions have modelled it so people a certain number of the most popular people are connected together (increases their degree by 1 but ignorable). The starting point where component size increases is the same as the previous question but the rate of increase is much higher.