25 Community Detection

Finding relationships and communities from a social network has been a holy grail of the internet data explosion since the late 90s. Before Facebook, before MySpace. This information is important for targeted advertising, for identifying influential people, and for predicting trends before they happen.

At the most general, a social network is a large directed graph $G = (V, E)$. For 50+ years, psychologists and others studied small scale networks (100 friends, seniors in a high school). Anything larger was too hard to collect and work with.

Also mathematicians studied large graphs, famously as properties of random graphs. For instance, the Erdös-Rényi model assumed that each pair of vertices had an edge with probability $p \in (0, 1)$. As p increased as a function $|V|$, they could study properties of the connectedness of the graph.

Now there are active social networks (Facebook, Twitter, G+), but in many cases the full versions of these networks are closely guarded secrets. But examples are available through inventive information retrieval techniques. How can we extract useful information out of these? Which of these models are correct/useful? We have some answers, but many questions are still open.

Example Question: *Why do people join groups?*

Consider a group of people C that have tight connections (say in a social network). Consider two people X (Xavier) and Y (Yolonda). Who is more likely to join group C?

- X has three friends in C, all connected. (safety/trust in friends who know each other)
- Y has three friends in C, none connected. (independent support)

Answer: X Nodes form tightly connected subset of graphs.

25.1 Communities / Importance

Finding communities requires a notion of what we are looking for. One option is spectral clustering (and its relatives); see **L11**.

Another view of importance is PageRank; see **L22**.

But there are other approaches, and they require careful build up.

25.1.1 Preferential Attachment

What makes a person important or part of a group? A first thought is to look at the number of in-coming and out-going edges. But this alone can be deceiving, and easy to spoof / spam.

A better approach turns out to be looking at the number of triangles in the graph.

- If edge (A, B) and (A, C) exist in a graph, then it is more likely than random for (B, C) to exist.
- B and C somehow already trust each other (through A).
- A may have incentive to bring B and C together.
- All edges may result from common phenomenon (e.g. church group).
- If A has few triangles (compared to degree), more likely to be depressed (empirical study).

This has led to definitions of several new models on how to generate a random graph (as model for algorithms), better than Erdös-Rényi model.

25.1.2 Betweenness

The betweenness of an edge (A, B) is defined as

$$\mathsf{betw}(A, B) = \text{fraction of shortest paths that use edge } (A, B).$$

A large score may indicate an edge does *not* represent a central part of a community. If to get between two communities you need to take this edge, its betweenness score will be high. A "facilitator edge", but not a community edge.

The betweenness of a node is the number of shortest paths that go through this node.

How can we calculate $\mathsf{betw}(A, B)$?
Reduces to all pairs shortest path problem. For each $v \in V$: Build DFS on entire graph: this builds a DAG. Then walk back up the DAG and add a counter to each edge.

This is a bit tricky since from the leaves, each path has count 1. But when paths merge, the count adds as more edges are crossed. Some paths will split, then the count splits as well. This handles ties.

This takes $O(|V||E|)$. Its pretty slow (on large graph). Sampling approaches provide unsatisfactory results. The development of efficient, scalable and robust approaches remains an open question.

How does this find communities? Remove high-betweenness edges, and the remaining connected components are communities.

Thus high-betweenness edges are important for keeping a network connected.

25.1.3 Modularity

Given a group of nodes $C \subset V$ its modularity is basically:

$$Q(C) = (\text{fraction of edges in group}) - (\text{expected fraction of edges in group})$$

So the higher the more tightly packed the community is.

Lets define this a bit more precisely. The adjacency matrix has $A_{i,j}$ as 1 if edge (i, j) exists, and 0 otherwise.

The expected value of an edges for two nodes i and j with degree d_i and d_j, respectively, is $E_{i,j} = d_i d_j / 2|E|$. Note this allows self edges. Now formally,

$$Q(\mathcal{C}) = \frac{1}{4|E|}\left[\sum_{C \in \mathcal{C}} \sum_{i,j \in C} (A_{i,j} - E_{i,j})\right]$$

Note that $Q(\mathcal{C}) \in [-1/2, 1]$. It is positive if the number of edges exceeds expectation, based on degree. Typically when $Q \in [0.3, 0.7]$ this is a significant group.

So, how can we find a clustering $\mathcal{C}$ that has high modularity?
Spectral clustering: Finding leading Eigenvector, use this to find best split. If this split increases modularity, the accept it and recurse on both halves. Otherwise stop.

If too slow, can use PageRank repetition to estimate eigenvector.

Alternative: Start with all nodes in singleton cluster (with no other vertices).

- **Greedy Nibble**: Add one best node, or pair of nodes (like hierarchical agglomerative clustering).
- **Greedy Chomp**: Add (or subtract) all nodes which individually improve modularity.

Both will find local optimum, but the Chomp tends to work faster and find better local optimum.

25.1.4 Smaller Core Communities

Another option is not to divide up entire graph, but just find small groups of tight communities. These are *cliques*, groups of nodes where each pair has an edge.

To find cliques, use something like Apriori Algorithm. Start with all edges. For a node v with several edges, check each neighbor u of v to see if u has neighbor that is also in v's neighbor list. All t-cliques for $t \geq 3$ must contain 3-cliques (triangles). This will greatly narrow down the search space. Then this trick can be applied recursively to keep growing the existing cliques.