Networks - Week 3 - Community Detection and Katz Centrality for Dynamic Networks

Antonio León Villares

October 2023

Contents

Based on the notes by Renaud Lambiotte and Pete Grindrod, Chapters 4 & 5

1 Community Detection

1.1 Graph Partitioning

- **What is community structure within a network?**
  - when a **network** is composed of **intertwined** groups of **vertices**
  - these vertices are very **densely connected** within their own group (community)
  - for example, in **Stochastic Block Models** each block presents community structure
- **What is graph partitioning?**
  - process of splitting vertices into groups, minimising number of edges between different groups
  - given c groups, and n total nodes ,the time complexity of such algorithms is of

$$\mathcal{O}\left(n^{c^2}\right)$$

- **How are community detection and graph partitioning related?**
  - **community detection** seeks to split graphs into **densely connected groups**, which are **sparsely connected** amongst each other

1.2 Partitioning via Spectral Methods

1.2.1 Proposition: Approximate Bi-Partition of Vertices

*Let A be an **adjacency matrix** for an **undirected graph**.*
*Say we want to **partition** the graph into 2 **communities** (B_1 and B_{-1}).*
*Let $\underline{f}$ be the **Fiedler Eigenvector** corresponding to the **combinatorial Laplacian** of A. Then, an **approximately optimal** partition is given by:*

$$v_i \in B_{\text{sgn}(f_i)}$$

for any vertex v_i in the graph, and where sgn *is the **sign function**.*

Proof. For a vertex v_i, denote with s_i the community to which it belongs, such that:

$$v_i \in B_{s(i)}$$

If we do a bi-partition on our graph, the number R of edges between the two communities (known as the cut size) is given by:

$$R = \frac{1}{2} \sum_{i,j \mid s(i) \neq s(j)} A_{ij}$$

Alternatively, since:

$$\frac{1}{2}(1 - s_i s_j) = \begin{cases} 0, & v_i, v_j \text{ in the same community} \\ 1, & v_i, v_j \text{ in different communities} \end{cases}$$

we also have that:

$$R = \frac{1}{2} \sum_{i,j \mid s(i) \neq s(j)} A_{ij} = \frac{1}{2} \sum_{i,j} \frac{1}{2}(1 - s_i s_j) A_{ij} = \frac{1}{4} \sum_{i,j} (1 - s_i s_j) A_{ij}$$

If d_i denotes the degree of v_i, then:

$$\sum_{i,j} A_{ij} = \sum_{i,j} \delta_{i,j} d_i = \sum_{i,j} s_i s_j \delta_{i,j} d_i$$

since:

$$s_i s_j \delta_{i,j} = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}$$

so:

$$\begin{aligned} R &= \frac{1}{4} \sum_{i,j} (1 - s_i s_j) A_{ij} \\ &= \frac{1}{4} \left(\sum_{i,j} A_{ij} - \sum_{i,j} s_i s_j A_{ij} \right) \\ &= \frac{1}{4} \left(\sum_{i,j} s_i s_j \delta_{i,j} d_i - \sum_{i,j} s_i s_j A_{ij} \right) \\ &= \frac{1}{4} \sum_{i,j} s_i s_j (d_i \delta_{i,j} - A_{ij}) \end{aligned}$$

But now, we recognise that the ij entry of the combinatorial Laplacian of A is:

$$L_{ij} = d_i \delta_{i,j} - A_{ij}$$

so in fact we have the quadratic form:

$$R = \underline{s}^T L \underline{s}$$

The bipartition is given by the vector $\underline{s}$, which contains -1 or 1 (depending on the community). However, minimising $R = \underline{s}^T L \underline{s}$ subject to this constraint is hard. Instead, we relax our assumptions, and choose $\underline{s}$ to be some (normalised) real eigenvector. Then, we know that the $\underline{s}$ minimising the quadratic form is the eigenvector corresponding to the first non-zero eigenvalue of L; in other words, the fielder eigenvector $\underline{f}$. Moreover, since $\underline{f}$ is orthogonal to the eigenvector with 0 eigenvalue (which is the eigenvevctor full of 1s), it follows that choosing $s_i = f_i$ means that we have that:

$$\sum_{i=1}^{n} s_i = 0$$

Then, we partition v_i based on $\text{sgn}(f_i)$.

□

- **What are the advantages of the spectral method for bipartite community detection?**
  - it is relatively **simple** and **inexepnsive** (just need to compute an eigenvector)
  - as $n \to \infty$, the partition becomes more reliable
- **What are the disadvantages of the spectral method for bipartite community detection?**
  - it is only **approximate**
  - it only works for bipartite partitions

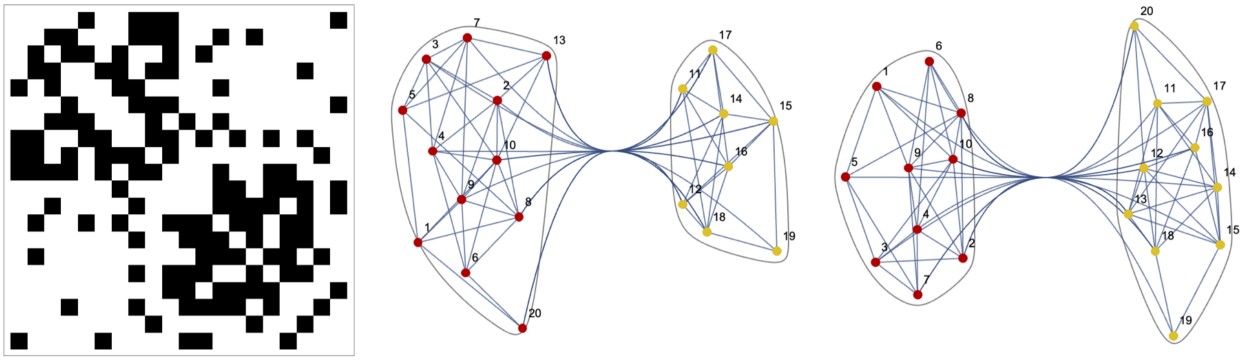

Figure 1: Example of applying the spectral method for a bipartite partition on a stochastic block model. To the right, the "correct" partition. The spectral method (centre) performs 2 missclassifications.

1.3 Partitioning via Modularity

1.3.1 Definition: Modularity

Modularity *is a* ***measure*** *of how good a partitioning is at identifying* ***communities****. In particular, it compares* ***edges within a community*** *with the* ***expected number of edges*** *in an appropriate null model.*

Modularity *is a* ***normalised sum*** *of a modularity measure for each community. For example, if we take the* ***configuration model*** *as our* ***null model****, the* ***probability*** *of an* ***edge*** *between* $v_i, v :_j$ *is:*

$$P_{ij} = \frac{d_i d_j}{2m}$$

where m *is the number of edges.*
If we have:

- n_{CM} *communities*
- *the* c*th community is* CM_C

then the ***modularity*** *is:*

$$Q = \frac{1}{2m} \sum_{c=1}^{n_{CM}} \left[\sum_{v_i, v_j \in CM_c} \left(A_{ij} - \underbrace{\frac{d_i d_j}{2m}}_{P_{ij}} \right) \right]$$

Equivalently, if we let g_i *denote the community of* v_i*:*

$$Q = \frac{1}{2m} \sum_{i,j} \left(A_{ij} - \underbrace{\frac{d_i d_j}{2m}}_{P_{ij}} \right) \delta(g_i, g_j)$$

where $\delta(g_i, g_j) = 1$ *iff* $g_i = g_j$*, and is 0 otherwise.*

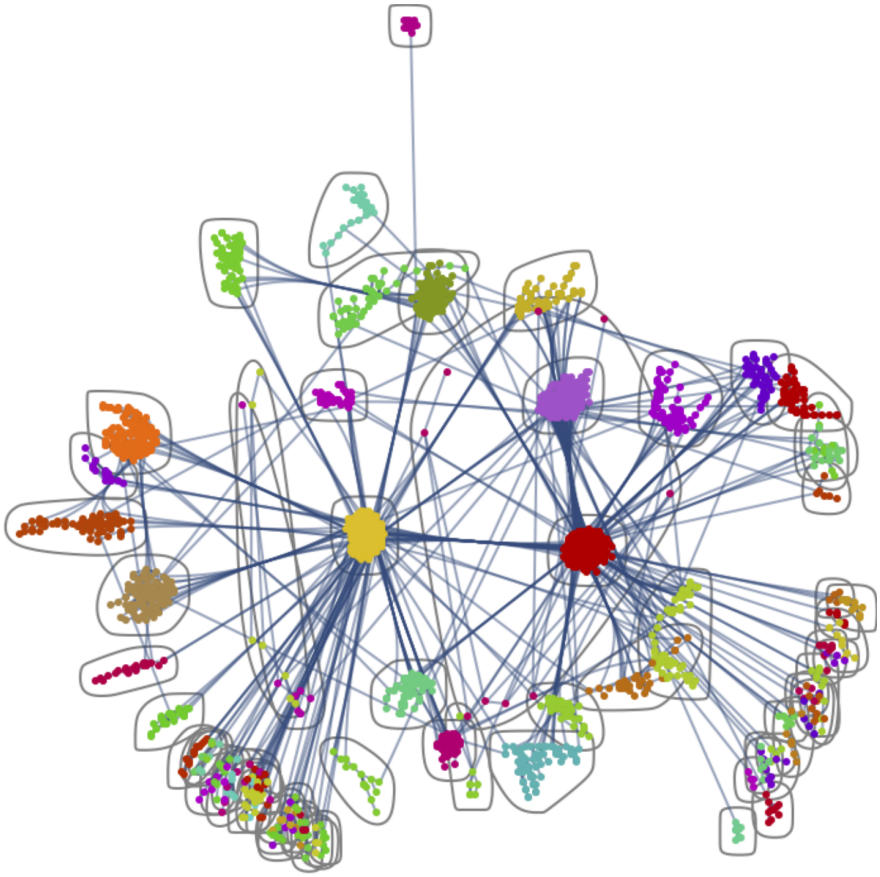

Figure 2: Twitter friendship network, partitioned into 74 communities by Mathematica's implementation of modularity maximisation.

Maximising modularity was the method used to derive the bipartition showcased above, which got 0 misclassifications.

- **What range of values does modularity take?**
  - **modularity** ranges in $[-0.5, 1]$
  - positive values indicate higher connectedness than expected by chance (i.e by the null model)
  - negative values indicate lower connectedness than expected by change (i.e a completely random model, or graphs with each node as a community)
- **When is modularity of a graph 0?**

– when the whole graph is taken as a single community (the **trivial partition**):

$$\begin{aligned} Q &= \frac{1}{2m}\sum_{i,j}\left(A_{ij} - \frac{d_i d_j}{2m}\right)\delta(g_i, g_j) \\ &= \frac{1}{2m}\left(2m - \frac{\sum_i^n d_i \sum_j^n d_j}{2m}\right) \\ &= \frac{1}{2m}\left(2m - \frac{(2m)(2m)}{2m}\right) \quad \textit{by Handshake Lemma} \\ &= 0 \end{aligned}$$

- **Is optimising modularity simple?**
  - it is an **NP-Hard** problem (c^n ways of producing a c-element partition)
  - in practice, **modularity** is optimised through **approximations**

1.4 Spectral Optimisation of Modularity

1.4.1 Definition: Modularity Matrix

*The **real symmetric matrix**:*

$$B_{ij} = A_{ij} - \frac{d_i d_j}{2m}$$

*is called the **modularity matrix**.*

- **How are the modularity and Laplacian matrices similar/dissimilar?**
  - *similar*: $(1, \ldots, 1)^T$ is an **eigenvector** of B with **eigenvalue** 0
  - *dissimilar*: the **eigenvalues** are can be **positive and negative** (in the Laplacian, all non-zero eigenvalues were positive)

1.4.2 Proposition: Modularity Matrix for Modularity Optimisation

*Let B be the **modularity matrix** corresponding to a **graph**. Then, the **sign** of the **components** in the **eigenvector** of B with the **largest positive eigenvalue** gives a near optimal **bipartition** of the graph.*

Notice, this strategy only seems to work for bipartite communities. In practice, the algorithm is applied recursively through each community. One stops partitioning when the modularity Q stops increasing.

Proof. As before, let s_i denote the partition to which vertex v_i belongs. We have that if g_i is the community of v_i:

$$\delta(g_1, g_j) = \frac{1}{2}(s_i s_j + 1$$

Hence:

$$\begin{aligned} Q &= \frac{1}{2m} \sum_{i,j} \left(A_{ij} - \frac{d_i d_j}{2m} \right) \delta(g_i, g_j) \\ \implies Q &= \frac{1}{4m} \sum_{i,j}^{n} \left(A_{ij} - \frac{d_i d_j}{2m} \right) (s_i s_j + 1) \\ &= \frac{1}{4m} \sum_{i,j}^{n} \left(A_{ij} - \frac{d_i d_j}{2m} \right) s_i s_j \end{aligned}$$

since we saw above that by the handshake lemma:

$$\frac{1}{2m} \sum_{i,j} \left(A_{ij} - \frac{d_i d_j}{2m} \right) = 0$$

But then, Q is nothing but a quadratic form in the modularity matrix:

$$Q = \frac{1}{4m} \underline{s}^T B \underline{s}$$

To maximise Q, note we can write $\underline{s}$ using the eigenvectors $\underline{u}_i$ of B (since they form an orthogonal basis):

$$\underline{s} = \sum_{i=1}^{n} a_i \underline{u}_i$$

where

$$a_i = \underline{u}_i^T \underline{s}$$

and we label the (real) eigenvalues in decreasing order:

$$\beta_1 \geq \beta_2 \geq \ldots \geq \beta_n$$

Hence, if β_i is the eigenvalue of B for $\underline{u}_i$, we have that the modularity is:

$$Q = \sum_{i=1}^{n} \beta_i a_i^2$$

Moreover, notice that by definition of $\underline{s}$, we have the normality constraint that:

$$n = \underline{s}^T \underline{s} = \sum_{i=1}^{n} a_i^2$$

Now, the optimal solution would solve this optimisation problem with the elemnts of $\underline{s}$ restricted to ± 1.

As before, we instead relax the problem, by choosing $\underline{s}$ to be a real vector. Then, we see from $Q = \sum_{i=1}^{n} \beta_i a_i^2$ that to maximise the sum, we want the component a_1 to be the largest (since β_1 is the largest eigenvalue). Hence, we want to pick $\underline{s}$ as close as possible (parallel) to $\underline{u}_1$. This can be achieved by setting:

$$s_i = \text{sgn}((\underline{u}_1)_i)$$

which will indeed satisfy the normality constraint.

□

- **What is the main issue with using spectral methods for modularity optimisation?**
  - need to estimate **eigenvectors** of the **modularity matrix**
  - this can be particulary expensive for graphs with more than 10^4 vertices

1.5 Definition: Louvain Method for Optimising Modularity

The Louvain method is a greedy, agglomerative algorithm, used to more cheaply compute communities in graphs.

*The **Louvain Method** generates a **hierarchical** sequence of **community partitions**. The hierarchy is generated **iteratively**, such that at each **pass** of the algorithm:*

- *the **average size** of the **communities***
- *the **modularity***

increases.

---

*Each **pass** consists of 2 steps:*

1. ***Local Optimisation**: for each vertex v_i, move v_i to the **community** which leads to the **maximal increase** in **modularity** (if no such community is found, v_i doesn't move). Repeat until no vertex moves. If there are n vertices, and n_c communities:*
   - *if $n > n_c$ (there's been vertex movement), proceed to next step*
   - *else, return the current partition*
2. ***Vertex Merge**: construct new graph with n_c **vertices** (corresponding with each of the **communities** found previously). **Edge weights** in new graph is the sum of edge weights between the vertices in each community node.*

- **How is the Louvain algorithm initialised?**
  - before the first pass, we start with **singleton communities**
  - each vertex constitues its own community (so the partition contains n communities)

1.6 Limitations of Modularity Optimisation

- **What are the 4 principal issues with modularity optimisation?**

1. **Overlapping Communities**: these tend to occur in **emprirical networks**, but won't be found by partitions
2. **Size Dependency**: size of graph influences the effectiveness of modularity:
   - **modularity-based emthods** favour communities of a certain size (which depends on the size of the network)
   - as $m \to \infty$ (number of edges), the **null model** is neglected (since it relies on a factor proportional to $1/2m$); instead, it uncovers the **connected components** of the network
   - the dependency of $1/2m$ means that dense node clusters might get neglected if smaller than a certain scale
3. **Marginal Gains**: as **high scoring partitions** are found, subsequent partitions will only be marginally better (due to the exponential number of possible high-scoring partitions which exist, one just jumps around amongst them)

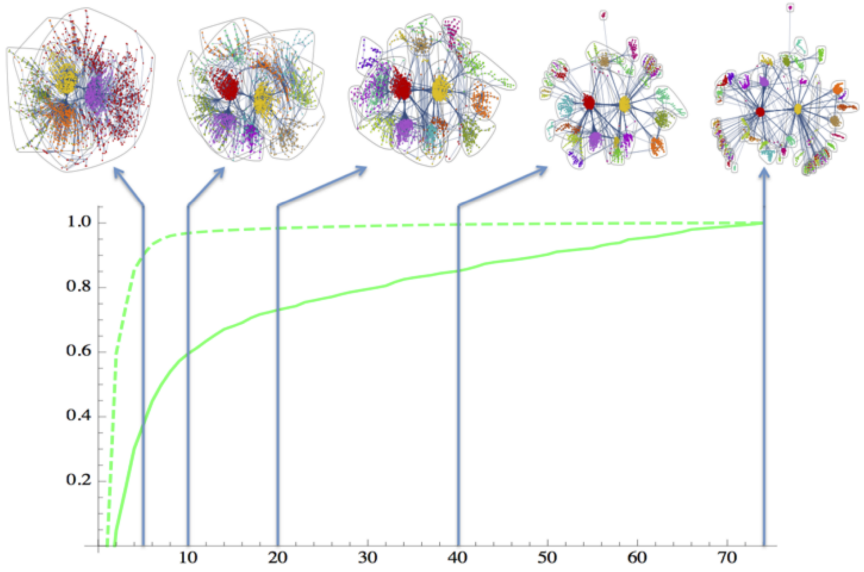

Figure 3: In dashed green, the **modularity**. In solid green, the entropy.

4. **Comparability**: a **modularity** score for a network can only be used to compare partitions of the same network, not partitions of different networks (for example, in a random network, the best partition can be found to have $Q \approx 1$, even if there is no modularity)

2 Katz Centrality for Observed Evolving Networks

2.1 The Katz Centrality Matrix

2.1.1 Definition: Evolving Network

*An **evolving network** consists of a sequence of **adjacency matrices***

$$\{A_k\}_{k\in[1,K}$$

with n vertices.
*Each A_k represents an **undirected graph**, without loops or double edges).*
At step t_k, the network is equal to A_k.

2.1.2 Definition: Dynamic Walk

*A **dynamic walk** of length m from vertex i_1 to vertex i_{m+1} consists of:*

- *a **sequence of edges***

$$(i_1, i_2), \ldots, (i_m, i_{m+1})$$

- *a sequence of **non-decreasing times***

$$t_{r_1} \leq t_{r_2} \leq \ldots \leq t_{r_m}$$

such that in an evolving network , the edge $(i_j, i_{j+1}$ is traversed at time t_{r_j}.
In other words:

$$(A_{r_j})_{i_j i_{j+1}} = 1$$

2.1.3 Definition: Katz Centrality Matrix

The Katz Centrality Matrix generalises the notion of Katz centrality for dynamic networks.

*Let $\{A_k\}$ be an **evolving network**. Its **Katz Centrality Matrix** is given by:*

$$Q = \prod_{i=1}^{K} (I - \alpha A_i)^{-1}$$

2.1.4 Proposition: Properties of the Katz Centrality Matrix

1. *Entry ij of Q counts **all** possible walks of **all** possibe combinations of **edges** taken from successive time steps that can be taken across time in the evolving network.*
2. *Q is only defined when:*

$$\forall k \in [1, K], \quad \alpha < \frac{1}{\rho(A_k)}$$

3. *Q is generally non-symmetric*

Proof.

(1)

The matrix product

$$\prod_{i=1}^{m} A_{r_i}$$

has entry ij counting the number of dynamic walks of length m from v_i to v_j, and with the jth step of the walk happening at t_{r_j}.

The matrix product:

$$\prod_{i=1}^{m} A_{r_i}^{m_i}$$

has entry ij counting the number of dynamical walks from v_i to v_j taking $m_k \geq 0$ edges in A_k at each time step t_k.

Hence, the product:

$$\prod_{i=1}^{\infty} \sum_{j=0} \alpha^j A_i^j$$

contains all possible walks of all possible combinations of edges taken from succesive time steps. Using the defintion of Katz Centrality, the result follows.

(2)

The Katz Centrality only converges when:

$$\alpha < \frac{1}{\rho(A_k)}$$

Hence, the product converges if and only if each of the product elements converges.

③

Since matrix multiplication isn't commutative, this matrix multiplication won't always yield a symmetric Katz Centrality Matrix (it depends on the order of the A_k).

□

2.1.5 Definition: Broadcast Centralities

*The **broadcast centralities** measure the outward influence of each vertex within the network, as a weighted sum of walks from each vertex to anywhere.*

---

*These are given by the **row sums** in the **Katz Centrality Matrix**.*

2.1.6 Definition: Receive Centralities

*The **receive centralities** measure the inwards influence of each vertex within the network, as a weighted sum of walks from anywhere to each vertex.*

---

*These are given by the **column sums** in the **Katz Centrality Matrix**.*