Network Analysis

Dr. Shaina Race
Institute for Advanced Analytics

Community Detection

• • •

i.e. Clustering

Clustering in Graphs

1. Can still use classical algorithms (i.e. **k-means**)

2. …*Or* choose an algorithm specifically for graphs

3. Fundamental Theorem: Nothing works best all the time!

Clustering in Graphs

1. Can still use classical algorithms (i.e. **k-means**)
   - **Edge weights should reflect similarity** and not distance
   - Use the **adjacency matrix like a data matrix**
   - The "observations" and "variables" are the same entities, but you simply characterize an observation by its similarity to others.

	A	B	C	D	E	F	G	H	I
A	9	9	9	4	3	1	1	0	0
B	9	9	9	8	1	0	0	0	3
C	9	9	9	0	0	0	0	1	0
D	4	8	0	9	1	9	1	1	0
E	3	1	0	1	9	0	0	0	5
F	1	0	0	9	0	9	8	8	0
G	1	0	0	1	0	8	9	9	7
H	0	0	1	1	0	8	9	9	7
I	0	3	0	0	5	0	7	7	9

2. …*Or* choose an … graphs
   - **Spectral** (Eigenvect…
   - **Modularity**
   - Minimum Spanning T…

3. Theorem: Nothi… ne!

Clustering in Graphs

1. Can still use classical algorithms (i.e. **k-means**)
   - **Edge weights should reflect similarity** and not distance
   - Use the **adjacency matrix like a data matrix**
   - The "observations" and "variables" are the same entities, but you simply characterize an observation by its similarity to others.

2. …*Or* choose an algorithm specifically for graphs
   - **Spectral** (Eigenvector) methods
   - **Modularity**
   - Minimum Spanning Trees

3. Theorem: Nothing works best all the time!

Spectral Clustering

• • •

(Spectral ➔ Eigenvalues/Eigenvectors. Yay!)

Not just for Graphs!

- Keep in mind the following methods operate on a similarity matrix (i.e. adjacency matrix).

- If you develop a notion of similarity using traditional data, these methods can be useful for clustering any data!

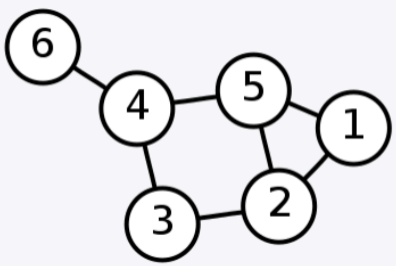
The Laplacian Matrix

- Spectral methods typically use a **Laplacian Matrix.**

- Let **A** be an adjacency matrix for a graph (or a similarity matrix for some data)
- Let **D** be a diagonal matrix containing the degrees (d_i of each node:
  - $D = diag\{d_1, d_2, \ldots, d_n\}$
- The **Laplacian matrix** is defined as **L = D – A**

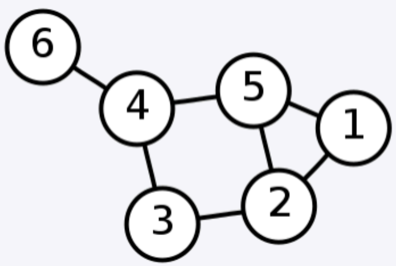
The Laplacian Matrix

Example

Labelled graph	Degree matrix	Adjacency matrix	Laplacian matrix
	$\begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 2 & -1 & 0 & 0 & -1 & 0 \\ -1 & 3 & -1 & 0 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 3 & -1 & -1 \\ -1 & -1 & 0 & -1 & 3 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \end{pmatrix}$

The Laplacian Matrix

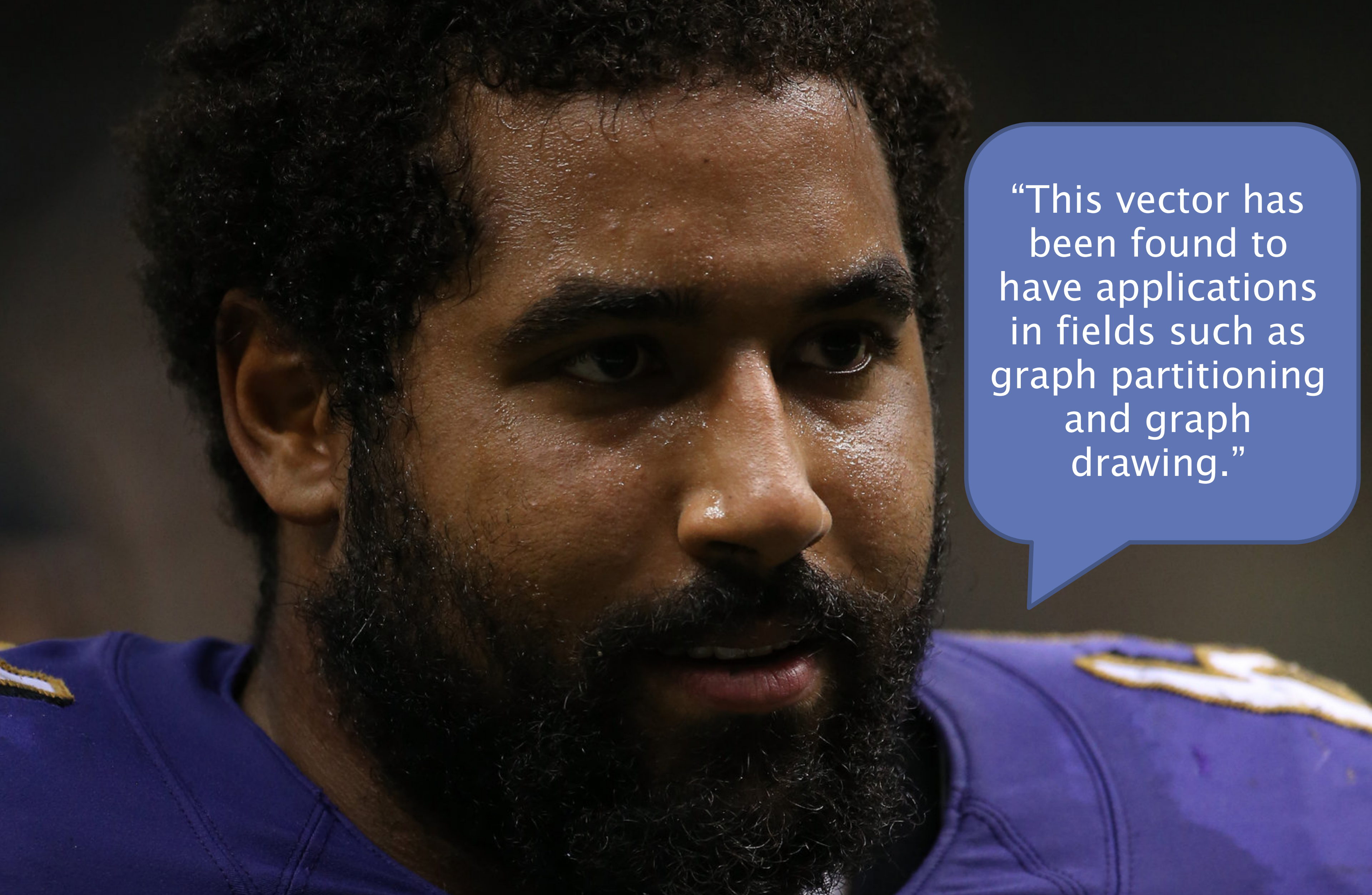
Example

Labelled graph	Degree matrix	Adjacency matrix	Laplacian matrix
	$\begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 2 & -1 & 0 & 0 & -1 & 0 \\ -1 & 3 & -1 & 0 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 3 & -1 & -1 \\ -1 & -1 & 0 & -1 & 3 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \end{pmatrix}$

Simple Spectral Clustering

- The **Laplacian matrix** is defined as L = D-A
- The **Fiedler vector** is the eigenvector associated with the ***second smallest*** eigenvalue of L.
- The Fiedler vector is known to contain information about optimally partitioning a graph

Baltimore Ravens' ex-Offensive Lineman John Urschel can tell you about it

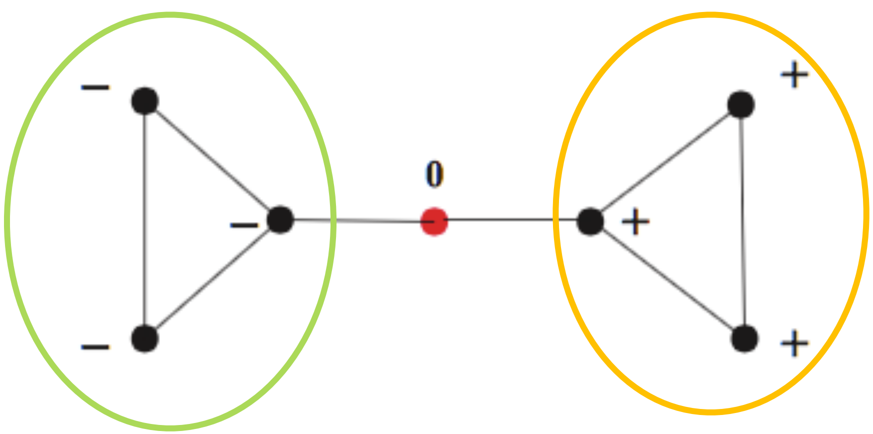


A Cascadic Multigrid Algorithm for Computing the Fiedler Vector of Graph Laplacians
John C. Urschel, Xiaozhe Hu, Jinchao Xu, Ludmil T. Zikatanov 12/2014

Simple Spectral Clustering

Use the signs of the entries in the Fiedler vector.

- Nodes associated with positive entries in one cluster
- Nodes associated with negative entries in second cluster
- Arbitrarily assign nodes associated with zero entries
  (often called **Articulation Points** – sometimes they are brokers)

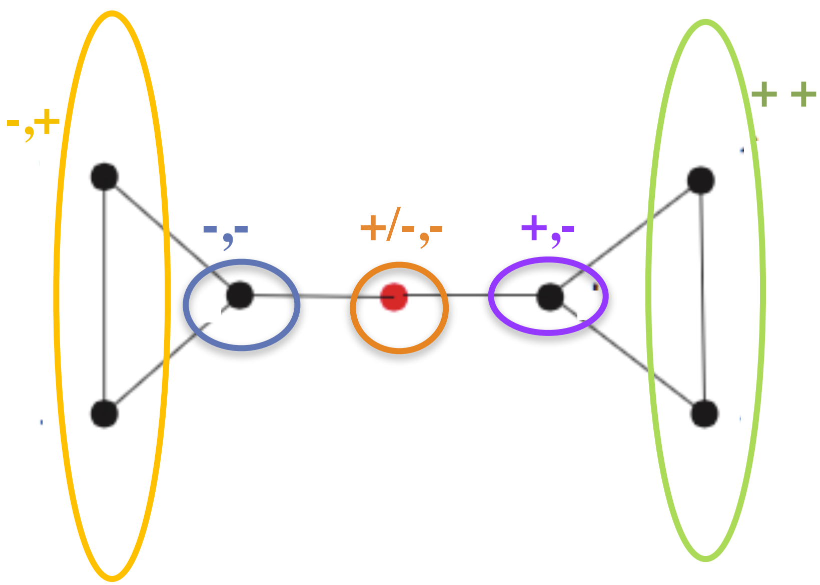


$$\mathbf{v}_6 = \begin{pmatrix} -0.44 \\ -0.44 \\ -0.33 \\ 0 \\ 0.33 \\ 0.44 \\ 0.44 \end{pmatrix}$$

Simple Spectral Clustering

How to get more than two clusters? (Two Ways)

- Repeat process on each cluster.
- Use additional eigenvectors (Two Ways)
    - Use the sign patterns (shown below)
    - Cluster the rows of the eigenvectors with k-means (Next Slide)



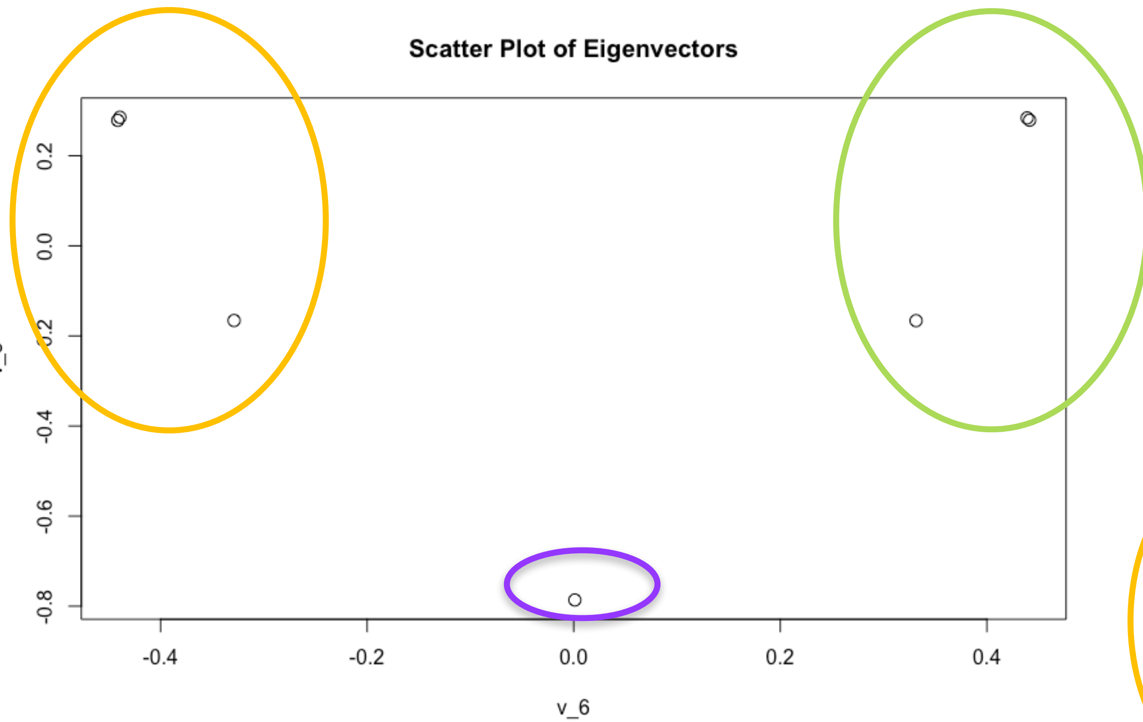
$$\mathbf{v}_6 = \begin{pmatrix} -0.44 \\ -0.44 \\ -0.33 \\ 0 \\ 0.33 \\ 0.44 \\ 0.44 \end{pmatrix} \quad \mathbf{v}_5 = \begin{pmatrix} 0.28 \\ 0.28 \\ -0.16 \\ -0.79 \\ -0.16 \\ 0.28 \\ 0.28 \end{pmatrix}$$

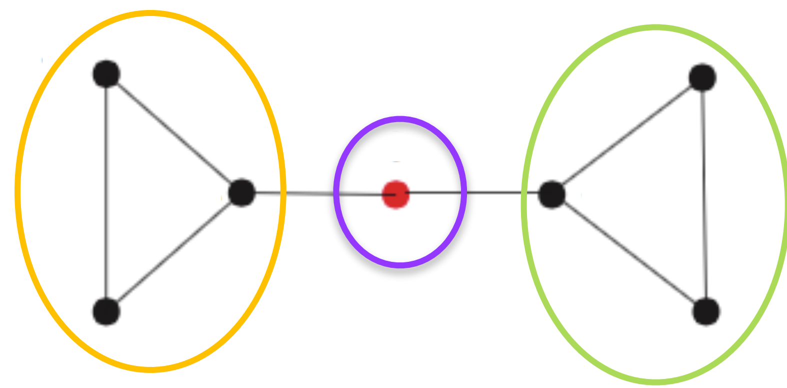
Simple Spectral Clustering

Use k-means to cluster the rows of the eigenvectors

- (Get to choose k in this case)

$$\mathbf{v}_6 = \begin{pmatrix} -0.44 \\ -0.44 \\ -0.33 \\ 0 \\ 0.33 \\ 0.44 \\ 0.44 \end{pmatrix} \quad \mathbf{v}_5 = \begin{pmatrix} 0.28 \\ 0.28 \\ -0.16 \\ -0.79 \\ -0.16 \\ 0.28 \\ 0.28 \end{pmatrix}$$





Advanced Spectral Clustering

- NCUT
- Ratio Cut
- Normalized Spectral Clustering
- ...Long list of algorithms

- *Most* involve the Laplacian Matrix (normalized in different ways), and k-means run on Eigenvectors (normalized in different ways)

Modularity Maximization

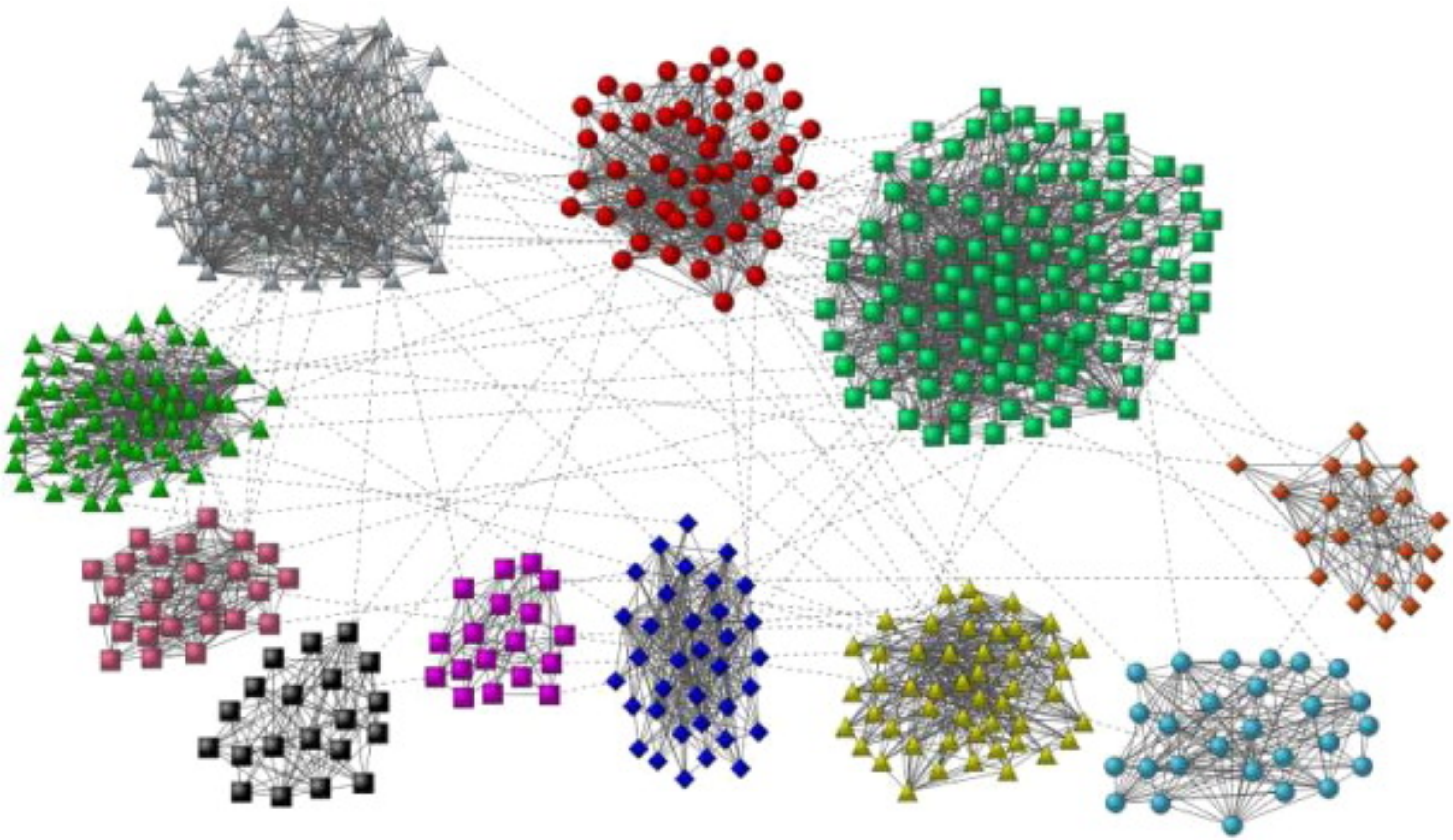
• • •

Mark Newman et. al

Modularity Maximization

- Currently the **most popular** algorithm for community detection.
- Developed in 2006 by Mark Newman (UMichigan)
- Algorithm **Intuition**:
  - Compare the observed network to what you would expect to find at random.
  - Where are there more edges than expected?
  - These areas may define communities.

Modularity Maximization

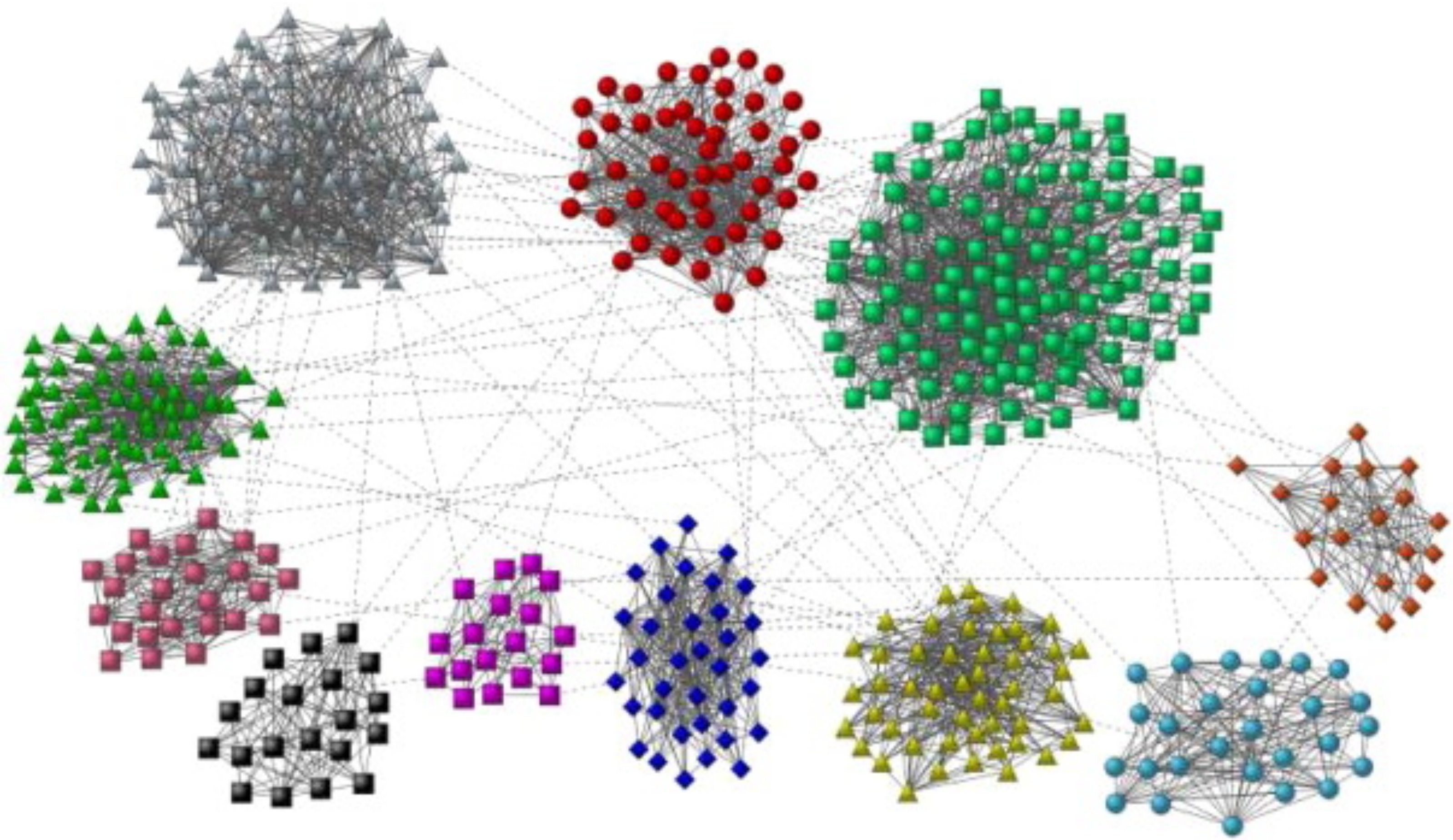


Modularity

- **Modularity** is a number that describes the ***extent to which given groups form communities in a graph.***
- Fraction of edges within groups minus the *expected* fraction if edges were distributed at random.
- Number in range [-1, 1)
  - **negative $\longrightarrow$ random partition**
    - (We'd expect to find *more* edges within our groups if they were distributed at random)
  - **nearer 1 $\longrightarrow$ better communities (=1 $\longrightarrow$ components)**
    - (We see far more edges within our groups than we'd expect to find at random)

Modularity Maximization

Picks the partitioning of the vertices that maximizes the modularity.



Algorithm 13 Modularity Procedure for Network Community Detection (Newman) [94]

Input: $n \times n$ adjacency matrix $\mathbf{A}$ for an undirected graph to be partitioned

1. Let d_i be the i^{th} row sum of $\mathbf{A}$. Let $d = \sum_{i=1}^{n} d_i$
2. Form the matrix $\mathbf{P}$ with $\mathbf{P}_{ij} = d_i d_j / d$.
3. Form the modularity matrix $\mathbf{B} = \mathbf{A} - \mathbf{P}$.
4. Compute the largest eigenvalue λ_1 and corresponding eigenvector $\mathbf{u}_1$ of $\mathbf{B}$.
5. If $\lambda_1 < 0$, stop. There is no partition of this graph.
6. Otherwise partition the vertices of the graph into 2 clusters as follows

$$\begin{aligned} C_1 &= \{i : \mathbf{u}_1(i) < 0\} \\ C_2 &= \{i : \mathbf{u}_1(i) \geq 0\} \end{aligned} \tag{3.11}$$

7. Determine further partitions by extracting the rows and columns of the original adjacency matrix corresponding to the vertices in each cluster to form $\mathbf{A}'$ and repeat the algorithm with $\mathbf{A}'$ until each created cluster fails to partition in step 5.

Output: Final clusters.

Modularity Maximization

Advantages

- **Automatically determines number of clusters**
- **Intuitive** rationale for/definition of a community
- **Easy** to program and compute

Disadvantages

- Node can belong to only one community **(hard clustering)**
- If first eigenvalue of "modularity matrix" is negative – no clusters.
    - (could be an advantage!)

Minimum Spanning Trees

• • •

An alternative

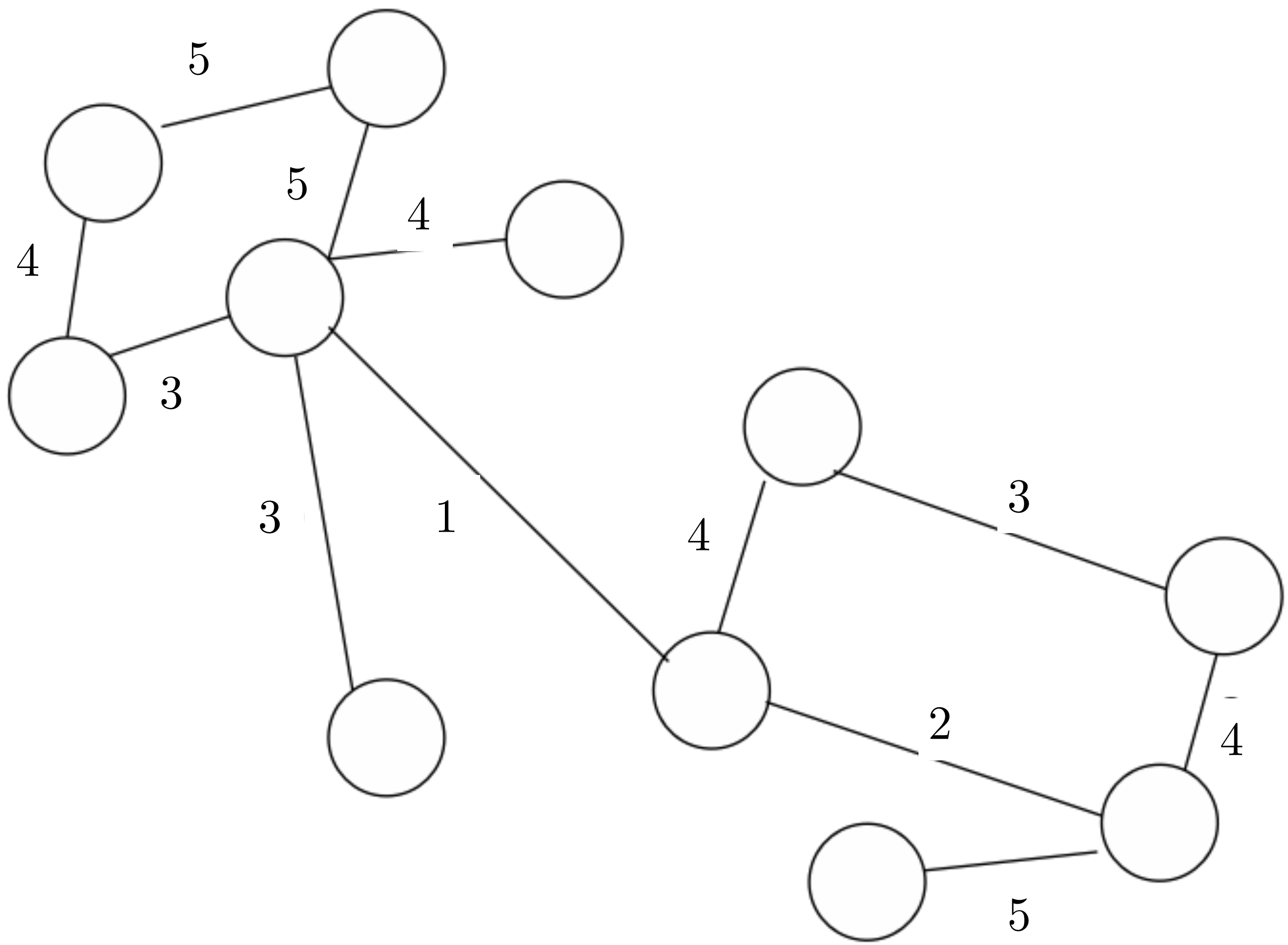
Minimum Spanning Trees

(or maximal spanning trees in the case of network similarity)

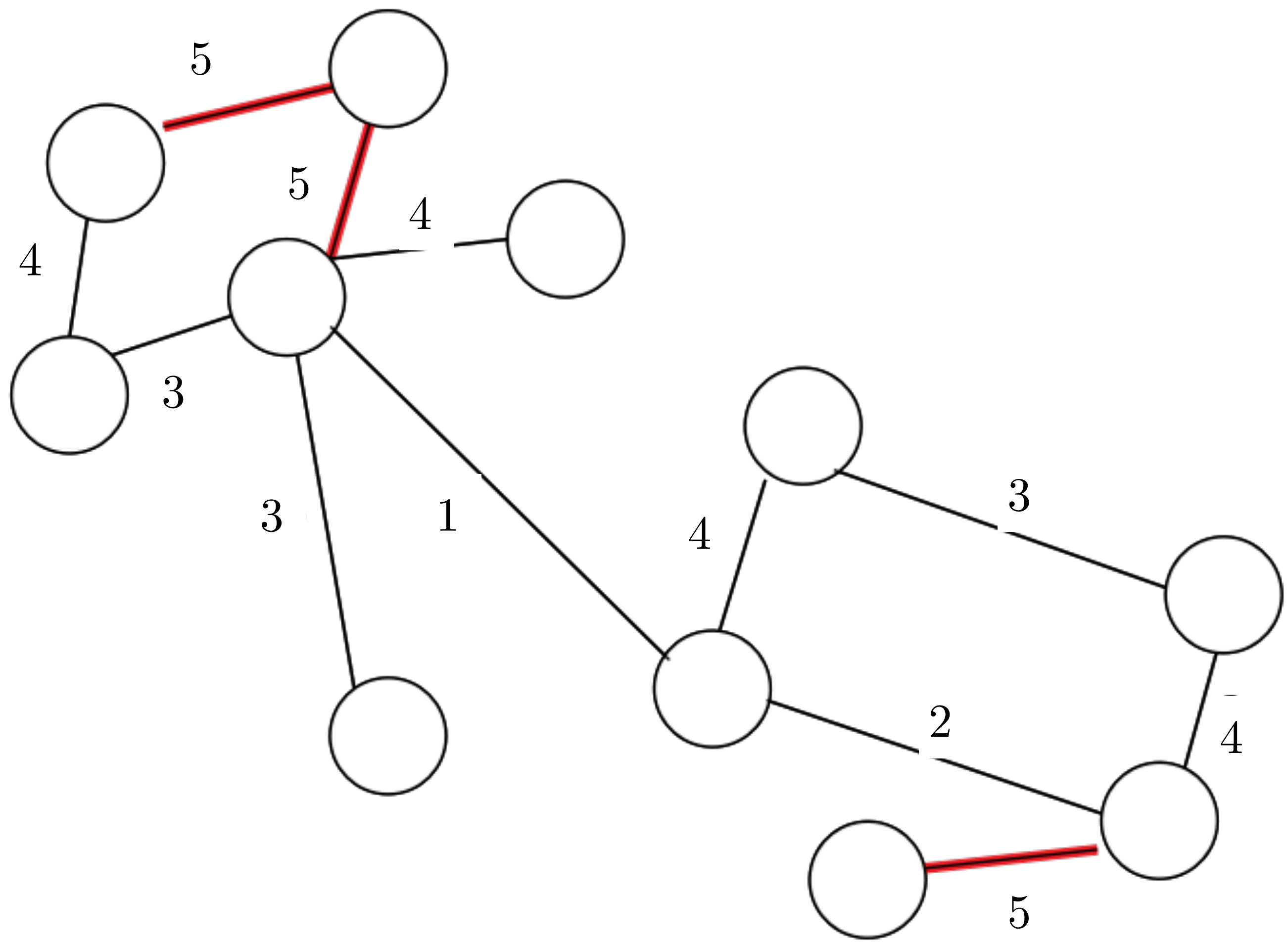
- Equivalent to **Single Linkage hierarchical clustering**.
- Creates a **tree** (graph with no cycles) that connects every node.
- Cutting all edges of the tree whose weight doesn't meet a pre-specified threshold will result in clusters.
- Changing threshold changes the number of clusters.

Building the *Maximum* Spanning Tree

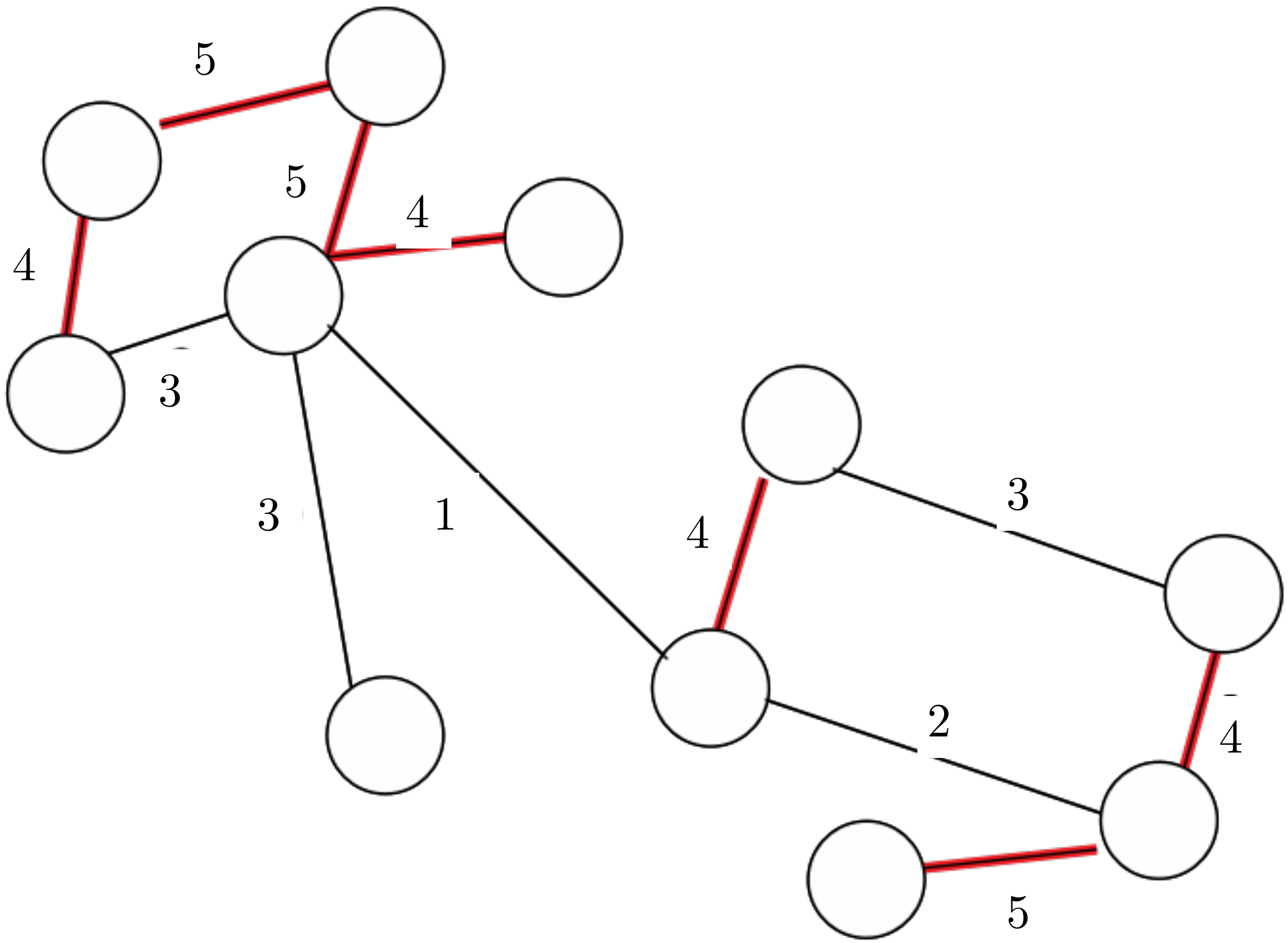
(In most graphs, edge weights reflect similarity)



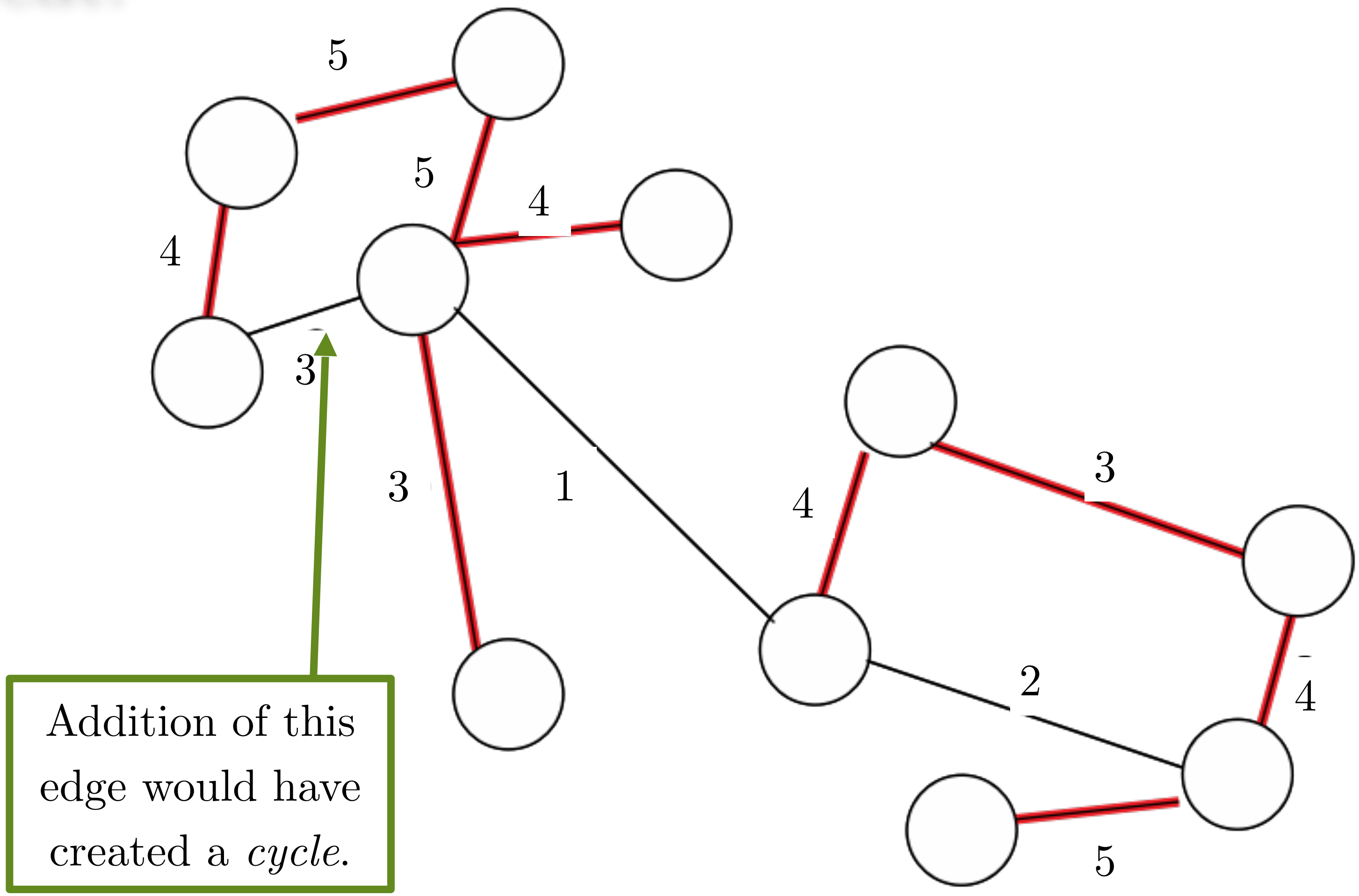
Add the most similar edges to the spanning tree, as long as no cycles are created. Repeat.



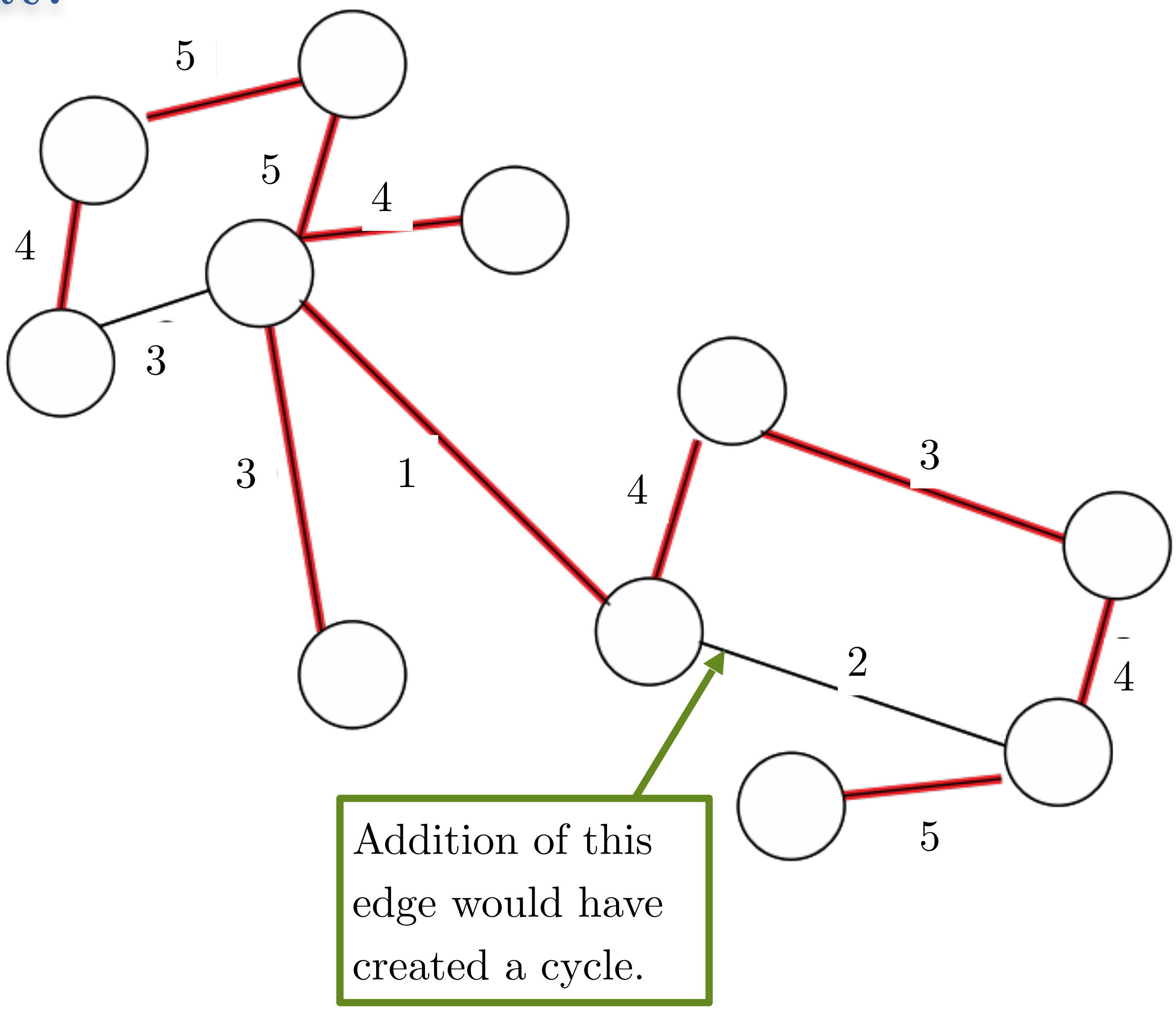
Add the most similar edges to the spanning tree, as long as no cycles are created. Repeat.



Add the most similar edges to the spanning tree, as long as no cycles are created. Repeat.

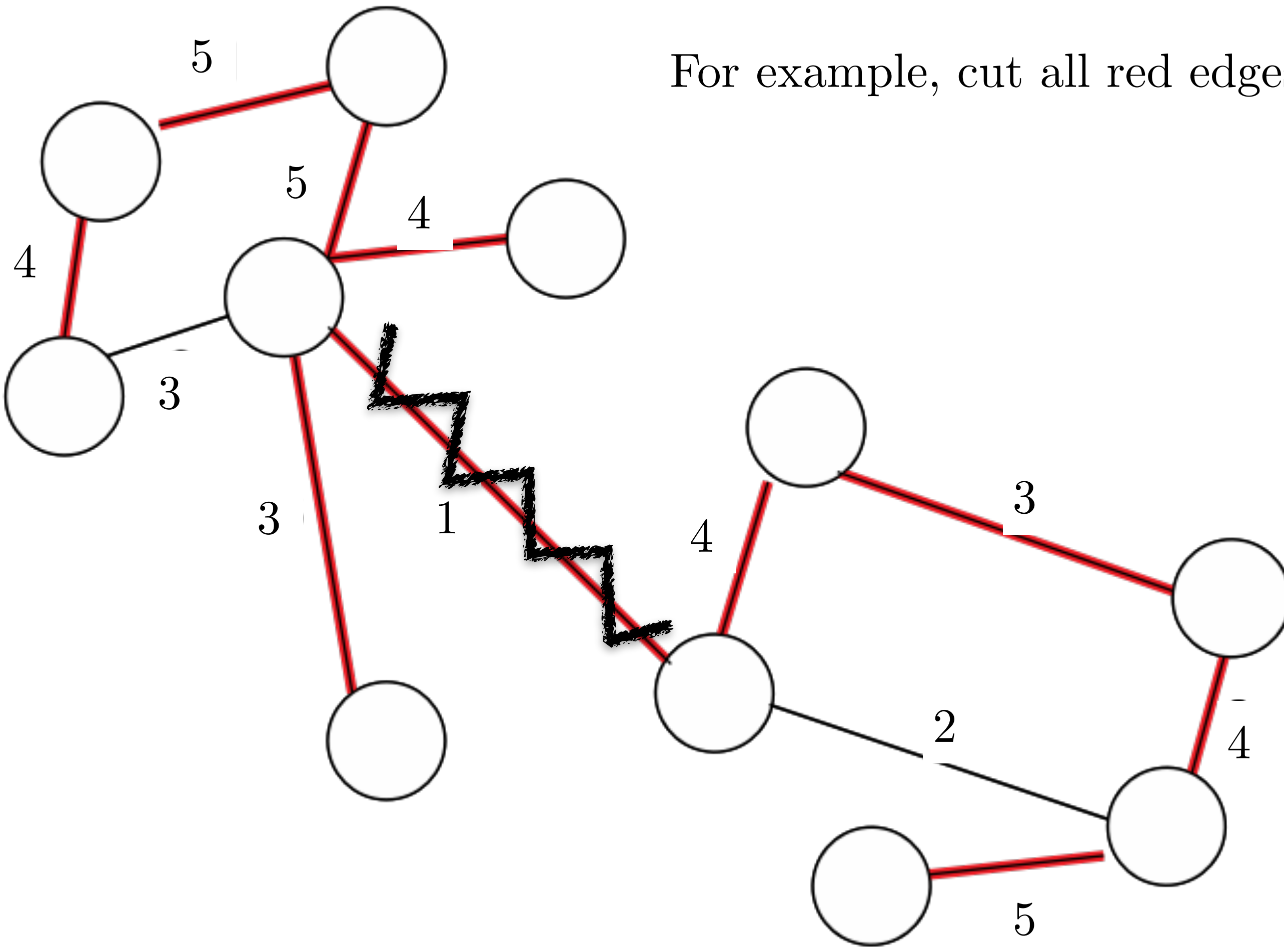


Add the most similar edges to the spanning tree, as long as no cycles are created. Repeat.



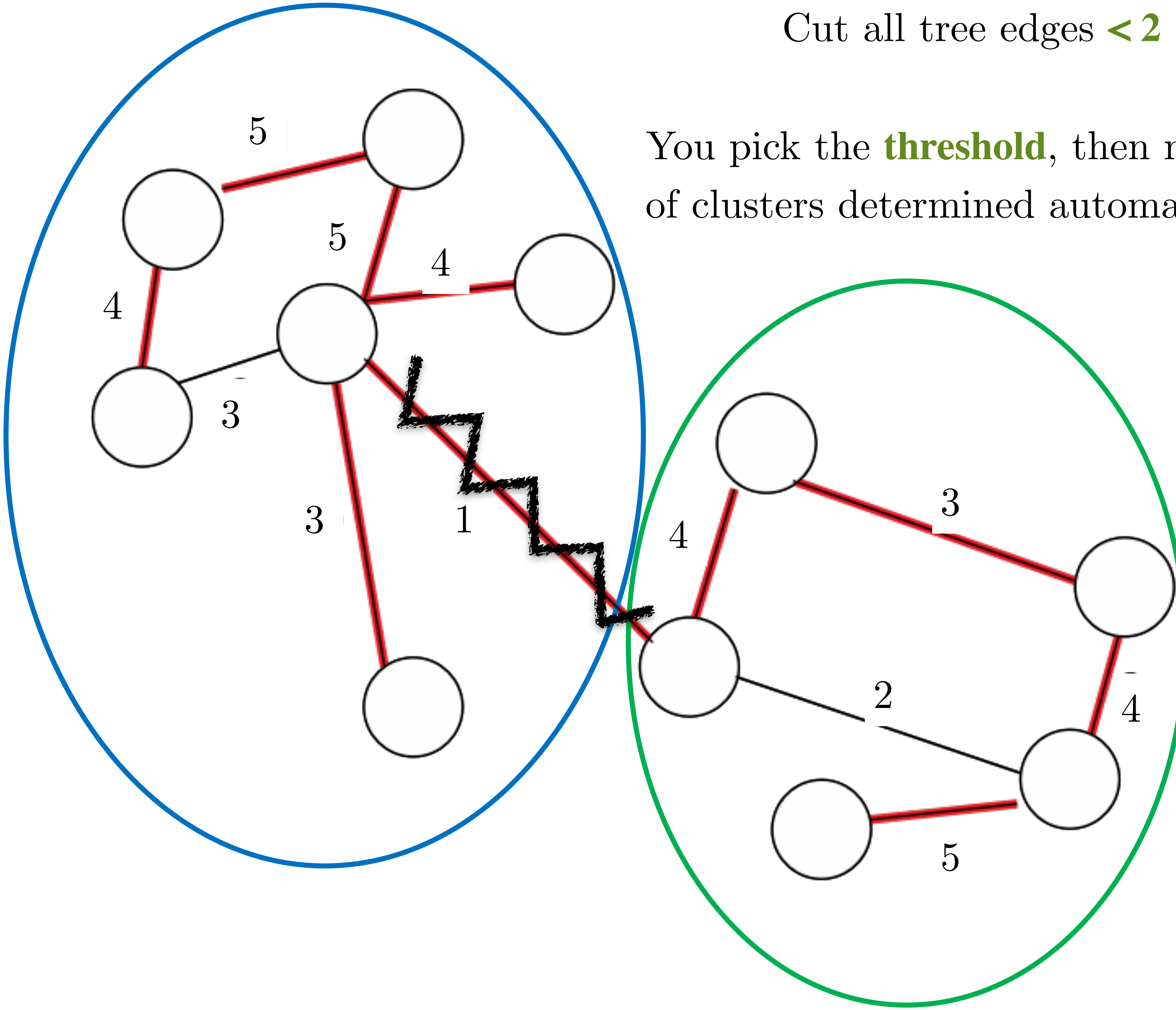
Cut the tree at some threshold.

For example, cut all red edges < 2



Cut all tree edges **< 2**

You pick the **threshold**, then number of clusters determined automatically.



Ensemble Clustering

- Try many different clustering algorithms
- (Or even k-means with different starting points)
- **Create a network where the weight of the edge connecting object i to object j is the number of times that object i was clustered with object j.**
- Cluster the resulting network using any method

Ensemble Clustering