Inference, Models and Simulation for Complex Systems CSCI 7000-003
Lecture 11 12 October 2010
Prof. Aaron Clauset

1 Large-scale network structures

The structural measures we met in Lectures 7 and 8, e.g., the degree distribution, the diameter, the clustering coefficient and the various centrality measures, can tell us a great deal about the structure of a network. However, these measures generally describe patterns as if they were spread uniformly throughout the network. In many real-world network, however, networks are heterogeneous but *not* in an evenly distributed way. Instead, the patterns are arranged into larger-scale structures of tens, hundreds or even thousands of vertices. These kinds of structural patterns, which may be very subtle at the vertex-level, are called *large-scale structures*.

One way to think about this idea is to recall the notion from spatial data analysis of "clumps" or "clusters" of data. That is, perhaps the observed heterogeneity is simply a byproduct of mixing or averaging across several distinct parts of the network that are, within themselves, relatively homogeneous. (We'll revisit this analogy in Section 2.5.) This kind of pattern is called a *modular organization* or *community structure*. In biology, such structures could be functional clusters of genes or proteins, they might represent targets of natural selection above the level of individual genes but below the level of the whole organism, and, in the social sciences, they could be human social groups with common interests, histories or behaviors.

A natural generalization of modular structure, which we'll study in more detail in Lecture 12, is that of *hierarchical* community structure, in which vertices divide into groups that further subdivide into groups of groups, and so forth over multiple scales. This structure is often described using a *dendrogram* or tree that shows how the smallest groups are nested within larger groups, and they in turn are nested within still larger groups, etc. A common assumption for this kind of large-scale organization is that two vertices who are "closely related", that is, are separated by a short distance on the tree, are more likely to be connected. This is typical, for instance, of human social structures, for instance: our class sits within the larger Computer Science department, which itself sits inside the larger College of Engineering, which sits inside CU, which sits inside Colorado schools, etc. This kind of hierarchical structure is distinct from the large-scale structure represented by a military hierarchy or "org chart", which describes the hierarchical flow of control or information.

A third kind of large-scale structure is a *core-periphery* structure, in which a dense core is surrounded by a more diffuse, weakly interconnected periphery. This notion has much in common with traditional measures of centrality, which can be viewed as proxy measurements for identifying "core" vertices.

2 Modular or "community" structures

Most of the past work on studying modular structure has been in two directions. The first and much larger is in developing algorithms for identifying the presence of such structures from interactions alone. The second direction is in understanding the impact that the presence of modular structures in a network has on the dynamics of certain processes such as the diffusion of information or a disease across a network. In both areas, modules or communities are typically defined vaguely as large-ish groups of vertices that are more densely connected to each other than they are to the rest of the network. Different researchers often formalize this loose notion in different ways, which can make results and algorithms difficult to compare. Here, we'll only give a brief summary of a few of the methods for identifying modular structures in a network.

2.1 Network partitions

Most community detection methods can be thought of as trying to solve the following problem. First, assume some *score function* f that takes as input a network and a *partition* of its vertices (defined below) and returns a scalar value. Now, find the partition P that maximizes f. The idea is that f encodes our beliefs about what makes a *good* partition with respect to representing modular structures. If f gives higher scores to partitions that place more edges within groups than between them, then maximizing f over all partitions yields the partition with the densest possible groups under f.

Network partitions are simply a division of a network into k non-empty groups (modules), such that every vertex v belongs to one and only one group (module). For a given value of k, the number of possible such partitions is given by the Stirling numbers of the second kind

$$S(n,k) = \left\{ \begin{array}{c} n \\ k \end{array} \right\} = \frac{1}{k!} \sum_{j=0}^{k} (-1)^j \binom{k}{j} (k-j)^n \ . \tag{1}$$

For instance, there are $S(4,2) = 7$ possible partitions of a network with $n = 4$ nodes (indexed as 1,2,3,4) partitioned into $k = 2$ groups:

$$\{1\}\{234\},\ \{2\}\{134\},\ \{3\}\{124\},\ \{4\}\{123\},\ \{12\}\{34\},\ \{13\}\{24\},\ \{14\}\{23\}\ .$$

The number of all possible partitions of a network with n vertices into k non-empty groups is given by the nth Bell number, defined as $B_n = \sum_{k=1}^{n} S(n,k)$, which grows super-exponentially with n.[1] That is, the universe of all possible partitions of even a moderately sized network is very very big and an exhaustive maximization of f is not typically feasible.

Instead, most algorithms use some kind of heuristic for estimating the maximum of f.

[1] The first 20 Bell numbers (starting at $n = 0$) are 1, 1, 2, 5, 15, 52, 203, 877, 4140, 21147, 115975, 678570, 4213597, 27644437, 190899322, 1382958545, 10480142147, 82864869804, 682076806159, 5832742205057.

2.2 The modularity score

One of the most popular and well-known choices of f is the *modularity function* Q, introduced by Mark Newman and Michelle Girvan in 2004, which is conventionally defined for simple, unweighted networks, but can be generalized easily to multi-edge, weighted graphs.

Let k_i denote the degree of vertex i, c_i denote the index of the group containing vertex i, and define a function $\delta(c_i, c_j) = 1$ if vertices i and j are placed in the same group (i.e., if $c_i = c_j$) and 0 otherwise, then the modularity score of a partition is defined as

$$Q = \frac{1}{2m} \sum_{ij} \left(A_{ij} - \frac{k_i k_j}{2m} \right) \delta(c_i, c_j) \ . \tag{2}$$

This summation ranges over all pairs of vertices, but only accumulates value when the pair i, j are in the same community. That is, it only counts the internal edges of a community identified by the given partition. Crucially, the inner term counts these internal edges but subtracts their expected number under a random graph with the same degree sequence (this is the familiar configuration model; given that two vertices have degrees k_i and k_j, the probability that they are connected under the configuration model is exactly $k_i k_j / 2m$).

Equivalently, and perhaps more intuitively, the modularity score can be rewritten as a summation over the structure of the modules themselves

$$Q = \sum_i \left[\frac{e_i}{m} - \left(\frac{d_i}{2m} \right)^2 \right] \ , \tag{3}$$

where now i indexes modules, e_i counts the number of edges within the ith module and d_i is the total degree of vertices in the ith module. Thus, the first term e_i/m measures the fraction of edges within module i and $(d_i/2m)^2$ is the expected fraction under the configuration model. A "good" partition—with Q closer to unity—represents groups with many more internal connections than expected at random; in contrast a "bad" partition—with Q closer to zero—represents groups with no more internal connections that we expect at random. Thus, Q measures the cumulative deviation of the internal densities of the identified modules relative to a simple random graph model.[2]

To give you a feeling for how the modularity function works, consider two partitions of the same small modular network with $m = 7$ edges shown in Figure 1. It's useful to construct a 2×2 matrix M based on each partition, where M_{ij} counts the number of "half-edges" that go from all vertices in module i to all vertices in module j:

[2]It should be pointed out that a high value of Q only indicates the presence of significant deviations from the null model of a random graph. A large deviation could mean two things: (i) the network exhibits genuine modular structure in an otherwise random graph or (ii) the network exhibits other non-random structural patterns that are not predicted by a random graph. See Section 2.5 for some brief discussion of what this means for practical applications.

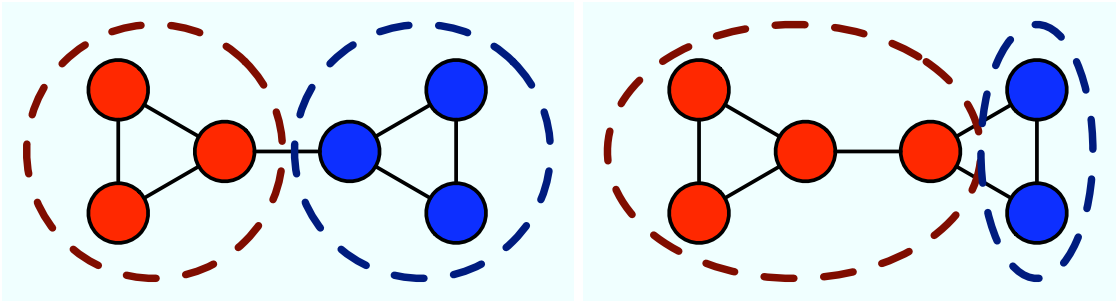

Figure 1: Two partitions of a simple network with modular structure. The modularity score of the first partition is $Q_{\text{good}} = 0.357$, while the score of the second is $Q_{\text{bad}} = 0.122$.

M_{good}	red	blue
red	6	1
blue	1	6

M_{bad}	red	blue
red	8	2
blue	2	2

The diagonals of M now give $2e_i$, the ith column (or row) sum gives d_i, and the matrix sum gives $2m$.[3] Reading values from these matrices, the first partition has $e_i = 3$ and $d_i = 7$ (for both modules), which yields $Q = 5/14 = 0.357$; the second has $e_1 = 4$ and $d_1 = 10$ while $e_2 = 1$ and $d_2 = 4$, which yields $Q = 6/49 = 0.122$. Clearly, the modularity function does a good job in this case of giving a higher score to the partition that yields more internally dense groups.

Recall, however, that there are B_n possible partitions of a network with n vertices (in this case, $B_6 = 203$, which is not so large), which means that we must typically search non-exhaustively over the space of all partitions to find those that yield high modularity scores. In general, it's known that maximizing Q is NP-hard but many heuristics seem to work well in practice. Some of these include greedy agglomeration, mathematical programming (linear and quadratic), spectral methods, extremal optimization, simulated annealing and various kinds of sampling techniques like MCMC. We won't cover any of them in detail; if you're interested, I can point you toward several recent review articles on them.

Many of these algorithms run quite slowly on real-world networks, taking time $O(n^2)$ or slower; a few run very quickly, for instance, taking time $O(n \log^2 n)$ and can thus be applied to very large networks of millions or more vertices. Because they are heuristics, of course, faster algorithms must make more aggressive assumptions about how to search the partition space and thus are less likely to find the globally optimum partition.

[3] Counting half-edges makes the math a little simpler but introduces these extra factors of 2. Equivalently, we could count only the full connections from vertices of type i to vertices of type j, which is the same as the above definition of M except that all the diagonal elements are half as big.

2.3 The stochastic block model

Another very popular and well-known choice of f is the likelihood function of the *stochastic block model* (SBM), which was first studied in mathematical sociology by Holland, Laskey and Leinhardt in 1983 and by Wang and Wong in 1987. Conventionally, this model is defined for simple unweighted networks, but can be generalized to directed networks. Adding weights requires adding an auxiliary model.

Unlike the modularity function, the stochastic block model is a parametric model that generates network structures; thus, the inference problem is to identify the choices of these parameters that maximizes the likelihood of generating the observed network data. Specifically, the SBM takes as input (i) a $n \times 1$ vector $\vec{c}$ where c_ℓ [or $c(\ell)$] gives the group index of vertex ℓ, and (ii) a $k \times k$ connectivity matrix M where M_{ij} gives the probability that a vertex of type i is connected to a vertex of type j. Thus, k denotes the number of groups or modules in the network.

Given c and M, we can draw an instance of the SBM model by flipping a coin for each pair of vertices u, v and letting the edge exist with probability given by $M_{c(u),c(v)}$. Thus, edges are independent but not identically distributed; edges *are* iid for a given pairing of i, j types. If $M_{ij} = p$ for all i, j, then the SBM reduces to the Erdős-Rényi random graph model $G(n, p)$. Otherwise, the SBM generates Erdős-Rényi random graphs within a community i and random bipartite graphs between pairs of communities i and j.

We can use this model to do inference if we are given some observed network structure G and ask what choices of model parameters c and M would maximize the likelihood of observing G? Let E_{ij} denote the number of observed connections from all vertices of type i to all vertices of type j and let N_i denote the total number of vertices of type i. The likelihood of observing exactly the edges in G is simply the likelihood of observing exactly the E_{ij} for each pair of groups i, j, and none of the other edges:

$$\mathcal{L}(G \,|\, M, c) = \prod_{i=1}^{k} \prod_{j=1}^{k} M_{ij}^{E_{ij}} (1 - M_{ij})^{N_i N_j - E_{ij}} \quad , \tag{4}$$

except for $i = j$, in which case the number of possible edges is not $N_i N_j$ but $\binom{N_i}{2}$. We can simplify the inference step a little by using a maximum likelihood estimator for the binomial distribution, $\hat{M}_{ij} = E_{ij}/N_i N_j$ [again, except when $i = j$, in which case the denominator is $\binom{N_i}{2}$], which allows us to derive M given a particular partition of vertices into groups c.

Returning to the example given in Figure 1, using the binomial MLE given above, we can tabulate the matrices M for each partition:

M_{good}	red	blue
red	3/3	1/9
blue	1/9	3/3

M_{bad}	red	blue
red	4/6	2/8
blue	2/8	1/1

The corresponding likelihoods under these SBMs are $\mathcal{L}_{\text{good}} = 0.001875\ldots$ and $\mathcal{L}_{\text{bad}} = 0.000244\ldots$, which means this graph structure is about 8 times more likely under the good partition than the bad one.

Note that in Eq. (4), the number of groups k is fixed. If we allow k to increase, then the size of M increases and when $k = n$, every vertex is in a group by itself, the matrix M becomes identical to the adjacency matrix A and the likelihood is maximized at $\mathcal{L} = 1$. This behavior occurs because k controls the model complexity (simplistically: the number of parameters) and larger models increase the likelihood of any data set of fixed size. Fortunately, because of the likelihood framework, we can choose k by using some kind of likelihood-based model selection technique to penalize larger models (larger k), e.g., a minimum-description length term.

As described above, the SBM decomposes a network into random bipartite graphs and Erdős-Rényi random graphs. Recall from Lecture 9, however, that such graphs have Poisson degree distributions. If fitted to a network with a heavy-tailed degree distribution, the SBM tends to favor partitions that place the highest degree vertices in a group together. This makes intuitive sense, but it also means that a network with modular structure *and* a heavy-tailed degree distribution cannot be well-modeled by an SBM—the SBM can model one or the other behavior, but not both. In a 2010 paper, Brian Karrer and Mark Newman introduced a generalization of the SBM that circumvents this problem, which is a very promising development.

2.4 Other approaches

There are, of course, many other approaches to identifying communities in networks. We won't cover any of them in detail. Instead, let's just mention two important classes.

The first is the *local* clustering algorithms. Note that the two approaches described above both require us to explicitly know the entire network structure. Thus, they require access to *global* information, e.g., the total number of edges in the network. Local techniques make no such assumptions and can be applied to networks like the WWW, which are too big to be fully known. Local methods often "crawl" outward from a particular starting point to identify the first "community" that encloses the starting vertex by looking for a boundary of vertices with relatively sparse connectivity to the rest of the network.

The second is the "soft" clustering algorithms for identifying *overlapping* communities, i.e., rather than sorting vertices such that they appear in one and only one cluster ("hard" clustering), vertices are allowed to belong to multiple communities, potentially with different strengths of association.

2.5 Three ill omens

Our first ill omen is the observation that despite a tremendous amount of activity on identifying clusters in networks, for the most part, we have no good explanations of what social, biological or technological processes should cause clusters to exist in the first place, or what functional role they play in those systems. (Presumably, these two things are related.) The best we have is a verbal argument due to Herbert Simon (the same Simon we met in Lecture 10) for why modular designs a good way to construct complex systems. The argument goes like this:

Suppose we are tasked with assembling many small, delicate mechanical parts into a beautiful Swiss watch, a masterpiece of complexity. And, suppose that we are interrupted every now and then, perhaps by the arrival of email or the need to have a snack, and than if we are interrupted before we have fully completed the assembly of a single watch, the partially assembled watch falls to pieces and we must start afresh after our break is over. If, on average, we experience one or more interruptions during the long process of assembling a watch, we will never produce very many watches. Now, however, consider the advantage of a modular design to the watch. In this case, the final watch is produced by combining smaller groups of parts or modules. If these modules are themselves stable products, then now we can tolerate interruptions and still produce watches because at worst, we only lose the work necessary to build a piece of the whole, rather than the whole itself. Thus, it seems entirely reasonable to expect that, in the face of "interruptions" of all different kinds, complex systems of all kinds will exhibit a modular or even hierarchical organization.

A second ill omen comes from recent observations that many of the popular global techniques for identifying modules in networks exhibit two bad behaviors: (i) *resolution limits* and (ii) *extreme degeneracies*. The first implies that the technique cannot detect dense clusters of vertices that are smaller than some preferred scale. In general, the mathematics showing the appearance of this behavior always points to the same root cause, which is the random-graph assumptions built into the techniques' score functions. The second implies that there are an exponential number of alternative, high-scoring partitions, which makes both finding the best partition computationally difficult (in some cases, NP-hard) and interpreting the particular structure of the best partition ambiguous. These two bad behaviors make it somewhat problematic apply network clustering techniques in contexts where it's not known *a priori* whether modules exist and what they should look like. For much more detail about these issues, see Good et al., "The performance of modularity maximization in practical contexts." *Physical Review E* **81**, 046106 (2010).

A third ill omen comes from the world of spatial clustering, a very old and still very active field. Many of the techniques for finding clusters in networks have antecedents in this domain. The general problem can be framed like so: we are handed a data set $\mathbf{x} \in \mathbb{R}^{\mathbf{n}}$ and our task is to identify the "natural" clusters of these data such that points in a cluster are closer (via some measure of distance) to each other than they are to all the other points.

However, in a 2002 paper titled "An Impossibility Theorem for Clustering", Jon Kleinberg showed, using an axiomatic approach, that no clustering function f can simultaneously satisfy three highly reasonably and practically banal criteria:

1. *Scale Invariance*: For any distance function d and any $\alpha > 0$, $f(d) = f(\alpha \cdot d)$. Intuitively, this requirement says that if we rescale all the distances between points by some constant factor α, the best clustering should stay the same.

2. *Richness*: Range(f) is equal to the set of all partitions of $\mathbf{x}$, which simply requires that all possible partitions of our data be potentially okay candidates.

3. *Consistency*: Let d and d' be two distance functions. If $f(d) = \Gamma$, and d' is a Γ-transformation of d, then $f(d') = \Gamma$. In other words, if a particular distance function d yields the clustering Γ, then if we make a new distance function d' that reduces all of the distances arising within clusters while increasing all those distances between clusters—that is, we make the clusters internally more dense and move them further away from each other—then we should get the same clustering Γ under d'.

Thus, any practical solution to identifying clusters in spatial data can, at best, only have two of these properties. One consequence of this fact is that we can categorize clustering techniques into broad categories, based on which single criteria they violate. It's unknown if a similar impossibility theorem exists for network clustering.[4]

[4]It is not hard to see that the second and third criteria have natural analogs in the context of network clustering. It's the first criteria that is problematic.