Network Centrality and Clustering

Overview

We previously introduced the concept of a graph to describe networks. To further analyze networks we will discuss network centrality and clustering.

Motivation

A common question in the analysis of networks is to understand the relative *importance* of the nodes in the network.

For example:

- in a social network, who are the most influential individuals?
- on the Web, which pages are more informative?
- in road network, which intersections are most heavily used?

The key idea is that the **structure** of the network should give us some information about the relative importance of the nodes in the network.

Centrality and Clustering

To analyze networks we want to be able to classify:

- important **nodes**, and
- important **groups of nodes**.

Determining which nodes are important nodes leads to the notion of **centrality**.

Determining which groups of nodes are important leads to the notion of **clustering** and **partitioning**.

In order to classify important nodes and groups of nodes, we will use graph theory and linear algebra.

Centrality

Do some nodes in the network have a special role? Are some nodes more *important* than others?

These are questions of **centrality** (or **prestige**).

Definition

- **Centrality** in graph theory and network analysis refers to measures that identify the most important vertices (nodes) within a graph.
- These measures assign numbers or rankings to nodes based on their position and influence within the network.
- Centrality can help determine which nodes are most influential, critical for connectivity, or central to the flow of information.

We will study three basic notions of centrality:

1. **Closeness Centrality**: A central node is close to all others.
2. **Betweenness Centrality**: A central node is on many paths through the network.
3. **Eigenvector Centrality**: A central node is connected to other central nodes (sometimes called **status** centrality).

We'll look at a very famous network analysis dataset: Zachary's karate club.

Karate Club Graph

The back story: from 1970 to 1972 the anthropologist Wayne Zachary studied the social relationships inside a university karate club.

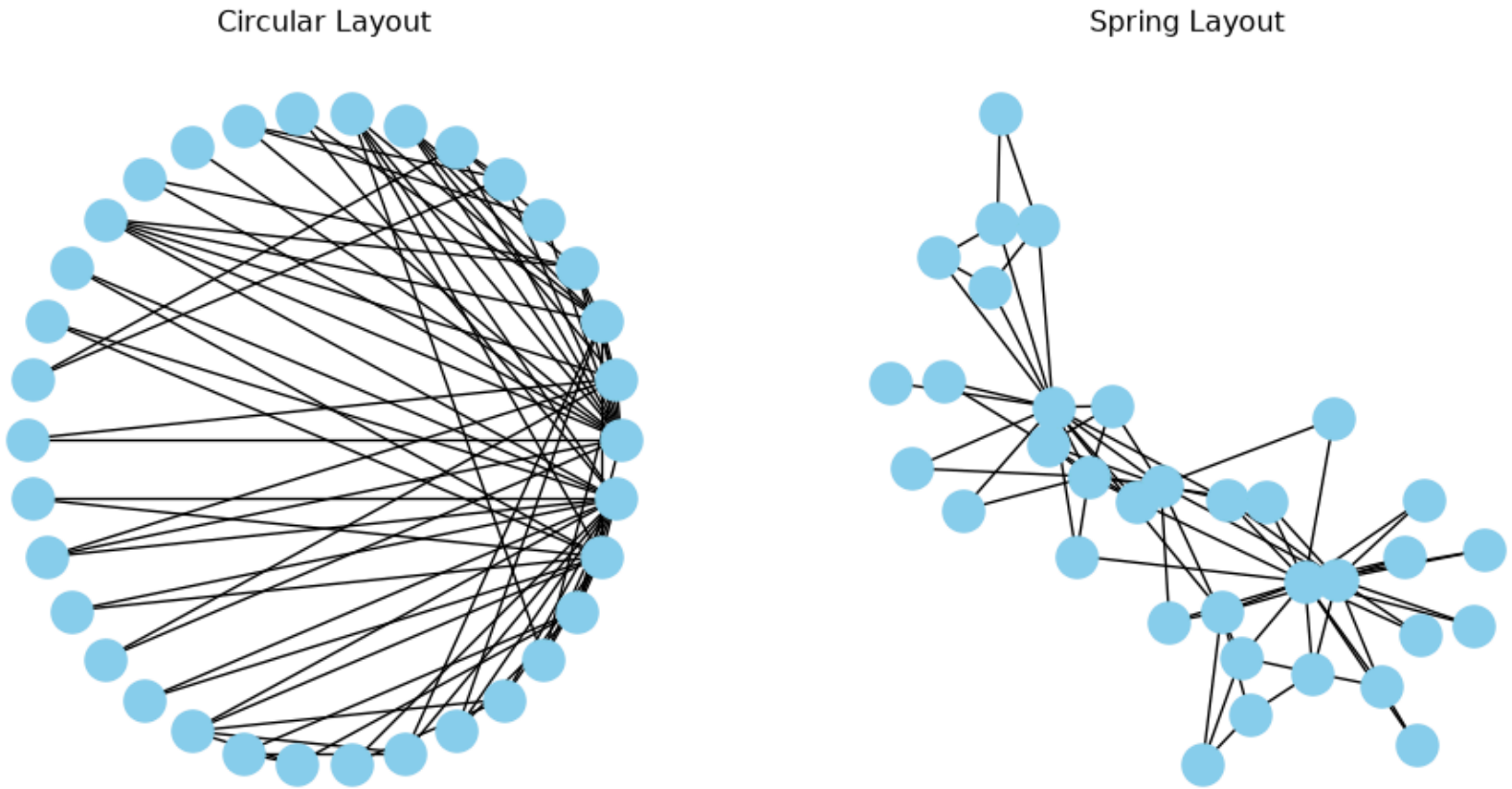
While he was studying the club, a factional division led to a splitting of the club in two.

The club became split between those who rallied around the club president and those who rallied around the karate instructor.

You can read the story of the Karate club here. This dataset has become so famous that it has spawned its own academic traditions.

Here’s a view of the social network of the karate club.

▶ Code



Closeness Centrality

The closeness centrality of a node i is an indicator of the proximity between node i and all the other nodes in the graph.

Let $G = (V, E)$ be a connected graph with n nodes.

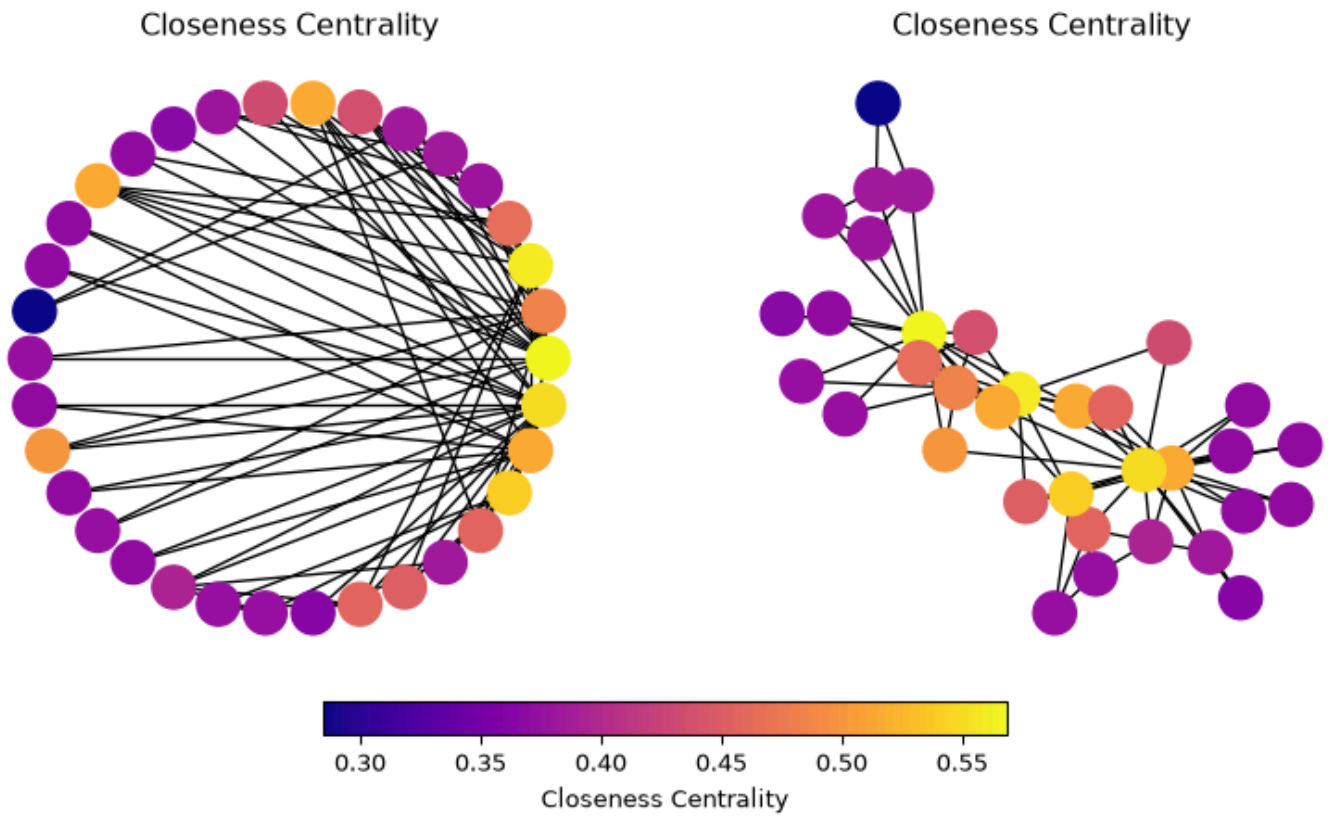
Let $d(i, j)$ be the shortest path distance between node i and node j in G.

The standard way of formulating closeness centrality is the reciprocal of the total distance to all other nodes

$$\text{closeness}(i) = \frac{1}{\sum_{j \in V} d(i, j)}.$$

Interpretation

▶ Code

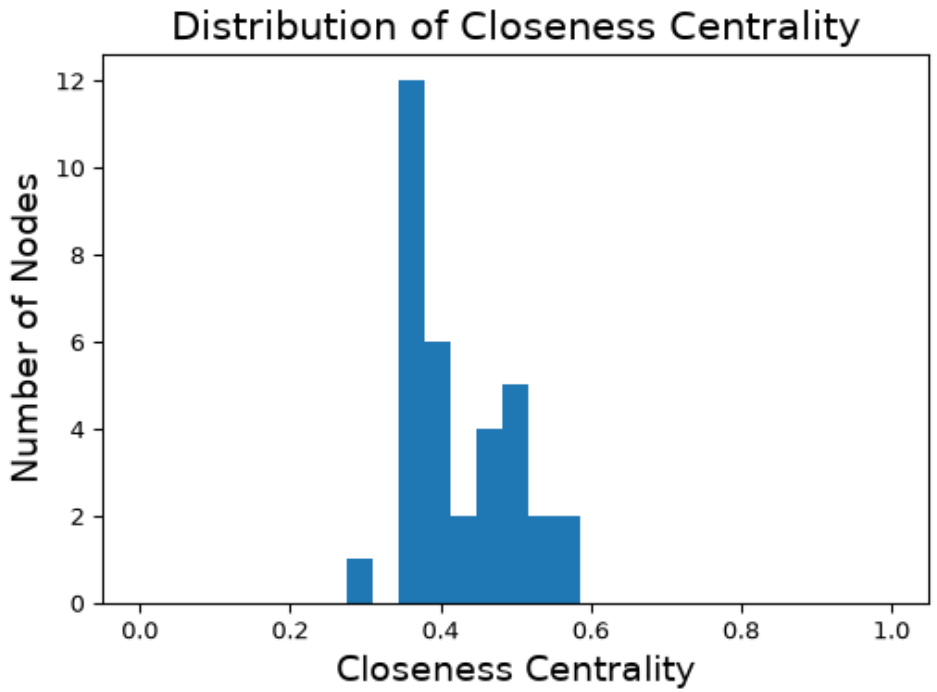


In this graph, most nodes are close to most other nodes.

However we can see that some nodes are slightly more central than others.

Distribution of Closeness Centrality

▶ Code



Higher closeness centrality means closer to all other nodes.

Betweenness Centrality

Alternatively we might be interested in the question, *is the node on many paths?*

Betweenness captures how important a node is to the communication process, or *how much* information passes through the node.

First, let's consider the case in which there is only one shortest path between any pair of nodes.

Then, the betweenness centrality of node i is the **number of shortest paths that pass through** i.

Mathematically we have

$$\text{betweenness}(i) = \sum_{i \neq j \neq k \in V} \begin{cases} 1 & \text{if path from } j \text{ to } k \text{ goes through } i, \\ 0 & \text{otherwise.} \end{cases}$$

For a graph with n nodes, we can normalize this to a value between 0 and 1 by dividing by $\binom{n}{2} = n(n-1)/2$.

General Case: Dependency

In a general graph, there may be **multiple** shortest paths between j and k.

To handle this, we define:

- $\sigma(i \mid j, k)$ is the number of shortest paths between j and k that pass through i, and
- $\sigma(j, k)$ is the total number of shortest paths between j and k.

Then we define the *dependency* of i on the paths between j and k as the *ratio of those two quantities*:

$$\text{dependency}(i \mid j, k) = \frac{\sigma(i \mid j, k)}{\sigma(j, k)}.$$

You can think of this as *the probability that a shortest path between j and k goes through i*.

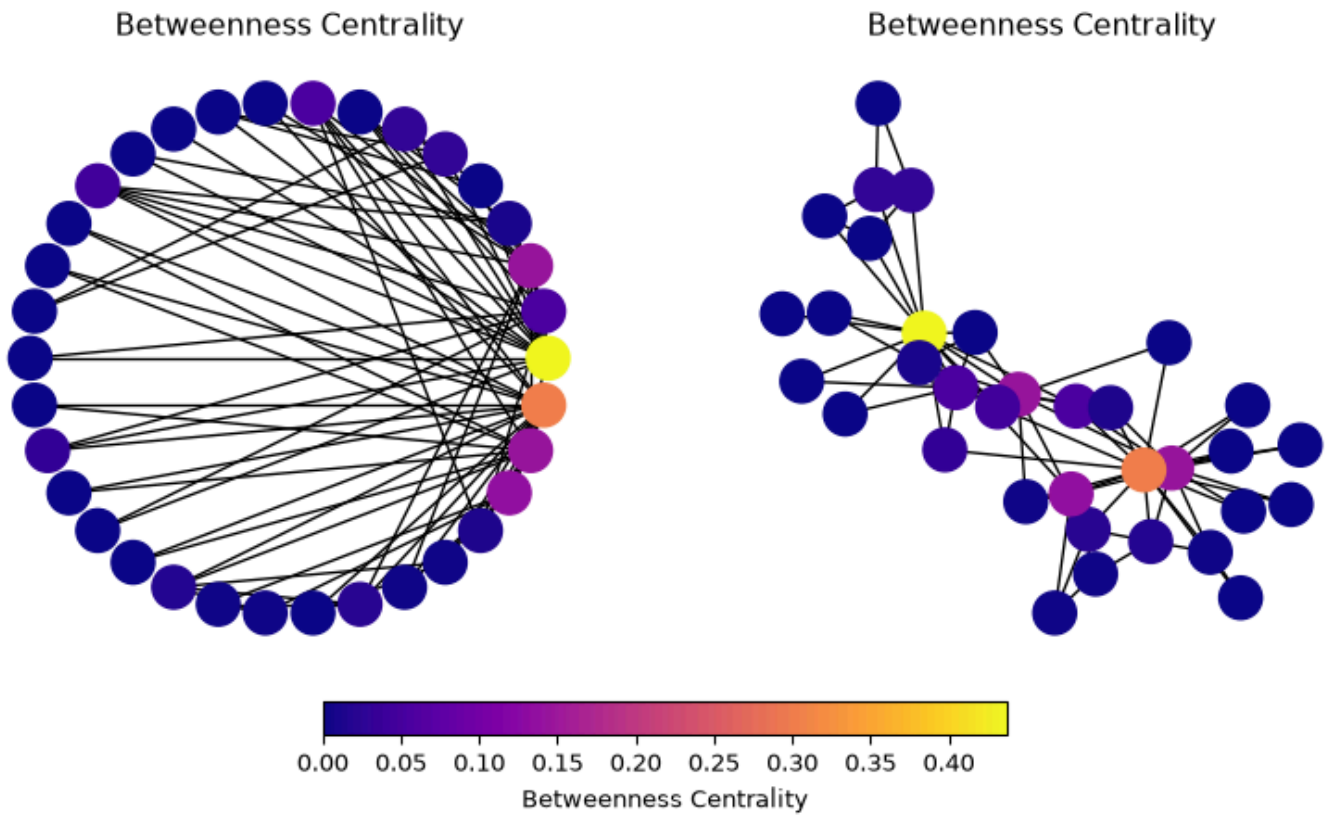
General Case: Betweenness Centrality

Finally, the betweenness centrality of node i is the sum of the dependencies of i on all pairs of nodes j and k.

$$\boxed{\text{betweenness}(i) = \sum_{i \neq j \neq k \in V} \text{dependency}(i \mid j, k).}$$

> Note that many nodes will have a betweenness centrality of zero – no shortest paths go through them.

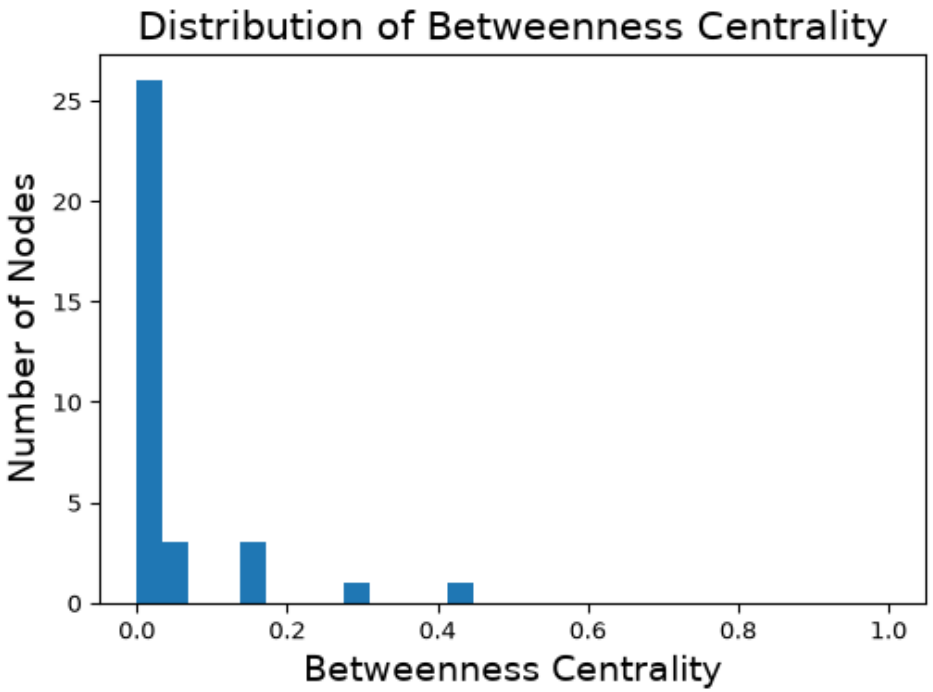
▶ Code



We start to see with this metric the *importance* of two or three key members of the karate club.

Distribution of Betweenness

▶ Code



The higher the betweenness centrality of a node, the more it is on many shortest paths.

Eigenvector Centrality

Eigenvector centrality is a measure used in network analysis to determine the influence of a node within a network.

The centrality score of a node is proportional to the sum of the centrality scores of its neighbors.

The main idea of eigenvector (or status) centrality is that ***high status* nodes are connected to *high status* nodes**.

We calculate this with some matrix algebra.

Adjacency Matrices

First, represent the graphs by its **adjacency matrix**.

Given an n-node undirected graph $G = (V, E)$, the adjacency matrix A is defined as

$$A_{ij} = \begin{cases} 1, & \text{if } (i,j) \in E \\ 0, & \text{otherwise} \end{cases} .$$

ID	0	1	2
0	0	1	0
1	1	0	1
2	0	1	0

Karate Club Adjacency Matrix

An important way to think about adjacency matrices is that **column j holds j's neighbors**.

The adjacency matrix is *nonnegative* and *symmetric*.

```
[[0 1 1 1 1 1 1 1 1 0 1 1 1 1 0 0 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 1 0 0]
 [1 0 1 1 0 0 0 1 0 0 0 0 0 1 0 0 0 1 0 1 0 1 0 0 0 0 0 0 0 0 1 0 0 0]
 [1 1 0 1 0 0 0 1 1 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1 0]
 [1 1 1 0 0 0 0 1 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [1 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [1 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [1 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1]
 [0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1]
 [1 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1]
 [0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1]
 [1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1]
 [1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 0 1 1]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 1 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1]
 [0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 1]
 [0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 1 1]
 [0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1]
 [1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 0 0 0 1 1]
 [0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 1 0 0 1 0 1 0 1 1 0 0 0 0 0 1 1 1 0 1]
 [0 0 0 0 0 0 0 0 1 1 0 0 0 1 1 1 0 0 1 1 1 0 1 1 0 0 1 1 1 1 1 1 1 0]]
```

Reminder: Properties of nonnegative symmetric matrices

A **nonnegative symmetric matrix** A has the following properties:

1. All eigenvalues of A are real.
2. All eigenvalues of A are nonnegative.
3. There is only **1 corresponding eigenvector with all positive entries**.

This is a consequence of the Perron-Frobenius Theorem.

Eigenvector Centrality

Remember:

- Eigenvector centrality is a measure used in network analysis to determine the influence of a node within a network.
- The centrality score of a node is proportional to the sum of the centrality scores of its neighbors.
- The main idea of eigenvector (or status) centrality is that ***high status*** **nodes are connected to** ***high status*** **nodes**.

Eigenvector Centrality Definition

Given a graph with n nodes and let A be the adjacency matrix of the graph.

The eigenvector centrality x_i of node i is defined as proportional to the sum of the centrality scores of its neighbors.

$$x_i = \frac{1}{\lambda} \sum_{j=1}^{n} A_{ij} x_j,$$

where

- x_i is called the eigenvector centrality of node i.
- λ is a constant (which we'll show is an eigenvalue).

Put another way, the importance of node i is proportional to the sum of the importance of the other nodes j directly connected to i.

Matrix Form

In matrix form, this can be written as

$$\mathbf{A}\mathbf{x} = \lambda \mathbf{x},$$

where $\mathbf{x}$ is the eigenvector corresponding to the the eigenvalue λ of the adjacency matrix $\mathbf{A}$.

The eigenvector $\mathbf{x}$ gives the centrality scores for all nodes in the network.

- In general, there may be multiple eigenvalues λ for which the above equation holds.
- In practice, the largest eigenvalue is used because it captures the most significant mode of influence propagation in the network.

Geometric Intuition

- **Eigenvector as a direction**:
  - The eigenvector is a direction in the network space. Applying the adjacency matrix (representing connections between nodes) to this vector, *stretches* the vector along that direction. The amount of stretch is determined by the eigenvalue.
- **High centrality, high stretch**:
  - A node with a large eigenvector centrality corresponds to an eigenvector that experiences a large *stretch* when transformed by the adjacency matrix. This indicates that it is connected to many other nodes that are also considered high-status.
- **No direction change**:
  - The eigenvector doesn't change its direction when transformed, only its magnitude. This means that a node with a high eigenvector centrality remains relatively *central* even when considering its connections to other central nodes.

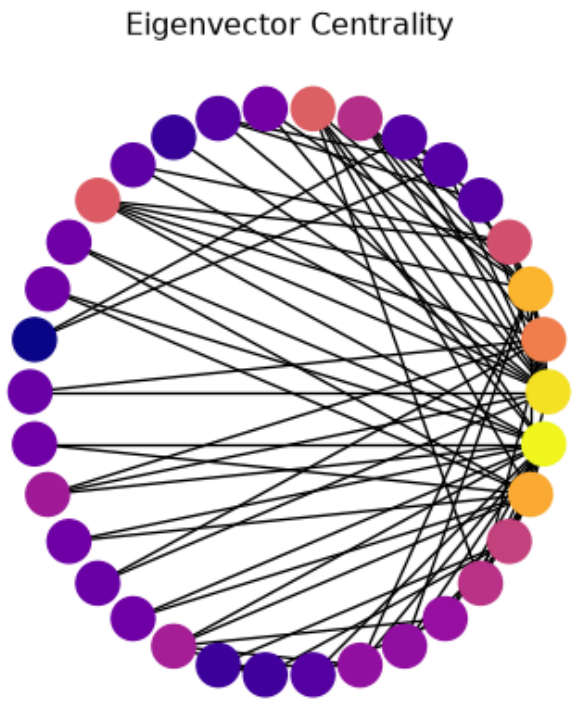
Key Points on Eigenvector Centrality

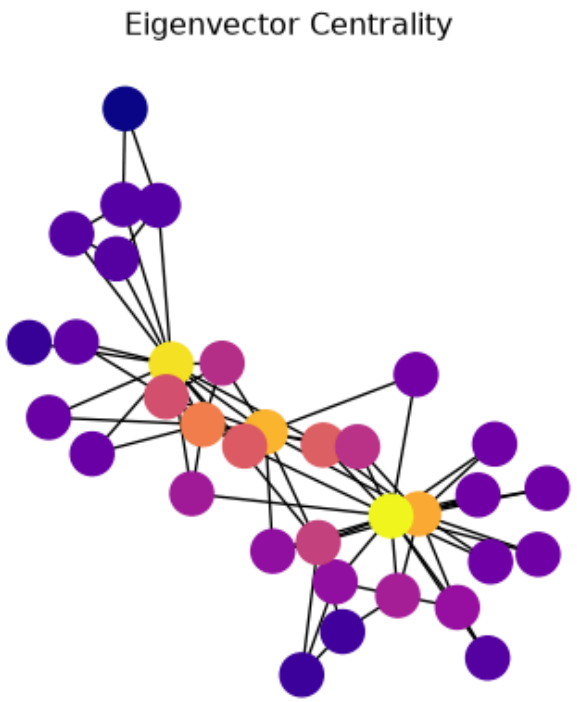
- **Eigenvector**: The vector $\mathbf{x}$ represents the centrality scores.
- **Largest Eigenvalue**: The centrality scores are derived from the eigenvector corresponding to the largest eigenvalue of the adjacency matrix.
- **Perron-Frobenius Theorem**: For a connected graph, the largest eigenvalue is positive. In addition, there is only 1 corresponding eigenvector with all positive entries.

The Perron-Frobenius Theorem is an important theoretical tool when working with adjacency matrices, which are by definition nonnegative and positive.

Eigenvector Centrality: Example

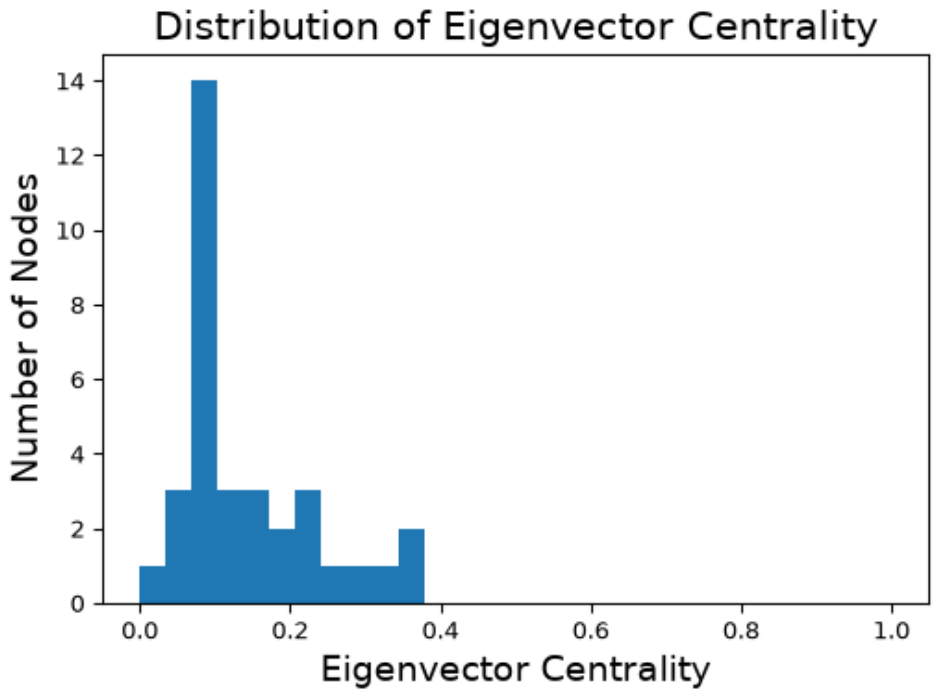
▶ Code





Distribution of Eigenvector Centrality

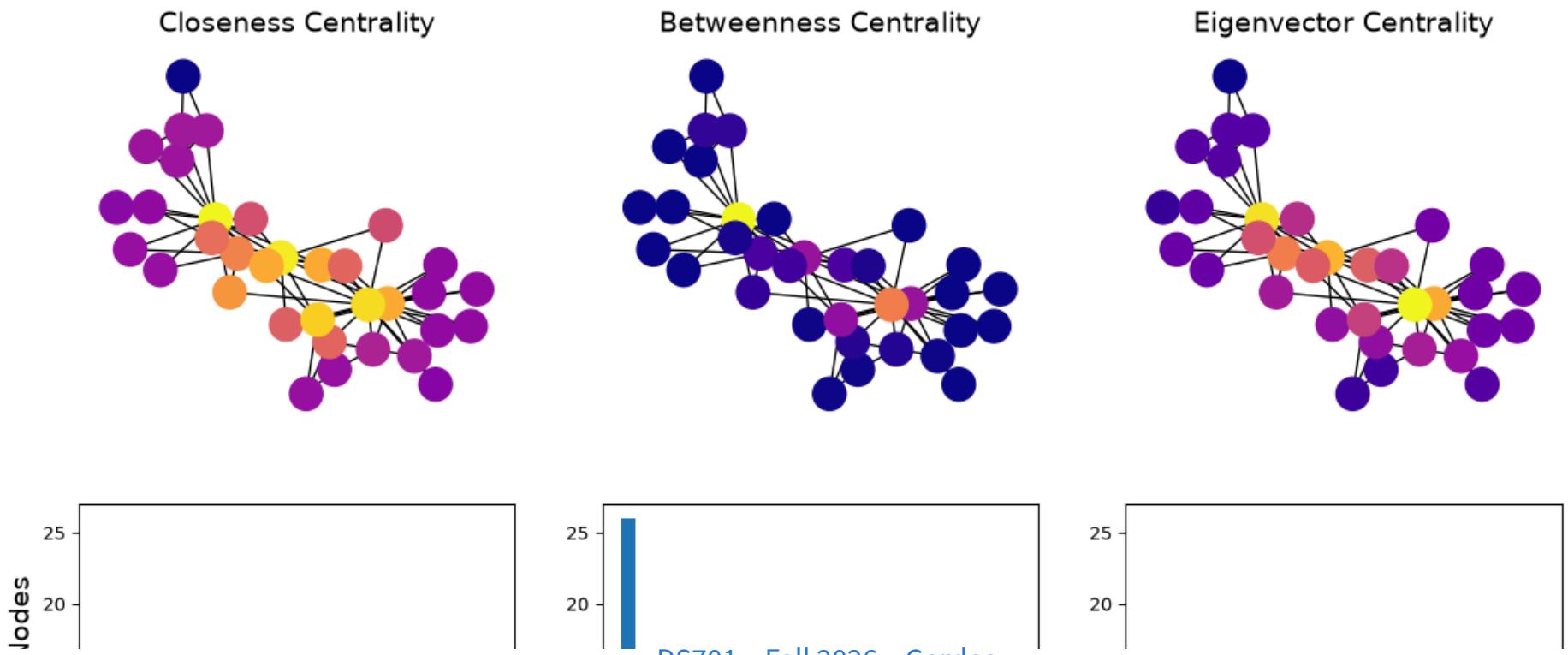
▶ Code



Centrality Comparison: Karate Example

Let’s compare the three versions of centrality we’ve looked at so far.

▶ Code



Clustering and Partitioning

Clustering and Partitioning

We now turn to the question of finding important **groups** of nodes.

Why might we want to cluster graph nodes?

- Assigning computations to processors in a parallel computer
- Segmenting images (finding boundaries between objects)
- Clustering words found together in documents, or documents with similar words
- Divide and conquer algorithms
- Circuit layout in VLSI
- Community detection in social networks

Min s-t cut

We'll start with a problem that is fundamental to many other problems in graph analysis.

It involves finding the smallest set of edges that, if removed, would disconnect a specified source node (s) from a target node (t) in a graph.

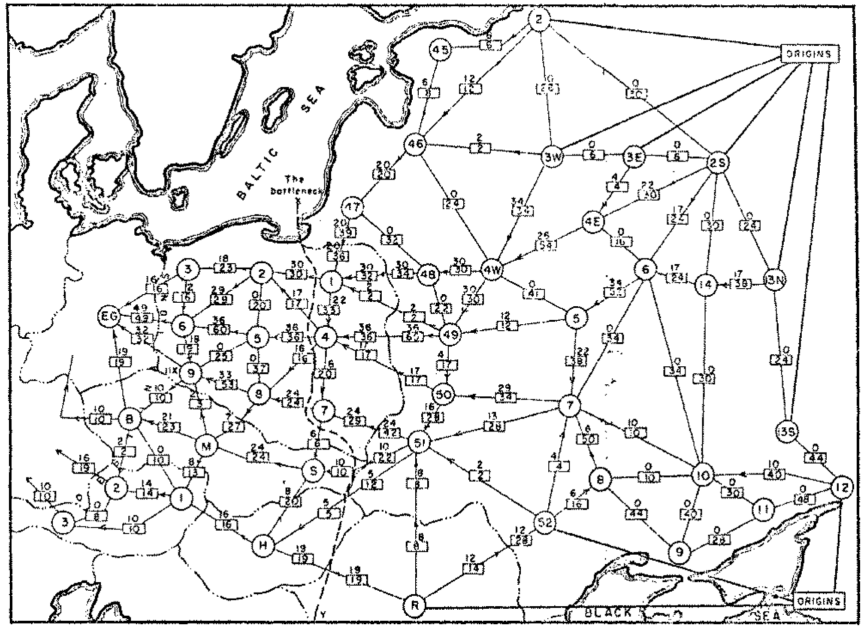
The goal of a min s-t cut is to partition the graph into two disjoint subsets such that:

- s is in one subset and t is in the other, and
- the total weight (or number) of the edges that need to be removed to achieve this separation is minimized.

Practical Example: Soviet Railway Network

Here is a schematic diagram of the railway network of the Western Soviet Union and Eastern European countries in the Cold War era.

The idea was to determine a minimum cut, which would best halt the flow of materials in the railway network.



For an interesting historical perspective on the min-cut problem and its relation to the Cold War, see "On the history of the transportation and maximum flow problems," by Alexander Schrijver, in Mathematical Programming 91.3 (2002): 437-445.

Cost of a Cut

Let $G = (V, E)$ be a weighted graph.

A weighted graph assigns a weight $w(e)$ to each edge $e \in E$.

An s-t cut C of G is a partition of V into $(U, V - U)$ such that $s \in U$ and $t \in V - U$.

Min s-t Cut Problem

Let $G = (V, E)$ be a weighted graph with edge weights $w(e)$ for each $e \in E$.

An s-t cut C is a partition of V into $(U, V - U)$ such that $s \in U$ and $t \in V - U$.

The **min s-t cut problem** seeks to find the partition that minimizes this cost:

$$\min_{C} \text{Cost}(C) = \min_{U \subseteq V : s \in U, t \notin U} \sum_{e(u,v),\, u \in U,\, v \in V - U} w(e).$$

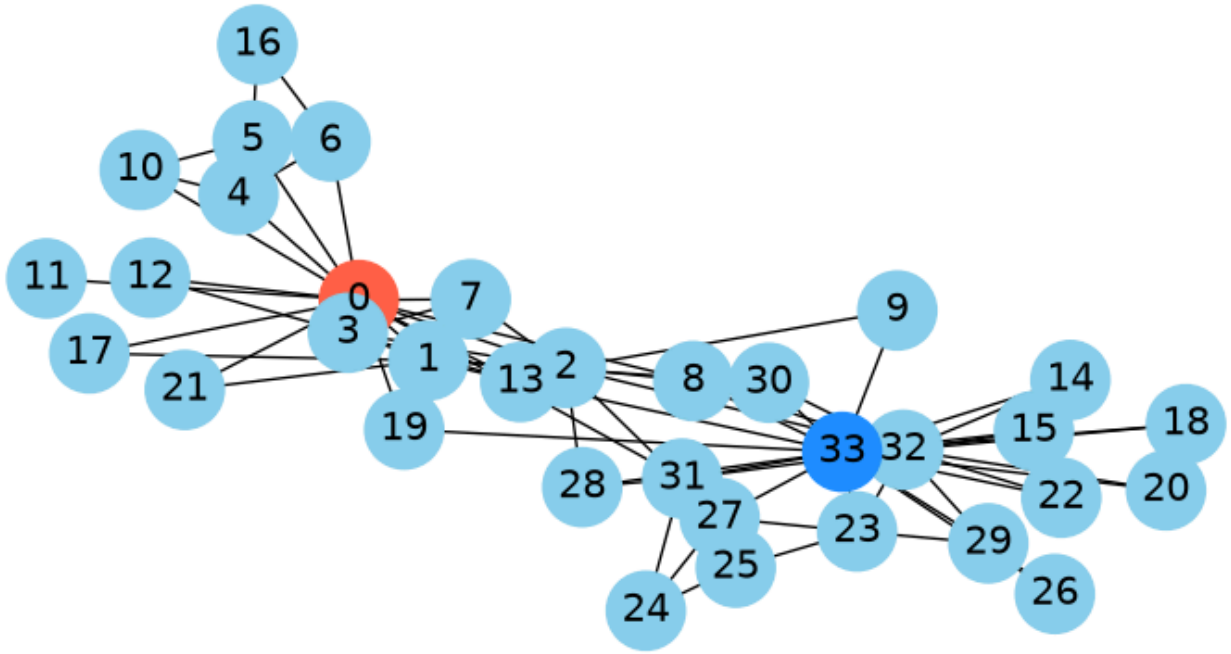
This is a very famous problem that can be solved in time that is polynomial in the number of nodes $|V|$ and edges $|E|$.

Increasingly better solutions have been found over the past 60+ years.

What can a min s-t cut tell us about a graph?

Let's look at the karate club, in which we've highlighted the president and the instructor (in red and blue, respectively).

▶ Code

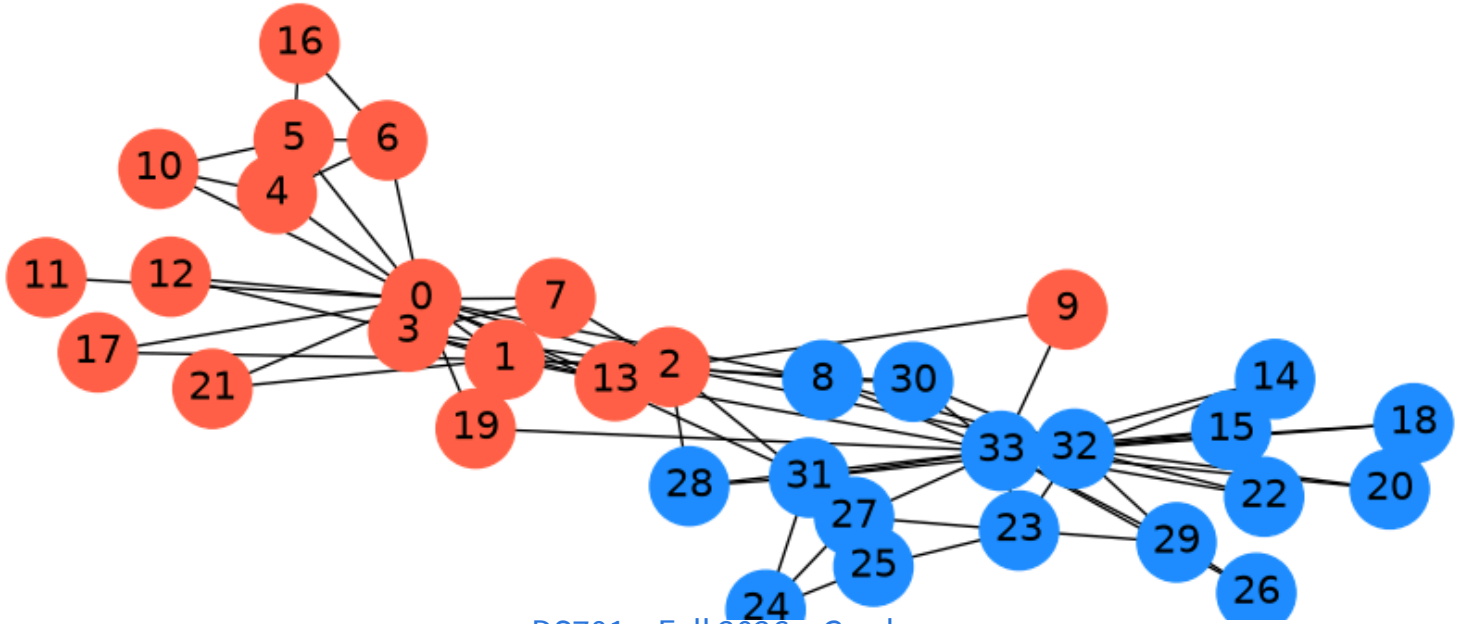


As mentioned, when Wayne Zachary studied the club, a conflict arose between the instructor and the president (nodes 0 and 33, respectively).

Zachary predicted the way the club would split based on an s-t min cut between the president and the instructor.

In fact, he **correctly** predicted every single member's eventual association except for node 8.

▶ Code

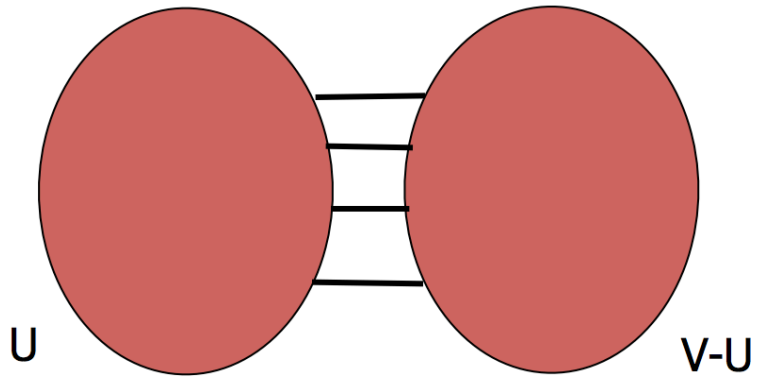


Minimum Cuts

In partitioning a graph, we may not have any particular s and t in mind.

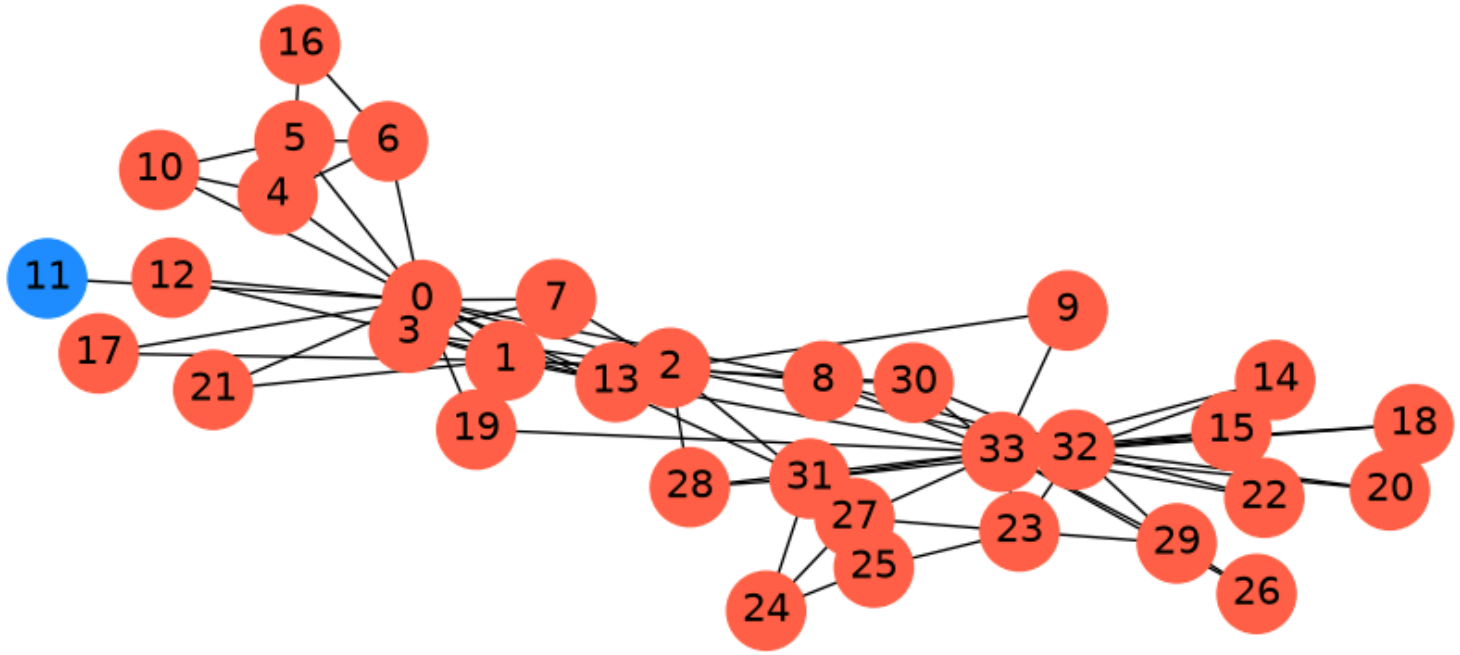
Rather, we may want to simply find the *minimal* way to disconnect the graph.

Clearly, we can do this using an s-t min cut, by simply trying all s and t pairs.



Let's try this approach of finding the minimum $s - t$ cut over all possibilities in the karate club graph.

▶ Code



This is in fact the minimum cut. Node 11 only has one edge to the rest of the graph, so the min cut is 1.

As this example shows, minimum cut is not, in general, a good approach for clustering or partitioning.

To get a more useful partition, we need to define a new goal. The new goal will be to find a **balanced cut**.

Balanced Cuts

The idea to avoid the problem above is to normalize the cut by the size of the **smaller** of the two components. The problem above would be avoided because the smaller of the two cuts is just a single node.

This leads us to define the **isoperimetric ratio**

$$\alpha = \frac{E(U, V \setminus U)}{\min(|U|, |V \setminus U|)},$$

and the **isoperimetric number of G**

$$\alpha(G) = \min_{U} \frac{E(U, V \setminus U)}{\min(|U|, |V \setminus U|)},$$

where $E(U, V \setminus U)$ is the number of edges between the two parts.

The idea is that finding $\alpha(G)$ gives a *balanced cut* – one that maximizes the number of disconnected nodes per edge removed.

Unfortunately, this is a challenging problem that is not computable in polynomial time.

However, we can make good approximations, which we'll look at now.

To do so, we'll introduce **spectral graph theory**.

Spectral Graph Theory

Spectral graph theory is the use of linear algebra to study the properties of graphs.

To introduce spectral graph theory, we define some terms.

Let G be an undirected graph with n nodes. We define the $n \times n$ matrix D as a diagonal matrix of node degrees, i.e., $D = \text{diag}(d_1, d_2, d_3, ...)$ where d_i is the *degree* of node i.

Assuming G has adjacency matrix A, we define the **Laplacian** of G as

$$L = D - A.$$

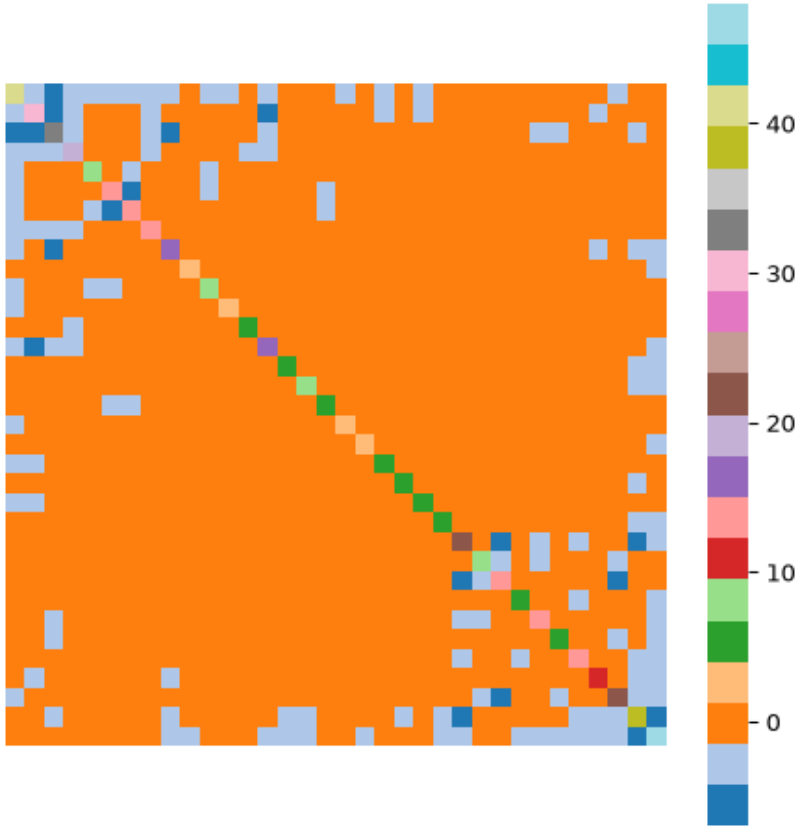
Where the matrix has positive entries on the diagonal and negative, symmetric entries off the diagonal.

If you want to study this in more detail, some excellent references are

- Allow Me to Introduce Spectral and Isoperimetric Graph Partitioning by Jonathan Shewchuck, which has outstanding visualizations and physical intuition.
- Spectral and Algebraic Graph Theory by Daniel Spielman which provides proofs and much more detail.

Below we show the Laplacian matrix L for the karate club network as a heatmap.

▶ Code



Laplacian Matrix Interpretation

Now let us think about an n-component vector $\mathbf{x} \in \mathbb{R}^n$ as an **assignment of values** to nodes in the graph G.

For example, $\mathbf{x}$ could encode node *importance* or *strength* or even a more concrete notion like *temperature* or *altitude*.

Laplacian Quadratic Form

Here is an amazing fact about the Laplacian of G.

The quadratic form

$$\mathbf{x}^T L\mathbf{x} = \sum_{(i,j)\in E} (x_i - x_j)^2,$$

i.e., $\mathbf{x}^T L\mathbf{x}$ is the sum of squared differences of $\mathbf{x}$ **over the edges in** G**.**

A proof of this is given in the web notes.

Now, let's think about a vector $\mathbf{x}$ that **minimizes** the differences over the edges in the graph.

We can express this in terms of the graph Laplacian:

$$\min_{\|\mathbf{x}\|=1} \sum_{(i,j)\in E} (x_i - x_j)^2 = \min_{\|\mathbf{x}\|=1} \mathbf{x}^T L \mathbf{x}.$$

From linear algebra, we know that when

$$\lambda = \min_{\|\mathbf{x}\|=1} \mathbf{x}^T L \mathbf{x},$$

then λ is the **smallest eigenvalue of** L and $\mathbf{x}$ is the corresponding eigenvector.

We are connecting functions on the graph G with eigenvectors of the matrix L. This is quite remarkable.

This is the standard eigenvalue equation with the constraint that $\mathbf{x}$ is a unit vector.

What do we know about L?

1. L is **symmetric.** Therefore the eigenvectors of L are orthogonal and its eigenvalues are real.
2. L is **positive semidefinite**, e.g. $\mathbf{x}^T L\mathbf{x} \geq 0$ for all $\mathbf{x}$.
    - Therefore the eigenvalues of L are all positive or zero.

We can order the eigenvalues from largest to smallest $\lambda_n \geq \cdots \geq \lambda_2 \geq \lambda_1 \geq 0$.

How do we know that L is positive semidefinite?

- Consider $\sum_{(i,j)\in E} (x_i - x_j)^2$. This is always a nonnegative quantity.
- As a result, $\mathbf{x}^T L\mathbf{x} \geq 0$ for all $\mathbf{x}$.

Assume that G is *connected* (e.g. there is a path between any two nodes).

Then L has a single eigenvalue of value $\lambda_1 = 0$. The corresponding eigenvector is $\mathbf{w}_1 = \mathbf{1} = [1, 1, 1, ...]^T$.

It is easy to verify that

$$L\mathbf{1} = 0 \cdot \mathbf{1} = \mathbf{0}.$$

Recall that row i of L consists of d_i on the diagonal, and d_i -1s in other positions.

The second-smallest eigenvalue of L, λ_2, is called the **Fiedler value.**

We know that all of the other eigenvectors of L are orthogonal to $\mathbf{1}$, because L is symmetric.

As a result, a definition of the second smallest eigenvalue is:

$$\lambda_2 = \min_{\|\mathbf{x}\|=1,\ \mathbf{x}\perp\mathbf{1}} \mathbf{x}^T L\mathbf{x}.$$

Note that $\mathbf{x} \perp \mathbf{1}$ means that the sum of the entries in $\mathbf{x}$ is zero, which implies that $\mathbf{x}$ is **mean-centered**.

Fiedler Vector

The corresponding eigenvector to λ_2 is called the **Fiedler vector.**

It minimizes

$$\mathbf{w}_2 = \arg \min_{\|\mathbf{x}\|=1,\ \mathbf{x} \perp \mathbf{1}} \sum_{(i,j) \in E} (x_i - x_j)^2.$$

If we think of x_i as a 1-D coordinate for node i in the graph, then choosing $\mathbf{x} = \mathbf{w}_2$ (the eigenvector corresponding to λ_2) puts each node in a position that minimizes the sum of the squared stretching of each edge.

Spectral Partitioning

This leads to key ideas in node partitioning.

The basic idea is to partition nodes according to the Fiedler vector $\mathbf{w}_2$.

This can be shown to have provably good performance for the **balanced cut** problem.

There are a number of options for how to split based on the Fiedler vector.

If $\mathbf{w}_2$ is the Fiedler vector, then split nodes according to a value s:

- bisection: s is the median value in $\mathbf{w}_2$
- sign: separate positive and negative vaues ($s = 0$)
- gap: separate according to the largest gap in the values of $\mathbf{w}_2$
- ratio cut: s is the value that maximizes α

See Spectral and Algebraic Graph Theory by Daniel Spielman, Chapter 20, where it is proved that for every $U \subset V$ with $|U| \leq |V|/2, \alpha(G) \geq \lambda_2(1-s)$ where $s = |U|/|V|$. In particular, $\alpha(G) \geq \lambda_2/2$.

Example: Bisection Method

Consider a graph with 8 nodes where the Fiedler vector has values (sorted for convenience):

$$\mathbf{w}_2 = [-0.6, -0.4, -0.2, -0.1, 0.15, 0.3, 0.45, 0.5]$$

Bisection approach: Use $s = \text{median}(\mathbf{w}_2) = \frac{-0.1+0.15}{2} = 0.025$

Result:

- **Group 1** (values < 0.025): nodes 1, 2, 3, 4
- **Group 2** (values ≥ 0.025): nodes 5, 6, 7, 8

This ensures balanced partition sizes (4 nodes each).

Example: Sign Method

Using the same Fiedler vector:

$$\mathbf{w}_2 = [-0.6, -0.4, -0.2, -0.1, 0.15, 0.3, 0.45, 0.5]$$

Sign approach: Use $s = 0$ to separate positive from negative values

Result:

- **Group 1** (negative values): nodes 1, 2, 3, 4
- **Group 2** (positive values): nodes 5, 6, 7, 8

For this example, the sign method gives the same partition as bisection, but this won't always be the case.

Example: Gap Method

Using the same Fiedler vector:

$$\mathbf{w}_2 = [-0.6, -0.4, -0.2, -0.1, 0.15, 0.3, 0.45, 0.5]$$

Gap approach: Find the largest gap between consecutive sorted values

Gaps: $0.2, 0.2, 0.1, 0.25, 0.15, 0.15, 0.05$

Largest gap is 0.25 (between -0.1 and 0.15), so $s = 0.025$

Result:

- **Group 1** (values < 0.025): nodes 1, 2, 3, 4
- **Group 2** (values ≥ 0.025): nodes 5, 6, 7, 8

This identifies the most natural "break" in the data.

Example: Ratio Cut Method

Consider a different scenario where we try several cut values:

For $s = -0.3$:

- Group 1: nodes 1, 2 (size = 2)
- Group 2: nodes 3, 4, 5, 6, 7, 8 (size = 6)
- Cut edges: 5, so isoperimetric ratio $\alpha = \frac{5}{\min(2,6)} = 2.5$

For $s = 0.025$:

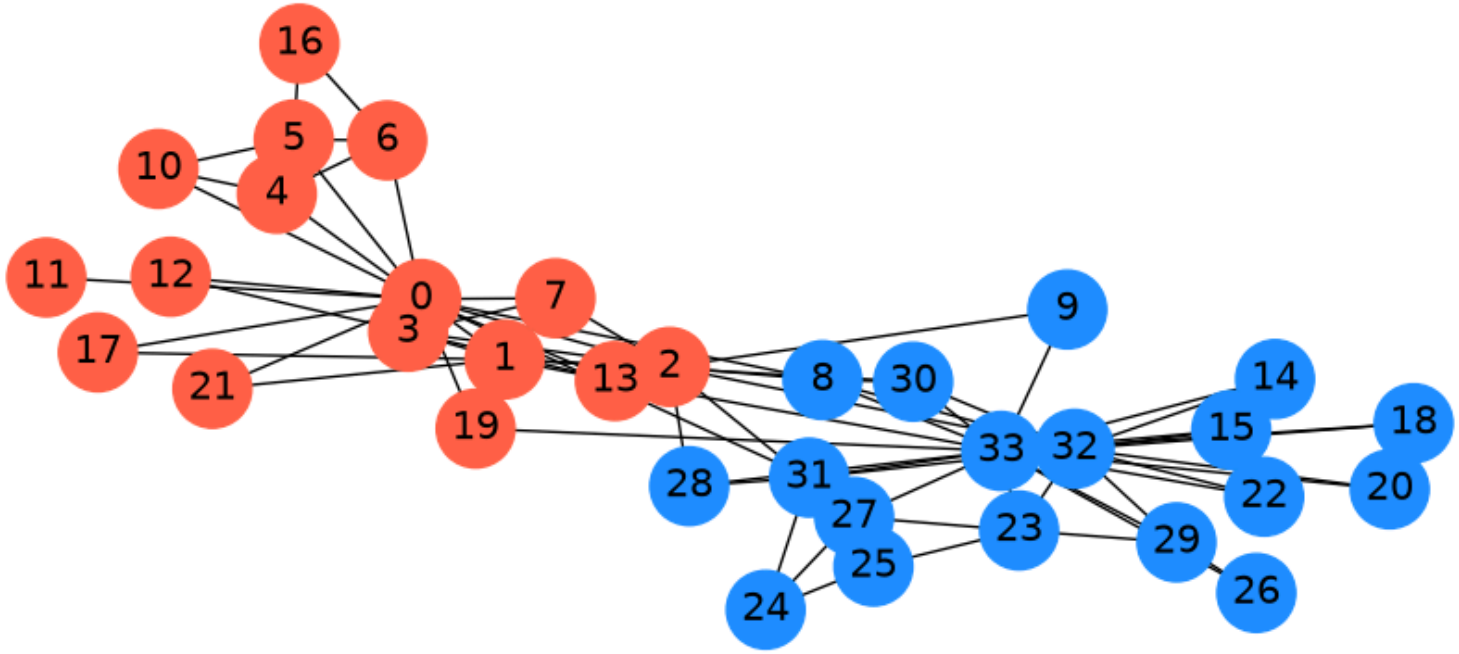
- Group 1: nodes 1, 2, 3, 4 (size = 4)
- Group 2: nodes 5, 6, 7, 8 (size = 4)
- Cut edges: 8, so isoperimetric ratio$\alpha = \frac{8}{\min(4,4)} = 2.0$

Ratio cut approach: Choose s that maximizes α

The actual implementation searches over possible cut values to find the optimal partition

Here is a spectral parititioning for the karate club graph.

▶ Code



Interestingly, this is almost the same as the s-t min cut based on the president and instructor.

Spectral Clustering

In many cases we would like to move beyond graph partitioning, to allow for clustering nodes into, say, k clusters.

The idea of spectral clustering takes the observations about the Fiedler vector and extends them to more than one dimension.

See the notes for more on spectral clustering.

Recap

Networks are present in many interesting applications.

We have covered the following topics

- Graph representations
- Cluster coefficients
- Centrality measures
- Clustering – (see notes)

In addition we used the NewtorkX package to visualize our networks in the following formats

- circular
- spring
- spectral