Random Networks via Monte Carlo Simulations on the Raspberry Pi Zero W

A Computational Physics II project by Jeremy Kazimer (jdkazime@buffalo.edu)

Introduction

Whether it's obvious or not, networks are everywhere. Well, to be more precise, the information that they encode is everywhere. What I mean by this is that networks on their highest level are composed of two properties: nodes, N, and edges, M. More will be elaborated upon in the theory section, but essentially each node represents some object in a pre-defined space. This could be a person, a neuron, or... well, anything, really (Bianconi 2015). Anything that takes on an identity and can be classified by said identity. As for the other property, the edges, these encode the interactions between any set of nodes. If, for example, two people follow each other on Twitter, then they have an edge between them. These are the surface level properties of a network. There are of course more, but these are the essentials.

Thus, taking a step back, it is more obvious that networks are everywhere. To elaborate, with the right data as defined by information theory (Anand and Bianconi 2009), a network can be constructed such that the nodes encode any set of objects and the edges the subsequent interactions. To continue the Twitter analogy, take any N twitter users. These are the nodes, the objects, of our system. Each of theses users, these nodes, have unique properties, such as their username, their profile picture, etc. This is what it means for a node to encode information. They don't have to encode information, but they inherently have the structure to do so. Regardless, we can then define the existence of an edge on whether a user follows another user. If they do and follow each other, then they share an edge. If one person follows another, but not the other way around, then there exists only one edge from that user to the followed user. These are directional edges. There are also weighted edges such that perhaps an interaction between two nodes is more important than the other nodes. Either way, if they do not follow each other, then there is no edge between that node.

Really, what an edge represents here is the ability for information to spread; in a closed system, that is to say that there is no possibility of two completely separated nodes of communicating, information travels from node to node by their edges. As a physical analogy, think of this like a set of interacting particles (Barré et al. 2018); if there is no collision, that is to say particles are not interacting with each other, then energy is not spread around the system, assuming the lack of physical entropy. However, if they are interacting through collision, then this energy, the information, is spread via this transfer.

Ultimately, with the above assessment, the goal of this project is to characterize these networks on a purely random level. That is to say, for a network of size N it is generated randomly with edge probability p, the probability of an edge between two nodes, so that both the configuration and the number of edges M is random each time, within the probability constraints (Li, Mucha, and Taylor 2017). With that, we can create an analogy between this and that of actual particle interaction via information-theoretic entropic analysis, among other things. In order to make definitive statements about these random networks, the goal of this assignment, I will employ the use of Monte Carlo methods so that we can validate known theory and also make observations of our own. As a further constraint, this will also be created such that it can be reasonably ran on a Pi Zero W, which introduces its own margin of error.

Limitations of the Raspberry Pi Zero W

The deployment of this application on the Raspberry Pi Zero W is not a particularly difficult feat; none of the involved libraries here are difficult to install or unable to be installed on the Raspberry Pi Zero W. Nor are the calculations time intensive; most of these calculations are upper bounded by $\mathcal{O}(N^4)$ such that N is the network size.

Really, the limiting factor here is space. Since graph structures are represented in y dimensional arrays such that $y \geq 2$, their space usage does not scale particularly well. Notably, since most graphs are represented symmetrically, at least the ones we'll be dealing with, there will always be a spatial demand of at least $N \times N$. If this is a multilayer structure (Kobayashi and Onaga 2021), then this is brought into the third dimension by

some number of layers l. Further, since each float is at minimum 4 bytes and maximum 16 bytes, the space allocation S scales $4 \cdot (N \times N \times l) \leq S \leq 16 \cdot (N \times N \times l)$.

As such, much of this experimentations for the Raspberry Pi Zero must be done with smaller networks, say 100 nodes. Although this seems like a lot, physical systems are in the thousands to millions of nodes so we will have difficulty presenting any tangible system in our experimental environment.

There are, however, experimental libraries in Python designed for the Raspberry Pi Zero W, such as VideoCore, to optimize libraries such as NumPy. The issue is that it's rather experimental so any incorrect configuration could evaporate the board. Not literally, but it would at worst brick the system. Especially in such a complicated set of calculations, it's not worth the risk of trying to set it up. With that, the solution we'll be utilizing is instead to just limit space usage. And by virtue of working on smaller networks, the computation time will also speed up since it depends on network size.

With that, discussion on theory can begin.

Network Theory

Graph Structure

In order to generate networks, we must first define all of its properties such that the properties are understood. First, speaking of nodes we define

$$\mathcal{V} = \{1, 2, \ldots, n\} \tag{1}$$

for $n \in \mathbb{N}$ such that $\mathcal{V}$ is the set of all nodes (Li, Mucha, and Taylor 2017) and 1 and 2 represent the first and second nodes, for example. The number of nodes N is then just the cardinality of $\mathcal{V}$ such that $N = |\mathcal{V}|$. Then, for a single-layer graph Erdős–Rényi G_{NP} , (Li, Mucha, and Taylor 2017) that is to say a two-dimensional graph with no coupling constant D_x (Kobayashi and Onaga 2021), this is defined as the adjacency matrix A such that

$$G_{NP}(\mathcal{V}, E) = A = \left[A_{ij}\right] \tag{2}$$

for $0 \leq i, j < N$. Then, the number of elements in the adjacency would then be

$$|G_{NP}| = |A| = |N| \times |N| \tag{3}$$

Note that this is not the number of edges, but rather the number of all entries for this adjacency matrix or, in computer science terms, the size of the array. From this point onward, we'll only refer to G_{NP} as A. We can then define the values of A_{ij} such that

$$A_{ij} = \begin{cases} 1 : (i, j) \in E \\ 0 : (i, j) \notin E \end{cases} \tag{4}$$

where E is the set of all edges (Li, Mucha, and Taylor 2017) such that

$$E = \{(i, j) : p_{ij} \leq p\} \tag{5}$$

for some particular edge probability p_{ij} and a total edge probability p. Basically, each entry of A, A_{ij}, has an associated probability p_{ij} such that if it's less than the fixed parameter probability p then there exists an edge. We can then define the total number of edges as

$$M = |E| \tag{6}$$

For an unweighted and undirected matrix, this is the same as saying $2M = \sum_{ij} A_{ij}$, due to the symmetry of the adjacency matrix (Li, Mucha, and Taylor 2017). The expected number of edges then, for an A graph, would be

$$\bar{M} = \frac{N(N-1)}{2} \cdot p \tag{7}$$

because, without self edges, that is we disallow a node from being connected to itself, there are N nodes. However, we remove the locations where a node can connect itself such that there remains $N-1$ spots remaining. Then, we divide by 2, due once again to the symmetry of an undirected and unweighted adjacency matrix (n.d.). This leaves us with $\frac{N(N-1)}{2}$ edges. Since this is a purely probabilistic process, we multiply this term by p since generally an expectation function takes on the form

$$\bar{x} = x \cdot p \tag{8}$$

We can then relate this to the number of potential edges P, that is to say areas where edges don't exist such that they could exist if under some stochastic process, can be represented as

$$P = \frac{N(N-1)}{2} - M \tag{9}$$

since $|A|$ counts edges twice. We can then define the degree matrix, that is the number of edges connected to each node, as

$$D = \text{diag}[d_1, d_2, \dots d_N] \tag{10}$$

where $d_i = \sum_i A_{ij}$ for the i-th node (Li, Mucha, and Taylor 2017). The unnormalized Laplacian matrix L is then given by the equation

$$L = D - A \tag{11}$$

This is particularly useful to us in that this is the foundation of spectral analysis. That is to say, we use the Laplacian in order to extract its eigenvalues λ and eigenvectors $\vec{v}$ for a variety of uses. This also appears in fields such as machine learning and computer vision, but for our uses this is simply the basis of spectral analysis.

Stochastic Block Model

A stochastic block model (SBM) is a structure of graphs that describe random graphs. Here, it is structured in such a way that the diagonal blocks of the random graph are the communities, so to speak, of these graphs and the off diagonals represent the edges connecting these communities together (Lee and Wilkinson 2019).

More generally, we look at the case where all of the communities are equally sized; that is to say, the size of each community n_k is defined by

$$n_k = \frac{N}{k} \tag{12}$$

where k is the number of communities. It must be true then that $\sum_i^k n_k = N$. Although this describes the network size, this does not describe the number of edges. Typically, we describe the probability structure of the graph by the matrix

$$\begin{bmatrix} p_{in}^{(0)} & p_{out}^{(0,1)} & \dots & p_{out}^{(k-1,k)} \\ p_{out}^{(1,0)} & p_{in}^{(1)} & \dots & \dots \\ \dots & \dots & \dots & \dots \\ p_{out}^{(k,k-1)} & \dots & \dots & p_{in}^{(k-1)} \end{bmatrix} \tag{13}$$

Note that the sum of

$$\sum_{i}^{k-1} \sum_{j}^{k} p_{out}^{(i,j)} + \sum_{i} p_{in}^{(i)} \tag{14}$$

does not have to sum to unity, 1. Here, p_{out} describes the probability of edges forming between the communities or what we'll describe as the connecting edges. Then, p_{in} describes the edges within communities, or what we'll describe as the community edges. We can then decide the total edges within a community and connecting a community separately such that

$$\bar{M_{in}} = \sum_{i}^{k-1} \frac{n_k(n_k+1)}{2} \cdot p_{in}^{(k)} \tag{15}$$

and

$$\bar{M_{out}} = \frac{1}{2} \sum_{i}^{k-1} \sum_{j}^{k} n_k \cdot p_{out}^{(i,j)} \tag{16}$$

Here, each off-diagonal block is capable of having diagonal elements and our summation assumes that the graph is unweighted and undirected, so we can just divide by two to account for duplicate blocks (i.e., $p_{out}^{(0,1)} = p_{out}^{(1,0)}$, or more simply a symmetric system). Of course, then, the total number of edges would then just be

$$\bar{M} = \bar{M_{out}} + \bar{M_{in}} \tag{17}$$

with the reminder that we look at the expectation here, since forming a G_{NP} is inherently random. Overall, an SBM can be viewed as the superposition of several different random graphs of equal network size, but varying edge probabilities. More intuitively, an SBM is analogous to, say, a typical physical partition problem.

Suppose each community represents a separate partition (Reichardt and Bornholdt 2006). Each community edge then represents the potential that exists between two particles or nodes. In a closed system, i.e. all $p_{out} = 0$, there is no energy exchange between the partitions, hence a lack of edges. As such, as p_{out} increases throughout overall the system, the more energy distribution that there is since the partitions are opened and now the separate communities are allowed to interact.

That is all that will be said for now. Later, this will be elaborated upon.

Rewiring

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

Von Neumann Entropy

Spectral Analysis

To begin, spectral analysis is the examination of eigenvalues from some arbitrary system (SOVEREIGN, NOLAN, and MANDEL 2007). Since we are dealing with graph structure, any spectral analysis deals primarily with that of the eigenvalues of the Laplacian matrix L (Li, Mucha, and Taylor 2017). There are two types of the Laplacian we typically handle, the unnormalized Laplacian L and the normalized Laplacian L^{sym}.

Here, we'll only speak of the unnormalized Laplacian. There is nothing particularly wrong with the normalized, it's just that the unnormalized appears more commonly in nature (Li, Mucha, and Taylor 2017). Then, the eigenvalues $\{\lambda_i\}$ and eigenvectors $\{\vec{v}\}$ can be found arbitrarily with any typical algorithm.

In network theory, the eigenvalues by themselves are indicative of a few things. The first, the edge probability and number of nodes. This is because in a typical G_{NP}, the average eigenvalue floats around

$$\bar{\lambda_i} \approx N \cdot p \tag{18}$$

For an SBM, this is

$$\bar{\lambda_i} = N \cdot p_{in} \tag{19}$$

assuming that each $p_{in}^{(k)}$ is of equal value. This is because p_{out}, assuming that $p_{out}^{(k-1,k)}$ is the same for all k, is more indicative of the community structure than the edge probabilities and nodes.

Speaking of which, we can determine, for a G_{NP}, how many communities there are by the number of zero eigenvalues (Newman 2006). Or more specifically

$$k = |\{\lambda_i : \lambda_i = 0 \text{ for all } i \in N - 1\}| \tag{20}$$

Besides that, we often study how eigenvalues change under rewiring. If, for example, we were examining a two community SBM and randomly rewire it such that it becomes a one community SBM, we would anticipate that the number of zero eigenvalues decreases from 2 to 1 (Newman 2006).

If examined on a much smaller scale, eigenvalues just in general give insight to the changing structure of the network. It must be noted that we don't typically examine eigenvectors; this deals more with principal component analysis (PCA) (Jolliffe and Cadima 2016) than, say, network theory's branch of spectral analysis.

Of course, these can be examined, but there is no established theory to do so in the direction that this project is headed. And, admittedly, I don't know about them to develop theory on my own.

Otherwise, spectral analysis tends to ascend its eigenvalues by going into a more scalable form - von Neumann Entropy.

VNE

Von Neumann Entropy (VNE) is but one way to interpret eigenvalues. It originally comes from the equation of Shannon entropy (Li, Mucha, and Taylor 2017) such that

$$H = -\sum_i p_i \log p_i \tag{21}$$

where H is our entropy and p_i is the probability of some event happening. From a quantum perspective, think of an example such that we measure the probability of the location of a particle. If we know that it exists, then it must exist in one such location so that

$$\sum_i p_i = 1 \tag{22}$$

However, VNE takes this one step further by defining the probabilities as the trace of some density matrix, $\mathrm{Tr}(\rho)$ (Li, Mucha, and Taylor 2017) such that ρ is the density matrix. The entropy is then

$$H = -\mathrm{Tr}(\rho) \log_2 \mathrm{Tr}(\rho) \tag{23}$$

where base 2 comes from the binary representation of information (n.d.). Recent publications have claimed that $\mathrm{Tr}(\rho)$ is really a function of the eigenvalues (Li, Mucha, and Taylor 2017) such that

$$\mathrm{Tr}(\rho)_i = \frac{\lambda_i}{2M} \tag{24}$$

which would create an entropic model such that

$$H_M = -\sum_i \frac{\lambda_i}{2M} \log_2 \frac{\lambda_i}{2M} \tag{25}$$

We denote H_M as such, since it is an entropic model that depends on the number of edges. Note that it is true (Li, Mucha, and Taylor 2017)

$$\sum_i \frac{\lambda_i}{2M} = 1 \tag{26}$$

Since this comes from the trace of the density matrix, it is rather evident that this entropy serves its purposes as a means to convey the density of the Laplacian. Ultimately, this is indicative of the edge probability.

As such, in a totally disconnected system such that $M = 0$, the entropy is minimized at $H_M = 0$ because all eigenvalues would be zero such that we define $0 \log_2 0 = 0$. However, when $p = 1$, that is to say the maximum M, all of the eigenvalues take on the same value such that

$$\forall_{i,j \in N-1} \lambda_i = \lambda_j \tag{27}$$

So, with any typical entropy model, when all of the probabilities are the same the entropy is maximized. It is reasonable then to conclude that the usefulness of this model isn't particularly high; it really just indicates the density of any Laplacian, which can be observed regardless if this Laplacian is known. As such, it is only useful when the Laplacian is not known, in the rare case where only the eigenvalues are known.

Because of this, data scientists sought to find a better entropy model; one that has more application to physical systems. Recently, the following function was validated as being correct (Domenico and Biamonte 2016):

$$H_\beta = -\sum_i \frac{e^{-\beta\lambda_i}}{Z} \log_2 \frac{e^{-\beta\lambda_i}}{Z} \tag{28}$$

Here, Z is the partition function (Domenico and Biamonte 2016) from statistical mechanics such that

$$Z = \sum_j e^{-\beta\lambda_j} \tag{29}$$

where β is some time-scale parameter. Because of this parameter, we denote H_β as such. Note that in statistical mechanics β is defined as

$$\beta = \frac{1}{k_B T} \tag{30}$$

where k_B is the Boltzmann constant and T is the system temperature. However, here it refers to the time domain as opposed to the energy domain, especially since we're dealing with information theory here. Via quantum mechanics, this makes sense since energy and time cannot be known simultaneously (Libretexts 2020), so in a way this is the complement to the energy-based parameter.

Regardless, it is apparently that $\sum_i \frac{e^{-\beta\lambda_i}}{Z} = 1$, since Z is but the total of all of the denominator. This model of entropy in particular is great for community detection (Domenico and Biamonte 2016). That is to say, for a dense enough network with community structure, an SBM for example, the entropy approximates to the following form:

$$H_\beta \approx \log_2 k \tag{31}$$

For example, a dense G_{NP} with no community structure will evaluate to zero entropy. However, totally independent nodes, that is to say $k = N$, will evaluate to maximal entropy, or the maximum number of communities. As such, it is apparent how this could be useful in community detection. This is where the model breaks away from the older model; it can detect community effortlessly giving it purpose above determining matrix density.

Really, the question here remains about how β effects this entropy. Consider β as such: this parameter is indicative of how long the nodes get to interact through edges. So, small β means that there is less time for the nodes to interact and large β means that there is plenty of time to interact.

Later, this will be useful in edge ranking, more specifically in determining the most important edges. Nevertheless, this is the extent of the entropy model; it is rather new, so the technicalities of it have yet to be worked out. Especially since this partition of network science isn't the most popular.

Edge Ranking

The way that we rank edges is a rather simple process. To begin, removing one edge will produce some entropy ΔH_β, which we can measure after modifying the Laplacian. What is measured here is really just the difference in the entropy of the original graph, $H_\beta^{(1)}$, and the entropy after one edge has been removed, $H_\beta^{(1)}$. More generally, we can denote the entropy for t removals as

$$H_\beta^{(t)} \tag{32}$$

such that it reflects the change in the Laplacian. However, for this we only deal with one removal. This is because after we remove the edge and calculate its entropy, we return the graph to its initial state. The goal is to calculate the entropy of each individual edge removal, as opposed to procedurally bringing the graph down to zero edges (Taylor et al. 2015).

Therefore, the process is as follows:

- We collect the set of all edges.
- Then, we iterate over this set of all edges.
- After, remove the edges in any order and calculate the entropy following a single removal.
- Following this, we restore the edge that was removed and move onto the next edge.
- We do this for all edges.

The main consideration is that we only have to do this for the upper triangular or lower triangular of the graph because, due to symmetry, the edge $(1, 2)$ for example will have the same entropy as $(2, 1)$. This is a nice feature of undirected graphs. Once we have collected the set of entropies for all individual edge removals, we can then sort them from greatest to least such that the edge with the lowest ranking (meaning most important) would have the highest entropy.

The logic for this is that an edge removal will cause a change in the eigenvalues. If this removal entropy is low, then it has not perturbed the structure of the eigenvalues. However, the higher that this entropy is (a reminder that this is relative to the base entropy) the more that the eigenvalue structure has been perturbed. As such, we treat the edges that generate the most entropy as the most important edge as it implies some deep structural connection to the system (Taylor et al. 2015).

The application of edge ranking to a single-community G_{NP} is not particularly interesting; there is not really a community structure to be squeezing insight out of. Rather, looking for something is really just the Jesus in the toast phenomena (Pappas 2014). However, for $k \geq 2$, there certainly exists a distinction between community and connecting edges. Because of that, edge ranking in an ideal scenario would be able to distinguish between community and connecting edges without any prior impetus. In saying this, we mean that without any bias the entropy model should be able to identify the two different edge types. There is no citation here, as it is currently ongoing research, unpublished.

Hence, the beauty of the H_β model is as such. In entangled systems, it can tell us the most important edges and we can work backwards to understand the community structure of rather implicit networks. The brain, as one example, where the community boundaries are not clearly defined (Betzel et al. 2019). Or, in somewhat recent years, a Congressional network (Zhang et al. 2007), where we can confirm party lines as the two communities. In either instance, this entropy model is intensely powerful for a well-defined network.

The one caveat to this model is that it depends intrinsically on the time-scale parameter β. This parameter determines the hierarchy of the edges, seemingly independent of the actual eigenvalues. For example, when β is infinitely small, our $H_\beta \approx H_M$. We can use a Taylor expansion to show this:

$$\beta \approx 0 : Z = \sum_j e^{-\beta\lambda_j} \approx \sum_j -\beta\lambda_j = -\beta \sum_j \lambda_j = -\beta(2M) \tag{33}$$

Note that the statement $\sum_j \lambda_j = 2M$ actually comes from our original model (Li, Mucha, and Taylor 2017). Further,

$$\beta \approx 0 : e^{-\beta\lambda_i} \approx -\beta\lambda_i \tag{34}$$

Then,

$$\beta \approx 0 : H_\beta \approx -\sum_i \frac{-\beta\lambda_i}{Z} \log_2 \frac{-\beta\lambda_i}{Z} \approx -\sum_i \frac{-\beta\lambda_i}{-\beta(2M)} \log_2 \frac{-\beta\lambda_i}{-\beta(2M)} = -\sum_i \frac{\lambda_i}{2M} \log_2 \frac{\lambda_i}{2M} \tag{35}$$

Ultimately, this explains why small β is unable to distinguish important edges; it no longer acts an entropic model for community detection, but rather the density of the network. More technically, the eigenvalues that would normally have the most important value to the entropy model are now zeroed out, meaning that they all weigh the same. The rich structure that we can observe from spectral analysis vanishes, basically.

There are still two cases, yet. Specifically, $\beta = 1$ and $\beta >>> 1$. In the first case, there is really nothing to say. It is almost like the time-scale parameter isn't there, so it doesn't influence the eigenvalues by any means. In the second case, however, we treat this as β being very large, or really

$$\lim_{\beta \to \infty} H_\beta \tag{36}$$

As such, we first look at $e^{-\beta\lambda_i}$. There are two cases to consider here:

$$e^{-\beta\lambda_i} \approx \begin{cases} 1: & \lambda_i \approx 0 \\ 0: & \lambda_i \not\approx 0 \end{cases} \tag{37}$$

For a k-community SBM, the first condition occurs for the first k eigenvalues. The second condition occurs for the $N-k$ eigenvalues that form a Gaussian distribution about $N \cdot p_{in}$. As such, Z can be written as such

$$Z \approx \sum_{j}^{k} e^{-\beta\lambda_j} \tag{38}$$

However, we know that here $e^{-\beta\lambda_i} \approx 0$ such that

$$Z = k \tag{39}$$

This is fine because most eigenvalue algorithms have them sorted from least to greatest. If not, we would have to impose more conditions. Regardless, this would mean that our entropy takes on the approximate form

$$H_\beta \approx -\sum_{i}^{k} \frac{1}{k} \log_2 \frac{1}{k} = \log_2 k \tag{40}$$

which is where an earlier formulation comes from. As such, this detects community structure strongly; it almost discards the latent organization of the network to exclusive phone in on the community-determining eigenvalues. Because of that, when an edge is removed, the entropy produced here reflects the community structure. With that, the connecting edges are always going to have highest priority because they determine the number of near-zero eigenvalues (Domenico and Biamonte 2016).

Of course, computers can only handle so large a β, so this is purely a theoretical application. Since β is being exponentiated, it quickly approaches the limits of machine precisions and oftentimes reaches float overflow if β is not properly calibrated.

Using Monte Carlo as an Explorative Tool

Since a G_{NP} and, by extension, an SBM are inherently random in nature, there exists permutations for any given configuration. That is to say for an initial N and p, a graph can be rearranged in finite ways while still preserving the qualities. As such, there emerges an application of Monte Carlo analysis to first validate known quantities and then, after knowing that the Monte Carlo simulations are valid, using it to validate behavior expectations.

Taking a step back, Monte Carlo simulations, named after the location in Monaco, is a set of methods to try and quantify random processes (Education, n.d.). Or, more simply, we group together random behavior such that we can potentially classify it by its average. This is particularly useful in cases where the phenomena doesn't necessarily have a closed form solutions. One such practical application of this is integrals; not all integrals can be described easily in algebraic terms. So, a Monte Carlo simulation can be used to find an approximate solution to this calculus (Mishra and Gupta 2009).

The real catch here, however, is that Monte Carlo simulations become increasingly better at describing the phenomena as the sample size increases. That it so say, the data being averaged over will be nearly accurate if there are infinitely many iterations. As the number of iterations increases, the accuracy asymptotically approaches zero. Unfortunately, due to computer limitations such as machine precision, eventually the most accurate of systems will fall apart from the computer not being able to carry about basic arithmetic properly. So, there exists a iteration-accuracy tradeoff such that there needs to be decided a number of iterations where the accuracy doesn't fall under machine precision. This, however, is not the goal of this project.

Rather, our use of Monte Carlo simulations here, as aforementioned, are to first validate known quantities. For example, we can use Monte Carlo simulations to validate that

$$\bar{\lambda_i} \approx N \cdot p \tag{41}$$

or

$$\bar{M} \approx \frac{N(N-1)}{2} \cdot p \tag{42}$$

Since these are known, we can use them to validate that Monte Carlo simulations will in fact operate well on graph structures. Once that has been done, we can move onto something more ambitious, say understanding how SBM structure affects edge rankings for different β, p_{in}, or p_{out}. Our focus will mostly be on p_{in} and p_{out}, since we've already described out how the behavior of β should operate.

Regardless, once this established we can set up a framework for a toy model experiment.

Potential Applications to Real Physics

Results

Comparison to other Methods

Although there aren't many other analytical methods that are similar to network theory, spectral theory (Domenico and Biamonte 2016) in particular, the one that stands out the most is principal component analysis (PCA) (Jolliffe and Cadima 2016). This family of algorithms in particular examines the eigenvalues and eigenvectors so that a fuller picture of the data is formed. Typically, this is used to reduce down data to a scalable form such that loss of information is minimized.

Network theory and, by extension, spectral theory is a bit better than PCA in some cases because it allows for a more complex relationship to be developed by the data. That is to say, complex networks are able to encode very specific interactions between nodes, especially when multiple layers, a multilayer or multiplex network, are introduced. Since PCA reduces these features down to only the eigenvectors and eigenvalues typically, many of these rich relationships are lost in translation.

Spectral theory is particularly important, especially with the introduction of Shannon entropy via von Neumann Entropy (VNE) (Li, Mucha, and Taylor 2017). This could be perhaps be treated as an extension of both PCA and network theory, since entropy takes on the form of its eigenvalues. Regardless, VNE is far more common a tool of analysis in network theory than it is in PCA. Really, it in itself is reducing data down to a single value, akin to PCA, but this is used rarely to more so describe the entire system, as opposed to its individual components. Perturbation theory (Li, Mucha, and Taylor 2017) is also useful in studying how the network changes under slight modifications to the network, something that PCA doesn't exactly cover.

Nevertheless, it is the amount of information that can be gleaned from network theory that sets it above PCA. The complex systems can also be captured, but at the cost of runtime complexity. It is true that in general PCA is faster than network theory, since its main bound is calculating eigenvalues. Whereas, with network theory, there are many methods from modularity, to rewiring, to edge ranking, which all have significant runtime complexities. However, their availability as an analysis tool is more important than runtime, in a larger scale. Of course, this may not be the case for the Raspberry Pi Zero W, but initial conditions can be set such that runtime complexity isn't a major factor.

Conclusion

DISCUSS RESULTS

So, now that all of the outstanding issues have been addressed, where would I see this project going? Foremost, it would be good to develop models for different types of random networks, such as a multilayer network

(Kobayashi and Onaga 2021) or scale-free network (Broido and Clauset 2019) Both certainly appear in nature; arguably more so than an SBM, since a typical SBM has rather fixed constraints in terms of community size.

After that? I would like to apply this to real physical data. One such example of something with rich community structure can be found in biology or chemistry. What I refer to here is really any large molecule (Sun 2017). With initial conditions, the actual bond energies can be calculated such that we can form something with visible community structure. The main reason that I wasn't able to apply this here is that, since this is designed for the Raspberry Pi Zero W, the space limitations are fairly obvious. You can look at the protein database (https://www.rcsb.org/) and see that much of these molecules are in the thousands, if not tens of thousands, of atoms or what we would classify as the nodes.

Otherwise, I think that this was a clear success. There exists now a rather evident bridge between the information-theoretic world and some of the most classical of physics problems, namely the partition problem that pops up in statistical mechanics and thermodynamics. Interestingly enough, those problems also deal in entropy, but a kind different from the one described here. Regardless, this was enjoyable and I look forward to studying this more as I ascend the academic ranks.

References

Anand, Kartik, and Ginestra Bianconi. 2009. "Entropy Measures for Networks: Toward an Information Theory of Complex Topologies." *Physical Review. E, Statistical, Nonlinear, and Soft Matter Physics* 80 (October): 045102. https://doi.org/10.1103/PhysRevE.80.045102.

Barré, Julien, J. Carrillo, Pierre Degond, Diane Peurichard, and Ewelina Zatorska. 2018. "Particle Interactions Mediated by Dynamical Networks: Assessment of Macroscopic Descriptions." *Journal of Nonlinear Science* 28 (February). https://doi.org/10.1007/s00332-017-9408-z.

Betzel, Richard, Maxwell Bertolero, Evan Gordon, Caterina Gratton, Nico Dosenbach, and Danielle Bassett. 2019. "The Community Structure of Functional Brain Networks Exhibits Scale-Specific Patterns of Inter- and Intra-Subject Variability." *NeuroImage* 202 (July). https://doi.org/10.1016/j.neuroimage.2019.07.003.

Bianconi, Ginestra. 2015. "Interdisciplinary and Physics Challenges of Network Theory." *EPL (Europhysics Letters)* 111 (September). https://doi.org/10.1209/0295-5075/111/56001.

Broido, Anna, and Aaron Clauset. 2019. "Scale-Free Networks Are Rare." *Nature Communications* 10 (March). https://doi.org/10.1038/s41467-019-08746-5.

Domenico, Manlio, and Jacob Biamonte. 2016. "Spectral Entropies as Information-Theoretic Tools for Complex Network Comparison." *Physical Review X* 6 (September). https://doi.org/10.1103/PhysRevX.6.041062.

Education, By: IBM Cloud. n.d. "What Is Monte Carlo Simulation?" *IBM*. https://www.ibm.com/cloud/learn/monte-carlo-simulation.

Jolliffe, Ian, and Jorge Cadima. 2016. "Principal Component Analysis: A Review and Recent Developments." *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 374 (April): 20150202. https://doi.org/10.1098/rsta.2015.0202.

Kobayashi, Teruyoshi, and Tomokatsu Onaga. 2021. "Dynamics of Diffusion on Monoplex and Multiplex Networks: A Message-Passing Approach," March.

Lee, Clement, and Darren Wilkinson. 2019. "A Review of Stochastic Block Models and Extensions for Graph Clustering." *Applied Network Science* 4 (December). https://doi.org/10.1007/s41109-019-0232-2.

Li, Zichao, Peter Mucha, and Dane Taylor. 2017. "Network-Ensemble Comparisons with Stochastic Rewiring and von Neumann Entropy." *SIAM Journal on Applied Mathematics* 78 (April). https://doi.org/10.1137/17M1124218.

Libretexts. 2020. "7.3: The Heisenberg Uncertainty Principle." *Physics LibreTexts*. Libretexts. https://phys.libretexts.org/Bookshelves/University_Physics/Book:_University_Physics_(OpenStax)/Book:

_University_Physics_III_-_Optics_and_Modern_Physics_(OpenStax)/07:_Quantum_Mechanics/7.03:_The_Heisenberg_Uncertainty_Principle.

Mishra, Mrinal, and Nisha Gupta. 2009. "Monte Carlo Integration Technique for Method of Moments Solution of Efie in Scattering Problems." *Journal of Electromagnetic Analysis and Applications* 1 (December): 254–58. https://doi.org/10.4236/jemaa.2009.14039.

Newman, M. 2006. "Finding Community Structure in Networks Using the Eigenvectors of Matrices." *Physical Review. E, Statistical, Nonlinear, and Soft Matter Physics* 74 (October): 036104. https://doi.org/10.1103/PhysRevE.74.036104.

Pappas, Stephanie. 2014. "Why It's Perfectly Normal to See Jesus in Toast." *LiveScience*. Purch. https://www.livescience.com/45414-brain-face-pareidolia.html.

Reichardt, Jörg, and Stefan Bornholdt. 2006. "Bornholdt, S.: Statistical Mechanics of Community Detection. Physics Review E 74(1), 016110." *Physical Review. E, Statistical, Nonlinear, and Soft Matter Physics* 74 (August): 016110. https://doi.org/10.1103/PhysRevE.74.016110.

SOVEREIGN, MICHAEL, RICHARD NOLAN, and JAMES MANDEL. 2007. "Application of Spectral Analysis." *Decision Sciences* 2 (June): 81–105. https://doi.org/10.1111/j.1540-5915.1971.tb01595.x.

Sun, Weitao. 2017. "The Relationship Between Low-Frequency Motions and Community Structure of Residue Network in Protein Molecules." *Journal of Computational Biology* 25 (September). https://doi.org/10.1089/cmb.2017.0171.

Taylor, Dane, Sean Myers, Aaron Clauset, Mason Porter, and Peter Mucha. 2015. "Eigenvector-Based Centrality Measures for Temporal Networks." *Multiscale Modeling & Simulation* 15 (July). https://doi.org/10.1137/16M1066142.

Zhang, Yan, A.J. Friend, Mason Porter, James Fowler, and Peter Mucha. 2007. "Community Structure in Congressional Cosponsorship Networks." *Physica A: Statistical Mechanics and Its Applications* 387 (August): 1705–12. https://doi.org/10.1016/j.physa.2007.11.004.

n.d. *Avrim Blum: CS 598 (CRN 62819) Topics in Algorithms (2015)*. https://www.cs.cmu.edu/~avrim/598/.

n.d. *Quantiki*. https://www.quantiki.org/wiki/von-neumann-entropy.