Instructor: E.Dimitrova
Name: Joseph McGuire Class: Math 599 (W2022)

Notes on Graph Theory

1 Fundamentals of Graph Theory

A graph is fundamentally about relations between objects in a finite set.

Definition 1.1. *A **relation** on a set S is a rule $R \subset S \times S$:*

$$(u, v) \in R \iff u \; R \; v.$$

Example 1.2. *Let $S = \mathbb{Z}$ and*

$$u \; R \; v \iff u \leq v.$$

Then

$$(1, 2) \in R \qquad (-100, 1) \in R \qquad (0, 1) \in R.$$

A graph then gives us a natural way of understanding relationships between objects in sets.

Definition 1.3 (**Simple Graph**). *Let V be a set and E be a set, where E (in the case of un-directed graphs) $E \subset \{\{v, w\} : v, w \in V\}$, then $G = (V, E)$ is a simple graph on V.*
Alternatively, we can also define $E \subset V \times V$ as a relation on V, where $(u, v) \in E$ if and only if $u \; E \; v$. Where there's an additional requirement of symmetry:

$$\text{If } (u, v) \in E, \text{ then } (v, u) \in E.$$

The first definition of E is notationally convenient, while the second definition will allow us to generalize the concept to the non-symmetric case.

Example 1.4. *In Figure 1 is a graph with node set $V = \{0, 1, 2\}$ and edge set*

$$E = \{\{0, 1\}, \{1, 2\}, \{0, 2\}\}$$

Simple graphs are useful in describing a variety of situations, because of their links to simple relations. We'll see in the next example one of the most well studied graphs in network theory is a simple graph.

Notes on Graph Theory

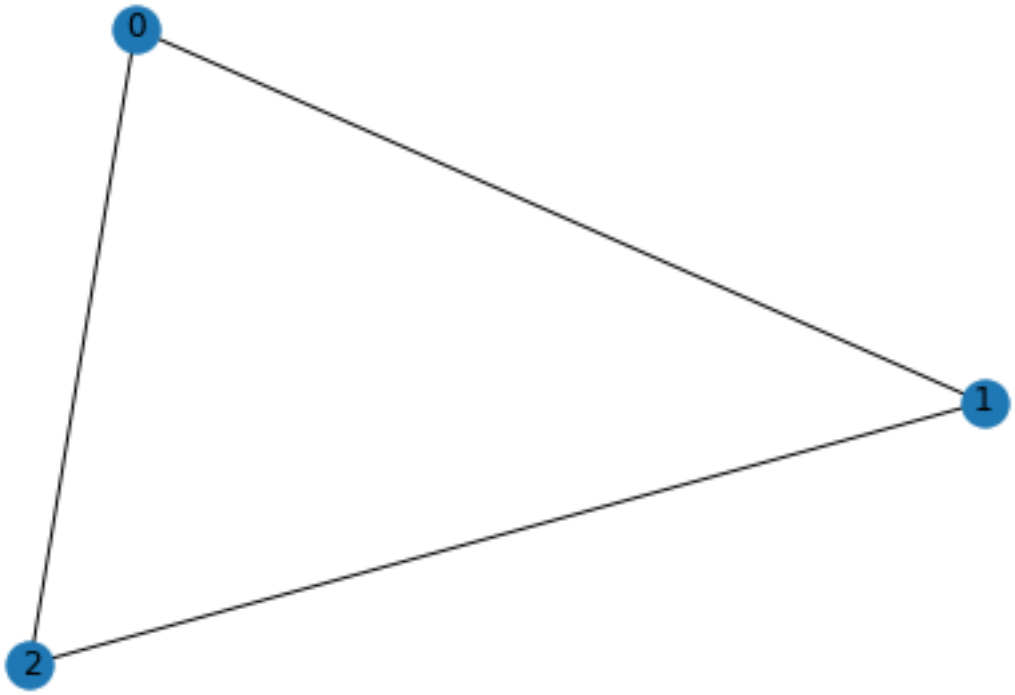

Figure 1: An undirected simple graph

Example 1.5. *A PhD student, Wayne Zachary, followed the interactions of the members of a Karate club over the course of 3 years (1970 - 1972). Over this span, there was a conflict in the club, resulting in two administrators 'Mr. Hi' and 'John A' (pseudonyms) split the group into two rival karate clubs (Think 'Karate Kid').*

The graph is created by treating each individual as a node, and then every edge is a social connection between two members of the club. The result being the graph in Figure 2.

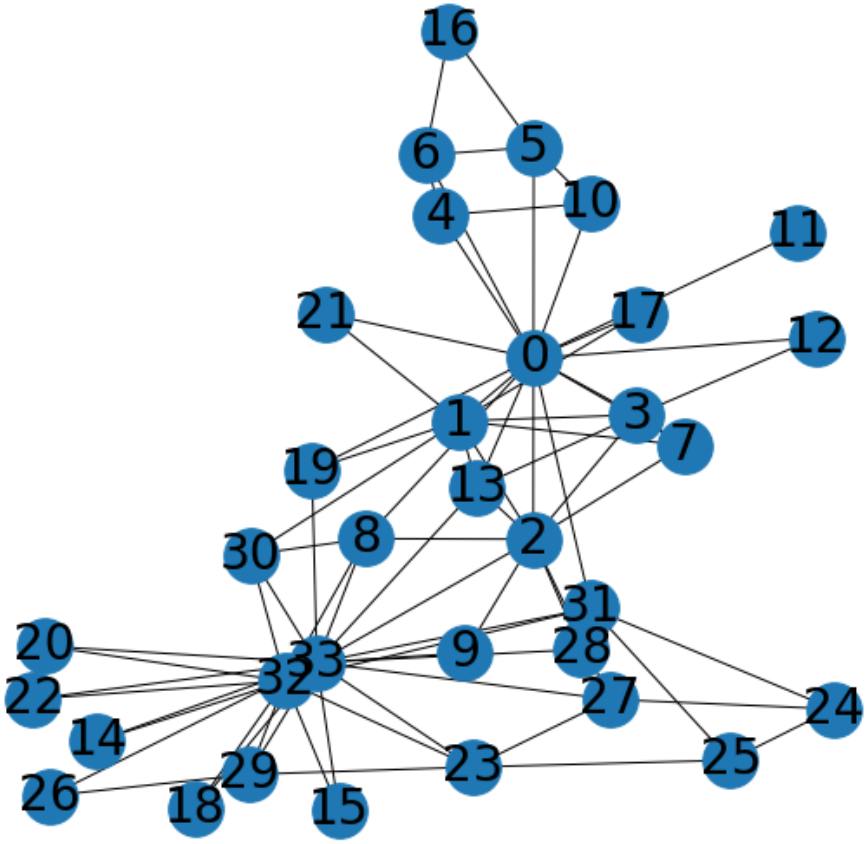

Figure 2: The Zachary Karate Club Network

A useful thing to know in a network is what are the total amount of connections a single node has. In Example 1.5, the number of connections that a node has tells us how connected that the node is, and thus how connected the person that node is representing. Formally, this is the notion of the degree of a node. And as with most things in mathematics, there is

a natural and elegant link from this to linear algebra.

Definition 1.6 (**Degree**). *Let $G = (V, E)$ be a graph. If $v \in V$, then:*

$$\deg(v) = \#\ \textit{of edges connected to } v$$

*is the **degree** of node v.*

Definition 1.7 (**Degree Sequence**). *Let $G = (V, E)$ be a graph with $|V| = n$. Then define the **degree sequence** to be the decreasing sequence:*

$$(\deg(v_1), \deg(v_2), \ldots, \deg(v_n)) \qquad \textit{with } \deg(v_i) \leq \deg(v_{i+1})\ \forall i.$$

Note: The labeling of $v_1, \ldots, v_n$ maybe done arbitrarily to fit the inequality requirement above.

Example 1.8. *In example 1.4:*

$$\deg(0) = \deg(1) = \deg(2) = 2.$$

In the Zachary Karate Club example (1.5),

$$\deg(16) = \deg(15) = \deg(20) = \deg(22) = 2 \qquad \deg(11) = 1.$$

Definition 1.9 (**Adjacency Matrix**). *Let $G = (V, E)$ be a graph with $|V| = n \in \mathbb{Z}^+$. Then we define the $n \times n$ matrix A by:*

$$A_{ij} = \begin{cases} 1 & \textit{if } i \textit{ and } j \textit{ share an edge} \\ 0 & \textit{otherwise} \end{cases}$$

Notes on Graph Theory

Example 1.10. *The adjacency matrix for Example 1.4 will be:*

$$A = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

Exercise 1.11. *Determine the eigenvalues and eigenvectors of A in 1.10*

Theorem 1.12. *Let $G = (V, E)$ be a simple graph. If $i \in V$ is a node, then:*

$$\deg(i) = \sum_j A_{j,i}.$$

In plain words, the degree of node i is the column sum of column i in the adjacency matrix A.

Note 1.13. *An important thing to notice with the adjacency matrices of simple graphs are that they are always symmetric:*

$$A_{i,j} = A_{j,i}$$

which follows by Definition 1.9.

We'll explore more about the structure of these matrices in the homework.

If we relax the constraint that the adjacency matrix needs to be symmetric the result will be a 'directed' graph. Directed, in that now the relationship that an edge in E describes is no longer required to be reciprocal (i.e goes both ways).

Definition 1.14 (Directed Graph). *Let V be a set and $E \subset V \times V$. Then $G = (V, E)$ is a **directed graph** or **di-graph**.*

Usually to distinguish between simple and directed graphs, arrows are used for directed graphs. So the edge $(0, 1)$ would represented by the arrow starting at 0 and landing at 1.

Notes on Graph Theory

Definition 1.15 (**Degrees for Directed Graphs**). *Let $G = (V, E)$ be a di-graph. If $v \in V$, then:*

$$\deg_O(v) = \#\ \textit{of edges pointing out of } v \qquad deg_I(v) = \#\ \textit{of edges pointing in to } v,$$

*are the **out-degree** and **in-degree** of v, respectively.*

Note 1.16. *Notice there's no additional requirements on a directed graph as compared to a simple graph, the edge set can be strange and non-symmetric.*

Example 1.17. *The adjacency matrix of the graph given in 3 is:*

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

Exercise 1.18. *Find the eigenvectors and eigenvalues of the matrix in Example 1.17. What do you notice about your answers here compared to 1.11?*

In many real world applications edges are not created equally, for example, the connections in a friend group might be weaker or stronger based on the quality of that friendship. So to model this, we introduce a generalization of both a simple and directed graph.

Definition 1.19 (**Weighted Graph**). *Let V be a set and $E \subset \{(u, v, w) : u, v \in V, w \in \mathbb{R}\}$. Then*

$$G = (V, E) \textit{ is a weighted graph.}$$

*The value w for an edge $u \to v$ is called the **weight** of that edge pair.*

Note 1.20. *It will be useful to talk about graphs where edges like (v, v) are allowed, these are called graphs with **self-loops**.*

Notes on Graph Theory

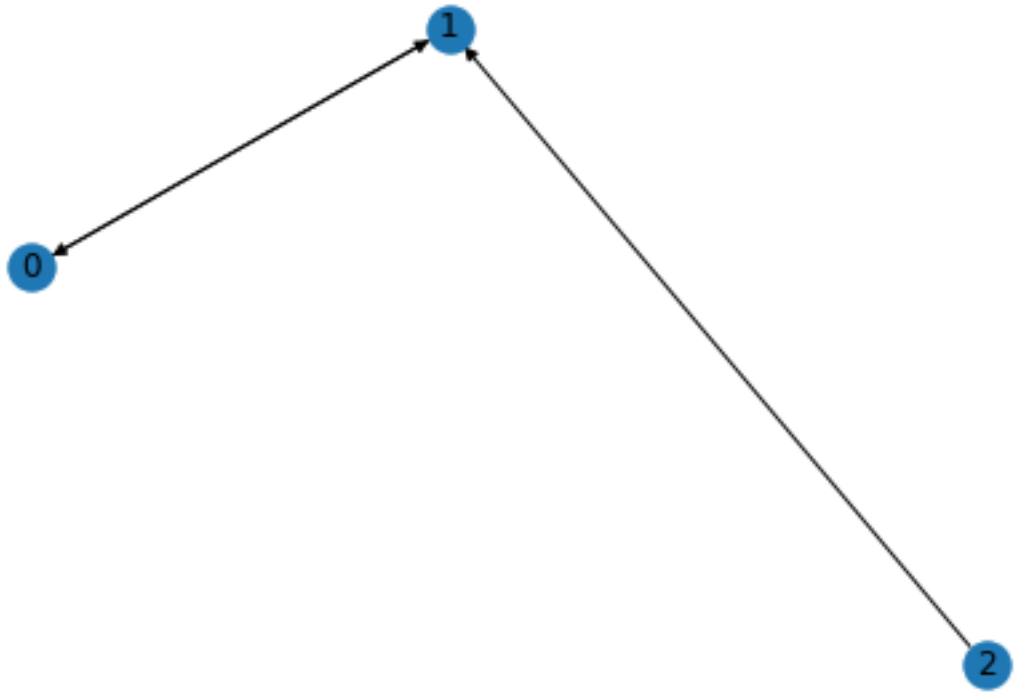

Figure 3: A randomly generated directed graph

Example 1.21. *Let $G = (V, E)$ be a weighted di-graph with self-loops, with adjacency matrix:*

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 9 & 0 & 0 \\ 3 & 0 & 2 & 3 \end{bmatrix}$$

The visualization of this in Figure 4.

Definition 1.22 (Walk). *On a graph $G = (V, E)$, a walk of length m on the vertices of G is a sequence:*

$$(v_1, \ldots, v_m)$$

such that

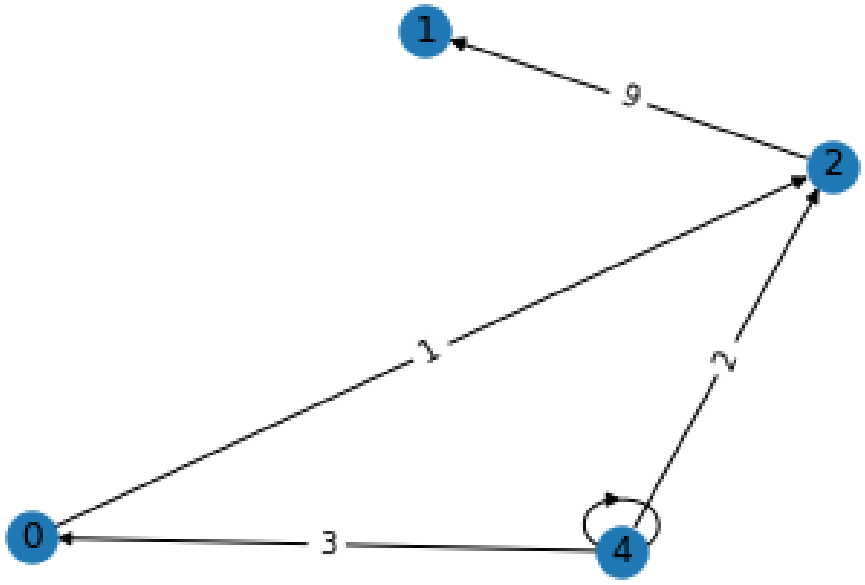

Figure 4: A weighted directed graph with self-loops

Definition 1.23 (**Connected**). *A graph G is **connected** if*

$$\textit{for all } u, v \in V \textit{ there exists a path between } u \textit{ and } v.$$

*A graph that isn't connected is **disconnected**; that is, there are two nodes $u, v \in V$ such that there is not path between them.*

Definition 1.24 (**Multi-Graph**). *Let V be a set, then G is a **multi-graph** if E is a multi-set (e.g. a multi-set: $\{(1,2),(1,2),(1,3),(3,1),(3,1)\}$).*

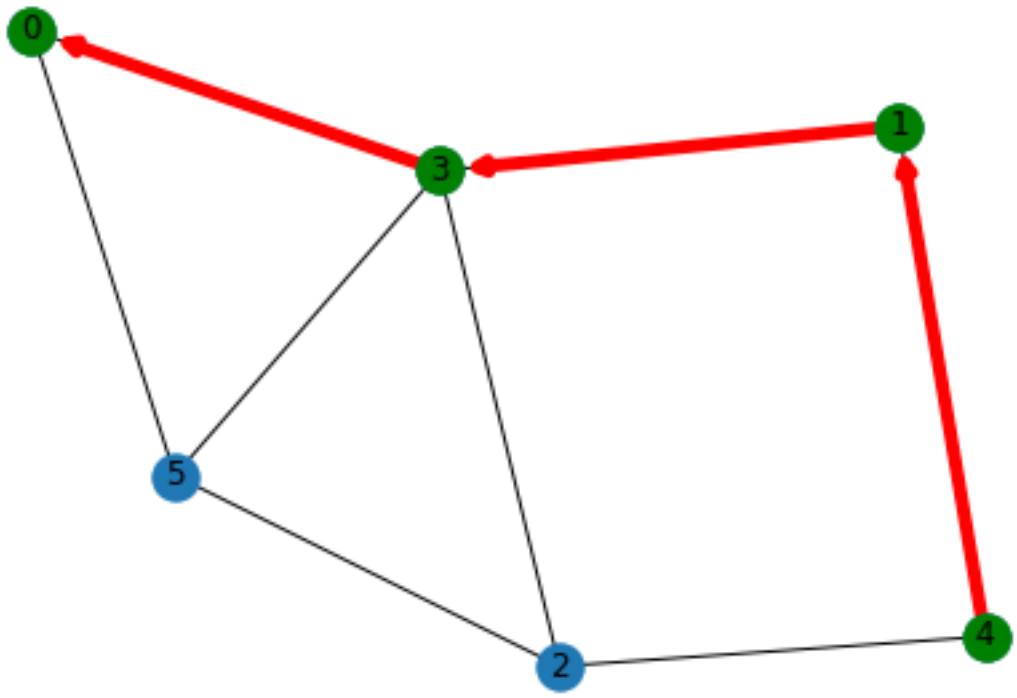

Figure 5: A path of $(4, 1, 3, 0)$, visited nodes are colored green and traversed edges are red and thicker than other edges.

Figure 6: An example of disconnected graph

Definition 1.25 (**Bipartite Graph**). *Let U, V be distinct sets such that $U \cap V = \emptyset$, then G is a* ***bipartite graph*** *if $E \subset U \times V$.*

That is, any edge in E must go between a member of U and V, it can't go to two members of U or two members of V. Example in 8

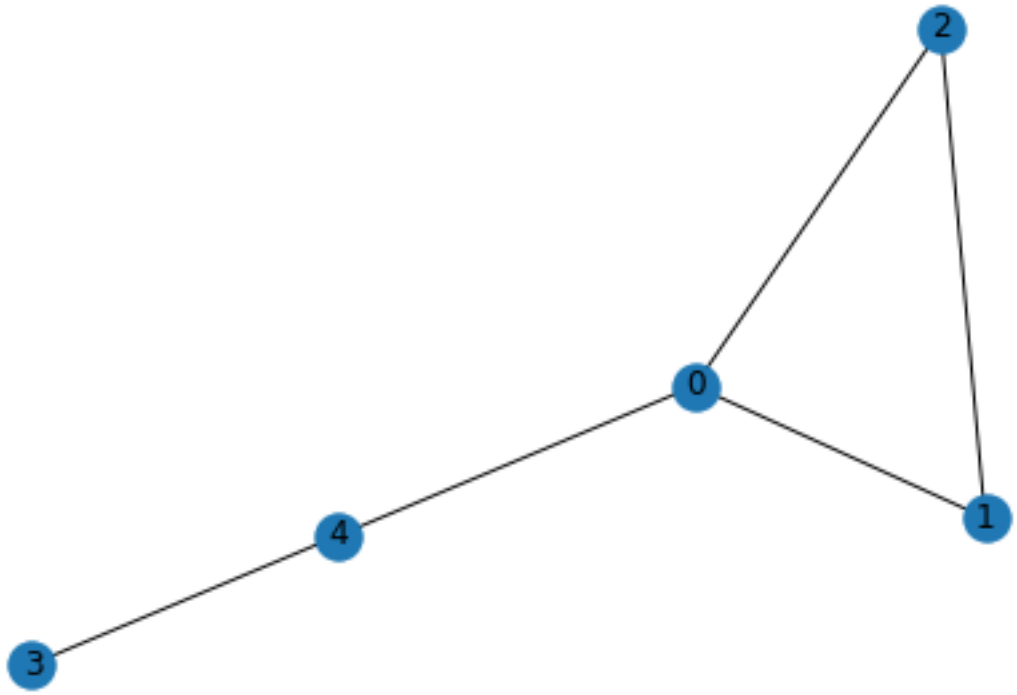

Figure 7: An example of a connected graph

2 Stochastic Block Models(SBMs)

Definition 2.1. *Let V be a set, $C = \{C_1, \ldots, C_r\}$ be a partition of V, an element of C is called a* ***community****, and P be $r \times r$ matrix such that*

$$P_{i,j} \in [0,1] \qquad P_{i,j} = P_{j,i} \qquad \forall i, j \in \{1, \ldots, r\}.$$

Then $S = (V, C, P)$ is a ***Stochastic Block Model****.*

From S, we generate a simple graph $G = (V, E)$ by generating E with our matrix P. For all vertices $v \in C_i$ and $w \in C_j$, there is a probability of $P_{i,j}$ that the edge $(v, w) \in E$.

- *In the case where $i = j$, $P_{i,i}$ determines the internal connections of the community C_i; all of the edges that are between two members of the same community.*

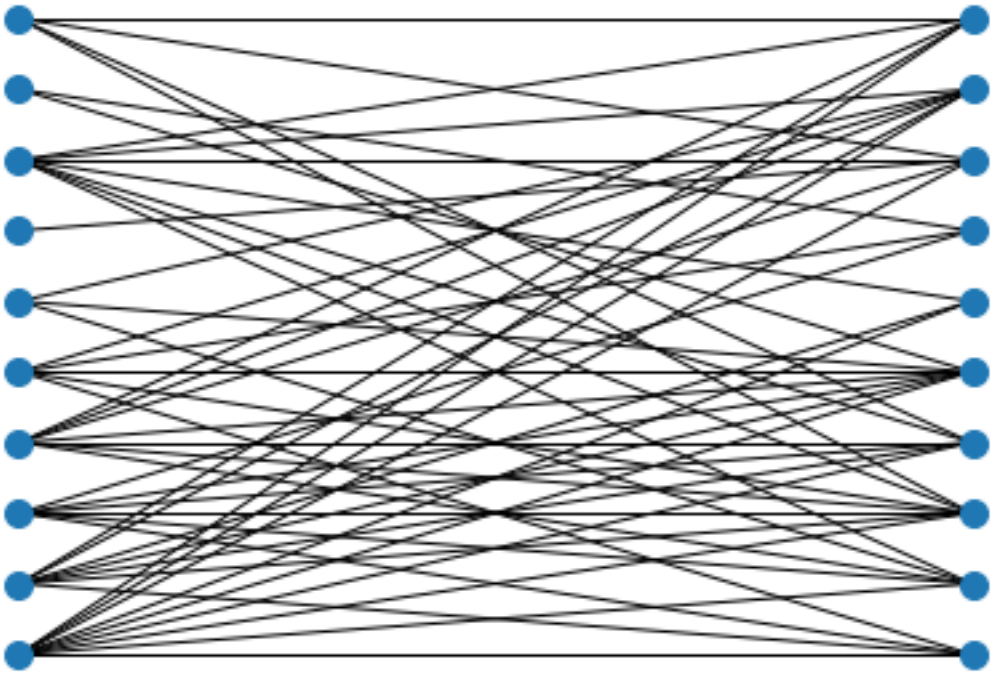

Figure 8: An example of a bipartite graph

Stochastic Block Models(SBMs)

- *In the case where $i \neq j$, $P_{i,j}$ determines the external connections between communities C_i and C_j; the connections joining the two communities.*

We can allow $P_{i,j} \neq P_{j,i}$ for some $i, j \in \{1, \ldots, r\}$, however we must drop the requirement that the generated graph will be simple. So we'll be sticking this requirement for this class, to see more see project on community detection in di-graphs.

Stochastic Block Models(SBMs)

Example 2.2.

- *Figure 12 is a graph generated by the stochastic block model:*

$$V = \{0, \dots, 29\} \qquad C_1 = \{0, \dots, 9\}, C_2 = \{10, \dots, 19\}, C_3 = \{20, \dots, 29\}$$

and

$$P_{i,j} = 0.5 \qquad \text{for all } i, j \in V.$$

This is an example of an ***Erdos-Reyni Model****, these satisfy:*

$$P_{i,j} = p \qquad \text{for all } i, j \in V.$$

Erdos-Reyni models are one model used to generate random graphs, which act as a probability distribution over graphs.

- *Figure 10 has a graph generated by the stochastic block model:*

$$V = \{0, \dots, 29\} \qquad C_1 = \{0, \dots, 9\}, C_2 = \{10, \dots, 19\}, C_3 = \{20, \dots, 29\}$$

and

$$P_{i,j} = 0.75 \ (i = j), \qquad P_{i,j} = 0.05 \ (i \neq j)$$

This is an example of a ***Planted Partition Model****. These SBM's satisfy:*

$$P_{i,j} = \begin{cases} p & \text{if } i = j \\ q & \text{if } i \neq j \end{cases} \qquad \text{for some values } p, q \in [0, 1].$$

Note the very strong 'clustering' that we're seeing in this graph, this is happening because our $p >> q$.

Example 2.3. *Returning to the example of the famous 'Zachary Karate club'. In this example, we see natural communities arising in the group.*

Namely the communities centered around 'Mr. Hi' (Node 0) and the communities centered around 'John A' (Node 33). Colored in Figure 11

In a variety of social, biological, and physical situations identifying these communities has significant implications.

- In sociology, there is the concept of *homophily*, which is the phenomena of people with similar interests/roles/gender/age tend to establish connections. Meaning that many social networks will naturally exhibit community sturcture.

Stochastic Block Models(SBMs)

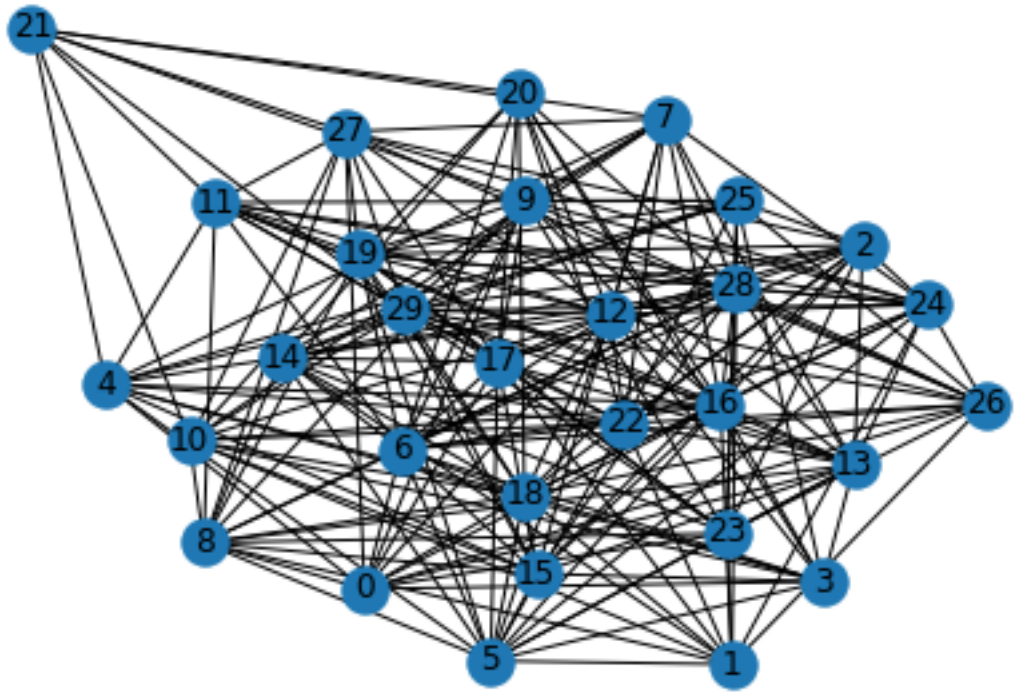

Figure 9: A graph generated by an Erdos-Reyni model

- In social networks, determining communities can provide a powerful tool for marketing and recommendation algorithms; people in the same community are more likely to be friends or like the same things [**?**]
- In neuroscience, community structure in brain networks are being investigated for treatment of epilepsy [**?**]. As well as variations in community structure informing neuroimaging. [**?**]
- In biology, community structure in protein-protein interaction networks help reveal potential drug discoveries and treatment plan [**?**]

Definition 2.4 (**Community Detection**). *Let $G = (V, E)$ be a graph. Assume G was generated by a Stochastic Block Model, $S = (V, P, C)$.*
Then ***community detection*** *is the task of recovering the partition C.*

Stochastic Block Models(SBMs)

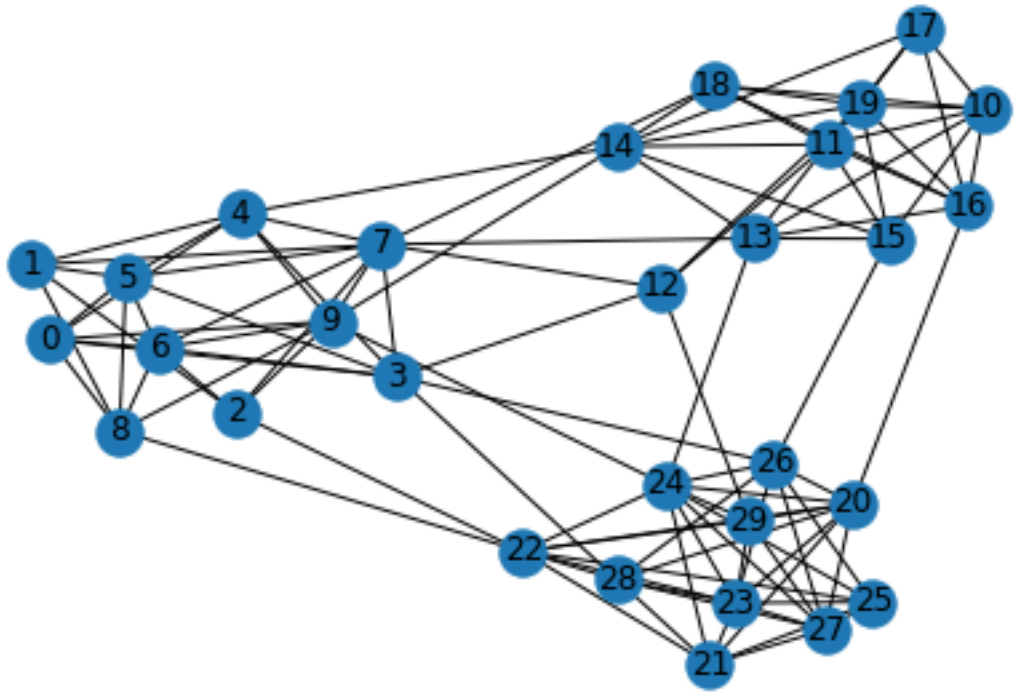

Figure 10: A graph generated by a planted partition model

Note 2.5. *As we'll discuss community detection in this class, our goal will be to find a partition C that gives the strongest possible clustering behavior such as we see in the Zachary Karate Club example (Figure 11) or as in the planted partition model (Figure 10).*

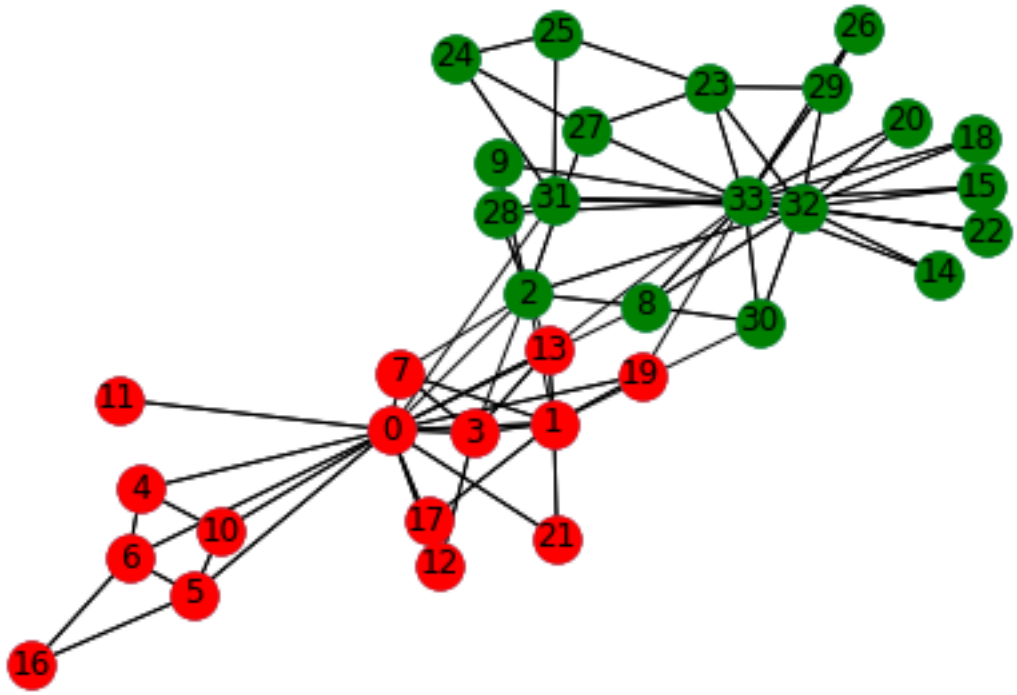

Figure 11: The Zachary Karate Club communities

3 Background

Before we can start talking about community detection, we need a way to actually define the 'clustering' behavior in graphs. (We'll restrict our discussion to undirected and unweighted graph.)

This '*clustering*' behavior is strongest when given a graph $G = (V, E)$ and a partition C of V, if $C_i, C_j \in C$ with $v \in C_i$ and $w \in C_j$ then:

- If $i = j$, then w and v are likely to be connected by an edge;
- If $i \neq j$, then w and v are not likely to be connected by an edge.

So to formalize this, we'll compare this to random **null model** assumption.

Assumption 3.1. *A randomly and uniformly generated graph has no community structure. (i.e a graph generated by an Erdos-Reyni model)*

Community Structure

Definition 3.2 (**Null Model**). *A **null model** for a simple graph $G = (V, E)$, with $|V| = n$ and $|E| = m$, is a multi-graph $G' = (V, E')$, defined by:*

1. *For each edge, $(v, w) \in E$, cut the edge in half, so we now have $2m$ 'dangling' **stubs**.*
2. *Reconnect the stubs by randomly and uniformly pairing two stubs together and joining the stubs to form an edge, store the edge in E'.*

*G' is the **null model** of G.*

Theorem 3.3. *Let $G' = (V, E')$ be the null model of $G = (V, E)$. Then G' and G have the same degree sequence and*

$$|E| = |E'|.$$

Proof. Let $G = (V, E)$ with v being a node in G. We'll show the result by going through the construction of G'.

Suppose v has degree k, there are k edges touching v. Now create the stubs attached to v, there will be precisely k stubs attached to v. Take a stub from v, there are $2|E| - 1$ choices of other stubs to attach to (the stub can't connect to itself). If the stub attaches to another stub of v, then this creates a self loop and that is a connection. If the stub attaches to stud not attached to v, then this is a connection for v. We do this k times for all nodes $v \in V$ and we have $G' = (V, E')$ the null model for G. By construction, the degree of v has not changed and since this is true for any $v \in V$, the degree sequence of G is the degree sequence of G' □

The intuition is that a graph generated by a Erdos-Reyni model has no clustering behavior: the entire graph is one big cluster. So this null model acts a null hypothesis for our community testing.

To summarize the other assumptions we've made so far:

Assumption 3.4.

1. *Graphs with community structure are generated by a stochastic block model*
2. *Two nodes are much more likely to be connected inside a community than out*

And with Assumption 3.1:

Community Structure

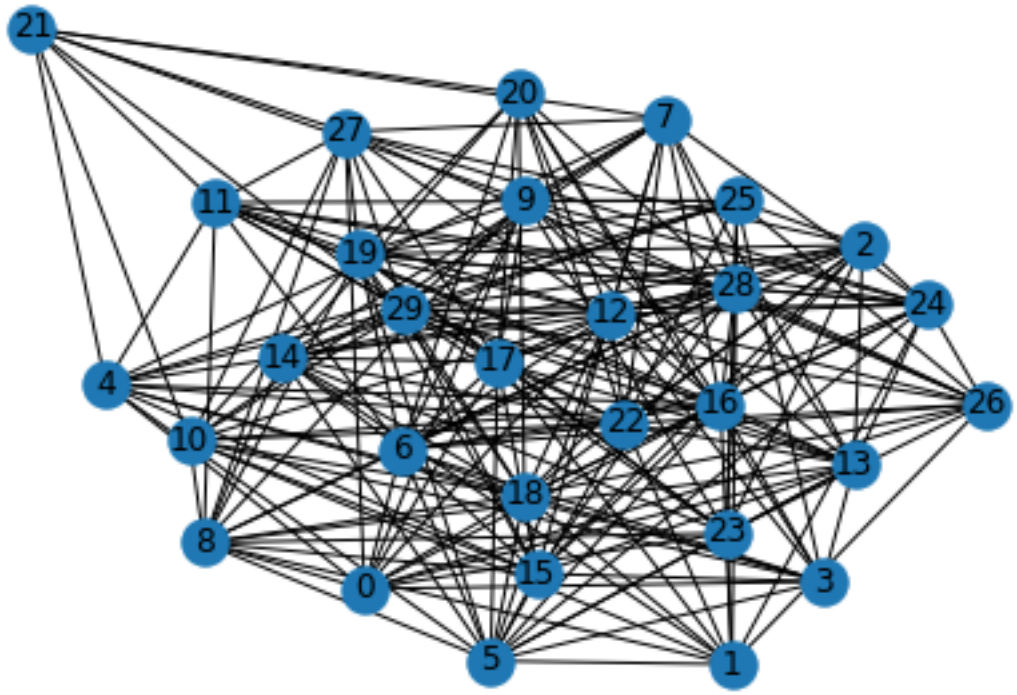

Figure 12: A graph generated by an Erdos-Reyni model has no community structure.

3. A null model of a graph has no community structure.

Lemma 3.5. *Let $G' = (V, E')$ is the null model of $G = (V, E)$, where $|E| = |E'| = m$, v, w be a vertex in G'. Define $I_i^{(v,w)} = 1$ if v and w are paired by a stub $i = 1, \ldots, k_v$. Then:*

$$p(I_i^{(v,w)} = 1) = \frac{k_w}{2m - 1}$$

Proof. Let $G' = (V, E')$ is the null model of $G = (V, E)$, where $|E| = |E'| = m$, and v, w be a vertex in G'. Define $I_i^{(v,w)} = 1$ if v and w are paired by a stub $i = 1, \ldots, k_v$.

For the stubs starting at v, there are a total of $2m - 1$ choices, since stubs can't connect to themselves but they can connect to any other stub in the network. These number $2m$ by their construction in the null model.

Out of those choices there are k_w many stubs that will connect v to w and hence make

$I_i^{(v,w)} = 1$. Thus:

$$p(I_i^{(v,w)} = 1) = \frac{k_w}{2m-1}.$$

□

Theorem 3.6. *Let $G' = (V, E')$ is the null model of $G = (V, E)$, where $|E| = |E'| = m$, v, w be a vertex in G'.*

$$\mathbf{E}[\ \#\ \textit{of edges between } v \textit{ and } w\ ] = \frac{k_w k_v}{2m-1}.$$

Proof. Let $G' = (V, E')$ is the null model of $G = (V, E)$, where $|E| = |E'| = m$, v, w be a vertex in G'. Note that:

$$\#\text{ of edges between v and w} = \sum_{i=1}^{k_w} I_i^{(v,w)}.$$

So that:

$$\mathbf{E}[\ \#\text{ of edges between v and w }] = \mathbf{E}\left[\sum_{i=1}^{k_v} I_i^{(v,w)}\right] = \sum_{i=1}^{k_w} \mathbf{E}[I_i^{(v,w)}] = \sum_{i=1}^{k_w} \frac{k_v}{2m-1} = \frac{k_w k_v}{2m-1},$$

where we used the linearity of $\mathbf{E}[\cdot]$ in the second equality. □

Note 3.7. *For large networks, where $|E| = m >> 1$, it's typical to just approximate this as:*

$$\frac{k_w k_v}{2m}.$$

So this gives us an average number of links between two nodes v and w in a Null Model.

4 Metric: Modularity

Definition 4.1 (**Modularity**). *For a graph $G = (V, E)$ with a partition C on V, define*

Community Structure

***modularity of** G **by the partition** C to be:*

$$Q(G,C) = \frac{1}{2m}\sum_i \sum_j \left(A_{i,j} - \frac{k_i k_j}{2m}\right)\delta(C_i, C_j) \qquad \delta(C_i, C_j) = \begin{cases} 1 & \text{if } C_i = C_j \\ 0 & \text{otherwise} \end{cases}$$

Note 4.2 (Interpretation of Modularity)**.**

- *If $Q(G,C) \leq 0$, then partition C gives no community structure at all, it might as well have been completely randomly generated.*
- *If $Q(G,C) > 0$, then the graph and partition both exhibit stronger community structure.*

As a working limit, $Q(G,C) > 0.3$ typically means strong community clustering.

Modularity will be used as the working definition of community structure, and we'll add this on to our list of assumptions for community detection:

Community Structure

Assumption 4.3.

1. *Graphs with community structure are generated by stochastic block models*
2. *Two nodes are much more likely to be connected inside a community than out*
3. *A null model of a graph has no community structure.*
4. *Community structure will correspond to graph with high modularity.*

Exercise 4.4. *Let $G = (V, E)$ be a graph, with $|V| = n$, $|E| = m$, with adjacency matrix A. If $C = \{C_1, \ldots, C_r\}$ is a partition of V, then define the $n \times r$ matrix S by*

$$S_{i,j} = \begin{cases} 1 & \text{if } i \in C_j \\ 0 & \text{otherwise} \end{cases}$$

and the $n \times n$ matrix B by

$$B_{i,j} = A_{i,j} - \frac{k_i k_j}{2m}.$$

Prove that:

$$Q(G, C) = \frac{1}{2m} Tr(S^T B S),$$

where $Tr(\,\cdot\,)$ is the trace operator.

With assumptions 4.3 our task of community detection can then be formulated as a maximization problem.

$$\arg\max_C Q(G, C) = \arg\max_C \frac{1}{2m} \sum_i \sum_j \left(A_{i,j} - \frac{k_i k_j}{2m} \right) \delta(C_i, C_j) \tag{1}$$

Community Detection in Networks

Note 4.5 (Brute Force Methods)**.** *A random brute force method is not practicable in most cases, even in the case of finding two communities C_1 and C_2 of equal size ($|C_1| = N_1$, $|C_2| = N_2$ where $N_1 = N_2 = \frac{N}{2}$), we have*

$$\frac{N!}{(N_1!)^2}$$

total possible partitions, using Stirling's approximation this is:

$$\frac{N!}{(N_1!)^2} \sim \exp\left((N+1)\ln(2) - \frac{1}{2}\ln(N)\right) \qquad N \to \infty.$$

*For even a reasonably small graph ($N = 100$) with these assumptions, there are approximately 10^{29} total partitions of this graph [**?**].*

So we need smart algorithms to find these partitions in a computationally efficient manner, as in real world networks it's not uncommon to see networks with the number of nodes exceeding 10^6. In the case of Facebook or other social media networks, the number of nodes might exceed 10^9 (as of 2021 Facebook has 2.91 billion users).

Note 4.6 (NP-Completeness)**.** *It has been shown that finding a global maximum for modularity is a NP-complete problem [**?**]. Meaning that the true global maximum of modularity can only be attained through an algorithm that takes non-polynomial time (i.e exponential time or higher).*

So all the algorithms that we'll be discussing here will only lead to local maximums of modularity, but then again something is better than nothing, so we'll see how different algorithms compare on different graphs.

5 The Louvain Algorithm

In these notes, we'll discuss one of the most widely used community detection algorithms called the **Louvain Algorithm** [**?**].

The goal of the Louvain algorithm and many other community detection algorithms is to maximize the modularity of a network by finding a local optimal community partition C [**?**, **?**].

Definition 5.1. *Let $G = (V, E, W)$ be a simple weighted graph and C be part of a partition of V. We'll define:*

$$\Sigma_{in} = \sum_{i \in C} \sum_{j \in C} A_{i,j}$$

total number of edges that land inside the community

$$\Sigma_{tot} = \sum_{i \in C} k_i$$

the total number of edges in a community, including those between a node in the community and to another community

$$\Sigma_{out} = \Sigma_{tot} - \Sigma_{in}$$

the total number of edges starting in C and landing outside of C

$$k_{i,in} = \sum_{j \in C} A_{i,j} + \sum_{i \in C} A_{i,j}$$

the number of edges a node has that land in C.

Instructor: E.Dimitrova
Name: Joseph McGuire Class: Math 599 (W2022)

Louvain Algorithm

Algorithm 5.2 (The Louvain Algorithm)**.**
Let $G = (V, E)$ be a simple weighted graph.
Phase 1

1. *Assign an initial guess of community structure of G:*

$$C = \{C_1, \ldots, C_r\},$$

a typical guess is that each node is a community; i.e. if $V = \{v_i\}_{i=1}^N$, then $C_i = \{v_i\}$ for all $i = 1, \ldots, N$.

2. *Start with a random community, then for each $v \in V$ with $v \in C_i$ move $v \to C_j$ (for some $j = 1, \ldots, r$) and calculate the change in modularity $\Delta Q_{i\to j}(v)$. We want to calculate each term of:*

$$\Delta Q_{i\to j}(v) = \Delta Q_{i\to\{v\}}(v) + \Delta Q_{\{v\}\to j}(v),$$

that is the modularity from moving v from C_i to C_j is the same as making an intermediate step of moving v to its own community $\{v\}$.

We'll prove these steps rigorously below.

We'll show that:

$$\Delta Q_{i\to j}(v) = \left[\frac{\Sigma_{in} + 2k_{i,in}}{2m} - \left(\frac{\Sigma_{tot} + k_i}{2m}\right)^2\right] - \left[\frac{\Sigma_{in}}{2m} - \left(\frac{\Sigma_{tot}}{2m}\right)^2 - \left(\frac{k_i}{2m}\right)^2\right]$$

For each node v, choose C_j such that:

$$C^* = \arg\max_j \Delta Q_{i\to j}(v),$$

that is pick the community C_j that maximizes the increase in modularity.

3. *Form the new partition C', by moving v to C^*.*

If the quantity in the previous step is $\Delta Q_{i\to j}(v) \leq 0$ (we lose or don't gain any modularity by moving v), then we don't move v.

4. *Repeat (1-3) for all $v \in V$ until no modularity can be gained.*

Phase 2

Louvain Algorithm

1. *Construct the new graph G' by taking $C_p \in C$, for each $p \in \{1, \ldots, r\}$, assign a self-loop to C_p with weight:*

$$w(C_p, C_p) = \sum_{v \in C_p} k_{v,in}$$

then for each $C_i, C_j \in C$ assign an edge with weight:

$$w(C_i, C_j) = \sum \text{ of edges starting in } C_i \text{ and ending in } C_j$$

2. *Repeat Phase 1 with the new graph G'.*

Note 5.3. *Note that the result of this algorithm, will then be a local maximum attained by the 'greedy' gathering of modularity by each community.*

Lemma 5.4. *Let $G = (V, E, W)$ be a simple weighted graph and $C = \{C_1, \ldots, C_r\}$. Then the modularity is:*

$$Q(G, C) = \frac{\Sigma_{in}}{2m} - \left(\frac{\Sigma_{tot}}{2m}\right)^2$$

Proof.

$$\begin{aligned} Q(G, C) &= \frac{1}{2m} \sum_i \sum_j \left(A_{i,j} - \frac{k_i k_j}{2m}\right) \delta(C_i, C_j) \\ &= \frac{1}{2m} \sum_i \sum_j A_{i,j} \delta(C_i, C_j) - \frac{1}{2m} \sum_i \sum_j \frac{k_i k_j}{2m} \delta(C_i, C_j) \\ &= \frac{\Sigma_{in}}{2m} - \frac{1}{4m^2} \left(\sum_i k_i\right) \left(\sum_j k_j\right) \\ &= \frac{1}{2m} \Sigma_{in} - \left(\frac{\Sigma_{tot}}{2m}\right)^2 \end{aligned}$$

□

Community Detection in Networks

Theorem 5.5. *Let $G = (V, E, W)$ be a simple weighted graph and $C = \{C_1, \ldots, C_r\}$. Suppose $v \in V$ and $C_p \in C$, then:*

$$\Delta Q_{\{v\}\to p}(v) = \left[\frac{\Sigma_{in} + k_{i,in}}{2m} - \left(\frac{\Sigma_{tot} + k_i}{2m}\right)^2\right] - \left[\frac{\Sigma_{in}}{2m} - \left(\frac{\Sigma_{tot}}{2m}\right)^2 - \left(\frac{k_i}{2m}\right)^2\right]$$

Proof. By definition, this is:

$$\Delta Q_{p\to\{v\}}(v) = Q_{After} - Q_{Before},$$

where Q_{After} is the modularity after we place v in C_p, and Q_{Before} is before this occurs and v is in its own isolated community $\{v\}$.

We'll have that:

$$Q_{Before} = \left[\frac{\Sigma_{in}}{2m} - \left(\frac{\Sigma_{tot}}{2m}\right)^2\right] + \left[0 - \left(\frac{k_i}{2}\right)^2\right]$$

this is the sum of the modularity of the graph without the community $\{v\}$ and then the modularity of the isolated community $\{v\}$.

Then

$$Q_{After} = \left[\frac{\Sigma_{in} + k_{i,in}}{2m} - \left(\frac{\Sigma_{tot} + k_i}{2m}\right)^2\right]$$

So that:

$$\Delta Q_{\{v\}\to p}(v) = \left[\frac{\Sigma_{in} + k_{i,in}}{2m} - \left(\frac{\Sigma_{tot} + k_i}{2m}\right)^2\right] - \left[\frac{\Sigma_{in}}{2m} - \left(\frac{\Sigma_{tot}}{2m}\right)^2 - \left(\frac{k_i}{2m}\right)^2\right]$$

□

Example 5.6. *Let's use this with the Zachary Karate Club:*

- *The original graph Figure 13*
- *Phase 1's result Figure 14*
- *Phase 2's result Figure 15*

Community Detection in Networks

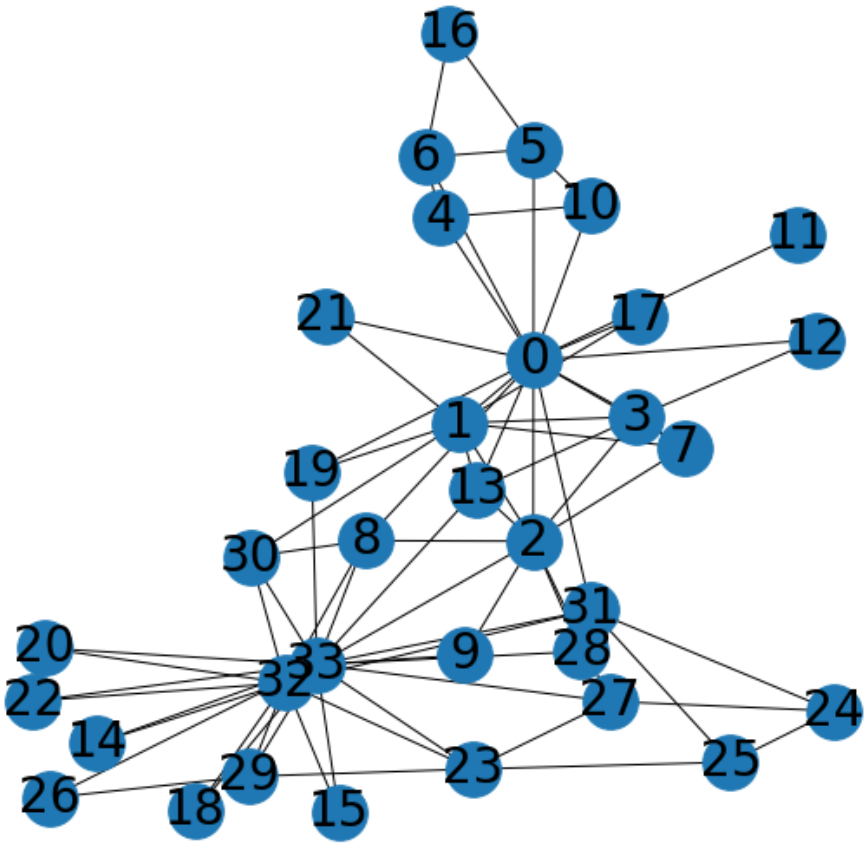

Figure 13: The Zachary Karate club network

Community Detection in Networks

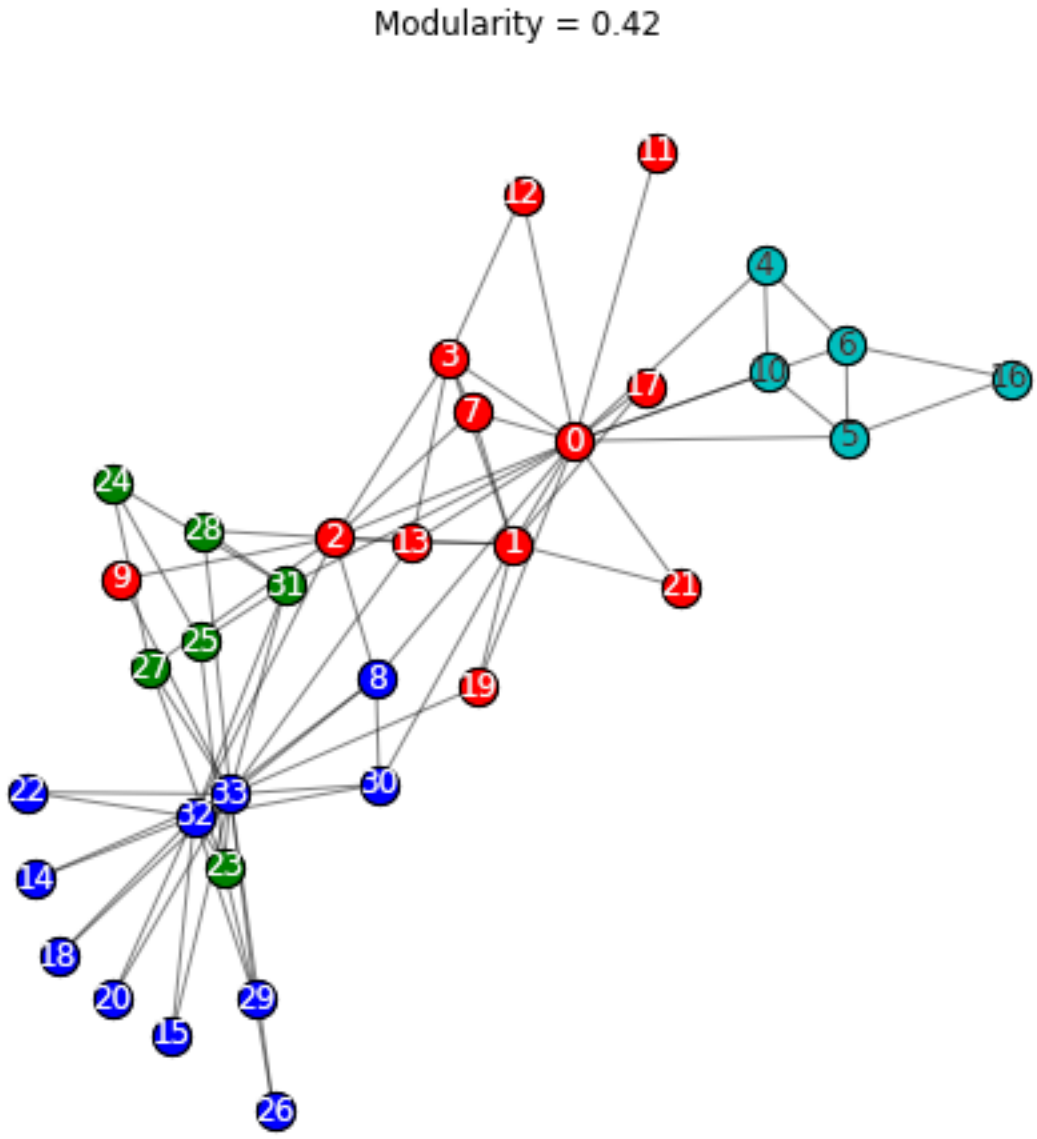

Figure 14: Phase 1 of the Louvain Algorithm on the Zachary Karate Club Network

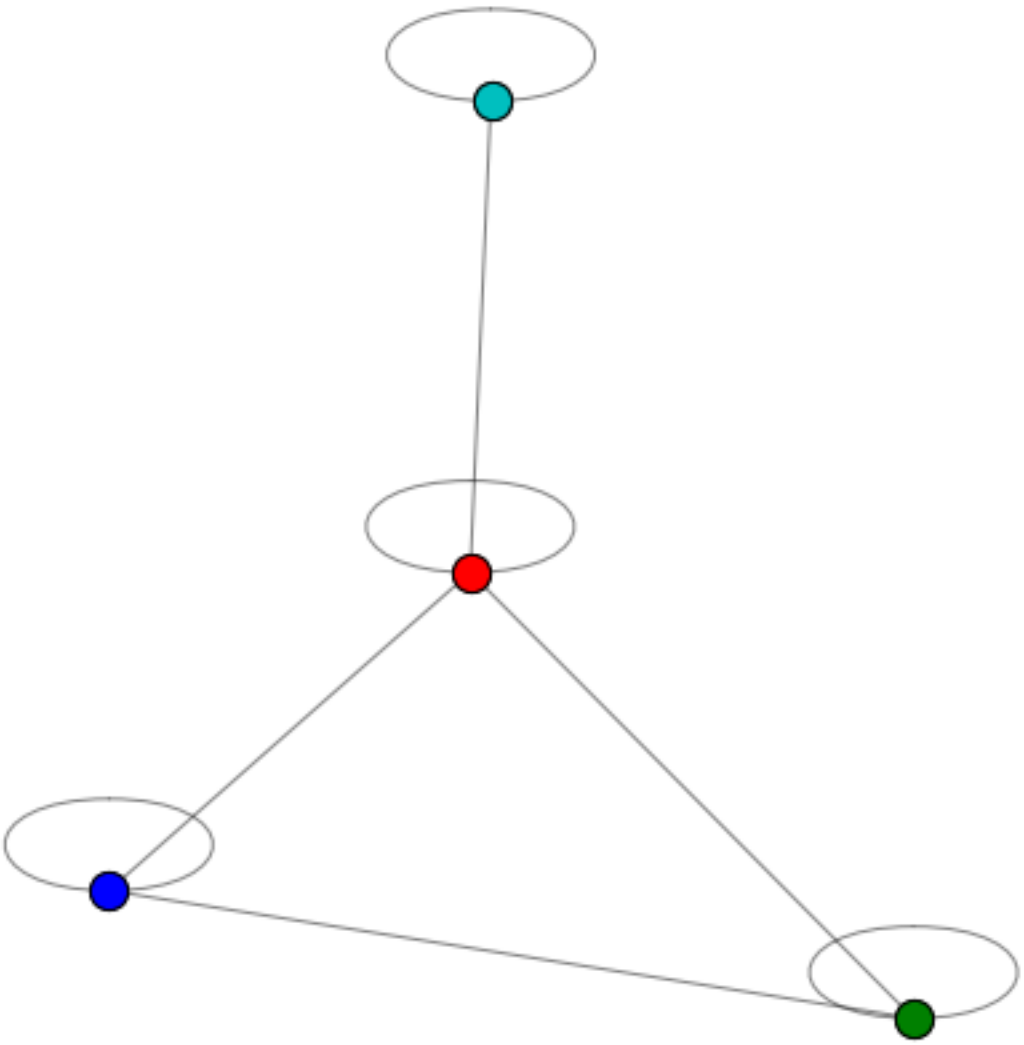

Figure 15: Phase 2 of the Louvain Algorithm on the Zachary Karate Club Network

6 K-Means Clustering

Let X be a data set $X = \{x^{(i)}\}_{i=1}^{N}$ such that $x^{(i)} \in \mathbb{R}^m$ for each $i = 1, \ldots, N$ and $m \in \mathbb{N}$. **K-Means Clustering** deals with the problem of finding clusters of densely packed groups in this data set, this is a topic in *unsupervised machine learning* (*supervised machine learning* deals with prediction of a target variable usually denoted $y^{(i)}$ for each $x^{(i)}$, while unsupervised machine learning has no target variables for the data points $x^{(i)}$)

This algorithm is sometimes referred to as Lloyd's algorithm or *naive k-means* [**?**].

Algorithm 6.1. *Let $k \in \mathbb{N}$ be fixed.*

1. *Partition your data set into k-groups, $S_1, \ldots, S_k$.*
2. *For each $i = 1, \ldots, k$:*

   Calculate the mean vector for S_i, denote this $m^{(i)}$; that is:

   $$m^{(i)} = \frac{1}{|S_i|} \sum_{x^{(j)} \in S_i} x^{(j)}$$

3. *For each $x^{(i)}$, $i = 1, \ldots, k$:*

   Assign each $x^{(i)}$ to the set S_{j^*} where $j^* = 1, \ldots, k$ and:

   $$j^* = \arg\min_j \|x^{(i)} - m^{(j)}\|_2^2$$

4. *Repeat steps until in previous step, no change occurs.*

Example 6.2. *We generate a random data set $X = \{x^{(i)}\}_{i=1}^{1000}$ where $x^{(i)} \in \mathbb{R}^2$ for each $i = 1, \ldots, 1000$ using a uniform distribution on $[0, 1] \times [0, 1]$. One half of the data is kept in the rectangle $[0, 1] \times [0, 1]$, the other half is shifted diagonally to be contained in $[1, 2] \times [1, 2]$. We run k-means with $k = 2$ using the Python library Scikit-Learn and we end up with the result seen in Figure 16.*

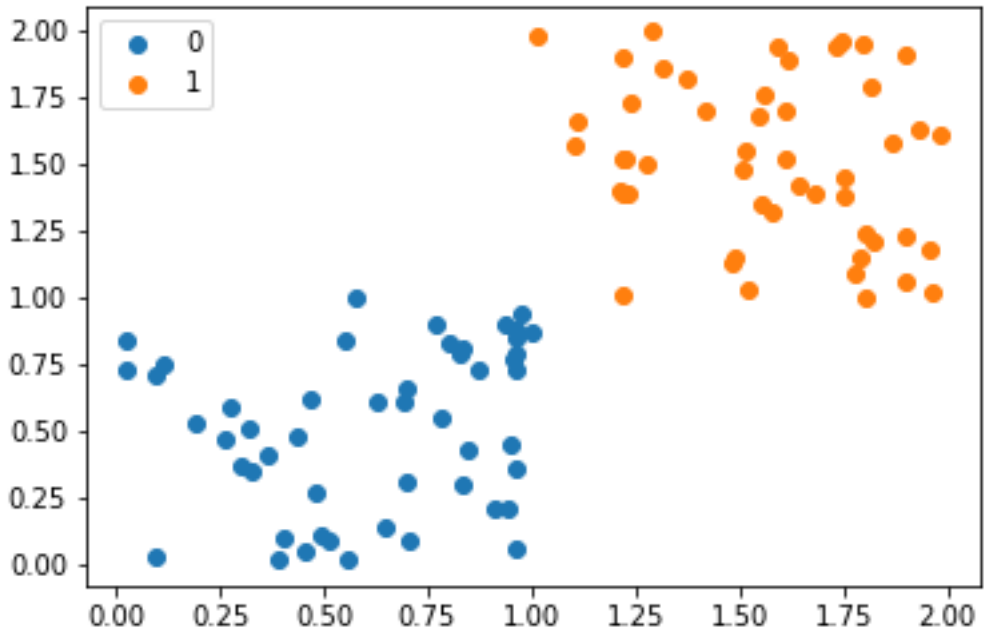

Figure 16: K-means ran with $k = 2$

7 Spectral Clustering

As we've seen with the Louvain algorithm, the problem of community detection can a difficult problem. One way to tackle the problem of community detection is to find a mapping:

$$f : G \to \mathbb{R}^m,$$

for some small $m \in \mathbb{N}$ and analyze the problem a problem in $\mathbb{R}^m$, then map the solution back to the graph G.

The advantage of this is that the study of clustering in continuous spaces like $\mathbb{R}^m$ is a well studied area of research; there exists dozens of techniques in unsupervised machine learning that tackle this problem [**?**]. The tricky task we have to tackle before we can leverage this machinery against community detection, we need the map $f : G \to \mathbb{R}^m$ to decide what information about the graph G we need to preserve.

Instructor: E.Dimitrova
Name: Joseph McGuire Class: Math 599 (W2022)

Spectral Clustering

Definition 7.1 (**Laplacian**). *Given a weighted simple graph $G = (V, E, W)$, define the Laplacian matrix of G to be:*

$$L = [D_{i,j} - W_{i,j}]_{i,j},$$

where W is the weighted adjacency matrix of G and D is the diagonal degree matrix of G:

$$D_{i,j} = \begin{cases} \sum_j w_{i,j} & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}.$$

This matrix both encodes degree information of the graph, as well as the adjacency matrix of the graph (since $W_{i,i} = 0$ for graphs without self-loops). So the diagonal entries will always tell us the degree of a node, while the off-diagonal entries tell us about the connections between nodes.

*The study of the Laplacian matrix is a very well studied area of research known as Spectral Graph Theory [**?**].*

Theorem 7.2. *Let G be a simple graph and L be the Laplacian of the graph. Then*

1. *L is positive semi-definite and symmetric*
2. *0 is an eigenvalue of L with eigenvector $\mathbf{1}$*

We relate community detection back to the Laplacian by posing community detection as a problem of finding a line that minimizes the number of edges it passes through, the line acts as a cut on the graph. In particular, we'll first look at a simple case of a partition into two sets of the same size.

Definition 7.3 (**Cut & Cut Size**). *Let $G = (V, E, W)$ be a simple weighted graph and let l be a cut on the edges of G; i.e $l \subset E$ that defines a partition $A \cup B = V$, $A \cap B = \emptyset$. Let $X, Y \subseteq V$.*

*For any X, Y define: **weight**:*

$$w(X, Y) = \sum_{i \in X, j \in Y} w_{i,j}$$

***the normalized cut size** between X and Y:*

$$ncut(X, Y) = \frac{w(X, Y)}{w(X, V)} + \frac{w(X, Y)}{w(Y, V)}$$

Spectral Clustering

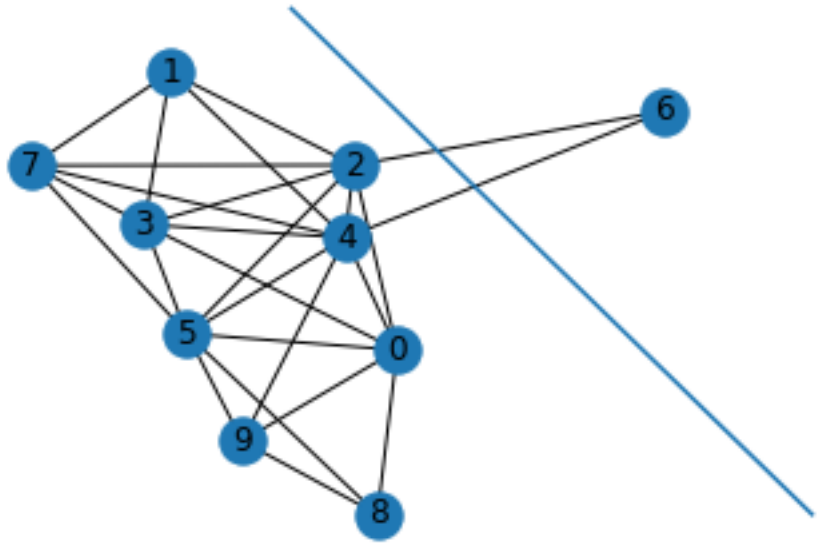

Figure 17: A cut equalling $\{(2,6),(4,6)\}$ and defines a partition $A = \{6\}$ and $B = V \setminus \{6\}$.

This is the size of the cut l normalized so that $0 \leq ncut \leq 2$

The following theorem relates the eigenvector of this matrix L to a suitable map from $G \to \mathbb{R}^m$.

Theorem 7.4. *Let $G = (V, E)$ be a simple graph.*
Suppose $V = A \cup B$ is a partition of V, with $|A| = |B|$ where $A \cap B = \emptyset$. Define

$$y = \begin{cases} 1 & \text{if } i \in A \\ -b & \text{if } i \in B \end{cases}, \qquad b = \frac{k}{1-k}, \qquad k = \frac{\deg(A)}{\deg(V)}.$$

Then

$$\min_{(A,B)} ncut(A, B)$$

is equivalent to:

$$\begin{cases} \min\limits_{y} \dfrac{y^T L y}{y^T D y} \\ y^T D \mathbf{1} = 0 \end{cases},$$

where $\mathbf{1}$ *is the vector of all 1's.*

Proof. The proof is a lot of computation, for details see [**?**]. □

Note 7.5. *If* D *is a diagonal matrix:*

$$D = diag\{d_1, \ldots, d_M\},$$

then for $\alpha \in \mathbb{R}$

$$D^\alpha = diag\{d_1^\alpha, \ldots, d_M^\alpha\}.$$

To solve:

$$\begin{cases} \min_y \dfrac{y^T L y}{y^T D y} \\ y^T D \mathbf{1} = 0 \end{cases}$$

We're going to transform this problem into:

$$\begin{cases} \min_z \dfrac{z^T B z}{z^T z} \\ z^T z_0 = 0 \end{cases}, \qquad B = D^{-1/2} L D^{-1/2}$$

where $z = D^{1/2} y$ and $z_0 = D^{1/2}\mathbf{1}$. This new matrix $B = D^{-1/2} L D^{-1/2}$ is called the **normalized Laplacian**. The reason for this transformation is that we can now pose this as an eigenvalue problem:

$$Bz = \lambda z \iff z^T B z = \lambda z^T z \iff \lambda = \frac{z^T B z}{z^T z}$$

and the condition:

$$z^T z_0 = 0$$

ensures that z is orthogonal to the eigenvector z_0.

Exercise 7.6. *Verify that* z_0 *is in fact an eigenvector of* B*.*

Spectral Clustering

Theorem 7.7.

$$\begin{cases} \min_z \dfrac{z^T B z}{z^T z} \\ z^T z_0 = 0 \end{cases},$$

then the eigenvalues of B satisfy:

$$\lambda_0 = 0 < \lambda_1 < \dots$$

then the minimization problem is solved by the eigenvector with eigenvalue $\lambda_1 \neq 0$.

This method then can be generalized to k-cuts in a number of different ways, outlined in [**?**]. The general algorithm for spectral clustering is then given below

Algorithm 7.8 (**Spectral Clustering for Graphs**).
Let $G = (V, E)$ be a simple weighted graph with $|V| = n$.

1. *Calculate the Laplacain for G, L.*
2. *Find the first k eigenvalues of L and their corresponding eigenvectors.*
3. *Form a matrix U where the column vectors of U are the k-eigenvectors of L; U is a $n \times k$.*
4. *For each $i = 1, \dots, n$:*
   (a) *Map the node $i \in V$ to the row vector $v_i \in \mathbb{R}^k$ obtained from U.*
5. *Use K-means clustering or other clustering algorithms in $\mathbb{R}^k$ to find clustering.*
6. *Map cluster membership in $\mathbb{R}^k$ back to the graph G.*

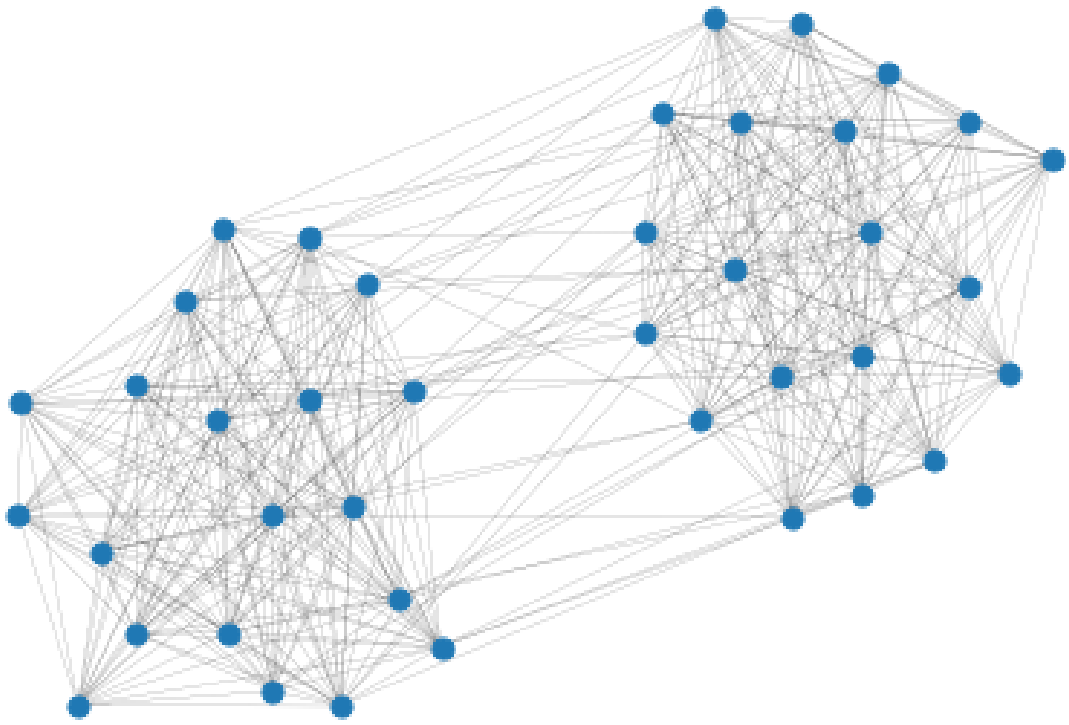

Figure 18: The graph we'll be preforming spectral clustering on.

Example 7.9. *Let's take a planted partition model with inter-community probability* $p = 1$ *and a intra-community probability* $q = 0.1$ *with two communities of equal size and a vertex set of size* 20*; that is, nodes inside the same community must be connected and nodes from the two different communities have a one-in-ten chance of being linked. The graph can be seen in Figure 18.*

Choosing just the 2 smallest non-zero eigenvectors of the Laplacian for G*, and using* 2*-means clustering we end up with Figure 19.*

Finally, we map this clustering back to our graph we end up with Figure

Spectral clustering is also applied to problems of clustering in $\mathbb{R}^n$ as a data reduction method, where a mapping is found from $\mathbb{R}^n \to G$, where G is typically graph defined by a distance function in $\mathbb{R}^n$, then a second map is found from $G \to \mathbb{R}^m$ $(m \leq n)$. So spectral clustering can also be used a dimensional reduction technique where we map data in $\mathbb{R}^n \to G \to \mathbb{R}^m$ where $m \leq n$.

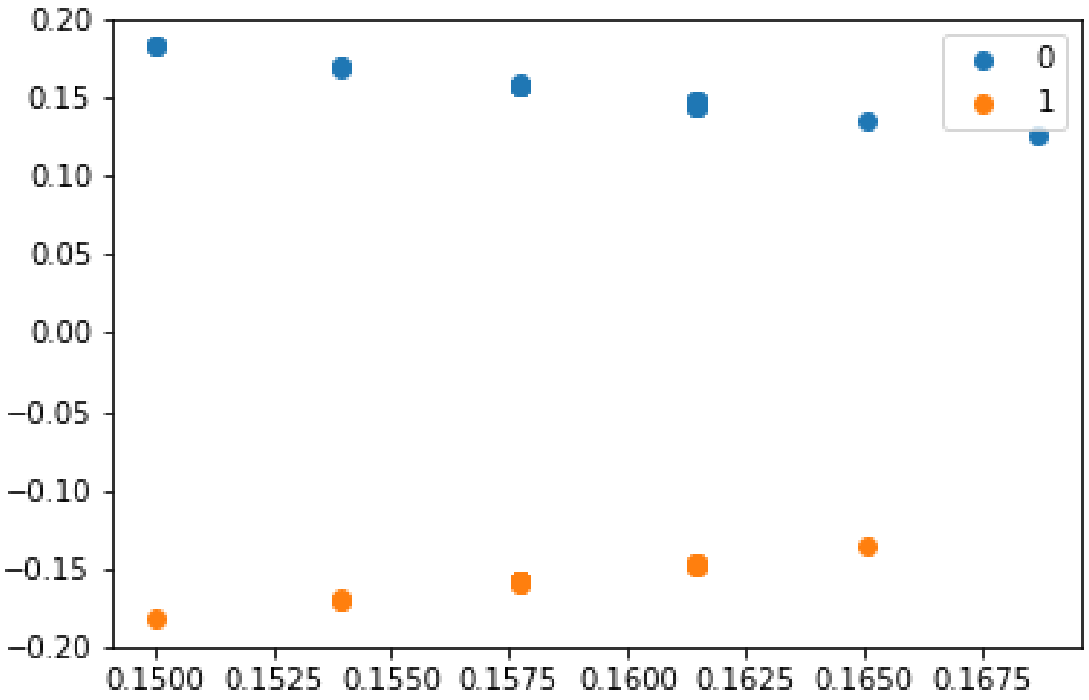

Figure 19: The eigenvectors of the normalized Laplacian of the graph 18.

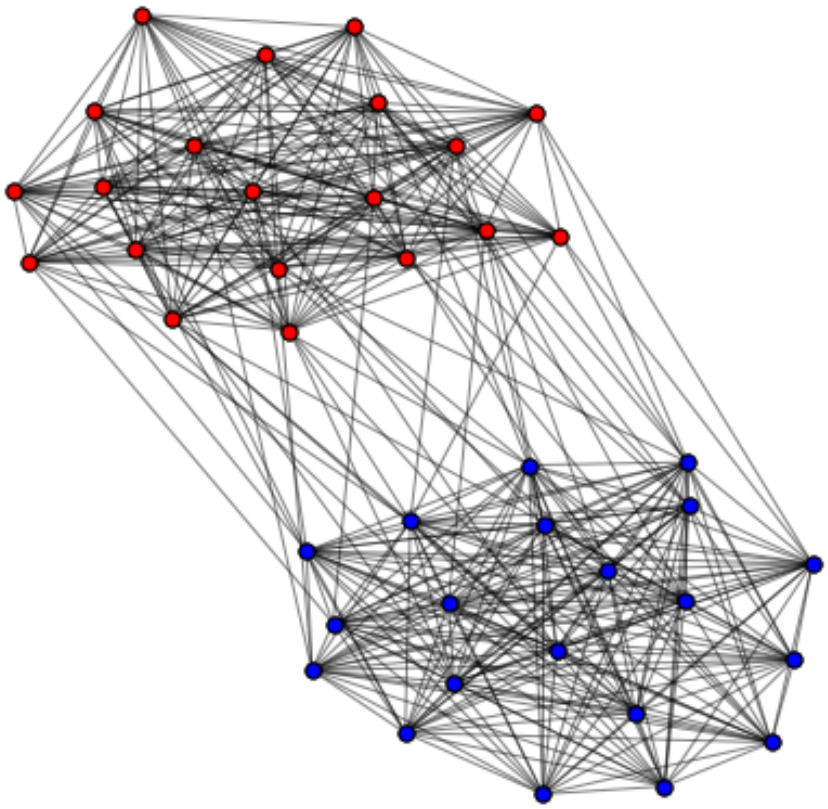

Figure 20: The result of spectral clustering on the graph 18, the modularity of this clustering is approx 0.4.