Dynamics of Networks

For Euler

November 18, 2020

Terminology

- **Nodes/vertices**: elements of the node set V, where $V = \{A, B, C, D...\}$. Note, $A, B, ...$ are labels, not variables.
- **Edges/links**: the connection between two nodes. Represented by an unordered pair in the edge set $E = \{(A, B), (C, D),$
- **Network/Graph**: The combination of a vertex set V and an edge set E (which contains elements of V).
- **Empty graph**: Edge set $E = \emptyset = \{\}$ (E is empty), i.e. the network has no connections.
- **Null Graph**: $V = E = \emptyset$. The network has no nodes or connections.
- **Fully connected Network**: Every possible link between nodes exists.
- **Component**: A disconnected part of a network. A component is a set x of nodes and links that meets the following conditions: 1. All links that are incident to nodes in x are also in x. 2. For each link in x the endpoints of the links are also in x. 3. For every pair of nodes $a, b \in x$ there is a walk starting in a and ending in b.
- **Forest**: A network that has no cycles. Every component is a tree.
- **Trees**: Components of Forests. In a tree the number of links K is one less than the number of nodes N (otherwise there would be cycle).
- **Spanning Tree**: A tree that contains every node in the network. Or a Forest with only one component.
- **Distance Matrix/Weight Matrix**: A matrix of the weights/distances of the links between nodes. We use the term 'weights' if we are dealing with numbers associated to the links.
- **Labeled Graph**: A network where the nodes are labeled.
- **Incident on**: Edge (a, b) is incident on vertices a and b. Just mathematical term for 'connects'.
- **Endpoints**: Nodes a, b are endpoints of the edge (a, b).
- **Neighbours/Adjacent**: Vertices a and b are neighbours if there exists an edge (a, b).
- **Neighbourhood** of a, is the set of all nodes that are adjacent to a.
- **Walk**: An alternating sequence of vertices and edges that starts and ends with a vertex (where every edge is incident on the nodes before and after it in the sequence), e.g. $a, (a, b), b$ is a walk. Sometimes only the nodes, or links are listed.
- **Trail**: A walk in which every edge/link appears at most once.
- **Path**: A walk in which every node/vertex appears at most one
- **Closed**: The first node is also the last in the walk. Otherwise it is **open**.
- **Circuit**: A closed trail.
- **Cycle**: A closed walk in which every vertex occurs only once, except the starting vertex which occurs exactly twice.
- **Length** (of a walk): The sum of the weights that appear on a walk.
- **Connected Network**: The network contains at most one component. Otherwise it is considered **disconnected**.
- **Arc**: A unidirectional link.
- **Directed Graph/Digraph**: A network that contains at least one arc.
- **Directed walks** adhere to the directions described by arcs, **undirected walks** do not.
- **Strongly connected component**: A component that has arcs and thus directed walks. **Weakly connected component** if it doesn't have arc and thus undirected walks.
- **Simple Graph**: A network that does not contain loops or multilinks.
- **Loops**: Links that connect a node back to itself.
- **Multilinks/Parallel Edges**: Multiple links that connect the same two nodes.

- **Multi-graph**: A network that contains multilinks.
- **Pseudograph**: A network that contains loops (can also contain multilinks).
- **Adjacency Matrix**: A matrix A that contains the *number* of connections between nodes. It specifies the **topology** of the network, i.e. what is connected to what.
- **Node Degree**: The number of undirected links coming out of a node: $k_i = \sum_j A_{ij} = \sum_j A_{ji}$.
- **In-degree**: The sum of incoming links, $k_i^{in} = \sum_j A_{ij}$. **Out-degree**: the number of outgoing links, $k_i^{out} = \sum_j A_{ji}$.
- **Eulerian Walk**: A walk in which every link of the network is used exactly once. These can only exist if (and only if) the network contains at most two nodes of odd degree. If the number of odd nodes is 2, then the walk is a trail, and starts at one odd node and ends at the other. If there are no odd nodes, the Eulerian walk is Eulerian circuit.
- **Hamiltonian Path**: a path that visits every node in a network exactly once.
- **Hamiltonian cycle**: a path that visits every node exactly once, except that it return to the starting point.
- **Distance**: The length of the shortest between two nodes. If no path exists between a pair of nodes, their distance is considered ∞. Average distance is $d(i) = \sum_j d(i,j)/N$.
- **Diameter**: The longest distance between any pair of nodes in a network: $D = \max d(i,j)$. The diameter is the length of the longest shortest path in the network, if that makes sense. There are other definitions of diameter like taking the average distance between nodes...
- **Ensemble**: A collection of all the different possible permutations of links of a network.
- **Degree distribution**: The probability distribution p_k of node degrees, i.e. if we pick a random node in the network, the probability that it has degree k is p_k.
- **Mean degree**: The expectation z of the degree distribution, i.e. the mean degree of all nodes. For ER $z = 2K/N = \sum_k kp_k$
- **Excess degree distribution**: The probability q_k of finding a node that has k additional links if we follow a random link (in a random direction). $q_k = (k+1)p_{k+1}/z$.
- **Mean excess degree**: The expectation/mean q of the degree distribution. $q = \sum_k kq_k$. Find it by choosing a link and going one way to compute excess degree then other way and divide by 2.
- **Heterogeneous Network**: The degree distribution is wider than the Poisson distribution (from the random network) of the same mean, which implies that $q > z$.
- **Homogeneous Network**: The degree distribution is narrower than the Poisson distribution, and thus $q < z$.
- **Regular Graph**: The most homogeneous network, all nodes have the same degree, and thus $q = z - 1$. Thus the degree distribution is $p_k = \delta_{k,z}$ (Kronecker delta).
- **Degree Sequence**: A sequence containing the node degrees of a networ (usually ordered in a non-increasing manner).
- **Graphical Degree Sequence**: There exists at least one simple graph which has the sequence as its degree sequence. Basically, a degree sequence that can generate a graph.
- **Subgraph**: Given a graph $G = (V,E)$, $G' = (V',E')$ is a subgraph if $V' \subset V$ and $E' \subset E$.
- **Motif**: Given a graph $G = (V,E)$, $G' = (V',E')$ is a motif if $V' \subset V$.
- **Identical/Isomorphic Networks**: Two networks are considered identical/isomorphic if their nodes can be labelled such that their adjacency/weight matrices are identical.
- **Triangle**: Fully connected simple graph with three nodes.
- **Cumulative degree distribution** c_k describes the probability that a randomly selected node as *at least* k links.
- **Dynamical system**: Used to compute the degree distributions of an evolving network. The system consists of a set of **variables** X, which changes according to an **evolution rule** f, which potentially depends on a set of **parameters** P.
- **Stationarity**: A state in which the variables do not change anymore (can occur for **long-term dynamics**). If the system exhibits stationary dynamics we say that it is in a stationary state (or **steady-state**).
- **Adaptive Networks**: Networks in which there is an interplay between the dynamics on the network and a topological evolution of the network.
- **Bipartite Networks**: Networks in which the nodes can be colored in two different colours such that node has a neighbour of the same colour - a checkerboard like effect.
- **Cluster and Modularity**: A cluster is a relatively tightly connected group of nodes - they are tightly connected internally and weakly connected externally. Modularity is a quantity of the goodness of partition of a network into different clusters.

Useful Combinatorix Rules

- If we have a number of different decisions to make, and the number of options for each decision is fixed, then the total number of options is the product of the number of options for the individual decisions. E.g. in a network with 3 nodes, we have $2 \cdot 2 \cdot 2 = 2^3 = 8$ different options for possible link combinations that will connect the nodes.
  - The maximum number of links in a fully-connected network with N nodes is $N(N-1)/2 = N + (N-1) + (N-2) + ... + 1$. So the max number of possible combinations is $2^{N(N-1)/2}$

Useful Mathematics

- $\sum \frac{n^k}{k!} = e^n$

1 L1: Kruskal's Greedy Algorithm

$ Good for minimum spanning tree problems (MST problems). Not for optimal paths from A to B, more like choosing electricity gird.

% What this algorithm is trying to solve: For a given set of nodes and a given distance matrix we want to find the edge set such that there is only **one component** and the **sum of the weight of the edges is minimal**.

- We can optimise a network to satisfy these two conditions (bold font in the line above) by using the conditions themselves.
  1. Start with a set G that represents the links in an empty graph (i.e. no links for now) $\Rightarrow$ means we have a large number of components, but minimal weight.
  2. Create a set A with all links available in that network (or take from weight matrix).
  3. Remove the cheapest link from the set of candidate links A. If adding this link to G reduces the number of components then add it to G, otherwise discard it.
  4. Repeat 2 and 3 until there is only 1 component.
- So we start with max components and min total weight and then increase the total weight until we only have 1 component.
- This a **Greedy Algorithm**: A greedy solution works as follows: 1. Find a way to break the construction of the solution into individual steps. 2. Go through the steps and try to make an optimal choice in every step. Greedy algorithms are always fast, but they do not always result in the optimal solution.

2 L2 Dijkstra's Algorithm

$ Good for find optimal paths between two nodes. Used for Google maps, etc... Not good for finding minimium spanning tree.

% The algorithm tries to solve problems of the form: Given a weighted network, find the shortest path from a given starting node to a given end node.

- Go over the Algorithm when it comes to actual revision.
- This is a branch and bound solution strategy.
- Check out $https://graphonline.ru/en/$

3 L3 Hierholzer's Algorithm, Chinese Postmen

$ Good for solving Eulerian walk problems, i.e. seeing if its possible to reach all nodes in a network.

- Eulerian walk conditions:
  1. Nodes of degree one must be a starting or ending node, so an Eulerian walk cannot exist if there are more than two nodes with degree 1.

2. A Eulerian walk started with a degree two node (or any even degree number) must end at that same node, and the walk is circuit.
3. If there are more than two nodes of odd degree a Eulerian walk cannot exist.

- To create a Eulerian walk we can add links between pairs of odd degree nodes, until there are only two odd degree nodes. To create a Eulerian circuit we add links until all nodes have even degree.
- **Chinese Postman Problem**: Find the shortest walk that uses every link *at least* once. Solve by adding links to shortest distances between odd degrees.
- **Finding the Eulerian Walk**:
  - Hierholzer's Algorithm:
    * In my own words: If you know a route is possible, set off from an odd degree node, and if you get stuck, choose a node on the path you just traced that still has links coming out of it and create a new route. Do this until the network is covered by routes and then combine the separate routes into one.

4 L4 Travelling Salesman, Hamiltonian Walks

$ Good for travelling salesman (TSP) and Hamiltonian path problems, i.e. finding the best route that covers a set of nodes (not like Dijkstra which just finds the quickest route between two nodes irrespective of the nodes needed on the way).

% There are many ways to solve these types of problems, but there no one optimal solution (except brute force, which is not time efficient).

- **Rahman-Kaykobad Theorem**: A simple graph with n vertices has a Hamiltonian path if, for every non-adjacent vertex pairs the sum of their degrees and their shortest path length is greater than n. [not sure what they mean with shortest path length here exactly].
- Trial and error method (non-optimal) and brute force method (optimal, but doesn't scale well), could also use Dijkstra's (branch and bound method like this effective if we have heuristics, but not necessarily optimal), or try greedy MST solution and adjust to create paths.
- **Christophides Algorithm**: Construct MST, connect nodes of odd degree to create eulerian circuit. Follow Eulerian circuit but if a node appears for a second time, skip it, i.e. don't go there.
- **Local Search Algorithm**: Divide network into local clusters, use Christophides Algorithm locally to find hamiltonian path and then finally merge paths together.

5 L5 Degrees of Separation

$ Good for identifying important nodes, strategies for combating epidemics, policy decisions..

- **Erdos-Reyni random graphs**: Ensemble graph $G(N, K/p)$ is created by generating N nodes, and K links or links with probability p for every pair of nodes (i, j).
  - Mean degree: $z = \frac{2K}{N} = (N-1)p \approx Np$ [not sure about the N-1]
  - Degree distribution: Binomial but can be approximated with Poisson: $p_k = \frac{z^k e^{-z}}{k!}$
  - Excess degree distribution: $q_k = p_k$ [simple proof].
  - Mean excess degree $q = z$.
- Diameter of Networks:
  - Assuming locally the network is a tree then the number of nodes at a given distance from a source n_d is $n_d = zq^{d-1} \approx q^d$, where d is the diameter.
  - Assuming diameter D is the distance at which we reach every node, then $D = \frac{\log N}{\log q} = \frac{\log N}{\log z}$. Note that if $q \to 0$ then $D \to \infty$.
  - This estimate of a networks diameter generally works well. In the real world cycles exist which reduces the excess degree but this is cancelled by there actually being larger excess degrees.
  - $\frac{dD}{dq} = -q\frac{\log N}{(\log N)^2} \to$ an error caused in excess degree q causes only a tiny error in diameter D.

- **Closeness centrality**: Finding the most important nodes.
  - Use Dijkstra's to compute shortest distances $d(i,j)$ for ever pair of nodes.
  - For each node i compute $C_i = \left(\sum_j \frac{d(i,j)}{N}\right)^{-1}$.
  - The node with the highest C_i is the most central.

6 L6 Giant Components, Percolation, Fragmentation

$ Good for assessing risk of epidemic spread.

- **Percolation Argument**: If something produces in average less than one offspring per generation, it will exponentially decline. If something produces more than one produces in average more than 1 offspring per generation it will exponentially grow. Anything that does not grow out of bounds or disappears must eventually produce one offspring per generation.
- Using the percolation argument we define:
  - If mean excess degree $q < 1$ then we are in a small component
  - If $q = 1$ we are giant component transition
  - If $q > 1$ we are in a giant component.
- **A component size** is given by $S = 1 + \frac{1}{1-q}$ [Two derivations for this in lecture].
- **Finding the size of the giant component**:
  - Let the probability that a node is drawn from the giant component be s, and its negation be $u = 1 - s$. Also, a neighbour of a node not in the giant component is also not in the giant component, hence $u = u^z$. A solution to this is $z = 1$ ($u = 0$ and $u = 1$ also work, but aren't useful), and for ER graphs $z = q$ which means its right at the giant component transition.
  - If we use the degree distribution we get $s = 1 - e^{-zs}$ and we can use this to estimate the proportion of nodes in the giant component [learn derivation closer to exam time - although questions will be multiple choice...]

7 L7 Attacking Networks

$ Good for stopping epidemics, computer viruses etc...

- Types of attacks:
  - Precisely targeted - not covered in lecture, requires topological knowledge
  - Targeted - attack nodes with certain properties (requires information about nodes, not entire topology)
  - Viral - Attack propagates over entire network (no detailed info needed)
  - Random - Attack randomly selected nodes (no detailed info needed)
- **Configuration model**: A way of creating an ensemble of networks with a given number of nodes N and degree distribution p_k.
  - Generate degree sequence from degree distribution, then generate nodes with stubs corresponding to the degree sequence and finally connect the stubs randomly.
- **Havel-Hakimi Theorem**: A non-increasing degree sequence is graphical if subsequent degree sequences, whereby the largest degree in the sequence is used up on subsequent degrees in the sequence, is graphical.
- **Random Attack**: If a percent of the nodes remain, the probability that a link remained is a^2.
  - After the attack ('): Mean degree and mean excess degree become $q' = z' = \frac{2K'}{N'} = \frac{2a^2K}{aN} = \frac{2aK}{N} = az = aq$
  - To destroy the giant component in an ER graph, we need to destroy $b = 1 - a = 1 - 1/z = 1 - 1/q$ percent of the nodes.
  - To destroy a giant component in a regular graph we need $b = 1 - 1/q = 1 - 1/(z-1)$ percent of the nodes. Likewise for ER graphs $z' = az$ and $q' = aq$ (except they are different now). Regular graphs are more fragile to begin with but then get better after attacks because the network becomes less homogeneous.
  - Generally $\rightarrow$ estimate surviving proportion of nodes $\rightarrow$ compute degree distribution of surviving nodes $\rightarrow$ compute

excess degree distribution $\rightarrow$ compute mean and excess mean degree after attack. Check existence of giant component, size of giant component and diameter of network after attack.
  - Heterogeneous networks ($q > z$) are very robust initially. The random attacks reduce the heterogeneity, but the network generally remains robust.
  - To avoid computing the degree distributions, just use $z' = az$ and $q' = aq$ where a is the proportion of nodes that survive after the attack.

- Targeted attacks: For a summed Poisson distributed degree distribution remove terms z_T (T being the targeted node degree). Then, the total endpoints before the attack is $e = Nz$, and after the attack its $e' = N_{destroyed}z_T$. The surviving network then has a mean degree of $z' = q' = (1 - e/e') * z$. Shows that heterogeneous networks are not necessarily robust to targeted attacks.
- Viral attacks:
  - Basic reproductive number: $R_0 \approx q$.
  - Vaccinate $b = 1 - 1/q$ of the network randomly until $q < 1$.
  - Other solutions - Vaccinate high degree nodes, vaccinate contacts of infected person, anyone who receives the vaccination can nominate three other people to receive the vaccination.

8 L8 Generating Functions

$ Good for working with distributions, computing norms, means and variances quickly, solving series, curve fitting, solve differential equations, combining systems of ODE's.

- **Generating Function**: A function whose coefficients (found by differentiating the function) generate a desired sequence of numbers.
  - Given a sequence of numbers $p_k = (p_0, p_1, p_2, ...)$ we can define a function $G(x) = \sum_{k=0}^{\infty} p_k x^k = p_0 + p_1 x + p_2 x^2 ...$, so that $p_k = \frac{1}{k!}\frac{\partial^k}{\partial x^k}G(x)|_{x=0}$.
  - $G(1) = \sum_{k=0}^{\infty} p_k$ is the norm of p_k.
  - $G'(1) = \sum_{k=0}^{\infty} kp_k = z = \langle k \rangle$.
  - $G'(1) + G''(1) = \langle k^2 \rangle$ (second moment of p_k)
  - Generally $((x\partial_x)^n G(x))|_{x=1}\langle k^n \rangle = \sum_{k=0}^{\infty} k^n p_k$ to get any moment. Note that $(x\partial_x)^n$ means apply $x\frac{\partial}{\partial x}$ n times to $G(x)$.
  - Example: If the degree distribution $p_k = \frac{e^{-z}z^k}{k!}$ (Poisson, like for ER graphs), then the generating function $G(x) = \sum_{k=0}^{\infty} p_k x^k = \sum_{k=0}^{\infty} x^k \frac{e^{-z}z^k}{k!} = e^{(x-1)z}$. Then the norm $G(1) = 1$, the mean degree is $G'(1) = z$ and the second moment is $z + z^2$.
  - The excess degree distribution can be generated too: $Q(x) = \sum_{k=0}^{\infty} q_k x^k = \sum_{k=0}^{\infty} x^k \frac{(k+1)p_{k+1}}{z} = \frac{G'(x)}{G'(1)} \Rightarrow Q(x) = G'(x)/G'(1)$.
  - The mean excess degree q is $Q'(x) = \frac{G''(1)}{G'(1)} = \frac{\langle k^2 \rangle}{\langle k \rangle} - 1$.
  - Generating function have cool properties, which you can write down once you've done a couple of questions. For example, we can write a generating function $S(x) = ax + b$, where a is a probability that a node survives an attack to the network, and b is the negation of a. Then, given a generating function for the degree distribution, we can just combine the two generating functions to get the degree distribution after the attack: $G_{aft}(x) = G_{bef}(S(x))$, which is pretty neat.
    - From this we can show that $z' = az$ and $q = aq$.

9 Viral Marketing

- Very confusing lecture – may have to watch on media site.
- Show that the cluster size in a network can be found using Generating functions to give same result as earlier $S = 1 + \frac{z}{1-q}$

10 L10 Navigating Small Worlds

$ Clustering coefficient, small world, watts-strogatz model navigability and assortativity

- In this lecture, we try to minimise the effect of the assumptions used in the previous lectures: that networks are tree-like and that there are no long range correlations between nodes (nodes are sufficiently random).
- **Clustering Coefficient**: $c = \frac{3n_\Delta}{n_{--}}$, where $3n_\Delta$ is the number of triangles in the graph and n_{--} is the number of three node chains (0-0-0). Can also use to see whether assuming a random network is a good assumption if the clustering coefficient of a real network is known.
- For Erdos-Renyi graphs (for large N) $n_\Delta = \frac{N^3}{6}p^3 = \frac{z^3}{6}$, $n_{--} = \frac{N^3}{6}3p^2 \Rightarrow c = p$ from G(N,p), i.e. the clustering coefficient is just the probability of connections in the network.
- General form for the number of small subgraphs of a certain type in a network: $n_* = \sigma N^n p^k = \sigma z^k N^{n-k}$, where n and k are the number of nodes and links in a network and σ is a constant factor that depends on the kind of subgraph.
  - For every cycle $n = k$, $n_{cycle} = \sigma(Np)^k = \sigma z^k$. I believe n is the number of cyles/triangles or whatever the $*$ is.
  - All subgraphs that contain exactly one cycle ($n = k$) remain constant. All subgraphs that contain more than one cycle $n > k$ disappear and all trees $k < n$ explode [not sure I get this].
  - If z is not constant and $z = N^\alpha$, then $\alpha = 1 - \frac{n}{k}$, $(\alpha k + n - k = 0)$ for the subgraph to remain at a constant level as $N \Rightarrow \infty$.
- Many real-world networks are very sparse (low mean degree, we only know a handful of people), highly clustered (cyclical, my friends are friends) and have a low diameter (I can connect to almost anyone quickly).
- Watts and Strogatz showed how a network can be sparse, clustered,and small in diameter at the same time. The Watt-Strogatz model is as follows:
  - Create N node and link every node n to nodes $n+1, n-1, n+2$ and $n-2$, then for every link rewire it with probability p. Allows you to get low diameter, sparse, but clustered networks.
  - 'The "weak" long-distance links play an important role in society' because they reduce the diameter.

11 L11 Growth of the Internet

$ This lecture is about the evolution of networks, differential equations, Barabasi-Albert model, rich-get-richer dynamics, scale-free networks and the science of success.

- Many real world networks have power-law degree distributions: $p_k \sim k^{-\gamma}$, where $\gamma \in [2, 3]$. These types of networks are called **scale-free** networks.
- We can represent $\sum_k k^{-\gamma}$ as $Li_\gamma(1)$, where $Li_\gamma(x)$ is a γ-polylogarithm function.
- Scale free networks with $\gamma \in [2, 3]$ have a finite mean degree but an infinite excess degree.
- In a network that creates links between pairs of nodes at rate a and destroys existing nodes at rate b, the dynamics of the links can be described by

$$\dot{K} = aN - bK.$$

  We can thus write $\dot{z} = 2a - bz$. To find the stationary state of the mean degree we set $\dot{z} = 0 = a - bz/2 \Rightarrow z^* = 2a/b$.
- For the same case as the bullet point above, we can derive that the degree distribution changes as $\dot{p}_k = 2a(p_{k-1} - p_k) + b((k+1)p_{k+1} - kp_k)$. In terms of the generating function $G = \sum p_k x^k \Rightarrow \dot{G} = \sum \dot{p}_k x^k$, then $\dot{G} = 2a(x-1)G - b(x-1)G'$, where $G' = \frac{\partial G}{\partial x}$. In steady state $G(x) = e^{z(x-1)}$ - which shows that the network in steady state is a ER graph!
- For Barabasi-Albert networks the cumulative distribution dynamics is expressed as $\frac{dc_k}{dt} = \frac{kp_k}{2} - c_k$. From the steady state $\dot{c}_k = 0$ we find that $p_k = \frac{2c_k}{k}$. Note that $\frac{dc_k}{dk} = -p_k$, which means we can find $c_k = Ak^{-2}$, and thus $p_k = A'k^{-3}$, showing that the Barabasi-Albert networks are scale-free networks with $\gamma = 3$.
- The internet is a scale free network with $\gamma = 1.6$.

12 L12 - non-existent

13 L13 Criticality

$ Mean field approximation, critical states, bifurcations, and self-organized 'criticality'.

- Assume we are investigating a forest fire. Different patches in the forest can either be B burning, A at risk, or C burnt. Let there be N patches in the forest. Using bracket notation, e.g. $[B] = \frac{B}{N}$, is the normalised form (a proportion).
- We can define a mean-field model as $\dot{B} = pl_{AB} - rB$, where p is the rate at which the fire spreads between burning and 'at risk' patches, r is the rate at which a burning patch becomes burnt, and l_{AB} is the number of links between burning and 'at-risk' patches. In bracket notation (normalised) we get $[\dot{B}] = p[AB] - r[B]$, where we define $l_{AB} =: [AB]$, the number of AB links per node.
- When we approximate a quantity with its system average, its referred to as **mean field approximation**, e.g. assuming $p(A|link from B) =\approx p(A) = A/N = [A] \Rightarrow [AB] \approx q[A][B]$. Substituting in the mean field approximation (and setting $[C] = 0 \Rightarrow [A] = 1 - [B]$ we get $[\dot{B}] = pq[B] - pq[B]^2 - r[B]$. For stability, eigenvalue $\lambda = \frac{\partial[\dot{B}]}{\partial[B]} = pq - r$. If $pq > r$, fire will spread (Note, $R_0 = \frac{pq}{r}$, shown earlier in the lecture using percolation).
- The phenomenon whereby a system brings itself back into a critical state (e.g. the forest regrows only to be burned again) is referred to as **self-organised criticality** I believe.
- **Critical Slowing Down** refers to the phenomena whereby a critical system returns to its stationary state slowly.

14 L14 - Epidemics

$ Flying corpses, adaptive SIS model, pair approximation, moment closure, discontinuous transitions and hysteresis.

- Epidemic Model names often consist of letters describing the sequence in which agents transition through epidemic states: S (susceptible), I (infected and infectious), R (recovered or removed), A (asymptomatic infected, can infect others), E (exposed to the disease, will be infectious soon). SIR model is very famous.
- Lecture goes into long winded explanation of deriving the SIS model. Have a look at the questions and watch the lecture if necessary.
- I believe (check this though) that $[A]$ is a proportion and $[AA]$ is a 'link density', so if we have two types of nodes, say A and B, then $[A] + [B] = 1$, and (not sure why exactly) but $[AA] + [AB] + [BB] = \frac{z}{2}$ - a conservation law apparently. And then you have 'moment closure approximations' e.g. $[ABA] \approx \frac{[AB][BA]}{[B]} = \frac{[AB]^2}{[B]}$? and $[AAB] = \frac{2[AA][AB]}{[A]}$. I believe the logic is somehow to do with $(A) - (B) - (A)$ or $(A) - (A) - (B)$ connection chains. Ah I think they are simply probabilities of a chain occurring, so $[A]$ is the probability that you find a node A which is $[A] = \frac{A}{N}$ and $[AAB]$ or whatever would be the probability of finding the chain $(A) - (A) - (B)$ or whatever, and all the mean field approximation does is estimate these probabilities! Oh and $[ABC]$ or whatever are also called motifs. You can also make it more specific $[A_2B_3]$ is the proportion (probability of finding) nodes of type A with degree 2 connected to nodes of type B with degree 3 (I believe).

15 L15 - Opinion Formation

$ High dimensional moment expansions, heterogeneous expansions, stability analysis, stability analysis for PDEs.

- If the degree distribution is (or can become) large (broad), use heterogeneous approximations, otherwise use homogeneous degree distributions. [What these approximations are exactly, I'm not sure.]
- You can go from an ODE system to a PDE system using generating functions with multiple variables, e.g. $G(x, y) = \sum_{k,l} A_{k,l}x^k y^l$.
- Given a PDE $\dot{G} = G' - G$, we can find a stationary solution by setting $\dot{G} = 0$ or $G' = 0$. For stability use Ansatz $G(x, t) = G_*(x) + v(x)e^{\lambda t}$, if $\lambda > 0$ then the stationary state is unstable.
- Voter model - A model that describes how opinions change in a network - essentially just a network where the nodes can be in two states, either opinion A or opinion B, and model then describes the dynamics of how opinion A nodes becomes opinion B opinions or vice versa. Using a mean field approximation we assume that a link is rewired at a rate

p and an opinion is copied at a rate $\bar{p} = 1 - p$.

16 L16 - Spectral Centrality

$ Spectral Centrality, PageRank, KatzCentrality, Gershgorin's Theorem. Good for finding accounts, people in social networks, pages in world wide web, etc...

- Recall that we have looked at degree centrality (high degree nodes are more important), closeness centrality (nodes that have short connections to other nodes are important.
- To compute **Spectral Centrality**, construct the adjacency matrix $\mathbf{A}$ for the network, compute the eigenvalues and the eigenvector corresponding to the largest eigenvalue $\mathbf{v}$ will contain the importance of each node, i.e. entry v_i of $\mathbf{v}$ will be the importance of node i. Pretty neat. 'Spectral' means reducing a problem into an eigenvector problem.
- You can also find the Spectral Centrality vector, by iteratively multiplying the adjacency matrix by a single vector and the output vector repeatedly - this will converge towards the eigenvector with the largest eigenvalue. This approach only works if the leading (largest eigenvalue) is also furthest from the origin, i.e. if you have $\lambda_1 = 5$, but $\lambda_2 = -6$ it might not work.
- Bipartite networks have symmetric positive-negative adjacency matrix eigenvalues, so you cannot compute the leading eigenvector using the multiplication technique. To overcome this issue we can use the multiplication technique to the matrix $\mathbf{B} = \mathbf{A} + \alpha\mathbf{I}$, where $\mathbf{I}$ is the identity matrix (all this does is shift the eigenvalues, but not change the eigenvalues). This multiplication method and spectral centrality is what is essentially going on in PageRank google searches.

17 L17 - Clustering

$ Tree clustering, Girvan-Newman, modularity maximization, spectral clustering, diffusion on networks and Zachary's Karate club. Good for finding similar groups of people, animals, etc... clustering. Might be good for geofence idea.

- Spanning tree clustering algorithm: Simply break network into M components, and construct Adjecency matrix. Then run Kruskal's algorithm until M components (clusters) are left.
- Girvan-Newman algorithm: Modularity maximization: Start will entire network and then successively remove links with the highest betweenness centrality until you are left with the desired number of clusters.
- Spectral Clustering: Imagine people 'walking' on the network, each node produces and receives walkers: We can then create the system of ODEs: $\dot{x}_i = -k_i x_i + \sum_j Aijx_j$, where k_i is the node degree of node i and $\mathbf{A}$ is the adjacency matrix. This is a linear system so we can re-write it as: $\dot{\mathbf{x}} = -\mathbf{L}\mathbf{x}$, where $\mathbf{L}$ the **Laplacian matrix** $L_{ij} = -A_{ij} + \delta_{ij} k_i$, where δ_{ij} is the Kronecker delta. Also $\mathbf{L} = \mathbf{K} - \mathbf{A}$, where $\mathbf{K} = diag\{k_1, k_2, ..., k_n\}$.
  - Given $\dot{\mathbf{x}} = -\mathbf{L}\mathbf{x}$ we know that the solution is $\mathbf{x}(t) = \sum_n c_n(t)\mathbf{v}_n$, where $\mathbf{v}_n$ are the eigenvalues of $\mathbf{L}$ and $c_n(t) = c_n(0)e^{-\lambda_n t}$
  - The Laplacian is a positive, semi-definite matrix - there are no negative eigenvalues.
  - The speed at which walker 'equilibrate' is associated with the value of the **second** leading eigenvalue λ_2. This eigenvalue is also called the **spectral gap** and is an important network metric. The larger this eigenvalue is the quicker the system goes to equilibrium.
  - We can split the network into clusters where $\mathbf{v}_2$ (corresponding to λ_2) has positive entries belong to one cluster and another for negative entries. For further splitting (clustering) we can use further eigenvectors $\mathbf{v}_3, \mathbf{v}_4, ...$
  - The eigenvalues can tell us how many meaningful clusters exist in the network. A large gap in value between λ_2 and λ_3 for example show that its probably best to just split the network into two clusters.
  - Really awesome way to cluster a network, have to say!

18 L18 - Diffusion Map

$ A way to make sense of large data sets. It creates a color coded map according to the spectral centrality value of the nodes (I think).

- Recall spectral methods: Problems that can be mapped to eigenvalue problems: Transmission (captured by Adjacency matrix), Diffusion (captured by Laplacian matrix), Stability (captured by Jacobian matrix).
- **Localization**: Eigenvectors that have large entries on almost all nodes are called **delocalized** eigenvectors. Eigenvectors that are zero except on a small number of nodes are called **localized** eigenvectors.
- **Absolute rowsum**: The absolute rowsum of a row of a matrix is the sumer of the absolute values of the entries in that row: $r_i = \sum_j |Mij|$. In a matrix there cannot be an eigenvalue that is greater than the largest absolute rowsum, which implies that the adjacency matrix cannot have an eigenvalue that is greater than the highest node degree.
- The absolute rowsum of the adjacency matrix of a regular graph is the same for each row and the largest eigenvalue is the absolute rowsum.
- Generally, in every network there is an eigenvalue close to the mean degree, i.e. $\lambda_1 \approx z$.
- In a network with a node of high degree surrounded by nodes with a much lower node degrees then $\lambda_1 \approx \sqrt{k}$.
- Diffusion maps: I think its literally just PCA.
  1. Standardise the data, mean of 0 and variance and standard deviation of 1
  2. Define similarities and construct weight matrix: Use Euclidean distance, for example, do find distance matrix $D_{ij} = \sqrt{\sum_n (x_{in} - x_{jn})}$ between vectors of data. Then compute the weight matrix as $w_{ij} = \frac{1}{D_{ij}}$ (means smaller distances have a larger weights).
  3. Threshold: Set all small entries to zero $\Rightarrow$ Curse of dimensionality (Euclidean distance starts to behave weirdly, or rather the Euclidean distance is useful for points that are close together, but not when the points are far apart $\Rightarrow$ so we threshold).
  4. Find the Laplacian matrix defined as $L_{ij} = -w_{ij} + \delta_{ij} \sum_j w_{ij}$.
  5. Compute Eigenvectors of Laplacian matrix. The most important eigenvector for diffusion is the one that corresponds to the smallest non-zero eigenvalue. The nth entry of this eigenvector corresponds to the nth data point.

19 L19 - Master-Stability Functions

$ Good for studying the synchronization of oscillators, including electricity grids, vibrations of machinery.

- Kronecker product of matrices: $\mathbf{X} \bigotimes \mathbf{Y} = \begin{bmatrix} X_{1,1}\mathbf{Y} & X_{1,2}\mathbf{Y} & ... \\ X_{2,1}\mathbf{Y} & X_{2,2}\mathbf{Y} & ... \\ \vdots & \vdots & \ddots \end{bmatrix}$. The have various properties (additive, distributive, etc...) you can find in the notes.
- If there is a solution that exists for a local system and non-local (global) system (I think?) then the solutions are considered homogeneous.
- Given a local and global system, represent the global Jacobian $\mathbf{J}$ in terms of the local one $\mathbf{P}$ using the systems Laplacian $\mathbf{L}$ and coupling matrix $\mathbf{C}$. Then $\mathbf{J} = \mathbf{I} \bigotimes \mathbf{P} - \mathbf{L} \bigotimes \mathbf{C}$. To find the eigenvalues λ and eigenvectors $(\mathbf{r} \bigotimes \mathbf{s})$ of J to determine stability (for a reaction-diffusion system for example)
  - Compute the eigenvalues λ of $\mathbf{P} - \kappa\mathbf{C})$ where $\mathbf{s}$ is the corresponding eigenvector
  - To find a κ simply compute the eigenvalues of $\mathbf{L}$, which will have a corresponding eigenvector $\mathbf{r}$
  - λ and $(\mathbf{r} \bigotimes \mathbf{s})$ will then be the eigenvalues and eigenvectors of $\mathbf{J}$.
- If κ is unknown, then $M(\kappa)$ is the real part of the leading eigenvalue of $\mathbf{P} - \kappa\mathbf{C}$, and is the so-called master-stability function.