INDIAN INSTITUTE OF TECHNOLOGY MADRAS

DEPARTMENT OF PHYSICS

SUMMER PROJECT REPORT

A Foray into Networks and Power Law Distributions

Author:
Gaurav VAIDYA

Supervisor:
Prof. Neelima M GUPTE

August 7, 2020

Contents

A Foray into Networks and Power Law Degree Distributions

August 7, 2020

1 Networks - Basics

A network is a collection of nodes and edges that is intended to be a one-to-one mapping between the components of a system and the interactions between these components.

A network essentially reduces a complex system to an abstract object whose various properties can be studied mathematically. An interesting point to note is that two completely unrelated systems may be modelled using the same kind of network. We can thus draw parallels between different systems. The network representation proves to be a useful method to study systems because of the wide range of mathematical tools that have been developed over the years in the study of graphs. A lot of network parameters have been used in this report to study some commonly encountered networks and a brief background on those parameters is given below. [2]

1. Average Shortest Path Length - the average number of steps along the shortest path for all pairs of network nodes.
2. Clustering Coefficient - the fraction of paths of length two that are closed.
3. Degree Distribution - the frequency distribution of node degrees, often normalised so that its value always lies between 0 and 1.

2 Small World Networks

Small World networks are systems that lie somewhere in the middle of random networks and regular networks. These systems have high clustering

coefficients like the regular lattices and yet have small average path lengths like random networks.

Small World networks can be used to model problems related to spread of diseases and social networks due to direct contact between pairs of individuals. Network metrics like average path lengths and the clustering coefficient are extremely useful. An interesting problem using Small World networks was modelling the rate of transport (in this case, the spread of diseases) [5].

3 Modelling of Small World Networks

In their paper, Watts and Strogatz devised a way to construct Small World networks by starting with a regular lattice with n nodes each with a degree k, ie. each node is connected to k of its immediate neighbours. Now, if each edge is rewired with a probability p ensuring that the graph does not become disconnected. (The following condition guarantees this.

$$k \gg \ln(n)$$

We see that the network progresses from an ordered system at $p = 0$ to a random one at $p = 1$. The intermediate values of p corresponds to the Small World region.

The behaviour of these networks can be characterized by two properties. The first being the clustering coefficient (a measure of how well the neighbours of a node are connected to each other), which is a local property and the second, the average shortest path length (the average shortest path between any two pairs of nodes) which is global property of the network. We see that an increase in p has a significant impact on L as it decreases in a nonlinear manner with p. Whereas, C is not significantly affected. These results suggest that at a local level, it is almost impossible to detect the change into a Small World network [5].

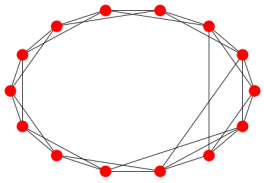

Figure 1: p = 0.1

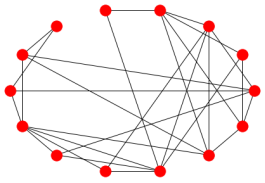

Figure 2: p = 0.6

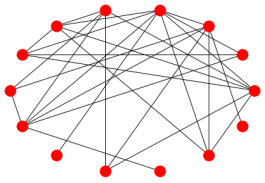

Figure 3: p = 1

To study these network's properties, a regular ring lattice can be rewired using a simple rewiring algorithm shown below and the parameters can be measured using built-in function in Python. The rewiring algorithm used resembles the Erdős–Rényi model; it does not preserve the degree sequence, but keeps the total number of nodes constant. Another advantage of this algorithm is that it ensures that the graph stays connected on random rewiring, even if the above condition between n and k (as given above) is not satisfied.

```
def rewiring(g,p):
    #g is the graph and p is
    probability of rewiring each edge
    global ng
    ng = g
    for i in ng.edges():
        pcomp = rnd.random()
        if pcomp > p:
            pass
        else:
            nodes = list(ng.nodes())
            while True:
                nedge = rnd.sample(nodes,1)
                if ng.has_edge(i[0],nedge[0]):
                    continue
                else:
                    ng.remove_edge(i[0],i[1])
                    ng.add_edge(i[0],nedge[0])
                    if nx.is_connected(ng) == True:
                        break
                    else:
                        ng.remove_edge(i[0],nedge[0])
                        ng.add_edge(i[0],i[1])
                        break
```

The properties of Small World networks generated using this algorithm resembled those that were done using the Watts Strogatz model, but the run time was reduced by 1 s (3 min 20s vs 4 min 28s).

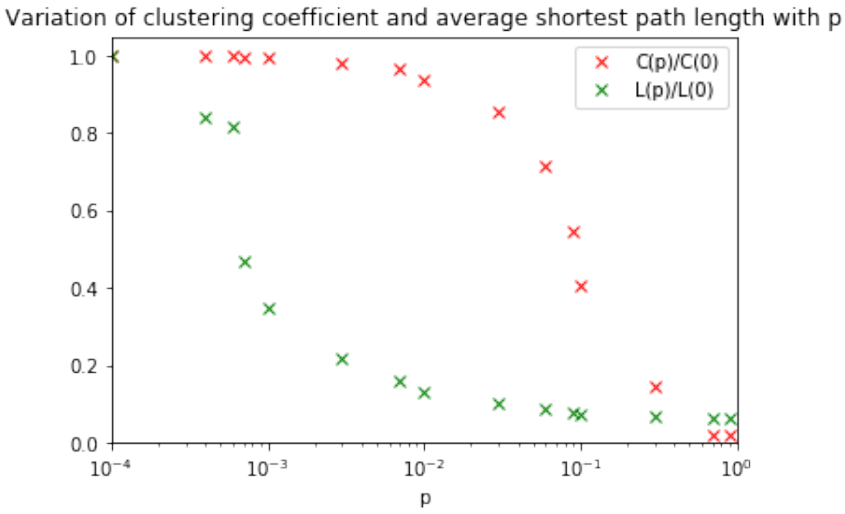

Figure 4: $n = 1000$ and $k = 10$

4 Real Small World Networks

Real world networks are often sparse, i.e. the number of edges is much smaller than the maximum number of edges the graph can have. In other words, $E \ll E_{max}$, where $E_{max} = N(N-1)/2$. This means that even a small number of shortcuts in the network can drastically reduce the average shortest path length, making the system small world. The neural network of the C. Elegans worm and the Power Grid of the United States [1] are two such networks whose L and C values are calculated below. To actually verify whether these networks are small world, we need to compare their L and C values with random networks ($p = 1$). In order to achieve this, a random rewiring of the network is done using the same algorithm as mentioned above.

Network	N	$k_{average}$	L(act)	C(act)	L(rand)	C(rand)
Power Grid	4941	2.67	18.9891	0.0801	9.0079	0.0003
C. Elegans	297	14.46	2.4553	0.2924	2.3124	0.1512

From the plot obtained in Figure 1, it is clear that for a network to be classified as small world, its average clustering coefficient must be significantly larger than the average clustering coefficient of a random network.

Moreover, the average shortest path length of the network must also be larger than that of the random network, but the inequality is a bit relaxed here as even small shortcuts in ring lattices lead to a drastic decrease in average shortest path length.

Since $L(actual) > L(random)$ and $C(actual) \gg C(random)$, we can classify these networks as Small World [5].

5 Random Networks and Degree Distributions

Before Small World Network and Scale Free Network models were introduced, real networks were studied using the Random Network models. One such model generates a network $G(N, p)$ which has N nodes and each node pair is connected with a probability p [2].The degree distribution of such a network resembles a Poisson distribution.

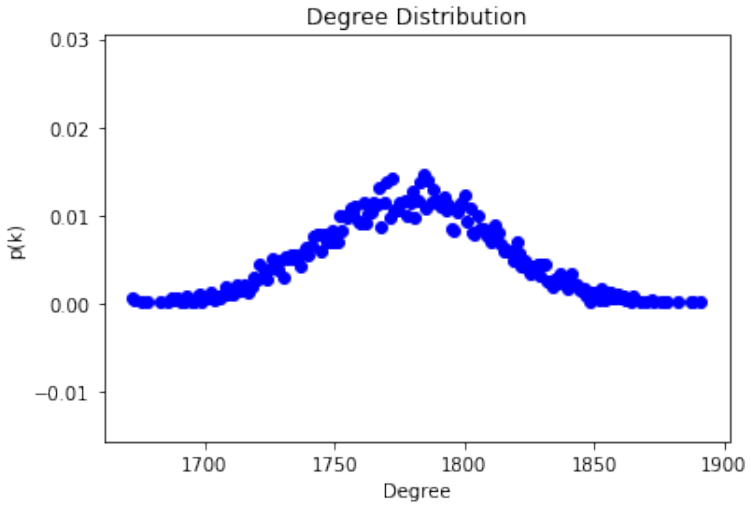

Figure 5: Degree Distribution of a Random network with N = 4941 and p = 0.2

As it is very apparent from Figure 2, the Poisson distribution does not permit nodes with large degrees (hubs) as it does not permit nodes with smaller degrees. But, hubs are, in fact, commonly observed in many real world networks such as the WWW network.

Despite this shortcoming, the Random Network Model does explain the small world phenomena, i.e. short average path lengths.

Consider a distribution with average degree $\langle k \rangle$. For this network, for any given node, the number of nodes at a distance 1 will be $\langle k \rangle$, at a distance 2 will be $\langle k \rangle^2$ and at a distance n will be $\langle k \rangle^n$. Hence, if d_{max} is considered to be the diameter of the network, the total number of nodes at a distance less than or equal to d_{max} has to be equal to N.

$$d_{max} = \ln(N) / \langle k \rangle$$

Since the maximum distance can be often influenced by extreme paths, the above relation is a better approximation for $\langle d \rangle$ in the limits of the Poisson distribution. Since $ln(N) \ll N$, we see that the average path is much smaller than the network size itself. The Power Grid network has a N = 4941 and $\langle k \rangle$ = 2.669. Using the approximation above, we get d_{max} = 8.664 which is close to what we got on random rewiring of the Power Grid Network (d_{max} = 9.0079)

6 Networks with Scale Free Degree Distributions

Networks with hubs are usually modelled by degree distributions that follow a Power Law.

$$p(k) = Ck^{-\alpha}$$

Plotted below are the degree distributions of two real world networks on a log-log scale. These networks do not follow a Poisson distribution, but rather follow a Power Law after a certain degree (cutoff). The explanation for emergence of scale invariance was attributed to preferntial attachment by Barabási [3], a phenomena that is observed in most real world networks. A new node added to a network, is more likely to share an edge with a node that has a high degree (more connected) rather than a node with a low degree (less connected node).

The emergence of scale free networks from preferential attachment was studied in the paper *Collective Dynamics of Scale Free Networks.* Barabási and Albert developed the model to show scale invariance as follows. They started with a random network of $m_0 = 5$ nodes. At each time instant, they grew the network by adding $m = m_0$ nodes . These new nodes were allowed to connect to k_{av} older nodes. The probability that a new node connects to an existing node i was taken to be proportional to the its degree.

It can be observed that no matter how many iterations of the node adding process we run, the degree distribution remains invariant with scale.

An important parameter pf the Power Law distribution is the scaling parameter α. The value of α often tells us a lot about the behavior of the

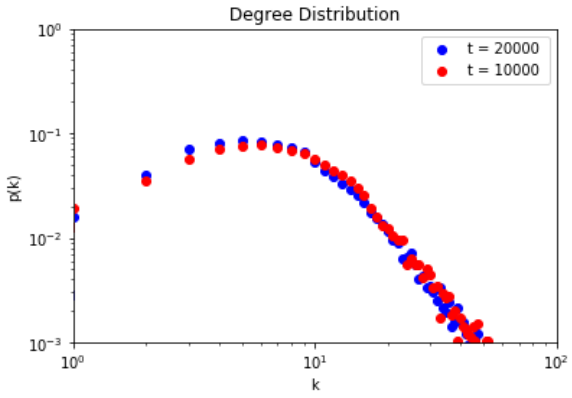

Figure 6: p(k) vs k plot for the constructed Scale Free network

network. Before we can see the significance of α, there are certain network parameters that need to be explained.

6.1 Largest Hub Size

For a degree distribution that follows a Power Law, the normalization condition gives us

$$\int_{k_{min}}^{\infty} p(k)dk = 1$$

where k_{min} is the minimum value of k for which the Power Law holds. To calculate k_{max} (The maximum degree for a given node in the network) we assume that there is one node in the network having a degree greater than k_{max}.

$$\int_{k_{max}}^{\infty} p(k)dk = 1/N$$

These two equations give us the relation between k_{min} and k_{max} as -

$$k_{max} = k_{min}N^{1/\alpha - 1}$$

6.2 Higher moments of k

$$\langle k^n \rangle = \begin{cases} \sum_{k_{min}}^{\infty} k^n p(k), & \text{for discrete distributions} \\ \int_{k_{min}}^{\infty} k^n p(k) dk, & \text{for continuous distributions} \end{cases}$$

Here, $n = 1$ gives us the mean of the distribution allowing us to predict the value around which a distribution is centered. Next, $n = 2$ gives $\langle k^2 \rangle$, which is used in calculating the variance of the distribution. Further, $n = 3$ gives $\langle k^3 \rangle$, which is used in to calculate how symmetric $p(k)$ is around its mean.

For a Scale Free network in particular, these moments can be simplified as

$$\langle k^n \rangle = C(k_{max}^{n-\alpha+1} - k_{min}^{n-\alpha+1})/n - \alpha + 1$$

The behaviour of higher moments is considered only as $k_{max} \to \infty$, and hence for all Scale Free networks, higher moments of k diverge. The convergence of $\langle k \rangle$ and $\langle k^2 \rangle$ depend on the value of α. [2]

1. $\alpha < 2$ is considered the anomalous regime where $\langle k \rangle$ and $\langle k^2 \rangle$ diverge and since the largest hub k_{max} grows faster than N itself, large networks don't exist in this regime.

2. $\alpha \in (2, 3)$ is considered the Scale Free regime where $\langle k \rangle$ is finite but $\langle k^2 \rangle$ diverges.

3. $\alpha > 3$ is a region where both $\langle k \rangle$ and $\langle k^2 \rangle$ are finite and hence, in the limits of large values of α, the degree distribution decays fast enough so that the network begins to resemble a random one.

Our interest lies in networks where $\alpha \in (2, 3)$. Also, we observe real world networks exhibiting power law degree distributions with their scaling parameters in this range.

7 Approximating the Scaling Parameter

7.1 Power Law Formalism

Power Law distributions are either discrete or continuous. For most calculations involving discrete distributions, it is often beneficial to approximate to a continuous one, especially when high accuracy is not needed. A continuous probability distribution is described by a probability density function as

$$p(x)dx = Cx^{-\alpha}dx$$

Since the density diverges as x approaches 0, the Power Law distribution only holds for a certain $x > x_{min}$. Applying the normalization condition for the distribution we can determine the value of the constant C to be:

$$C = (\alpha - 1)/(x_{min}^{\alpha-1})$$

In the discrete case, x can only take integer values

$$p(x) = Cx^{-\alpha}$$

and the normalization constant is calculate to be:

$$C = 1/\zeta(\alpha, x_{min})$$

where $\zeta(\alpha, x_{min})$ is the Hurwitz zeta function, defined as:

$$\zeta(\alpha, x_{min}) = \sum_{n=0}^{\infty}(n + x_{min})^{-\alpha}$$

7.2 Method of Maximum Likelihood

The study of a Power Law distribution involves finding two main parameters: the scaling parameter and the lower bound value x_{min} for the Power Law behaviour. First, we are going to focus our attention on calculating the scaling parameter using the method of maximum likelihood, which, as the name suggests, maximizing the likelihood function for the Power Law distribution.

The likelihood that the data is drawn from the Power Law distribution is simply

$$L(\alpha) = \Pi_{i=1}^{n}(\alpha - 1)(x_{min})^{\alpha-1}(x_i)^{-\alpha}$$

Maximizing the logarithm of this function we obtain

$$\widetilde{\alpha} = 1 + n[\sum_{i=1}^{n} \ln(x_i/x_{min})]^{-1}$$

Maximizing the likelihood for discrete data gives us a transcendental equation. This is often tedious to solve, and so the equation obtained above can be used for discrete data as well, after making a slight change to the expression, as explained in reference [4] and as shown below. This estimate for the scaling parameter is accurate to 1% for values of $x_{min} \geq 6$.

$$\widetilde{\alpha} = 1 + n[\sum_{i=1}^{n} ln(x_i/x_{min} - 1/2)]^{-1}$$

To test the efficacy of this approach, synthetic data sets were generated ($x_{min} = 7, n = 1000$) and the α values obtained were compared with the actual α values. The results are shown in Figure 7

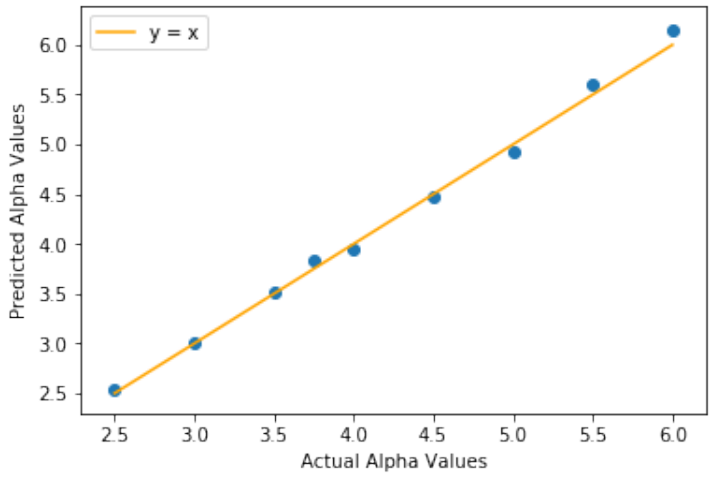

Figure 7: Actual versus predicted α values

7.3 Determination of x_{min}

The fundamental idea behind the determination of x_{min} is as follows: we choose the value of $\widetilde{x}_{min}$ (i.e. the estimate of x_{min}) such that the given probability distribution and the probability distribution obtained from the Power

Law model are as similar as possible, above the chosen value of $\widetilde{x}_{min}$. We do this by minimizing the distance between the two CDFs. This distance can be quantified and minimized using the Kolmogorov-Smirnov statistic (D) [6]. Our objective is to minimize D for $x > x_{min}$.

The Kolmogorov-Smirnov statistic D is given by

$$D = max|S(x) - P(x)|$$

Here, $S(x)$ is the function giving the fraction of data points to the left of a given value x_i in the data, and $P(x)$ is the cumulative distribution function of the distribution that the data has been fitted to. To test the method, a distribution was generated that follows a Power Law only above a certain x_{min} with $\alpha = 2.5$. We took $n = 10000$ random numbers from this distribution and $\widetilde{x}_{min}$ was determined as per the code given in Section 9.

The plot below (Figure 8) shows the values of predicted x_{min} ($\widetilde{x}_{min}$) against actual x_{min} obtained by minimizing the KS statistic.

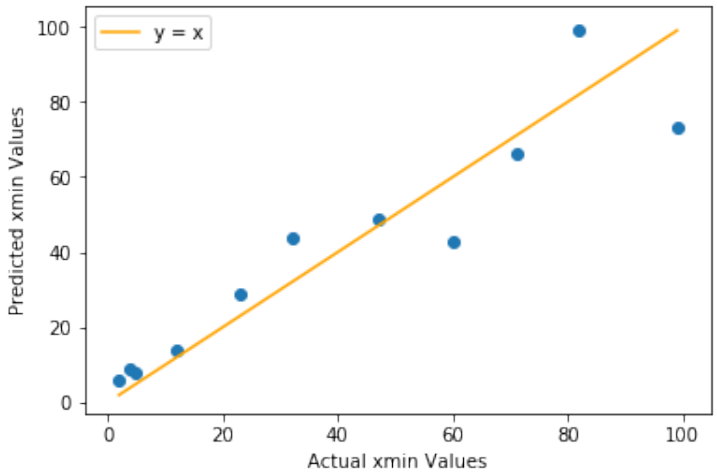

Figure 8: predicted vs actual $x_{min} for \alpha = 2.5$

8 Goodness of Fit Tests

Power Law distributions, log normal distributions, and exponential distributions all manifest as straight lines on a log-log plots. Hence, a detecting

a straight line on a log-log plot of the degree distribution is not a sufficient condition for Power Law behaviour. The goodness of fit tests tell us whether the Power Law distribution is a plausible fit to the data.

There are two main issues that the tests address [4]:

1. Data drawn from the Power Law may not follow an exact Power Law form due to the random nature of the sampling.
2. Due to the random nature of sampling, data drawn from a distribution which is categorically not a Power Law may appear to follow a Power Law distribution.

We need to measure how well the Power Law model fits the data. This is quantified using the goodness of fit parameter (p). To calculate this goodness of fit parameter (p), we first fit the data to the Power Law model and calculate the KS statistic. Next, we generate a large number of data sets ($n = 500$) with the same value of x_{min} and α, and fit these data sets to their own Power Law models to obtain the KS statistic. While generating these data sets, we not only generate n_{tail} observations that follow a Power Law distribution, but we also pick $n - n_{tail}$ observations from the original data set that belong to the $x < x_{min}$ region (where the Power Law distribution does not hold). The value of p is the fraction of the times where the KS statistic for the synthetic data is greater than the original data. [4]

If p is close to 1, we attribute the difference in the empirical data and the Power Law to statistical fluctuations. But, if the value of p is small, then the Power Law model is not a viable fit to the given data.

9 Fitting Real Data

The following are code snippets of functions used to fit Power Law to real data.

```
def finda(x, x_{min}):
    afind = np.log((x)/(x_{min}-0.5))
    a_ = 1 + len(afind)*((sum(afind)**(-1)))
    n = len(afind)
    #Finite correction
    a_ = a_*(n-1)/n + 1/n
    return a_
```

After obtaining the value of alpha, we look into a range of values around the guess, to widen our scope and obtain a more ideal value.

```
def alpharange(alpha, a_values):
    return np.linspace(alpha*0.90, alpha*1.10, a_values)
```

Among the values of alpha generated, the one that maximizes the log likelihood (As given in B.8) is the most ideal.

```
def llhood(alphavals, x_{min}, x):
    temp = x[x >= x_{min}]
    n = len(temp)
    return (-n*np.log(scipy.special.zeta(alphavals, x_{min})) -
    alphavals*np.sum(np.log(temp)))
```

The KS Statistic is calculated for all the (x_{min}, α) pairs and the one that yields the minimum difference between the two is chosen.

```
def ks_value(alpha, x_{min}, x):
    temp = x[x >= x_{min}]
    n = len(temp)
    ecdf = np.searchsorted(temp, temp, side = 'left')/n
    cdf = 1 - (temp/x_{min})**(-alpha + 1)
    return max(abs(ecdf - cdf))
```

All the unique values in the data are possible candidates for x_{min} and we iterate through all of them, and using the methods discussed above, find the best fit.

```
def findx_{min}(x):
    x.sort()
    ks = np.array([])
    alpha_list = np.array([])
    x_{min}vals = np.unique(x)
    for i in np.nditer(x_{min}vals[:-1]):
        temp = x[x >= i]
        x_{min} = i
        alpha = finda(temp, x_{min})
        alphavals = alpharange(alpha, 100)

        lhood = llhood(alphavals, x_{min}, temp)
        best_alpha = alphavals[np.argmax(lhood)]
        alpha_list = np.append(alpha_list, best_alpha)
        ks = np.append(ks, ks_value(best_alpha, x_{min}, temp))

    x_{min}_index = np.argmin(ks)
```

```
    x_{min} = x_{min}vals[x_{min}_index]
    alpha = alpha_list[x_{min}_index]
    pdata = x[x >= x_{min}]
    n = len(pdata)
    sigma = (alpha - 1)/np.sqrt(n)
    ks = min(ks)

    return x_{min}, alpha, n, sigma, ks
```

Generating random numbers that follow Power Law distribution according to the method described in Appendix D of *Power Law Distributions* [4] .

```
    def x_fn_above(u, x_{min}, alpha):
    # When r>=rmin.
    x = x_{min} * pow((1.-u), 1./(1.-alpha))
    return (x)
```

We generate *ntail* data points that follow a Power Law distribution using the function mentioned above. If n is the size of the data set, we pick $n - ntail$ observations from the original data set that do not follow a Power Law distribution.

Once the synthetic data is created, we fit it using the function above to calculate the KS value that is compared with that of the real data.

```
def genp(x_{min}, alpha, ntail, ks, degree):
    nexp = len(degree) - ntail
    p = 0
    # Normalization factor:
    C = (1./x_{min}) * pow((1./(alpha-1) + (np.exp(alpha)-1)/alpha), -1)

    for i in range(pcheck):
        # random numbers from uniform distribution in range [0, 1):
        r = np.random.uniform(low = 0., high = 1, size = ntail)


        y_above = []
        for u in r:
            y_above.append(x_fn_above(u,  x_{min}-0.5, alpha))

        y_temp1 = y_above
        y_temp2 = np.array(y_temp1)
        y = np.round(y_temp2 + 0.5)
```

```
        x = y.astype(int)
        x = x[x>0]

        exp_data = np.random.choice(degree[degree < x_{min}], nexp)
        x = np.append(x, exp_data)

        ks_synthetic = findx_{min}(x)[4]

        if ks_synthetic > ks:
            p += 1
        else:
            pass

    p /= pcheck
    return p
```

10 Results and Plots

Network	N	$k_{average}$	$xmin$	α	p
Protein	2018	2.90	6	2.855	0.502
Power Grid	4941	2.67	10	7.624	0.254
WWW (in)	325729	9.19	5	1.933	0
WWW (out)	325729	9.19	8	2.060	0
Metabolic Network	1039	11.17	7	2.64	0.082

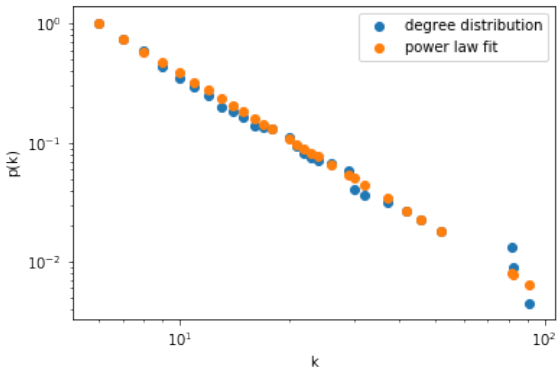

Figure 9: Protein Network

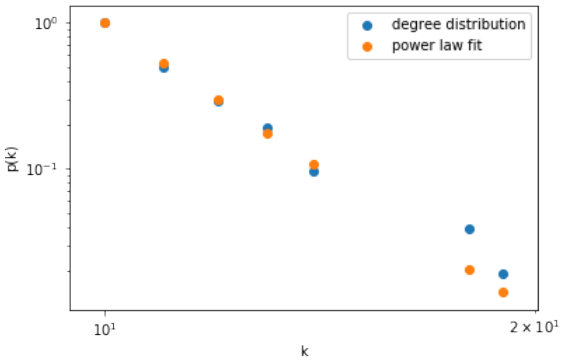

Figure 10: Power Grid of the USA

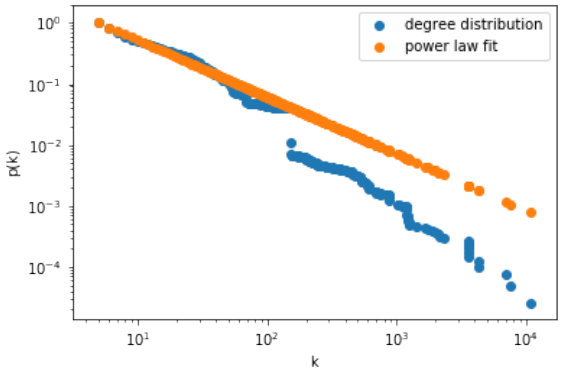

Figure 11: WWW (In degree)

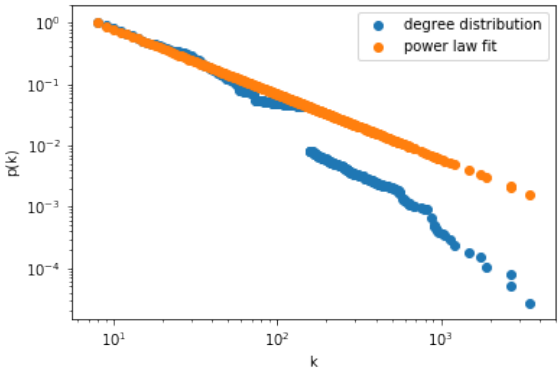

Figure 12: WWW (Out degree)

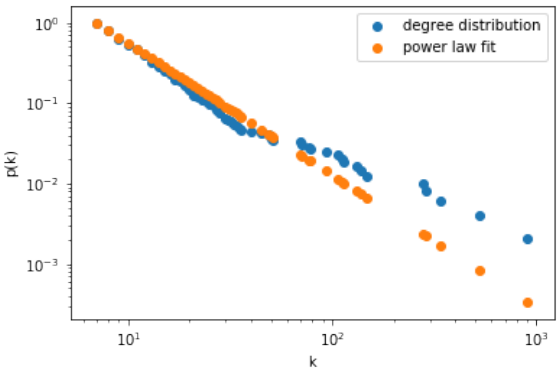

Figure 13: Metabolic Network of Yeast

References

[1] Albert-László Barabási. Network science book. `http://networksciencebook.com/translations/en/resources/data.html`.

[2] Albert-László Barabási. *Network Science*. Cambridge University Press, 2015.

[3] Albert-László Barabási and Reka Albert. Emergence of scaling in random networks. *Science*, 286:509–511, 1999.

[4] Aaron Clauset Cosma Rohilla Shalizi M. E. J. Newman. Power-law distributions in empirical data. *SIAM Review*, 51(4):661–703, 2009.

[5] Duncan J. Watts Steven H. Strogatz. Collective dynamics ofsmall-world networks. *Nature*, 393, 1998.

[6] William T. Vetterling Brian P. Flannery William H. Press, Saul A. Teukolsky. *Numerical Recipes in C*. Cambridge University Press, 2 edition, 1992.