# REVIEW MATERIALS FOR NETWORKS QUAL

DARYL DEFORD

## Introduction

This document contains basic materials collected in preparation for the networks qual. It is certainly not complete or exhaustive, but instead represents some of the basic definitions, concepts, and terminology of the field. The structure is currently based on the January 2015 qual syllabus outline and includes responses to some of the February 2015 sample questions.

## 1. Basic Constructions

1.1. **Graphs.** Graphs, or abstract collections of nodes and edges, form the main objects studied in the theory of complex networks. These models can be classified according to many different types of topological properties, but the underlying structures are quite simple. The general procedure for modeling with networks is to first identify the objects of interest (nodes), the crucial relations between then (edges), and then add more structure (such as weights or embeddings) depending on what is available in the original data.

Although there are many textbooks focusing on graph theory from a combinatorial perspective, these resources tend to be mostly irrelevant for the study of complex networks. The problem is that in real world settings many of the combinatorial properties and statistics are nonsensical. As an example, consider these basic graph theory questions: how many nodes does the internet have? does your router have an even or odd number of neighbors? what is a minimal independent set on Facebook? These are clearly ill–posed questions, but they highlights the disparity between the two fields. It is not possible to do much precise counting when the objects that you are studying are at best a noisy approximation to a fixed real world scenario, and more importantly analysis of one particular instance of the network has only small practical significance.

1.1.1. *Basic Graphs.*

(a) (undirected unweighted) These are the most basic abstraction underlying each of the other more complex network models. The application is modeled as a (finite) collection $V$ of nodes, together with a subset $E \subseteq V \times V$ that represents some relation of interest between the objects represented by the nodes. These relations are assumed symmetric and are referred to as edges. Examples include symmetric social networks, symbiote relationships, and physical computer networks. The corresponding adjacency matrix is $0-1$ and symmetric.

(b) (undirected weighted) In this case each edge is assigned a weight that represents some aspect of the underlying system. Examples include transportation networks with the weights representing capacities or co–authorship networks with weights representing the number of joint papers per pair of authors. The adjacency matrix is symmetric.

(c) (directed unweighted) Here the edges are no longer assumed to be symmetric (they are viewed as ordered pairs), so they carry more specific information. Directed edges are frequently referred to as arcs while the networks themselves are usually called digraphs. Examples include river flow networks, citation networks, and predation networks. The adjacency matrix need not be symmetric for directed graphs.

(d) (directed weighted) In this case the edges are both directed and weighted. An example is the WWW with nodes representing webpages and a directed edge of weight $w$ between two nodes $a$ and $b$ if there are $w$ links from page $a$ to page $b$.

*Date*: March 16, 2015.

1.1.2. *Bipartite Graphs.* A graph is said to be bipartite if the set $V$ can be partitioned into two distinguished subsets $A, B \subset V$ such that all of the edges connect a vertex in $A$ to a vertex in $B$ ($E \subseteq A \times B$). Bipartite graphs may also be weighted or directed. Graph theoretically, they are usually characterized as graphs with no odd cycles, but in the networks setting bipartite networks are usually specifically constructed with the sets $A$ and $B$ representing different types of objects. For example, networks matching actors to movies they have acted in or matching scientists to papers they have authored. In these cases the bipartition is selected at the beginning of the modelling process. Frequently, the networks are studied by projecting onto a single partite set, by associating all edges in $A$ that are attached to each node in $B$ (or vice versa).

1.1.3. *Hypergraphs.* In a hypergraph edges are allowed to connect more than two nodes at a time. Although this adds to the expressive power of the abstraction it comes at a significant practical cost. Many of the tools that work for analyzing simple networks fail for hypergraphs. Additionally, many of the standard algebraic techniques and algorithms for simple networks do not translate naturally to this setting.

1.1.4. *Multiplex Networks.* A multiplex network is a collection of different edge sets all associated to the node set. A natural example to consider is a set of individuals and their social networking behavior, using one collection of edges for each separate social network. Multiplex structures can also arise from disaggregation of data, for example, splitting up trade data by the type of good that is being exchanged.

1.1.5. *Trees.* A tree is a network with no loops. One distinguishing property is that given any two nodes in a tree there is a unique path connecting them. They are frequently used to model hierarchical data. Examples include citation networks and genealogical networks.

1.2. **Matrix Representations.** It is common to represent a network as a matrix to try to leverage the power of linear algebra in the study of these objects. Many of the fundamental results and algorithms of the field are derived in this fashion, and this document contains several of these examples.

1.2.1. *Adjacency Matrix.* The simplest matrix representation of a network is known as the adjacency matrix $A$. The entries $A_{i,j}$ are defined to be one if there is an edge in the graph between nodes $i$ and $j$ and zero otherwise. As mentioned in the previous sections this structure is simple to modify in order to capture the distinctions between the various (un)directed (un)weighted networks.

1.2.2. *Incidence Matrix.* Another valuable matrix representation of a network is the incidence matrix $N$[1]. This $n \times m$ matrix is formed by associating each column of the matrix to an edge and placing a one in the entries corresponding to the nodes that are connected by the edge. A slight modification of this construction gives the Cholesky decomposition of the graph Laplacian. We take the incidence matrix associated to a undirected netwrok and select one positive entry in each column to be scaled by $-1$. Then, $L = D - A = NN^T$. The signs can be assigned arbitrarily, as can be seen from writing out the matrix product in full.

This construction generalizes nicely to hypergraphs, where there are possibly more than two nodes on each edge, in a way that the adjacency matrix does not. In this case however, there is no consistent choice of signs that will allow for a matrix product like the Laplacian. A similar construction also generalizes for bipartite networks, such as the authorship network, where the two sets are authors and papers with edges representing authorship. In this case the columns correspond to the papers with entries of one corresponding to authors. Then, the node projections mentioned in 1.1.2 are $NN^T$ and $N^TN$.

1.2.3. *Biadjacency Matrices.* A representation that is commonly used for bipartite networks is the Biadjacency matrix, $B$[2]. Since there are no nodes within the partite sets, this representation condenses the wasted space in the standard adjacency matrix which will have a $2 \times 2$ block form with zero blocks along the diagonal. In $B$, he rows are indexed by the elements of the first partite set and the columns are indexed by the elements of the second, with entries representing edges as in the adjacency matrix.

[1]Newman and many other authors use $B$ for this matrix, but they also use $B$ for several other entirely unrelated matrices such as modularity matrices. In order to "reduce" the potential for confusion I will use $N$ throughout this document.

[2]At least here the $B$ makes sense.

## 2. Taxonomies

Network models of complex systems tend to exhibit several types of distinct topological structures. The importance and ubiquity of these features is one of the things that separates networks from standard graph theory. The two most commonly studied of the following characterizations (small world networks (2.2) and scale free networks (2.3)) are not perfectly formal in a mathematical sense, but have been adopted by the networks community. It can be difficult to empirically determine whether a particular network actually falls into one of these categories, but many broad classes of networks have been assumed to satisfy these conditions in the networks literature. The first subsection below defines some standard graph–theoretic families that are frequently used in the study of networks.

### 2.1. **Graph Families.** This section describes some of the most frequently studied graph families.

(1) (complete graph) A complete graph on $n$ nodes, denoted $K_n$ is the graph that has all possible edges. These are also known as cliques.
(2) (complete bipartite graph) A complete bipartite graph with partite sets of $m$ and $n$ nodes, denoted $K_{m,n}$ is the bipartite graph that has all possible edges.
(3) (cycle graph) A cycle graph on $n$ nodes, denoted $C_n$, is a graph where the nodes and edges form a cycle.
(4) (path graph) A path graph on $n$ nodes, denoted $P_n$, is a graph where the nodes and edges form a path. These are an example of a tree.
(5) (regular graph) A regular graph is one where every node has the same number of neighbors.
(6) (star graph) A star graph is a tree with one central node that is connected to some number of leaf vertices which are only connected to the center.
(7) (planar graph) A planar graph is a graph that can be drawn in the plane without any crossing edges. A graph is planar if and only if it does not have $K_5$ or $K_{3,3}$ as a minor.
(8) (hypercube graph) A hypercube graph is formed by taking the nodes to be all binary strings of length $n$ and connecting two nodes if their Hamming distance is one.
(9) (dense graph) A graph is called dense if it has $\mathcal{O}(n^2)$ edges.
(10) (sparse graph) A graph is called sparse if it has relatively few edges compared to the square of the number of vertices.

### 2.2. **Small World Networks.**

Small World networks are characterized by the property that the expected shortest path between two arbitrary nodes in the graph is about $\log n$. Sometimes they are also required to have a high clustering coefficient. This leads to the network having a fat–tailed degree distribution. The world wide web and most social networks (probably) satisfy this condition. Transportation networks with hubs, such as airline and train networks, also satisfy these conditions. Counterexamples include things like local transportation networks, which are too regular, or social networks that encompass a large time scale, which vastly increases the average path length.

The name comes from Milgram's small world experiments. The Watts–Strogatz model (see section 4a) is one of the most frequently studied generative networks models exhibiting this behavior. Assumptions about small world networks are used in many subjects to inform decision making. For example, small world properties of social networks are used to market products more effectively while the robustness profile (see section 3e) of small world networks have led to their use in technological networks.

### 2.3. **Scale Free Networks.**

A network is said to have the scale free property if its distribution of node degrees satisfies the power law, usually with parameter $2 \leq \gamma \leq 3$. That is, the fraction of nodes with degree $k$ is asymptotically $k^{-\gamma}$. Citation networks and other models usually assumed to form from some amount of preferential attachment tend to satisfy this property. These networks tend to have many "hubs" or nodes with relatively high degree and a clustering coefficient distribution that also follows a power law. The clustering distribution implies that the low degree nodes exist in small dense sub–networks connected by the hubs. The Albert–Barabsi model (see Section 4a) is one of the most frequently studied generative networks models exhibiting this behavior. However, this model does not give rise to the large clustering coefficient that is usually required of small world networks.

## 3. Network Statistics

This section contains descriptions of some of the basic tools used to analyze and differentiate networks.

3.1. **Degrees.** The simplest network statistic quantifies the number of edges per node in the network.

(a) (degree) The degree of a node is the number of edges incident to it.
(b) (weighted degree) The weighted degree of a node is the sum of the weights of each edge incident to it.
(c) (degree distribution) The degree distribution of a network is an ordered list of the degrees of each node in the network. The Erdös–Gallai Theorem characterizes when a particular list of integers can be a degree distribution. In network theory the distribution of the degrees is an important invariant of the structure. An interesting facet of complex networks is that while Erdös–Renyi random graphs have a Poisson or binormal degree distribution most observed networks have fat tailed distributions.
    (i) (power laws) A degree distribution is said to satisfy a power law if the probability of a node in the network having degree $k$ is $k^{-\gamma}$ usually for a value of $\gamma$ between two and three. These distributions have a scale free property (see 2.2) since multiplication by a constant scales the proportion by the constant to the $\gamma$. On a log–log plot these distributions appear linear.
    (ii) (heavy tailed) A long/heavy/fat tailed distribution has more nodes with high degree than a normally/exponentially/uniformly (hopefully clear from context) distributed collection of integers.

3.2. **Paths.** The next important class of network statistics deals with paths in networks.

(a) (geodesics) Shortest paths in a network lead to several important statistics, both for nodes and the entire network. Except in the case of tree networks these geodesics are rarely unique for a given choice of vertices. Many problems dealing with enumerating particular types of paths such as Hamiltonian (non–vertex repeating) or Eulerian (covering every edge) are of significant combinatorial interest. The following statistics are among some of the most frequently studied in complex networks.
    (I) Node Statistics:
        (i) The length of the shortest path between node $i$ and $j$ in a network is denoted $\ell_{i,j}$.
        (ii) The eccentricity of a node is $ecc(i) = \max_{i \neq j} \ell_{i,j}$.
        (iii) The closeness centrality of a node is $g_i = \frac{1}{\sum_{i \neq j} \ell_{i,j}}$.
        (iv) The number of shortest paths between $i$ and $j$ is $\sigma_{i,j}$.
        (v) The number of shortest paths between $i$ and $j$ through vertex $k$ is $\sigma_{i,j}(k)$.
    (II) Network Statistics:
        (i) The diameter of a graph is $d = \max_{i,j} \ell_{i,j}$.
        (ii) The average shortest path in a graph is $\frac{1}{n(n-1)} \sum_{i,j} \ell_{i,j}$.
        (iii) The radius of a graph is $\min_i ecc(i)$.
(b) (connectivity) A network is connected if there is a path between any two nodes. Since the nodes in different connected components cannot interact we usually analyze the components individually. The number of components in a network is given by the multiplicity of zero as an eigenvalue of the graph Laplacian. Connectivity is further characterized by the second smallest eigenvalue of the graph Laplacian (without multiplicities) (see Section 6 b). This is known as the Fielder value.

    Connectivity is frequently used as the defining condition in robustness/fragility measures (see Section 3e). Generally a relaxed definition of connectivity is preferable in many applications to prevent trivial solutions such as deleting a pendant edge to disconnect a network. These relaxations are directly related to clusering, also through the graph laplacian (see Section 7 c).

3.3. **Assortativity.** Assortativity (homophily in the social sciences) is the characteristic of network formation that suggests that nodes are more likely to be adjacent to nodes with similar properties. For example, in assortative networks, high degree nodes are expected to be connected to other hubs, while small degree nodes are more likely to be connected to other low degree nodes. In disassortative networks the opposite behavior is expected. This concept is usually studied in terms of the degrees of the nodes (because it is easier/tractable) although other types of correlations are possible, especially when outside correlations can be computed with respect to parameters not captured by the network abstraction itself. The most frequently used statistics are the Pearson Correlation Coefficient which is a statistical correlation and the neighbor connectivity which measures the average neighborhood size for nodes adjacent to nodes of fixed degree.

Assortativity can also be generalized to directed networks, although there are additional complications. In this case we must distinguish between in and out degrees and connections between nodes can be studied by pairing any combination of these, for example in degree with in degree along out edges. It can also be computed locally by considering the amount of assorativity that is contributed by each node individually. This consideration is used in iterative optimization algorithms that rely on maximizing assortativity by moving nodes between partition components. The most common model for explaining why this type of behavior is observed in many networks is described below, although there are other candidates.

(a) (preferential attachment) Preferential attachment is a model that attempts to explain the assortativity that occurs in many real networks such as the WWW and citation networks. Reduced to a buzz phrase, the idea is "the rich get richer." The idea is that during the network formulation process each new node has probability of connecting to any particular previous node proportional (in some sense) to the degree of the previous node. The most famous method making use of this process is the Albert–Barabasi algorithm. Given appropriate parameters, a preferential attachment process can generate a power law degree distribution, as is seen in many real world networks.

**Derive the formula for calculating the assortativity of a network.**

(This first approach follows the statistical methods in Newman's 2002 paper)

Assortativity in the network setting usually refers to the likelihood that nodes of similar degree are connected, though it can be applied more generally. In order to complete the derivation we need to define some notation. Let $p_k$ be the probability that a random node has degree $k$. The probability distribution for the degree of a node that is selected as a neighbor of a randomly chosen node is $kp_k$ since a neighbor is likely to have higher degree. Assortativity is traditionally computed in terms of the remaining degree of such a node which is the distribution of the number of edges minus the one that was used to arrive at that node.

The distribution of the remaining degree $q_k$ can then be determined to be $q_k = \frac{(k+1)p_{k+1}}{\sum_j jp_j}$ since the chosen node actually has degree one higher and the edge that carried us to the node could have been from a node of any degree. We further define $e_{j,k}$ to be the joint probability distribution that given an arbitrary edge the incident nodes have remaining degrees $j$ and $k$. Summing all the $e_{j,k}$ gives one, while fixing $k$ and summing over all $j$ gives $q_k$.

If there is no assortativity in the network we should expect that $e_{j,k} = q_j \cdot q_j$ so a measure of the assortativity can be realized as the average value over all edges of the network as: $\sum_{j,k} jk(e_{j,k} - q_j \cdot q_k)$. This value is traditionally normalized to allow for cross network comparisons by dividing by the maximum possible value which occurs when $e_{j,k} = q_k\delta_{j,k}$. Substituting this simplification in we see that $\sum_{j,k} jk(q_k\delta_{j,k} - q_j \cdot q_k) = \sum_k k^2 q_k - \left(\sum_k kq_k\right)^2 = \sigma_q$. Together we obtain a final formula: $r = \frac{\sum_{j,k} jk(e_{j,k} - q_j \cdot q_k)}{\sigma_q}$. This is equivalent to the Pearson Correlation Coefficient of the degrees of nodes incident to the same edge across the network. Note that this methodology only captures possible linear correlation between the degree connections, so it is important to be aware of other possible distributions of the correlations.

Here is a more natural[3] way to view assortativity as a special case of scalar modularity. Recall that for a partition of a network the (enumerative) modularity is defined as $Q = \frac{1}{2m}\sum_{i,j} B_{i,j}\delta(c_i, c_j)$, where $B_{i,j} + A_{i,j} - \frac{k_i k_j}{2m}$ and the $c_i$ represent the respective components of the vertices. Further this is compared to the maximum possible value on the network, where the edges all lie within components so $i \sim j$ implies $\delta(c_i, c_j) = 1$, giving $Q_{\max} = \frac{1}{2m}\sum_{i,j}(A_{i,j} - \frac{k_i k_j}{2m})$. The idea is capturing the overabundance of edges in the observed network that lie within instead of between the clusters.

Taking this idea to the case where our paritions are defined by scalar variables instead of enumerative classes we can proceed with a similar derivation, first computing the average over the edges as

$$\mu = \frac{\sum_{i,j} A_{i,j}x_i}{\sum_{i,j} A_{i,j}} = \frac{1}{2m}\sum_j k_j x_j$$

and then computing the covariance of the degrees across the edges. Although the following computation is messy the underlying idea is natural and the final result is equivalent to the standard modularity for enumerative partitions.

[3]less statistical

$$\begin{aligned}
cov(k_i, k_j) &= \frac{1}{2m}(\sum_{i,j} A_{i,j}(x_i - \mu)(x_j - \mu)) \\
&= \frac{1}{2m}(\sum_{i,j} A_{i,j}(x_i x_j - \mu(x_i - x_j + \mu))) \\
&= \frac{1}{2m}(\sum_{i,j} A_{i,j} x_i x_j - \mu \sum_{i,j} A_{i,j}(x_i + x_j - \mu)) \\
&= \frac{1}{2m}(\sum_{i,j} A_{i,j} x_i x_j - \mu(\sum_i \sum_j A_{i,j} x_i + \sum_j \sum_i A_{i,j} x_j + \sum_i \sum_j A_{i,j}\mu)) \\
&= \frac{1}{2m}(\sum_{i,j} A_{i,j} x_i x_j - \mu(\sum_i k_i x_i + \sum_j k_j x_k - \sum_{i,j} A_{i,j}\mu)) \\
&= \frac{1}{2m}(\sum_{i,j} A_{i,j} x_i x_j - \mu(2m\mu + 2m\mu - 2m\mu)) \\
&= \frac{1}{2m}(\sum_{i,j} A_{i,j} x_i x_j - 2m\mu^2) \\
&= \frac{1}{2m}(\sum_{i,j} A_{i,j} x_i x_j - \mu^2) \\
&= \frac{1}{2m}(\sum_{i,j} A_{i,j} x_i x_j - (\frac{1}{2m})^2 \sum_{i,j} k_i k_j x_i x_j)) \\
&= \frac{1}{2m}(\sum_{i,j} (A_{i,j} - \frac{k_i k_j}{2m}) x_i x_j)
\end{aligned}$$

Similarly, we consider for a perfectly assortative network the maximum value which occurs when we have $x_i = x_j$ for the coefficient of $A_{i,j}$ in the definition. This then gives a perfect mixing value of

$$\frac{1}{2m}(\sum_{i,j} A_{i,j}(x_i^2 - \frac{k_i k_j}{2m}) x_i x_j) = \frac{1}{2m}(\sum_{i,j} (k_i \delta(i,j) - \frac{k_i k_j}{2m}) x_i x_j)$$

Taking the quotient of the observed and ideal values gives the assortativity coefficient:

$$\frac{\sum_{i,j}(A_{i,j} - \frac{k_i k_j}{2m}) x_i x_j}{\sum_{i,j}(k_i \delta(i,j) - \frac{k_i k_j}{2m}) x_i x_j}$$

Then, we can obtain the standard assortativity for degrees by substituting in $k_i$ for $x_i$ into the expression above. It is easy to check that this definition agrees with the naïve notion of simply computing the covariance of degrees across the edges of the network as in the statistical interpretation.

3.4. **Centrality.** One of the most common ways to analyze important nodes in a network is through measures of centrality. Broadly these are metrics that attempt to characterize nodes through their importance to the network. The choice of importance metric determines the algebraic computations that are necessary.

(1) (degree centrality) In this metric the centrality of a degree is set equal to its number of neighbors. This is a simple measure to compute but it does carry some first order information about the network. Citation counts of papers and number of followers on Twitter are examples of this sort of centrality measure.
(2) (betweeness) Betweeness centrality is a measure of how well connected a node is to the other nodes in a network using the number of paths through the node as a proxy for centrality.
   (I) Node Statistics
      (i) The number of edges between neighbors of $i$ is $e_i$.
      (ii) The clustering coefficient of $i$ is $c(i) = \frac{2e_i}{deg(i)(deg(i)-1)}$.
      (iii) The betweeness centrality of $i$ is $b_i = \sum_{k \neq i \neq j} \frac{\sigma_{k,j}(i)}{\sigma_{k,j}}$.
   (II) Network Statistics
      (i) The clustering coefficient of the network is $\langle c \rangle = \frac{1}{n}\sum_i c(i)$.
      (ii) The $\delta$–clustering coefficient of the network is $c_\delta = \frac{\#\delta}{\binom{n}{3}}$.

   This quantity $b_i$ is measuring how often the node $i$ lies on a geodesic between two arbitrary vertices. For many networks this is a metric with a large range of values that clearly distinguishes between high and low values. The geodesic assumption can be misleading for some natural kinds of network dynamics, such as searching for information. It can also be adjusted to instead reflect the expected number of visits of an absorbing random walk between two arbitrary nodes. Note that this definition is no longer symmetric as can be observed by considering a pendant edge. This methodology has the opposite assumption that there is no optimization in the "flow" across the network.

(3) (closeness) Closeness centrality is related to the quantity $\ell_i = \frac{1}{n}\sum_j d_{i,j}$, where $d_{i,j}$ is the length of a geodesic from $i$ to $j$. This has small values when $i$ is close to many nodes so the centrality statistic is defined as the inverse of this number $c_i = \frac{1}{\ell_{i,j}}$. This is a metric that takes a very small range of values across small world type networks, because of the $\log(n)$ diameter assumption and high local clustering.
(4) (eigenvector) Eigenvector centrality ranks the centrality of nodes in the network by the size of their corresponding entries in the leading eigenvector of the adjacency matrix. Dynamically these processes model random walks on the network. Another technique introduced by Kleinberg is to rate authority centrality and hub centrality separately by computing the eigenvectors corresponding to $AA^T$ and $A^TA$ separately (note that this is only sensible for directed networks). For undirected graphs these are the same since $A$ is symmetric, but for undirected graphs they capture very different information. For examples consider Twitter or the WWW, where having many incoming nodes carries very different information than having many outgoing nodes. Similarly, the property of having arcs to/from hubs (in the power law case) distinguishes between authoritative nodes and standard nodes.
(5) (Katz) Best technique for directed acyclic graphs where eigenvector centrality cant be measured. The idea is to award each node some measure of centrality at each step to avoid absorbing states in the Markov process. It can be defined as

$$C_{Katz} = \sum_{k=1}^{\infty}\sum_{j=1}^{n} \alpha^k (A^k)_{i,j}$$

for some $\alpha$ less than the reciprocal of the largest eigenvalue of $A$ for a single node or

$$((I - \alpha A^T)^{-1} - I)1$$

for the entire network. Usually iterative methods are used to calculate these values since it is difficult to prove convergence bounds for this type of system. Google's PageRank algorithm is a version of this centrality measure, with the adjacency matrix normalized to make it stochastic.

3.5. **Robustness.** Robustness in a network is a notion of stability. We are usually concerned with the question "How does the perturbation of nodes or edges from the network change the network topology or dynamics?" In general, the connectivity of the network is used as a proxy for the health of the system. This is very natural in examples from biology or power systems, where disconnection of the network can have severe consequences. The main dynamical technique in the literature dealing with robustness is percolation although there are other approaches, such as in the world trade web paper.

(a) (fragility) Wildly underspecified definition ahead: A network is said to be fragile if it is easy to decompose by deleting/perturbing few edges or nodes and robust if many must be deleted/perturbed. Currently there is not a consistent general theory (in the mathematical sense) for what it means to be robust although for many particular applications natural notions present themselves. For example, in a power network connectivity is a natural measure since disconnection means there are homes without power. Alternatively, for weighted networks there may instead a natural measure of flow with a minimal quantity that must be preserved for the network to function.
(b) (attacks) Distinctions are usually made between different models of network degradation. Random failures such as those that occur over time due to normal physical dynamics are usually modelled by selecting edges or nodes with some natural distribution guided by the application and either deleting them entirely or degrading them by a constant multiple $0 \leq \alpha < 1$. On the other hand targeted attacks can be very damaging to some types of networks. For example, given an adversary that knows the abstract network structure they can select the "most important" nodes or edges, perhaps using some centrality or betweeness measure, to degrade. This in particular is damaging to networks with bridges or hubs that function as weak points in this analysis.

Attacks are frequently modeling using techniques from physics broadly classified as percolation theory. Them main idea is that some selection (either uniform or application determined) of the nodes (site percolation) or edges (bond percolation) are degraded or assumed to be non–functional and the properties of the networks are examined under these assumptions. A natural motivation is disease epidemics, where vaccinated indivudals do not interact with the disease dynamics.

(c) (taxonomies) Different types of networks have very different robustness properties. Trees for example are very fragile since any deletion is likely to separate the nodes into disjoint components. Scale free networks are weak to targeted attacks that destroy several nodes, but quite robust against random node or edge failure. This is why technological networks frequently employ construction techniques modelled on this behavior. On the other hand, Erdös–Renyi networks are quite robust against targeted attacks because in general there are not hubs in general. On the other hand, these graphs tend to serve as a poor model for most complex networks since usually their are underlying structural concerns such as preferential attachment, homophily, or design that enforce the construction of more significant and hence targetable nodes and edges.

## 4. Null Models

The idea behind null model analysis is to compare in a "principled" fashion a particular observed network with an ensemble of related random networks in order to determine the features of the observed network that are not likely to be caused by random behaviors. This is very intrinsically related to the idea that in most cases of interest the actual network that we are operating on is only an approximation of a snapshot of the underlying physical system. This section contains descriptions of the basic random network models and a discussion of how to use these models to discover significance from observed networks.

### 4.1. **Standard Random Network Constructions.**

(1) (Erdös–Renyi) Inputs: The number of desired nodes $n$ and a probability parameter $p$. Construct a network on $n$ nodes where each of the $\binom{n}{2}$ edges independently occur with probability $p$. Another standard version of this model selects an arbitrary graph from the collection of all graphs on $n$ vertices with $m$ edges uniformly, although this formulation makes it more difficult to calculate some standard network parameters since there is no assumption of independence on the edges.

(2) (Barabsi–Albert) Inputs: An initial network, a final number of nodes, and a fixed number $c$ of edges to add for each new node. This is an iterative process. At each step, until the final number of nodes is reached, add a new node to the network. Connect this new node to $c$ nodes already in the graph with probability $\frac{\deg(i)}{\sum_j \deg(j)}$. This preferential attachment process generates scale free networks.

(3) (Watts–Strogattz) Inputs: The number of desired nodes $n$, a probability parameter $p$, and the desired mean degree $d$. The construction begins with a ring lattice on $n$ nodes where each node is connected to its $\frac{k}{2}$ nearest neighbors. Visit the nodes sequentially and reattach each edge at that node with probability $p$. Select the new target for the reattached edges uniformly. This method produces small world graphs.

| Network Type | Average Degree | Average Path Length | Diameter | Clustering Coefficient | Degree Distribution |
|---|---|---|---|---|---|
| Erdös–Renyi | $np$ | $\log(n)$ | $\log(n)$ | $p$ | Binomial |
| Barabsi–Albert | $c$ | $\frac{\log(n)}{\log(\log(n))}$ | $\frac{\log(n)}{\log(\log(n))}$ | $n^{-\frac{3}{4}}$ | Scale Free |
| Watts–Strogattz | $k$ | $\log(n)$ | $\log(n)$ | $\frac{3}{4}$ | Poisson |

(4) (configuration models) Configuration models are a generalization of the Erdös–Renyi model that preserves the degree distribution of a network of interest. The idea is to cut each edge in the original network in half and reattach these "edge ends" at random. Since the number of ends at each node doesn't change the degree distribution is preserved. This is the null model used in the definition of modularity.

### 4.2. **Parameter Estimation.**

### 4.3. **Significance.** 

Null models are frequently used to determine if a computed parameter for a network of interest is interesting. This is usually done by identifying the parameter of interest, generating many random graphs that are structurally similar[4] to the observed network and comparing the values.

[4]also wildly undefined

## 5. Dimension Reduction

Dimension reduction techniques broadly fall under the heading of data analysis. They are used for many different purposes in all parts of applied mathematics. The key idea is to reduce in some fashion data that is presented is high dimensional space to a natural embedding in a smaller dimensional space that preserves as much of the original structure as possible. Mitigating the curse of dimensionality, providing useful visualizations, both for exploratory and explanatory purposes, and cleaning noisy data are all parts of dimension reduction techniques and methods. Many of these techniques and their results are intrinsically related to problems of clustering. We also note that standard network clustering techniques can be applied to data sets by interpreting the data as a network, perhaps by using thresholding or $k$ nearest neighbors to form a representative graph for the data.

5.1. **Multidimensional Scaling.** Multidimensional scaling (MDS) is a technique for embedding data into a Euclidean space so that the Euclidean distance (or another metric on $\mathbb{R}^n$) can be used as a "good" approximation to a given metric on a dataset. That is, given a dataset $X = \{x_i\}$ and a metric[5] $d : X \times X \to \mathbb{R}$ we wish to form an embedding $\varphi : X \to \mathbb{R}^k$ for some $k$ together with some metric $d'$ on $\mathbb{R}^k$ so as to minimize $\sum_{i,j} |d'(\varphi(x_i), \varphi(x_j)) - d(x_i, x_j)|$ or some similar variant. Usually we begin with a dissimilarity matrix $D_{i,j} = d(x_i, x_j)$ and state (and solve) the problem linear algebraically by forming a derived matrix that can represent the new inner products.

The choice of $k$ obviously influences the embedding process a great deal. For visualization of course, two or three dimensions are required, or alternatively there may be some intrinsic dimensionality to the data that is known in advance. For example, we may know or expect certain dimensions of our data are highly correlated and subtract these superfluous quantities from $k$. Alternatively, if our data stems from a Euclidean metric originally then the points can be embedded exactly using linear algebra and factoring a symmetric positive definite matrix. This method works for all data that comes from a metric and has the additional property of revealing the smallest dimension that provides an exact embedding as the rank of the derived matrix.

When the original measure is not a true metric it is not always possible to solve this problem exactly. One possible approach to correct this defect is to try to convert $d$ into a metric (or at least get it closer), such as is done with cosine dissimilarity measure. Regardless associated to any such embedding is a measure of stress that determines the effectiveness of an embedding at capturing the original information. These stress measures are frequently statistical and dependent on the particular data and can be used to search for the proper number of dimensions for an embedding.

Most of the standard software systems for data analysis (Matlab, R, Python, etc.) incorporate methods for doing MDS. These are usually computed iteratively from a random embedding, by adjusting the positions of the points slightly at each stage to minmize the stress function measuring the normalized difference between the embedded distances and the original dissimilarity matrix entries. Due to the random nature of the initialization it is possible that the software can return a local minimum instead of a global minimum, so it is recommended to perform the operation multiple times and select the embedding that minimizes the stress function.

**Show that if $D$ is a distance matrix giving distances between points in $\mathbb{R}^n$ the MDS will recover the coordinates of the points up to a rigid motion.**

Let $D$ be a matrix whose entries represent differences between $k$ points in $\mathbb{R}^n$. That is $D^2_{i,j} = \mathrm{d}(x_i, x_j)$. Since the points are assumed to be Euclidean already, we can realize the distance as an inner product: $D^2_{i,j} = ||x_i - x - j|| = \langle x_i - x_j, x_i - x_j \rangle$. If $X$ is a $n \times k$ matrix whose columns are the entries of the $x_i$ then the matrix $A = XX^T$ has entries $A_{i,j} = \langle x_i, x_j \rangle$. Thus, if we can construct the matrix $A$ from $D$ we can recover $X$ from the spectral decomposition of $A$ since it is symmetric and positive definite by construction.

Looking entrywise we see that $D^2_{i,j} = \langle x_i, x_i \rangle - 2\langle x_i, x_j \rangle + \langle x_j, x_j \rangle = A_{i,i} - 2A_{i,j} + A_{j,j}$. Considering the entries of $A$ as $k^2$ variables we obtain a system of equations that can be solved exactly to obtain $A$. Since we can obtain $X$ from $A$, this gives us at least a translate of the original vectors up to a rotation since the Euclidean distance is translation and rotationally invariant. In practice however, these problems are usually solved with iterative approaches for a fixed dimension, using a stress measure as an objective function.

[5]Need not be an actual metric, but the closer it is to satisfying all of the metric properties the better results obtained from it will be.

Another approach is to form the matrix $M_{i,j} = \frac{D_{1,i}^2 + D_{1,j} - D_{i,j}^2}{2}$. This is also a symmetric matrix whose spectral decomposition gives rise to another realization of $X$ as above. In this case we are shifting the first element in $X$ to the origin and then the entries of $M$ represent the differences from $x_0$ to each other point in the data set. The rank of $M$ captures the minimal dimension such that the distances can be realized exactly in Euclidean space with the standard metric.

5.2. **Principle Component Analysis.** Principle component analysis (PCA) is a technique from linear algebra for projecting a dataset of vectors onto a lower dimensional subspace in such a way as to capture most of the variance of the original data. The idea is that we can use the spectral decomposition of the covariance matrix of a data set (derived from the $D_{i,j}$ above) to suggest the most efficient vectors to project our information onto by selecting the eigenvectors corresponding to the largest eigenvalues. Specifically, we compute the sample covariance matrix $\frac{1}{n-1}(X - \hat{X})(X - \hat{X})^T$ and use its orthogonal decomposition.

In practice, parts of this decomposition may be done by SVD both for speed and numerical stability. The total variance is preserved under the diagonalization, but the new random variables are clearly uncorrelated in the change of variables. Thus, the eigenvalues, sorted descending by magnitude, describe decreasingly small proportions of the total variance explained by projecting onto the first $k$ eigenvectors. Note that these projections are orthogonal since the covariance matrix is symmetric. This means that the projections can be computed independently for the most meaningful $k$ eigenvectors to achieve the representation in $\mathbb{R}^k$ that explains the most variance possible.

## 6. Dynamics on Networks

In order to understand the effects of topological structure on real life systems it is frequently useful to consider dynamics on networks. Many of the most powerful descriptive invariants of network theory are revealed through considering the action of some operator across a network. Most commonly the operators are assumed to be linear in some fashion so that techniques from linear algebra, such as eigenvalue analysis, can be performed. As we shall see this leads to many fascinating results, especially in the case of the graph Laplacian.

6.1. **Markov Dynamics.** The first type of dynamics we consider are those governed by Markov chains. For our purposes a Markov process is a stochastic process where the conditional probabilities of each state depend only on the current state. In the context of networks, we are usually interested in discrete state Markov chains, this is a finite collection of states, together with a (sequence of) matrix(ces) whose entries capture the transition probabilities, i.e. $A_{i,j}$ is the probability of transitioning from state $j$ to state $i$.[6] Here we will generally only consider time homogeneous systems with a fixed transition matrix. This matrix is left stochastic and hence it can be shown (see next subsubsection) with Perron–Frobenius that one is the largest eigenvalue of the matrix and the corresponding eigenvector has all positive entries and can hence be interpreted as a probability vector (perhaps after normalizing). It is this steady state vector that we are most interested in.

We saw an example of this already in the context of centrality scores, when we considered the matrix $AD^{-1}$. This matrix represents the transition matrix of a Markov process defined on the nodes modeling a random walk on the network, where the edges are selected uniformly. This means that if the walker is at node $i$ then proceeding to the next step the walker has $\deg(i)$ choices or equivalently each edge is chosen with probability $\frac{1}{\deg(i)}$. This is a more interesting notion in the context of directed networks, since for undirected networks the steady state proportions are just given by $\frac{D\mathbf{1}}{||D\mathbf{1}||}$.

6.1.1. *Perron–Frobenius.* It is a little bit of a pet peeve of mine that most sources about networks gloss over how Perron–Frobenius actually applies to stochastic matrices so here is an outline. To begin with, Perron–Frobenius guarantees that for certain classes of non–negative matrices there exists a unique, real maximum eigenvalue of multiplicity one, whose corresponding eigenvector has all positive entries.[7] Moreover, this is the only eigenvector of the matrix with strictly positive entries. Before proceeding, we need to discuss which types of matrices actually satisfy the hypotheses of this theorem.

[6]It is traditional in the probability/statistics literature to have stochastic matrices act on the right of column vectors. I am not a fan of that convention.

[7]The theorem actually contains more content than this, including a very efficient formula for the projection onto the corresponding eigenspace. This portion of the theorem does not apply for operators that are irreducible and imprimitive.

The Perron–Frobenius theorem applies directly to any matrix with strictly positive entries. In our Markov example this means that at any time step any state may be reached from each other state. Additionally, the theorem applies without modification to primitive non–negative matrices. A matrix is primitive if there exists an integer $m$ so that the $m^{th}$ power of the matrix has all positive entries. Primitive matrices can also be characterized as irreducible, aperiodic matrices. A matrix is irreducible if for all $i, j$ there exists an $m$ such that the $m^{th}$ power of the matrix has a positive entry in the $i, j$ position. This can also be expressed by showing that the associated digraph is strongly connected.

The period of a non–negative matrix is the greatest common divisor of the lengths of closed paths in the associated digraph. An irreducible matrix is aperiodic and hence primitive if its period is equal to one. Perron–Frobenius for irreducible matrices has the same eigenvalue consequences described in the previous paragraph. Perron–Frobenius does not apply directly to reducible matrices. However, every reducible matrix may be permuted to an upper block diagonal form whose diagonal blocks are irreducible. Since the spectrum of the matrices must be the same sometimes information can be gleaned from this change of variables. This is the technique sometimes used for general stochastic matrices.

Unfortunately, many stochastic matrices, especially those attached to complex networks are neither primitve nor irreducible. In this case it is still true that they have leading eigenvalue one and non–negative (not necessarily positive) eigenvector. Since the matrix is left–stochastic the all ones vector is a right eigenvector corresponding to one. Gershogorin's Circle Theorem guarantees that one is at least a maximal eigenvalue in norm for the matrix. Additionally, since the matrix is non–negative we still have that every eigenvector corresponding to one has all entries with the same sign, since $A - I$ only has non–positive entries on the main diagonal. In this case there is no guarantee of multiplicity one. An easy example to see this failure is with a Markov chain that has several absorbing states[8]. In these cases there is an eigenvector corresponding to the eigenvalue one for each possible absorbing state. Clearly one is still the leading eigenvalue and this set of vectors is linearly independent of dimension equal to the number of absorbing states.

6.1.2. *Basic Definitions.* The transition matrix of a Markov chain can be interpreted as a weighted, directed graph with the states as nodes and the probabilities as edge weights. This provides convenient language for characterizing the properties of a given chain. For the remainder of this section we will not distinguish between a Markov chain and its associated digraph. This following are some basic definitions associated to Markov analysis.

(1) (essential) A state $i$ in the chain is essential if for every other state that can be reached from $i$ there is a path returning to $i$.
(2) (irreducible) A Markov chain is irreducible if it is strongly connected.
(3) (primitve) A Markov chian is primitive if there exists an integer $m$ such that any two (not necessarily distinct) vertices in the graph can be connected by a path of length $m$.
(4) (steady state) A stead state of a Markov chain is an eigenvector corresponding to one. For primitive chain this can be computed as the limit of the matrix powers. This property implies that for such a chain the initial distribution has no consequences in the limit.
(5) (period) The period of a state is the greatest common divisor of all lengths of cycles that begin at the state. A state is aperiodic if its period is one. Note that in bipartite graph, all periods must be even since there areno odd cycles.
(6) (hitting time) The hitting time $T_i$ of a state is the random variable associated to the chain counting the minimum number of steps until the path returns to the state.
(7) (transient) A state $i$ is transient if there is non–zero probability that a path from $i$ never returns to $i$. A non–transient state is called recurrent. If we define $q_i^n = P(T_i = n)$ then the state is transient if $\sum_{n=1}^{\infty} q_i < 1$.
(8) (mean recurrence time) The mean recurrence time of a node is the expected value of the hitting time $M_i = \sum_{n=1}^{\infty} n * q_i^n$. If $M_i$ is finite then $i$ is positive recurrent.
(9) (absorbing) A state is absorbing if it is impossible to leave. This occurs when a node has no out edges. These states are also known as sinks.
(10) (source) A state is a source if it has no in edges.
(11) (ergodic) A state is ergodic if it is aperiodic and positive recurrent.

[8]defined below

6.1.3. *Absorbing Markov Chains.* A Markov chain is said to be absorbing if it contains at least one absorbing state and there is at least one finite path from each state to an absorbing state. We expect the steady state solutions to these chains to distribute the probability only among the absorbing states. These chains are usually analyzed in terms of a fundamental matrix $F = \sum_{n=1}^{\infty} P^n$ where $P$ is the transition matrix relating only the transient states. Using Neumann series we can realize this matrix as $F = (I - P)^{-1}$. Then, we can interpret the $i, j$ entry of $F$ as the expected number of times state $j$ is visited if the initial state is $i$. The variance can be computed from a matrix derived from $F$ with the Hadamard product.

It is easy to see that the interpretation of $F_{i,j}$ implies that the expected number of steps beginning at $i$ until the flow is absorbed is captured by $F\mathbf{1}$. Furthermore, the likelihood of reaching one transient state beginning at another is captured by $[(F - I)\,\mathrm{diag}(F)^{-1}]_{i,j}$, while the probability of being absorbed by a particular absorbing state is $[FR]_{i,j}$ where $R$ is the transition matrix connecting the transient states to the absorbing states.

6.1.4. *Uses in Complex Networks.* Considering Markov chains on networks we usually use the $0, 1$ structure of the adjacency matrix of the graph to determine where to assign probabilities under the assumption that there is no possibility of flow between disconnected nodes, with the caveat that probabilities may be added along the diagonal to capture the possibility of remaining in place. The most common Markov process on networks uses only the information in the adjacency matrix to define the transition probabilities. This is sometimes known as the normalized random walk Laplcian: $AD^{-1}$.

This is model of a random walk on the network where the probability of moving from $i$ to one of its neighbors $j$ is $\frac{1}{\deg(i)}$. This weighting assumes that each edge is taken uniformly at random. In this case the interpretation of the steady state is as a limiting distribution of probabilities. We expect more highly connected nodes to have higher entries in this steady state, so the magnitudes of the steady state vector are frequently used as a measure of centrality on the network. Other probabilities on the edges can be enforced to reflect extra structure known from the application or edge weights associated to the network. In order to study flows across networks extra nodes can be added. For example, adding a source node to set the initial distribution, or adding a sink node to transform the walk into an absorbing Markov process.

This interpretation also shows up in several other network contexts. For example, normalized spectral clustering can be computed in terms of the eigenvectors of this matrix. Additionally, many other types of Markovian dynamics are defined on networks for specific applications because in general they admit simpler solutions than more general models, even if they are not fully representative of the actual application.

6.2. **Diffusion.** Diffusion on networks is modeled as a discrete version of continuous diffusion processes, such as those governed by the heat equation. The idea is that each node is affected only by its neighbors and that the quantities associated to each node "diffuse" or move from areas of high concentration to lower concentration areas. We model this by assuming that for any initial distribution $\varphi$ on the nodes of the network the values change by a scalar multiple (the diffusion constant $c$) of the sum of the differences between the value at each node and the values at each of its neighbors. Symbolically, this is $\frac{d\varphi_i}{dt} = c\sum_j A_{ij}\varphi\varphi_i - \varphi_j$ or $\frac{d\varphi}{dt} = c(D - A)\varphi$ in matrix form.

This matrix $L = D - A$ is called the graph Laplacian. It is one of the most important operators associated to a given network and its eigenvalues have deep connections to the structure of the graph. Returning to the diffusion model we see that $\frac{d\varphi}{dt} = cL\varphi$ is a linear differential equation and since $L$ is symmetric it is diagonalizable with a simple solution. Since all of the eigenvalues of $L$ are non–negative the solution converges to a steady state in the limit that corresponds to equidistribution of the initial vector across the network as we would expect from the continuous case. Since the solution or steady state is so simple we are usually more interested in the rate at which the function converges for various inputs. This rate is controlled by the eigenvalues.

6.2.1. *Normalized Diffusion.* The Laplacian $L = D - A$ defined in the previous section is not entirely standard. There are several other matrices that go by the name "graph Laplacian", usually normalized versions of the operator above. For example if we want the change at a node to be equal to the average of its neighbors instead of the sum we can define $\hat{L} = D^{-\frac{1}{2}}LD^{-\frac{1}{2}} = I - D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$. This version is usually preferred mathematically for its relation to invariants of the graph. It also has a natural connection to the random walk Markov matrix associated to a network which shows up in the context of clustering.

The second eigenvalue of this normalized Laplacian provides the solution to the normalized cut problem. It is also (up to a change of coordinates) equivalent to a scalar perturbation the random walk Laplacian, so depending on context we can translate eigendata between the matrices if one of the matrices is more computationally tractable. A comparison of the bounds of the eigenvalues of this normalization and the standard Laplacian shows that their proportions remain unchanged, but the magnitudes are compressed in this realization.

6.2.2. *Properties of the Laplacian.* The graph Laplacian can be constructed as $N^T N$ where $N$ is an incidence matrix for the graph. This implies that the Laplacian is positive definite. It is also clear that $L$ is symmetric from its construction so $L$ has several nice algebraic properties such as orthogonal diagonalizability (from the spectral theorem). However, $L$ is never invertible since $L\mathbf{1} = \mathbf{0}$. If the underlying network has $k$ connected components, $L$ can be permuted into a matrix with $k$ block diagonal components representing these subnetworks. A similar decomposition is possible for the normalized Laplacian. A famous theorem of Kirchoff shows that the determinant of any $n-1$ minor of $L$ is the number of spanning trees in the network.

6.2.3. *Eigenalues of the Laplacian.* The most interesting aspect of the graph Laplacian is that the eigenvalues of the operator have deep connections to many of the standard graph invariants. We mention just a few of the most common properties here, but the book "Spectral Graph Theory" by Fan Chung collects many more applications, although most of these properties tend to be of greater mathematical than practical interest. A gentler introduction to these topics is provided in Brualdi's book.

(1) Since $L$ is positive definite all eigenvalues are non–negative.
(2) The multiplicity of zero as an eigenvalue of the Laplacian is the number of connected components in the network.
(3) The next smallest eigenvalue is known as the Fiedler value or algebraic connectivity of the network.
(4) The edge connectivity of the graph is always greater than or equal to the Fiedler value. Additionally, the number of edges between a set of nodes $U$ and its complement is at least the product of the Fiedler value and $\frac{|U||V\setminus U|}{n}$.
(5) The Fiedler value is bounded below by the inverse of the number of edges in the graph and the diameter of the graph.
(6) The Fiedler value is also a measure of how fast the diffusion process across the network occurs.
(7) The synchronization of the network is defined to be the quotient of the largest eigenvalue of the Laplacian and the Fiedler value. This is also a measure of the rate of diffusion across the network.

**How can you use the graph Laplacian to determine the number of connected components of the network? Can you identify the groups of nodes in each component?**

The graph Laplacian is defined as $D-A$. It is clear that zero is an eigenvalue of this matrix since all of the row sums are zero. Thus, $L\mathbf{1} = \mathbf{0}$. The number of connected components is the number of zero eigenvalues of $L$. This can be seen from the fact that the restriction of $\mathbf{1}$ to the subspace spanned by the nodes in each connected component is annihilated by $L$ and that this collection of vectors is linearly independent. To see that there are no more vectors with the property we can proceed by induction using the fact that if a component is connected then the eigenvalue zero has multiplicity one. This fact can be seen by observing that if $v$ is a corresponding eigenvector, then $0 = v^t L v = \sum_{i\sim j}(v_i - v_j)^2$ so $v_i = v_j$ since the graph is connected.

The components can be discovered from an arbitrary basis of the null space of $L$ by identifying the components in each vector that have the same values, since we know that the vectors must be expressible as linear combinations of the characteristic vectors on the connected components. The underlying idea is the the matrices can be rearranged to be block diagonal representations of the connected components[9].

**Derive the graph Laplacian and the solution for the corresponding diffusion problem.**

The standard graph Laplacian can be realized in several separate ways that highlight different aspects of its usefulness. The various normalized versions of the Laplacian add even further complexity to this operator. Algebraically, it can be constructed as $BB^T$ where $B$ is the incidence matrix of the network, defined as a $n \times m$ matrix with each column representing an edge and signs ($\pm 1$) assigned to the columns arbitrarily. This construction gives the Laplacian many of its algebraic properties, such as symmetry and positive semi–definiteness.

[9] at least for $A$ and $L$.

On the other hand, the Laplacian arises quite naturally in the context of studying diffusion on graphs, as well as in the clustering case discussed in Problem 2, where the Laplacian arose as the constraints matrix for the relaxed cut problems. For diffusion, we consider a vector of values representing quantities on the nodes that spread along edges in the network proportionally to the difference between the values at the incident edges. This is a discretization of the continuous heat flow model that is solved by the standard Laplacian[10]. Letting $\varphi$ represent our vector function of interest, this gives the following system of differential equations:

$$\frac{d\varphi_i}{dt} = -c\sum_j A_{i,j}(\varphi_i - \varphi_j)$$

Where $c$ is the proportionality constant and the minus sign is for historical (in)convenience.

Distributing this expression, we obtain

$$\frac{d\varphi_i}{dt} = -c\varphi_i \deg(\mathrm{i}) + c\sum_j A_{i,j}\varphi_j = -c\sum_j(\delta_i(j)\deg(i) - A_{i,j})\varphi_j$$

Rewriting this expression in matrix form for the entire vector at once gives

$$\frac{d\varphi}{dt} = -k(D - A)\varphi = -cL\varphi,$$

which is exactly the form that we wanted. To solve this linear differential equation, we note that since $L$ is symmetric it is orthogonally diagonalizable so we can write $\varphi(0) = \sum_{k=1}^n c_k v_k$, where the $v_k$ are eigenvectors of $L$. Then, substituting into our matrix expression we have:

$$\begin{aligned} 0 = &\quad \tfrac{d\sum_{k=1}^n c_k v_k}{dt} + cL\sum_{k=1}^n c_k v_k \\ = &\quad \sum_{k=1}^n \tfrac{dc_k v_k}{dt} + cc_k\lambda_k v_k \end{aligned}$$

Since the $v_k$ are linearly independent, this implies that we have $\frac{dc_i}{dt} + c\lambda_i c_i = 0$ for all $i$. The solution to each of these linear equations is $c_i(t) = c_i(0)e^{-c\lambda_i t}$. We proved above that $L$ is positive semi–definite, so the $\lambda_i$ are all non–negative and hence the final solution for $\varphi$ converges to a steady vector in the limit, determined by coefficients of the components of the kernel of $L$. Furthermore, on each component of the graph the values of $\varphi$ converge to the average of the original values on that component since that is the limit of the projection onto the kernel, which is $\mathbf{1}$ times the projection onto each component.

A more general case can be considered if we allow forcing terms to act on our nodes. In this case we are allowing for the existence of constant sources or sinks across the network. This modifies the equations derived above by transforming the linear system to an affine one. We can still make use of the orthogonality of the eigenvectors of $L$ to obtain componentwise relations $c_i(t) = c_i(0)e^{-c\lambda_i t} + \frac{\gamma_i}{\lambda_i t}$ where the $\gamma_i$ is the $i^{th}$ coordinate of the forcing vector in the eigenvector coordinates.

6.3. **Epidemic Models.** One of the main uses of network dynamics is modeling the flow of some quantity (information or disease) across a network. These models have much in common with both the Markov and diffusion models. The simplest model is the SI model which is simply a reinterpretation of the standard diffusion model with an initial distribution consisting of point masses. Generally in this model the outcomes at each node are assumed to be binary or probabilistic. The most commonly used model is the SIR model (susceptible, infected, recovered) in which each node in the network is labelled with one of the properties at each stage and a Markov process probabilistically updates the labels. The main concerns of this model are the basic reproduction number and the threshold number, which represent the spreading rate of the disease and the rate at which the disease is expected to die out respectively. Many tweaks of this model are possible, each leading to a different (longer) acronym.

Another type of model is a contagion (or information) flow model. This is closely related to the diffusive model discussed above. Percolation techniques are also used to study the flow of diseases across networks. In general, for epidemic dynamics, the main concerns are understanding the equilibrium behavior of the model in terms of the proportion of infected individuals as well as the cyclic behaviors associated to the disease. The basic reproduction number is defined to be the expected number of new cases generated by a single infected individual in an entirely healthy population.

[10]hence the name

## 7. Clustering and Partitions

One of the fundamental problems in network analysis (and more broadly all of data analysis) is revealing meaningful substructures from large networks. This approach is complementary to dimension reduction since our goal is to associate individual nodes that are similar in some fashion into larger clusters that represent more homogeneous components of our data. Thus, we are attempting to minimize the number of relevant data points instead of minimizing the number of dimensions associated to each point. Some techniques such as $k-$means and linkage apply to data analysis more broadly, while spectral methods and modularity rely on the algebraic and graph theoretic structure of an underlying network.

However, there are ways to move in between these interpretations. For example, geodesic difference provides a metric on the graph. Conversely, if we use a dimension reduction technique to map arbitrary data into $\mathbb{R}^n$ we can use radial connections or nearest neighbor connections to form graphs from the embedding.

### 7.1. **Thresholding.**

Thresholding is a basic technique from data analysis for grouping data given with some application derived similarity metric. If no such metric is given some of the embedding techniques from dimension reduction can be employed to endow the data with a Euclidean structure. To perform clustering by thresholding an initial threshold $t$ is selected. Then, two points $i$ and $j$ are then put into the same cluster if $S_{i,j} > t$. This is a simple operation to perform, but can be very sensitive to the choice of $t$.

In general, proportionally high thresholds lead to more meaningful groupings while low thresholds tend to identify many more relationships but without obvious interpretability. Thus, determining an application appropriate threshold value (or collection of threshold values is essential to obtaining interesting information using thresholding. One option for selecting an appropriate threshold is to compute a histogram or proportion plot of threshold values and try thresholds at gaps or steady states respectively. Alternatively, since these groupings are efficient to compute a range of threshold values can be tested and interpreted individually.

### 7.2. **Linkage.**

Linkage is a hierarchical clustering method similar to thresholding. It is an iterative approach where at each step we combine the clusters that are the most similar. These iterations define a hierarchical structure, where clusters formed at earlier steps are more homogeneous than the connections made at later steps. There are several types of linkage grouping, but the basic outline of the iterative process is as follows, with the initialization placing each node into a separate cluster:

(1) Determine which pair of clusters has the highest similarity value
(2) Merge these identified clusters and reindex the cluster list
(3) Update the similarity values to incorporate the new conjoined cluster
(4) Repeat until there is only a single cluster

Obviously there are several details that must be specified before implementing this algorithm. For example, what if multiple pairs of clusters in step (1) have the same similarity score (usually one pair is chosen randomly, but this can lead to slightly perturbed results). Additionally, in step (3) there are several different methods that are used to compute the new similarity values:

- (single linkage) In single linkage the new similarity value between cluster $a$ and cluster $b$ is the maximum over all pairwise similarity scores between nodes in $a$ and nodes in $b$.
- (complete linkage) In complete linkage the new similarity value between cluster $a$ and cluster $b$ is the minimum over all pairwise similarity scores between nodes in $a$ and nodes in $b$.
- (average linkage) In average linkage the new similarity score is computed as the average over all pairs of nodes between the clusters.

Clearly, these different methods can have significant impacts on the final clustering results. Again, this is an efficient technique from a computational perspective and comparing the results from different versions of the method can yield interesting insights. Particularly, the distinction between results from single and complete linkage since depending on the underlying similarity distribution they can return quite different results.

### 7.3. $k$**-means.**

The $k$–means method is another iterative clustering approach from data analysis that relies on embedding the nodes in $\mathbb{R}^n$. In this approach we are attempting to determine a set of canonical representatives such that each data point can be associated to a representative while minimizing the total distance between points and representatives. To begin this process, the number of clusters, $k$, to be obtained must be specified in advance. Usually this is guided by some intrinsic knowledge about the data, but also may be experimented with numerically. The method then initializes with a random selection of $k$ points in $\mathbb{R}^n$.

Each node is then assigned to the cluster corresponding to the initial point that it is closest to. Each step of the algorithm then replaces the previous representative points with the centroid of each cluster from the previous step. The nodes are reassigned clusters based on distance to the newly computed centroids, and this updating continues until the clusters converge. At that point, the centroids and clusters are no longer changing, so the centroids are taken as representative elements, and the clusters are defined by this steady state.

Although for a given initial set of points this process is deterministic, the initial selection of points can lead to very differing results. Thus, in practice the algorithm is run many times on different selections of initial points, with the final results being given as the output with the smallest total distance between the points and centroids in the steady state. It is often valuable to examine the histogram of total distance values as each run of the algorithm finds a local minimum. Thus, modal data in the distance values may represent different types of clustering behavior. As the $k$ value must be set in advance, this is important because it may suggest a may appropriate value or highlight other features of the data.

7.4. **Spectral Clustering.** Spectral clustering is an algebraic method that applies directly to networks. The goal of spectral clustering is to find a partition of the nodes into $k$ sets such that the number of edges between the sets is minimized. This can be thought of as minimizing the amount of damage to the entire network if the clusters were disconnected from each other. However, solving this problem directly is $NP$–hard so the problem for large networks is traditionally relaxed to allow for an elegant solution in terms of eigenvalues.

We begin by describing the case where we wish to partition the network into two components. In this case we wish to find a vector $v$ with entries of $\pm 1$ representing the two clusters that minimizes the disconnection damage. Some simple algebraic manipulations after reformulating this expression in terms of the adjacency matrix, show that we are interesting in minimizing $v^t LV$, where $L$ is the graph Laplacian introduced previously. Unfortunately, this problem is still $NP$–hard.

However, if we relax the restriction that the entries of $v$ be $\pm 1$ and instead require that $|v| = 1$ and $v \perp \mathbf{1}$ then Lagrange multipliers show that the solution is given by the eigenvector corresponding to the smallest non–zero eigenvalue of $L$. Then, we can use either the signs or median of the entries of this eigenvector to partition our network. Sometimes, eigenvectors of other versions of the Laplacian are used instead, including the normalized Laplacian and the random walk matrix. The results obtained from these matrices are usually equivalent to those obtained from $L$. To partition the nodes into $k$ clusters, we usually use $k$ eigenvectors corresponding to the smallest $k$ non–zero eigenvalues and perform $k$–means on the associated node values.

**Derive the formula for the three spectral clustering methods.**

The ideas behind the three basic spectral clustering methods are that we wish to partition the network into two pieces such that if we disconnect the two components we do a minimal amount of "damage" to the network as a whole. The definition of the damage function is what distinguishes the various methods. The most naïve method uses the Cut formulation. In this case, we wish to minimize the number of edges that must be removed to disconnect the two components. Unfortunately, solving this optimization problem for the components is $NP$–hard, although it can be computed exhaustively for small networks. Instead, we approach a relaxed version of the problem. We begin by constructing a vector $v$ to represent the desired partition, with entries of 1 assigned to nodes in one component and entries of $-1$ assigned to the other component. This gives that $v_i v_j = 1$ if and only if $i$ and $j$ are in the same component. We can now formulate the damage condition algebraically as

$$\text{damage}(v) = \frac{1}{2} \sum_{i,j} \frac{1}{2}(1 - v_i v_j) A_{i,j}.$$

Proceeding by algebraic manipulation, we can reduce this expression to

$$\begin{aligned} \frac{1}{2} \sum_{i,j} \frac{1}{2}(1 - v_i v_j) A_{i,j} = & \quad \tfrac{1}{4}(\textstyle\sum_{i,j} A_{i,j} - v_i v_j A_{i,j}) \\ = & \quad \tfrac{1}{4} \textstyle\sum_{i,j} v_i \deg(i) v_j \delta_{i,j} - v_i A_{i,j} v_j) \\ = & \quad \tfrac{1}{4} v^T D v - v^T A v \\ = & \quad \tfrac{1}{4} v^T L v \end{aligned}$$

Now we have reduced our problem to minimizing this bilinear form over all vectors $v \in \{\pm 1\}^n$. Unfortunately, this is still an $NP$–hard problem so we relax our constraints to minimize over all real vectors of norm one, where the norm condition rules out trivial minimization solutions. We further require that $v \perp \mathbf{1}$ since $\mathbf{1}$ is in the kernel of $L$ and carries no information for our clustering. The theory of Lagrange multipliers then gives that the solution vector to our minimization problem is the eigenvector corresponding to the Fiedler value of $L$. We can assign nodes to components by separating the positive values and negative values.

The problem with the direct cut formulation is that it says nothing about the relative sizes of the partition. In order to rule out trivialities, such as disconnecting a pendant edge, two other types of damage metrics are frequently used in the literature. These are the RatioCut which scales the damage by the number of vertices in the subsets and the NormalizedCut which scales the damage by the number of edges in the subsets. Symbolically, these are

$$\text{RatioCut}(A,B) = \frac{1}{2}\left(\frac{E(A,B)}{|A|} + \frac{E(A,B)}{|B|}\right)$$

and

$$\text{NormalizedCut}(A,B) = \frac{1}{2}\left(\frac{E(A,B)}{\text{vol}(A)} + \frac{E(A,B)}{\text{vol}(B)}\right).$$

The RatioCut derivation leads to a very similar optimization structure as the (relaxed) Cut problem. In this case we assume that there is again a vector $v$ representing the partition, but this time the entries in $v$ are proportional to the size of the component that the vertex is selected from: $v = \begin{cases} \sqrt{\frac{|B|}{|A|}} & i \in A \\ -\sqrt{\frac{|A|}{|B|}} & i \in B \end{cases}$. We can simply compute that $v^T L v$ again gives exactly the cut value and that $||v||^2 = v^t v$ is exactly the denominator that appears in the definition of the RatioCut. Thus, we are minimizing the expression $\frac{v^T L v}{v^T v}$ which is the Rayleigh quotient associated to $L$. Since $v\mathbf{1} = \mathbf{0}$ by construction this expression is minimized by the second eigenvalue of $L$ as in the previous case. Thus, we again take the second eigenvector of $L$ to form our partition.

If we proceed as in the standard case for the NormalizedCut, we can again imagine a partition vector $v$, with $v_i = \frac{1}{\text{vol}(A)}$ if $i \in A$ and $v_i = \frac{-1}{\text{vol}(B)}$ if $i \in B$. Then, we see that $v^T L v = E(A,B)(\frac{1}{\text{vol}(A)} + \frac{1}{\text{vol}(B)})$, while $v^T D v = \frac{1}{\text{vol}(A)} + \frac{1}{\text{vol}(B)}$. Thus, the normalized cut corresponding to $v$ is the ratio of these two values. That is we wish to minimize $\frac{v^T L v}{v^T D v}$. Using the substitution $D^{-\frac{1}{2}} w = v$ this becomes $\frac{w^T D^{-\frac{1}{2}} L D^{-\frac{1}{2}} W}{w^T w}$ which is the Rayleigh quotient for the normalized Laplacian $I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$. Minimizing this expression is equivalent to finding the second eigenvector of this normalized Laplacian and we may partition the network with this vector as above.

Interestingly, the stochastic matrix $AD^{-1}$ can be used to solve the NormalizedCut problem. In this case if $\lambda, v$ are an eigenpair of $AD^{-1}$ then $(1-\lambda), v$ are an eigenpair for the normalized Laplacian. This relationship is more clear when we note that $AD^{-1} = D^{-\frac{1}{2}}(I - D^{-\frac{1}{2}} L D^{-\frac{1}{2}}) D^{\frac{1}{2}}$. The NormalizedCut value of a partition $Q$ can be realized in the random walk setting as $\text{NormalizedCut}(Q, \overline{Q}) = P(Q|\overline{Q}) + P(\overline{Q}|Q)$ which provides some intuition for the relationship. Thus, finding the second largest eigenvalue of $AD^{-1}$ and its corresponding eigenvector also gives a solution of the NormalizedCut problem on a network.

7.5. **Modularity.** Modularity is a measure of the proportion of connectivity with clusters to connectivity between clusters. Given a partition of the network into clusters, the modularity is defined as the difference between the number of edges that occur within each cluster and the number of edges that would be expected if the edges were distributed randomly between the nodes while keeping the same degree distribution. Other variants of the random edge distribution are possible, and can be adjusted based on the application. In general, the probability that two arbitrary nodes in the network are connected in a random network preserving degree distribution is approximately $\frac{\deg(i)\cdot\deg(j)}{2m}$ for large networks, where $m$ is the total number of edges in the network.

This is an example of a null model analysis with the configuration model providing the null model. Although there are many similarities between modularity and spectral clustering, the focus on density is a distinguishing feature because spectral clustering does not account (except incidentally) for the interconnectivity within the clusters it discovers.

This can be formulated and computed linear algebraically, by forming the modularity matrix $B$ from the adjacency matrix $A$ as $b_{i,j} = a_{i,j} - \frac{\deg(i)\cdot\deg(j)}{2m}$. Then, we form a matrix $C$ representing the clusters whose entries are $c_{i,j} = \begin{cases} 1 & \text{if node } i \text{ is in cluster } j \\ 0 & \text{otherwise} \end{cases}$. This allows us to compute the modularity value of the partition as $\frac{1}{2m}\,\text{trace}(C^t BC)$. Recasting this problem as an optimization problem as in spectral clustering allows us to attempt to determine the appropriate clusters for the network. In this case we are maximizing $s^t Bs$ and so the eigenvector corresponding to the leading eigenvalue is usually used to construct the initial clusters. Several algorithms such as simulated annealing, have been applied to give solutions to this optimization problem. This can also be recast as a spectral problem in terms of the modularity matrix $B$. A complete derivation of modularity for scalar characteristics can be found in the section describing assortativity above.

**How does modularity use a null model to determine communities in the network?**

The modularity of a network is a function that maps a partition of the network to a value measuring the proportion of connectivity that occurs within the clusters compared to the edges between clusters. Given a partition of the network into clusters, the modularity is computed as the difference between the number of edges that lie within the observed clusters to the number of edges that would occur if the edges had been distributed uniformly with the same degree distribution. This is usually normalized by the maximal value obtainable from the configuration model.

This is an example of a null model because the you are comparing the observed network to a theoretical model, of a network with same degree distribution and randomly placed edges, in order to determine the significance of the observed data. This particular model is formed by splitting each edge in half and reattaching the edge ends at random. Subtracting off the amount of clustering that appears in the null model leaves behind the extra (or deficient) amount of clustering that appears in the observed network.

The definition of modularity is also frequently extended to cover scalar partitions where each cluster is assigned a relative value. In this case the modularity can be interpreted as a covariance of the partition labels. Specifically, the notion of assortativity is a scalar modularity with the partitions defined by the degrees of the nodes.

Department of Mathematics, Dartmouth College
*E-mail address*: ddeford@math.dartmouth.edu