NETWORKS AND COMPLEX SYSTEMS QUAL PRACTICE SOLUTIONS

DARYL DEFORD

Introduction

This document consists of preliminary solutions to the 2/19/15 practice questions (with an additional bonus question that I needed an excuse to answer anyway). The answers are mostly not intended to be comprehensive, but instead sketches of the main ideas.

1. Easier Questions

(1) **Define the various centralities for a network.**

- (degree) Degree centrality ranks the nodes by the number of edges incident to each node.
- (closeness) Closeness centrality ranks each node by the length of geodesics to each other node. Symbolically, $C_i = \frac{n}{\sum_{i \neq j} d_{i,j}}$.
- (betweeness) Betweeness centrality ranks each node by the number of shortest paths between pairs of other nodes containing the node. Symbolically,$B_i = \sum_{i \neq j \neq k} \frac{\sigma_{j,k}(i)}{\sigma_{i,j}}$.
- (eigenvector) Eigenvector centrality ranks each node by the sum of the rankings of their neighbors. This corresponds to iterative multiplication by the adjacency matrix of the graph, so the nodes are ranked by their corresponding magnitudes in the leading eigenvector. This process can also be normalized to use the random walk Laplacian $D^{-1}A$, which is stochastic so the leading eigenvector is the steady state for the system. This corresponds to the probability that a random walk ends up at node j after the process is mixed.
- (Katz) Katz centrality is a modification of eigenvector centrality that works for directed networks. The idea is to add a constant value of reputation to each node at each step of the "Markov" process with final solution given by $(I - \alpha A)^{-1}\mathbf{1}$.

(2) **Describe the clustering algorithms.**

- (thresholding) Input: A (dis)similarity matrix and a threshold value. Place nodes i and j in the same cluster if the corresponding i, j entry is (above)below the threshold. Output: The clusters.
- (linkage) Input: A set of data points and a metric. This is a hierarchical, iterative process. Initialize with each point as a separate cluster. At each step, select the pair of clusters with the highest metric value (break ties randomly) and form a new cluster by taking their union. Update the similarity values according to one of the following three rules:
  - (single) Define the new similarity between clusters by taking the pairwise maximum of elements under the metric.
  - (complete) Define the new similarity between clusters by taking the pairwise minimum of elements under the metric.
  - (average) Define the new similarity between clusters by taking the average of all pairwise comparisons under the metric.

  Repeat this process until there is only a single cluster. Output: the ordered list of joins.
- (k–means) Input: A set of points embedded in $\mathbb{R}^n$ and an integer k that determines the number of clusters to find. This is also an iterative algorithm. Initialize with k arbitrary points in $\mathbb{R}^n$ as representative elements. For each point in the data set, assign it to the cluster associated to the nearest representative. Compute the centroid of each cluster and replace the original representative points with these centroids. Reassign the data points by Euclidean distance and repeat until the centroids do not change. Output: The k clusters.

Date: March 15, 2015.

- (spectral) Input: The adjacency matrix of a network. Eigenvector centrality in its various formulations clusters the nodes based on a discretization of a particular eigenvector corresponding to a particular Laplacian. Thus, based on the clustering desired compute the corresponding Laplacian and the eigenvector described below:
  - The second eigenvector of the Standard Laplacian gives a clustering that minimizes (relaxed) Cut and RatioCut formulations.
  - The second eigenvector of the Normalized Laplacian gives a clustering that minimizes the (relaxed) NormalizedCut.
  - The second highest eigenvector of the Random Walk Laplacian also minimizes the (relaxed) NormalizedCut.

  Usually, the nodes are clustered according to the sign of the entries in the eigenvector, although sometimes the median value is used as a division point instead.

(3) **What defines a scale free network?**

Scale free networks are characterized by an exponential degree distribution. This means that the fraction of nodes in the network with degree k is approximately $k^{-\gamma}$ for some fixed constant γ. For most empirical networks the variable γ is between two and three. Qualitatively, these networks tend to have high degree nodes (hubs) connecting densely connected sub networks of much lower degree. The Albert–Barabsi preferential attachment model is the most commonly used method for generating these networks.

(4) **Which centrality scores are best for very large networks?**

Honestly, none of them are great. Approximate sampling methods (i.e. looking at neighborhoods of neighborhoods of randomly selected nodes) are probably best for very large networks. Degree centrality is pretty reasonable especially if the data is expected to fall into a particular type of degree distribution. For example, if you expect the network to have scale free or small world properties, then information about specific nodes can be inferred with high probability from their degree. Eigenvector centrality is possible if the network is quite sparse using iterative methods to calculate the steady state.

Betweeness clustering is unreasonable because the numerator cannot be calculated efficiently although the denominator can be calculated in matrix multiplication time. Closeness centrality can be calculated in matrix multiplication time as well. To compute the length and number of shortest paths between every pair of nodes in the network requires being able to look entrywise at the collection of matrices $\{A, A^2, A^3, \ldots, A^{\text{diam}(A)}\}$, to determine the first time that every entry is non–zero and record the value that occurs. Unfortunately, matrix multiplication time is much too slow for large networks.

(5) **Describe the constructions of the standard random networks**

- (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.
- (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.
- (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.

2. Harder Questions

(1) **Derive the formula for eigenvector centrality.**

The idea behind the basic version of eigenvector centrality is that we should give more credit to nodes whose neighbors are high ranking. That is, we define the ranking of a node to be the sum of the rankings of each node adjacent to it multiplied by some scalar. If we consider the ranking values of the nodes as a vector, then multiplication by the adjacency matrix has the property of computing this value exactly, so we are looking for an eigenvector of the adjacency matrix with the scalar value the inverse of the eigenvalue.

Requiring that the entries of the resulting vector to be non–negative we can apply[1] the Perron–Frobenius Theorem to see that the eigenvector we want corresponds to the leading eigenvalue. We can also slightly rephrase the definition so that the matrix is stochastic, by dividing each column by the degree of the corresponding node. In this case, our matrix is the transition matrix for the associated finite state Markov process and we are computing its steady state solution. For undirected networks the Markov interpretation just gives rankings that are proportional to degree centrality. In either case, the correct eigenvalue can be computed by iterative methods for the leading eigenvector.

For directed graphs more care is necessary since weakly connected components tend to absorb all of the centrality in the network. To combat this problem the notion of Katz centrality is introduced, where at each step each node is granted some small amount of centrality. This becomes an affine system, but the solution can still be realized as a matrix vector multiplication using the machinery of Neumann series. Google's PageRank algorithm is a modified version of Katz centrality using the stochastic version of the adjacency matrix to prevent highly central nodes from distributing all of their centrality to their out neighborhood[2].

(2) **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

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

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.

[1] with some ... caveats detailed in my previous document.

[2] Just because your webpage can be found on Google doesn't make you important.

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

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

and

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

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 λ, 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.

(3) **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.

(4) **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[3].

(5) **Discuss the definition of a small world network. Give examples of networks that are/are not small world networks.**

The notion of a small world network does not have a perfectly standardized definition, but they are usually characterized by two properties: a low average geodesic length and a high clustering coefficient. Generally, the average geodesic length is expected to be about the log of the number of nodes in the graph. The clustering coefficient is defined to be $\frac{1}{n}\sum_{i=1}^{n} \frac{e_i}{\binom{\deg(i)}{2}}$ where e_i is the number of edges between neighbors of i. This is a measure of how connected the neighborhood of each node is compared to the complete graph. In general, to be considered a small world network, this value should be significantly higher than that of an Erdös–Renyi graph on the same number of nodes and edges.

For examples, transportation networks such as airline or train networks satisfy the small world properties due to their hubs, while local road networks do not(they tend to be fairly regular grids). Electrical power grids, neural networks, and social networks tend to satisfy the small world properties. The most common counterexample is social networks either all people, or people at an institution over an enormous time scale. In this case it is unlikely that there will be short path lengths between randomly selected individuals.

(6) **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.

Another approach is to form the matrix $M_{i,j} = \frac{D^2_{1,i} + D_{1,j} - D^2_{i,j}}{2}$. 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.

[3] at least for A and L.

(7) **Suppose you have a weighted directed network. Discuss the techniques you know that are applicable to such a network or could be adapted to this case.**

Weighted directed networks are one of the most general form of network models but the standard techniques for analyzing networks have to be adjusted to account for the weights and asymmetry. The trade off is between having finer grained information and having to adapt standard techniques to reflect thee additional complexities. Let's analyze the different types of statistics and dynamics individually.

For degree distribution, we need to consider in–degree and out–degree separately. We can also consider weighted degree vs. incidence degree to get finer grained information. For paths there is a similar dichotomy between the topological (binary) network and the weighted flow network. Depending on the data type the weights can either be thought of as costs or capacities which changes the definition of geodesic in each case. This is important for determining weighted variants of the standard statistics such as diameter, eccentricity, etc.

For centrality measures, modulo the definitions of geodesics and the edge weights as above, we can calculate betweeness and closeness centrality for both in and out edges. Eigenvector centrality must be modified significantly. There are two main approaches to this problem. One is a generalized notion of centrality due to Kleinberg that rates authority centrality and hub centrality separately as the eigenvalues of A^TA and AA^T. The other is to use a version of Katz centrality with a stochastic matrix that takes into account the edge weights.

For clustering the most standard approach is to simply assume that the network is undirected and apply basic techniques. Other approaches have been developed, such as redefining the objective functions for modularity and spectral clustering or projecting onto bipartite networks that preserve notions of the directed arcs and clustering in these settings. However, none of these techniques have become standard in the literature.

As far as dynamics, we can form modified versions of the Laplacian and random walk matrix incorporating the edge weights. Another approach is to only consider the out or in edges separately. The interpretation remains mostly the same in the case of the random walk dynamics but is slightly different for the Laplacian depending on what type of flow is being considered in the application.

(8) **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 $\sum_{j,k} jk(q_k\delta_{j,k} - q_j \cdot q_k) = \sum_k k^2q_k - \left(\sum_k kq_k\right)^2 = \sigma_q$. 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[4] 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:

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

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 obtain the standard assortativity for degrees by substituting in k_i for x_i into the expression above.

(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 (± 1) assigned to the columns arbitrarily. This construction gives the Laplacian many of its algebraic properties, such as symmetry and positive semi–definiteness.

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[5]. Letting φ 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.

[4] less statistical

[5] hence the name

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 λ_i are all non–negative and hence the final solution for φ 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 φ 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 γ_i is the i^{th} coordinate of the forcing vector in the eigenvector coordinates. Examples on 100 nodes can be see for the standard and forced cases below[6].

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

[6]only if you are using Adobe Reader, sorry Linux users...