Using a co-occurrance network

Jim Brunner

June 29, 2017

1 Network Building

We are looking at creating ecological networks of a microbiome. Right now, I have built two networks, in the form of graphs $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{E}_1)$ and $\mathcal{G}_2 = (\mathcal{V}_2, \mathcal{E}_2)$. In both graphs, vertex are labeled by taxa name. I'm going to conflate the vertices and their label. The edge sets are defined by co-incidence and co-occurence, respectively. Given a set of samples with abundancs of organisms, we map the abundances to discrete levels, as proportions of the maximum abundance *of that organism*. Precisely, let the samples be s_i and raw abundance of organism j in sample i be r_{ji}. I map the abundances according to

$$a(r_{ji}) = \begin{cases} \lfloor \left(\frac{r_{ji}}{\max_{s_k}(r_{jk})} \right) n \rfloor + 1 & \frac{r_{ji}}{\max_{s_k}(r_{jk})} \geq m \\ 0 & \frac{r_{ji}}{\max_{s_k}(r_{jk})} < m \end{cases}$$

where m is some minimum. Then, I create weighted edges between vertices (where 0 weight means no edge exists) where the weights of edges in $\mathcal{E}_1$ are

$$w^1_{jk} = \frac{\|\{i : a(r_{ji}) = a(r_{ki}) \neq 0\}\|}{S}$$

where S is the total number of samples. That is, we count the propotion of samples in which the two taxa appear at the same discritized level.

The second network accounts for random coincidence of taxa in a sample, following [4]. It begins with $\mathcal{G}_1$, and compares to a null model N. The null model is defined in the following way.

$$A_j = \sum_{s_i} \mathbf{1}_{a(r_{ji}) \neq 0}$$

and

$$S^l_i = \sum_{v_j} \mathbf{1}_{a(r_{ji}) = l}$$

Then, N assumes that if

$$X_{jil} \sim binom\left(A_j, \frac{S^l_i}{\sum_{il} S^l_i} \right)$$

then $P(a^N_{ji} = l) = 1 - P(X_{jil} = 0)$. This allows us to calculate the probability of co-incidence of taxa under the null model. Let w^N_{jk} be

$$w^N_{jk} = \|\{i : a^N_{ji} = a^N_{ki} \neq 0\}\|$$

This is the similar to the co-incidence model but now randomized. Then,

$$P(w^N_{jk} = K) = \sum_{\{A \subset \mathcal{V}_2 : |A| = k\}} \prod_{l \in A} a_{jl} a_{kl} \prod_{l \notin A} (1 - a_{jl} a_{kl})$$

Ideally, we would then define $\mathcal{E}_2$ by the weights

$$w^2_{jk} = \begin{cases} 1 & P(w^N_{jk} \geq w^1_{jk}) \leq t \\ 0 & P(w^N_{jk} < w^1_{jk}) > t \end{cases}$$

However, that probability is intractible to compute. Instead, we take

$$\widetilde{P}(w_{jk}^N = K) = \sum_{l=0}^{i} \binom{N_1}{l}\binom{N_2}{K-l} p_1^j p_2^{K-l}(1-p_1)^{N-l}(1-p_2)^{N-K+l}$$

where

$$p_1 = p_a - \left(\frac{N_2}{N_1}\frac{N(\mu-\sigma^2)-\mu^2}{N^2}\right)^{1/2}$$

and

$$p_2 = p_a - \left(\frac{N_1}{N_2}\frac{N(\mu-\sigma^2)-\mu^2}{N^2}\right)^{1/2}$$

Finally, N_1, N_2 are to ensure that $p_1, p_2 \in [0,1]$. It turns out we need:

$$\frac{\mu N(1-p_a)-N\sigma^2}{N(1-p_a)-\sigma^2} \leq N_2 \leq \frac{\mu^2}{\mu-\sigma^2}$$

with μ the mean of the real distrubution, σ^2 the variance, and $p_a = \frac{1}{S}\sum_i a_{ji}a_{ki}$. So, we take

$$w_{jk}^2 = \begin{cases} 1 & \widetilde{P}(w_{jk}^N \geq w_{jk}^1) \geq t \\ 0 & \widetilde{P}(w_{jk}^N < w_{jk}^1) > t \end{cases}$$

1.1 An alternative coincidence construction

A faster way would be to first normalize abundance vectors, and just take

$$W = AA^T$$

(with the diagonal reset to 0) to be the weighted adjacency matrix, where A is the matrix of abundances. Then, each entry is the cosine of the angle between the abundance vectors. This also allows us to look at negatively aligned abundance vectors.

How then would we construct the cooccurrence network? Basically, what would our null model be? Well, we would still randomize the sample data. That is, we could take X_{ij} be a random variable representing the abundance of taxa i in sample j, and require that $\|X_i\| = 1$. Then, we can ask $P(\boldsymbol{X}_i \cdot \boldsymbol{X}_k > w_{ik})$. I just don't know what distribution to put on the X_{ij} (the random abundances). I'll start with a binomial:

$$P(\tilde{X}_{ij} = k) = \binom{n}{k} p^1 (1-p)^{n-k}$$

where $p = \frac{deg(s_j)}{\sum_l deg(s_l)}$ and $n = \lfloor \frac{1}{\min_k(r_{ik})} r_i \rfloor$ where $r_i = \sum_k r_{ik}$. Then I normalize to the unit sphere. Finally, I can use a Monte-Carlo simulation to see which weights to check. To do this, I generate random matrices with the appropriate distributions, using a built in Numpy method. Then I check the proportion of random entries larger than the weights. If this is small, I keep the edge. Unfortunately, I can't use a control variate or otherwise try to reduce variance, because generating my own samples was far slower than the Numpy method. The Numpy method is pre-compiled in C. Variance was ~ 0.25, so 1000 samples should be enough.

Similary to the above but maybe on more established footing is Pearson's correlation coefficient. That is:

$$\rho_{xy} = \frac{1}{N}\frac{(\boldsymbol{x}-\mu_x\boldsymbol{1})\cdot(\boldsymbol{y}-\mu_y\boldsymbol{1})}{\sigma_x\sigma_y}$$

which can of course be constructed by modifying the data matrix and then multiplying with it's transpose. In expectation, this the covariance divided by the product of the variances. I can also use a Monte-Carlo approach to check significance. This ends up keeping a lot more edges than the binning procedure. I am considering doing a first pass of only keeping hgih weights. Low weights may be significant (i.e. higher than one could expect randomly) but are still low, and so maybe should be dropped. In the literature, people tend to only keep weights over 0.8 or something.

2 Network Analysis

We can cluster - community clustering, spectral clustering, comparing to random graphs. Comparison of clusters to grouping by sample type.

Community clustering minimizes a funciton of the graph called modularity [2][5]. That is

$$Q = \frac{1}{2m}\sum_{i,j}\left(w_{ij} + \frac{d_i d_j}{2m}\right)\delta(c_i, c_j)$$

where $m = \sum_{i\sim j} w_{ij}$, d_i is the (weighted) degree of vertex i, c_i is the community containing vertex i, and δ is the Kronecker δ. This done by a kind of gradient search/greedy algorithm.

Spectral clustering performs a k-means clustering on the rows of the matrix whose columns are the k eigenvectors of the graph laplacian corresponding to the smallest k eigenvalues [7] . Morally, this means it clusters points together that are close in the first k (slowest decaying!) modes of the diffusion equation on the graph.

3 Using the network to analyze a sample

The question now is what can we do these networks?

First, assesing GOTTCHA reads. A single GOTTCHA read would produce a network with each connected component complete. I think the co-incidence network is more appropriate. We want to assess the probability that you see this set together. We have the probability that you see any vertex pair (estimated) as the edge weights of $\mathcal{G}_1$. Precisely, the edge weights are

$$w^1_{jk} = P(j \,\&\, k \in S^l_i)$$

where S^l_i is sample i at discrete abundance level l. It might be useful to have the directed weight graph where

$$w^3_{jk} = P(j \in S^l_i | k \in S^l_i)$$

but that wouldn't be hard, because then

$$w^3_{jk} = S\frac{w^1_{jk}}{\|\{i : a(r_{ki}) > 0\}\|}$$

Anyway, let's start with the simplest case of one abundance level. Assume GOTTCHA found taxa $a, b, c, ..., n$. Maybe the first thing we would want is

$$P(a|b, c, d, ..., n),\ P(b|a, c, d, ..., n),\ etc$$

Clearly, we can see directly $P(a|b)$, etc. We can also get a bound for triplets (assuming $P(c, b) \neq 0$):

$$P(a|b, c) \leq \frac{\min_{(i,j)\subset\{a,b,c\}}(P(i, j))}{P(b, c)}$$

but we can't do any better than triplets explicitly, because we don't have any sort of independence (conditional or otherwise) and because our network is not acyclic.

We can probably learn something from asking about the connectivity of the induced subgraph. Notice that if it isn't complete, then one of the $P(a, b)$ is 0 above.

What does the connectivity of the induced subgraph of $\mathcal{G}_2$ tell us? If it is connected, that's good. We could use that network to identify vertices that are connected to many of the vertices in the induced subgraph - this might indicate that node should be in the sample.

I guess $\mathcal{G}_2$ is a markov random field [3]. This might give us a way to calculate the probability you see a group taxa (and maybe others). The main idea of a MRF is that nodes are conditionally independent of nodes they aren't neighbors of (conditioned on ones they are neighbors of). If c are the (maximal) cliques of the graph (complete subgraphs), then the probability of configuration $\boldsymbol{x}$ is

$$P(\boldsymbol{x}) = \frac{1}{Z}\prod_c \psi_c(x_c)$$

where Z is a normalizing constant and ψ_c are potential functions I don't know how to come up with yet. Anyway, if we have a sample that contains (maybe as a subset) s, we can calculate something. Let C_s be the cliques represented in s.

$$P(\boldsymbol{s}) = \frac{1}{Z}\sum_{\{\boldsymbol{x}:s\subset\boldsymbol{x}\}}\prod_{c\in C_s}\psi_c(\boldsymbol{x})$$

And we can ignore cliques not represented in s. We probably have to change Z. I guess we can also use neighbor pairs instead of maximal cliques. Either way we have to figure out what ψ_c are.

3.1 Determining ψ_c

Before figuring out ψ_c, it should be noted that we have a choice of configuration space of the network. We can use a binary $\{1,0\}^N$ space to denote presence or absence, or we can choose a continuous space to include abundances. To begin, I will only consider presence & absence.

Snowshoe hairs and Canadian Lynx, CRNT

Now I'm very tempted to think about how this approach relates to mass action dynamical systems. I'm going to think about my very favorite model.

$$\dot{\boldsymbol{x}} = \kappa_1 x \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \kappa_2 xy \begin{pmatrix} -1 \\ 1 \end{pmatrix} + \kappa_3 y \begin{pmatrix} 0 \\ -1 \end{pmatrix}$$

Well. Yes. But there is only one maximal clique so $P(\boldsymbol{x}) = \psi(\boldsymbol{x})$ and that's just whatever it is. We know that it evolves according to the stochastic mass action equations. In fact, the result that the stationary distribution is a product of poissons for a complex balanced system can be thought of as fitting into this framework. I wonder if you could use this theory to re-prove that? Recall that the stationary distribution for a complex balanced system with equilibrium $\boldsymbol{c}$ is

$$\pi(\boldsymbol{x}) = \frac{1}{Z} \prod_{i=1}^{d} \frac{c_i^{x_i}}{x_i!}$$

so here we have that

$$\psi_c = \prod_{i \in c} \frac{c_i^{x_i}}{x_i!}$$

or taking cliques to be just singletons

$$\psi_i = \frac{c_i^{x_i}}{x_i!}$$

I can't think of any way to arrive upon that directly, and so prove the result from this direction. Also, This is even more general than a MRF, as it has self-loops. Self loops play the role of time update. In fact, one might say that a MC is to a MRF as an ODE is to a PDE. Then, self loops in the MRF correspond to time derivatives or forcing appearing in the PDE.

It seems reasonable to use a pairwise RMF. It also seems reasonable to take a log linear function. My initial guess is

$$\psi_{s \sim t}(y_s, y_t) = \exp(\boldsymbol{\theta} \cdot \mathbf{1}_{y_s = i, y_t = j})$$

where $\theta = (0, \log(P(s)), \log(P(t)), \log(1/2(P(s|t) + P(t|s))))$. But that's a guess. It makes sense we have only unconnected nodes. It also also penalizes leaving out nodes that are connected to the nodes we do have. However, this penalizes us too harshly if we include a hub node.

Let's play with a small network to try to get a handle on this. Consider the network shown in fig. 1 I think we can assume that

$$p(\boldsymbol{x}) = \prod_{s \sim t} \psi_{st}(\boldsymbol{x}) \prod_{s} \psi_s(\boldsymbol{x})$$

To start, we should think about the distribution of an independent node (T_1). Clearly

$$P(T_1 = x) = \psi_1(x)$$

Similarly, we should have

$$P(T_2 = x) = \psi_2(x) = \int \psi_2(x)\psi_{23}(y)dy = \mathbb{E}(P(T_2 = x|T_3))$$

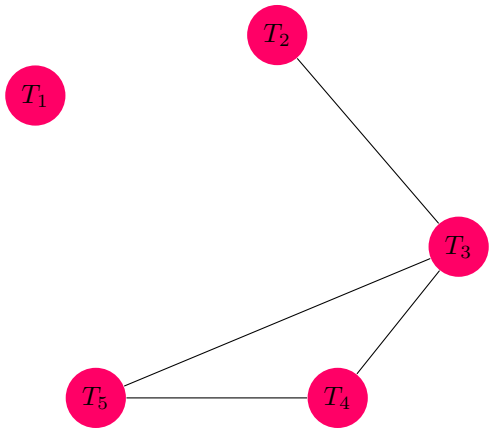


Figure 1: *A small network example*

Maybe a binomial is a good idea:

$$P\left(T_1 = \frac{k}{N}\right) = \binom{N}{k} p^k (1-p)^{N-k}$$

where p is something like the proportion of times T_1 appears in the data, the sum of abundances of T_1 in the data divided by N. N is some scaling parameter, like the largest total abundance or something. But then, it's natural to take N large and get a Gaussian by the CLT. So maybe we do a multivariate Guassian. Then

$$\psi_{s\sim t}(\boldsymbol{x}) = \exp\left(-\frac{1}{2} x_s \Lambda_{st} x_t\right)$$

and

$$\psi_s(\boldsymbol{x}) = \exp\left(\eta_s x_s - \Lambda_{ss} x_s^2\right)$$

[3]. We already have the covariance (normalized by the product of the standard deviations) matrix Σ, in the Pearson coefficient. So, we have $\Lambda = \Sigma^{-1}$, and $\boldsymbol{\eta} = \Sigma^{-1}\boldsymbol{\mu}$. Unfortunately, Σ is singular, it's dimension is bounded by the number of samples I have. This of course leaves us in the wind when it comes to Λ. So, I suppose we have to try to approximate Λ subject to the constraint that if $s \not\sim t$, $\Lambda_{st} = 0$. We can use the SVD:

$$\Sigma = MSN^* \Rightarrow \Sigma^+ = N\tilde{S}M^*$$

where $\tilde{S}$ is diagonal with $1/s_i$ until you get to zero singular values, where you just put a 0. Because Σ is symmetric, we have $M^* = N$. Then,

$$\Sigma^+ = N\tilde{S}N$$

I guess this is used, although it doesn't seem to preserve zeros. There's a function in scipy.

On the other hand, taking N large and p small we get an exponential random variable. So perhaps I want

$$\psi_1(x) = e^{-\lambda} \frac{\lambda^{Nx}}{(Nx)!}$$

or more directly

$$\psi_1(x) = \binom{N}{Nx} p^{Nx} (1-p)^{N-Nx}$$

and using Stirling's formula,

$$\log(\psi_1) \approx N\left[x \log\left(\frac{p}{x}\right) + (1-x)\log\left(\frac{1-p}{1-x}\right)\right]$$

3.2 Comparing configurations without specifying ψ

To compare two configurations $\boldsymbol{x}$ and $\boldsymbol{y}$, we can inspect

$$\frac{P(\boldsymbol{x})}{P(\boldsymbol{y})} > 1$$

without knowing them. For that, I would need to just identify which cliques change, and ask whether they increase or decrease. How exactly to decide that is a question. Probably single nodes we wouldn't change. For a clique, we could assume that ψ_c decreases if we have over half the nodes and reduce and still have at least half, but increase if the we have less than half and decrease. That is, let c be a clique with n members, and $\psi_c(k)$ be the value of ψ_c with k members "on". Then

$$\frac{\psi_c(k+1)}{\psi_c(k)} \begin{cases} > 1 & k > n/2 \\ < 1 & k \leq n/2 \end{cases}$$

Then, we are basically asserting the cliques like to either be all there or none there. Which seems good.

Here the algorithm for deciding which of two nodes should be on and which should be off would be:

(i) Identify cliques $c_1, ..., c_m$ which contain one or both nodes

(ii) compute $\frac{\psi_{c_i}(k^1)}{\psi_{c_i}(k^2)}$ for each $i \in \{1, ..., m\}$, where k^j is the number of members present in configuration $j \in \{1, 2\}$, and this fraction is computed according to a simple rule of the kind above.

(iii) multiply these ratios: $\frac{P(\text{configuration 1})}{P(\text{configuration 2})} = \prod_{i=1}^{k} \frac{\psi_{c_i}(k^1)}{\psi_{c_i}(k^2)}$

I need to specify that rule. The simplest reasonable thing is probably linear

$$\frac{\psi_c(k+1)}{\psi_c(k)} = \frac{2(1-r_{min})}{n-1}k + r_{min}$$

but one could imagine some kind of sigmoidal rule as well, so that around $n/2$ this ratio is close to 1. This makes sense on a 2 clique, as it implies that

$$\psi(2) > \psi(1) \;\&\; \psi(0) > \psi(1)$$

and in general makes sense for n even. It makes sense for n odd as well. It asserts that it is equally likely to have on $n/2 \pm 1/2$. Finally, if we need to compare k and $k+2$, we simply take

$$\frac{\psi(k+2)}{\psi(k)} = \frac{\psi(k+2)}{\psi(k+1)} \frac{\psi(k+1)}{\psi(k)}$$

That is, a discrete analogue to the chain rule.

To compare $P(v_1 = 1|\xi)$ and $P(v_2 = 1|\xi)$, let X be the configurations with ξ and $v_1 = 1$, and Y the configurations with ξ and $v_2 = 1$. Then,

$$P(v_1 = 1|\xi) = \sum_{x \in X} \frac{P(x)}{P(\xi)}$$

and so

$$P(v_1 = 1|\xi) > P(v_2 = 1|\xi) \Leftrightarrow \sum_{x \in X \setminus Y} P(x) > \sum_{y \in Y \setminus X} P(y)$$

but inspecting X and Y, we see that this is equivalent to

$$\sum_{x \in X \setminus Y} (P(x) - P(\tilde{x})) > 0$$

where $\tilde{x} = x$ for all nodes except v_1 and v_2, while $x(v_1) = \tilde{x}(v_2) = 1$ and $x(v_2) = \tilde{x}(v_1) = 0$. This can be written

$$\sum_{x \in X \setminus Y} \prod_{c : v_1, v_2 \notin c} \psi_c(x) \left(\prod_{c : v_1 \text{ or } v_2 \in c} \psi(x) - \prod_{c : v_1 \text{ or } v_2 \in c} \psi(\tilde{x}) \right) > 0 \tag{1}$$

where c are maximal cliques of the RMF.

3.3 Diffusion based method.

Here's an idea inspired by spectral clustering: Solve the diffusion equation on the graph:

$$\frac{\partial}{\partial t}u(v,t) = Lu(v,t)$$

where v takes values in the vertex set of the graph. Then, we can encode "known" information in three ways: initial values, boundary values, or a forcing vector.

Method 1 (Initial Value Problem)**.** *Let $u_i(t)$ be the solution at node v_i to the discete diffusion problem*

$$\frac{d}{dt}\boldsymbol{u}(t) = -L\boldsymbol{u}$$

where L is the graph laplacian with initial conditions $u_i = 1$ if node v_i is known to be "on", $u_j = 0$ if v_j is known to be "off", and $u_k = 0.5$ (or 0, or perhaps encoded with some confidence in $[0,1)$) if v_k is unknown. We then normalize the initial vector so that it represents a probability distribution on the nodes.
Then, if $\boldsymbol{K}$ is the information "known" and the values of v_k and v_l are unknown,

$$\int_0^\infty u_k(t)dt - \int_0^\infty u_l(t)dt > 0 \Rightarrow P(v_k = 1|\boldsymbol{K}) > P(v_l = 1|\boldsymbol{K})$$

We can easily compute these comparisons. Solutions to the diffusion equation are of the form

$$\boldsymbol{u} = \sum_{i=1}^{a} c_i\mathbf{1} + \sum_{i=a+1}^{n} c_i e^{\lambda_i t}\boldsymbol{\xi}_i$$

where a is the number of connected components of the graph, and $\lambda_i, \boldsymbol{\xi}_i$ are eigenvalue, eigenvector pairs of L. Note that each eigenvalue $\lambda_i \leq 0$, with $\lambda_i < 0$ for $i = a+1, ..., n$ [7]. Then, assuming there is some intial mass on the connected components containing v_1 and v_2,

$$\int_0^\infty u_k(t)dt - \int_0^\infty u_l(t)dt = \sum_{i=a+1}^{n} c_i(v_{ik} - v_{il}) \int_0^\infty e^{\lambda_i t}dt = \sum_{i=a+1}^{n} \frac{-c_i}{\lambda_i}(\xi_{il} - \xi_{ik})$$

and $\boldsymbol{c} = V^{-1}\boldsymbol{u}(0)$. Therefore, it is straightforward to compute and compare the transitive terms of the solution:

$$\int_0^\infty \left(u_k(t) - \sum_{i=1}^{a} c_i\right) dt = \sum_{i=a+1}^{n} \frac{-c_i}{\lambda_i}\xi_{ik}$$

We can regard this method as the Kolmogorov forward equation of a jump process that transitions from taxa to taxa which have an edge between them at a linear rate (see fig. 2). The state space of this process is the affine space $\mathbf{1}^\perp + \boldsymbol{b}_1 \cap \mathbb{Z}^n_{\geq 0}$, where $\boldsymbol{b}_i$ are the standard basis vectors. At any time t, $u_l(t)$ is the probability that the process is in the state $\boldsymbol{e}_l$. This model clearly admits no extinction events and is complex balanced deficiency zero. In the deterministic setting, it has globally attracting equilibrium $\frac{1}{n}\mathbf{1}$. Therefore, according to [1], it has stationary distribution on each connected component

$$\pi(\boldsymbol{x}) = \frac{1}{Z_{cc}} \prod_{i=1}^{n} \frac{\left(\frac{1}{n}\right)^{x_i}}{x_i!} e^{-\frac{1}{n}}$$

The state space of the system is $\{\boldsymbol{b}_j\}$, and we note that for $j = 1, ..., a$

$$c_j = \pi(\boldsymbol{b}_j) = \frac{1}{Z_{cc}n} e^{-\frac{1}{n}}$$

and so the distribution is uniform. We therefore must look at the transient behavior, which we take by the integral above, subtracting this stationary distribution. Unfortunately, this is going to allow us to compare nodes in the same connected component. We can add back the stationary distribution to get a better comparison. Then, we are choosing first the most likely connected components and then ranking within those by transient behavior.

Interestingly, we can regard this as the reaction network directly, deterministically modeled, which is the volume scaling limit of the jump process described above.

This method is the only one of the three that is well suited for judging the entire network, without confidence in the preknowledge encoded in the initial conditions. The other two require at least one (method 2) or two (method 3) confident assertion for each connected component of the graph.

Method 2 (Boundary Value Problem)**.** *Let $u_i(t)$ be the solution at node v_i to the discete diffusion problem*

$$\frac{d}{dt}\boldsymbol{u}(t) = -L\boldsymbol{u}$$

where L is the graph laplacian with fixed values (which can be regarded as boundary values) $u_i = 1$ if node v_i is known to be "on", $u_j = 0$ if v_j is known to be "off".
Then, if $\boldsymbol{K}$ is the information "known" and the values of v_k and v_l are unknown, and $\tilde{\boldsymbol{u}}$ is the equilibrium solution to the diffusion problem,

$$\tilde{u}_k dt > \tilde{u}_l \Leftrightarrow P(v_k = 1|\boldsymbol{K}) > P(v_l = 1|\boldsymbol{K})$$

First, we establish conditions under which the boundary value version will always have an equilibrium. A "boundary value" set on the graph is a set of fixed node values. We have seen already that the equilibrium of the diffusion problem is uniform, and so if we only have known "on" nodes $\tilde{\boldsymbol{u}} = \mathbf{1}$. If we specify off nodes as well, then there is not an equilibrium solution to the diffusion problem which gives those values at boundary nodes. However, this doesn't mean the boundary value problem (or, more precisely, the equvalent forced problem on the unknown subset) doesn't have an equilibrium solution. Take for example a complete graph on three nodes. If we prescribe one node as 1 and one as 0, the reduced problem is

$$\frac{d}{dt}u = -2u + 1$$

which has an equilibrium at $u = 1/2$. This is not an equilirium to the entire problem, because in that case

$$-L\begin{pmatrix} 1 \\ 1/2 \\ 0 \end{pmatrix} = \begin{pmatrix} -2 & 1 & 1 \\ 1 & -2 & 1 \\ 1 & 1 & -2 \end{pmatrix}\begin{pmatrix} 1 \\ 1/2 \\ 0 \end{pmatrix} = \begin{pmatrix} -3/2 \\ 0 \\ 3/2 \end{pmatrix}$$

The boundary value problem is equivalent to the forced problem on unknown nodes

$$\frac{d}{dt}\boldsymbol{u}|_{\boldsymbol{U}} = L|_{\boldsymbol{U}}\boldsymbol{u} + f_{\boldsymbol{K}}$$

where $\boldsymbol{u}|_{\boldsymbol{U}}$ is the projection of $\boldsymbol{u}$ onto the unknown set of nodes, $L|_{\boldsymbol{U}}$ is L with the rows and columns of the known nodes removed, and $f_{\boldsymbol{K}}$ is the forcing due to the boundary conditions.

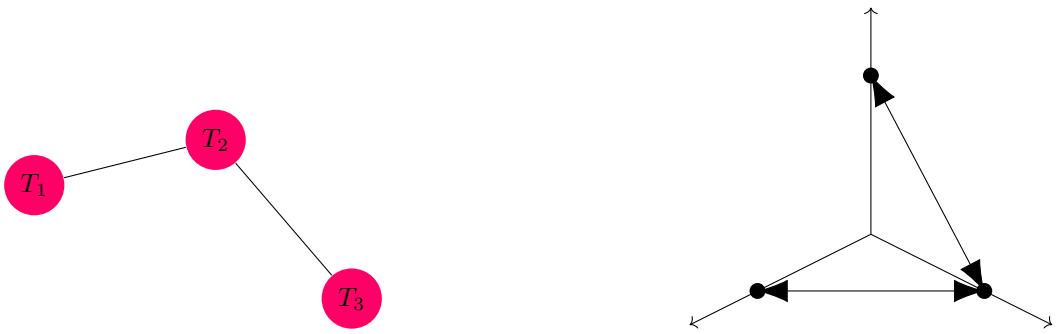
The nullity of L is the number of connected components of the graph. Removing a column from a matrix will not lower the rank of that matrix if the column was a linear combination of other columns, which is the case precisely when the column is the first one removed that corresponds to some connected component of the graph. Therefore, as long as we specify at least one "boundary value" from each connected component, $L|_{\boldsymbol{U}}$ is non-singular and there exists a unique equilibrium to the boundary value problem.

The equilibrium is simply computed by solving

$$0 = L|_{\boldsymbol{U}}\tilde{\boldsymbol{u}} + f_{\boldsymbol{K}}$$

The boundary value problem can be interpreted as a population walking around the graph, with the population at some nodes maintained at fixed values. We then assume that probability that a node is "on" is proportional to the size of the population at that node. Perhaps "voters" are a good analogy.

It can also be interpreted as the same reaction network as method 1, but the values of some species fixed.



(a) *A small network example* ***(b)*** *State transition graph for the IVP*

Figure 2: *Diffusion is equivalent to the Kolmogorov forward equations for the above system.*

Method 3 (Forced Problem)**.** *Let $u_i(t)$ be the solution at node v_i to the discete diffusion problem*

$$\frac{d}{dt}\boldsymbol{u}(t) = -L\boldsymbol{u} + f$$

where L is the graph laplacian and f a forcing vector with $f_i = \alpha_{cc}$ if node v_i is known to be "on", $f_j = -\beta_{cc}$ if v_j is known to be "off", where cc denotes a connected component of the graph. We choose α_{cc} and β_{cc} so that on any connected component cc, $\sum \alpha_{cc} = \sum \beta_{cc} = 1$.
Then, if $\boldsymbol{K}$ is the information "known" and the values of v_k and v_l are unknown, and $\tilde{\boldsymbol{u}}$ is the equilibrium solution to the diffusion problem,

$$\tilde{u}_k dt > \tilde{u}_l \Leftrightarrow P(v_k = 1|\boldsymbol{K}) > P(v_l = 1|\boldsymbol{K})$$

Notice that the kernel of L is $span(\{\mathbf{1}|_{cc}\})$, where cc denotes a connected component of the graph. For an equilibrium solution to exist, any connected component with an "on" node must also have an "off" node, so that $f \in span(\{\mathbf{1}|_{cc}\})^{\perp}$, which is the range of L because L is symmetric.

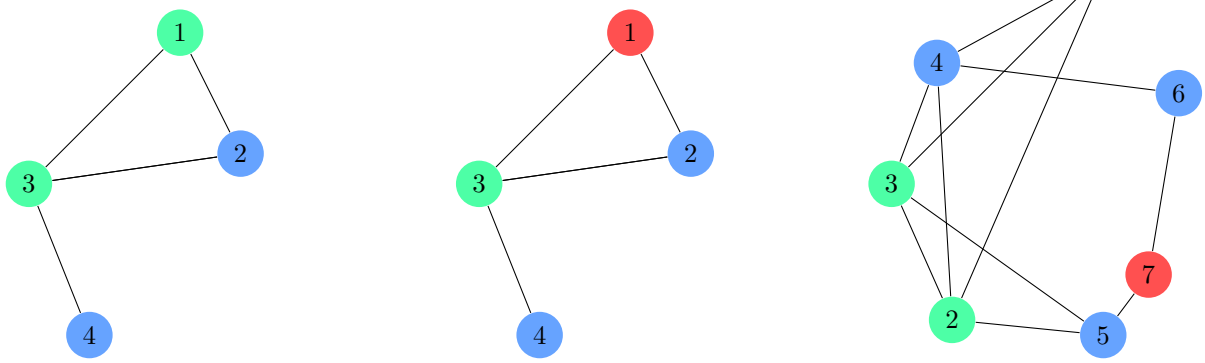
We can easily compute the equilibrium solution by solving, as long as we have guarenteed that $f \in range(L)$. On a practical note, we use numpy's least squares solver because the matrix L is singular. Unfortunately, if we do not specify an off node in a connected component in which we do specify an on node (or vice versa), there is no equilibrium solution.

The third method can be interpreted much like method 2. However, instead of the population being maintained at known nodes, we have a constant inflow or outflow from the graph at these nodes.

This can also be regarded as the same chemical reaction network model as method 1, but now with inflow and outflow.

3.3.1 Tests of these methods

I tested these three ideas on two small (unweighted) graphs, shown in fig. 3.



(a) *One tested configuration* ***(b)*** *One tested configuration* ***(c)*** *One tested configuration*

Figure 3: *Test networks. Blue nodes were "unknown", red nodes "off", and green "on".*

Configuration	Method	Ranking	Ties
fig. 3(a)	IVP	2, 4	none
	BVP	4, 2	4, 2
	Forcing	2, 4	none
fig. 3(b)	IVP	4, 2	none
	BVP	4, 2	none
	Forcing	4, 2	none
fig. 3(c)	IVP	4, 5, 6	none
	BVP	4, 5, 6	none
	Forcing	4, 5, 6	none

Table 1: *Results of ranking nodes by liklihood of being "on" by the three above methods*

4 Comparison of Networks

At some point here, I'm going to want to compare networks. Luckily, there's things to do. There are of course the simple and obvious: size, radius, various connectivity and clustering metrics. There are also more interesting things, like the random walk kernel on graphs. Graph kernels are similar to an inner product on two graphs.

A graph kernel $k(G_1, G_2)$ must be symmetric and positive definite (an inner product must also be bilinear) [6]. To compute the most common type (random walk kernels) we need to define the direct product of two graphs G and G'. That is the graph $G_\times$ with vertices

$$(v_i, v'_j) \in V \times V'$$

and edges

$$(v_i, v'_j) \sim (v_k, v'_l) \Leftrightarrow v_i \sim v_k \,\&\, v'_j \sim v'_l$$

We can compute the adjacency matrix of this by computing the Kronecker product of the adjacency matrices of G and G'. The Kronecker product of matrices A and B is

$$A \otimes B = \begin{pmatrix} a_{11}B & a_{12}B & \cdots & a_{1m}B \\ a_{21}B & \ddots & & \\ \vdots & & & \\ a_{n1}B & & & a_{nm}B \end{pmatrix}$$

A random walk on this graph is isomorphic to a random walk on either graph, and so one can consider a random walk on this graph as a simultaneous random walk on both. We can take the adjacency matrices normalized so row sums are 1, call those A, A', and then

$$W_\times = A \otimes A'$$

This gives a stochastic matrix $W_\times$. Given distributions p and p', define $p_\times = p \otimes p'$. Define the kernel

$$k(G, G') = \sum_{k=0}^{\infty} \mu(k) q_\times^T W_\times^k p_\times$$

where q, q' are stopping probabilities, and p, p' are initial distributions. This counts all the common random walks. The coefficients $\mu(k)$ serve two purposes. One is to make sure the sum is finite. The other allows one to tune the kernel to emphasize walks of certain lengths. I think it's called geometric if $\mu(k) = c^k$. However, this seems like it will overemphasize short walks, of which there will be many in common.

References

[1] David F. Anderson, Gheorghe Craciun, and Thomas G. Kurtz. Product-form stationary distributions for deficiency zero chemical reaction networks. *Bulletin of Mathematical Biology*, 72(8):1947–1970, Nov 2010.

[2] Aaron Clauset, M. E. J. Newman, and Cristopher Moore. Finding community structure in very large networks. *Phys. Rev. E*, 70:066111, Dec 2004.

[3] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.

[4] Heiko Mller and Francesco Mancuso. Identification and analysis of co-occurrence networks with netcutter. *PLOS ONE*, 3(9):1–16, 09 2008.

[5] M. E. J. Newman. Analysis of weighted networks. *Phys. Rev. E*, 70:056131, Nov 2004.

[6] S. V. N. Vishwanathan, Nicol N. Schraudolph, Risi Kondor, and Karsten M. Borgwardt. Graph kernels. *J. Mach. Learn. Res.*, 11:1201–1242, August 2010.

[7] Ulrike von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.