# Spectral Graph Embedding

## Introduction to Network Science

Instructor: Michele Starnini — https://github.com/chatox/networks-science-course

Universitat Pompeu Fabra Barcelona

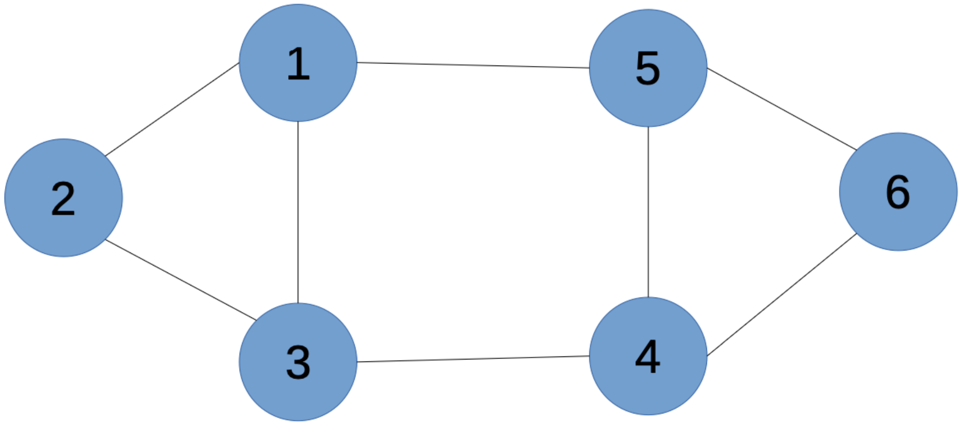
# Contents

- Graph Laplacian
- Application: Embedding a graph

# Graph Laplacian

# Adjacency matrix

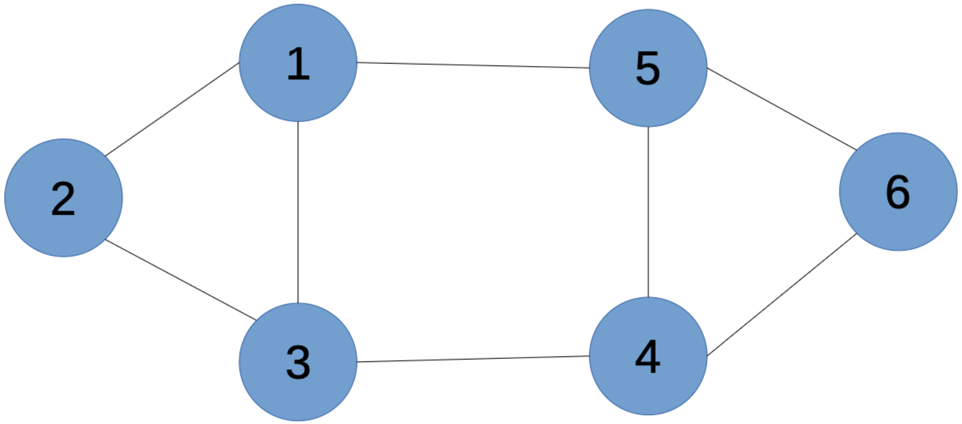
$$A_{ij} = \begin{cases} 1 & \text{if } (i,j) \in E \\ 0 & \text{otherwise} \end{cases}$$




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

# Degree matrix

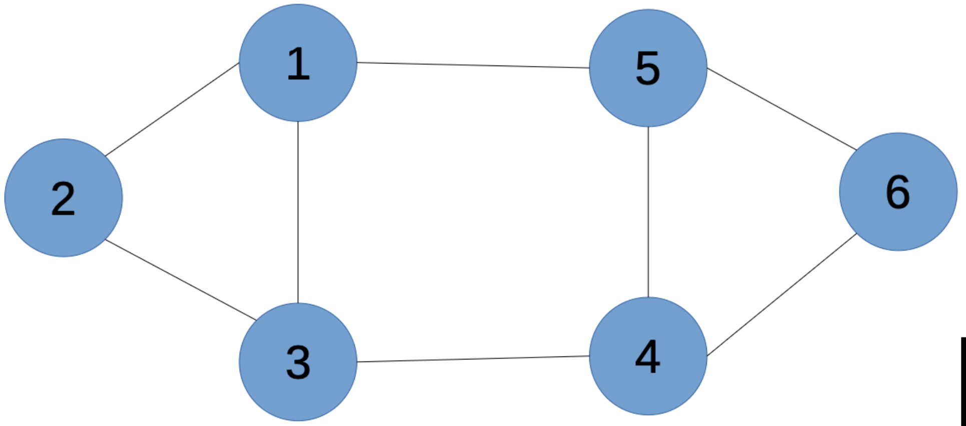
$$D_{ij} = \begin{cases} k_i & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$




$$D = \begin{bmatrix} 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 \end{bmatrix}$$

# Laplacian matrix

$$L = D - A$$




$$L = \begin{bmatrix} 3 & -1 & -1 & 0 & -1 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 \\ -1 & -1 & 3 & -1 & 0 & 0 \\ 0 & 0 & -1 & 3 & -1 & -1 \\ -1 & 0 & 0 & -1 & 3 & -1 \\ 0 & 0 & 0 & -1 & -1 & 2 \end{bmatrix}$$

Because A is symmetric, and we have only changed the diagonal, **L is symmetric.**

# Laplacian matrix L = D - A

$$L\vec{1} = \begin{bmatrix} 3 & -1 & -1 & 0 & -1 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 \\ -1 & -1 & 3 & -1 & 0 & 0 \\ 0 & 0 & -1 & 3 & -1 & -1 \\ -1 & 0 & 0 & -1 & 3 & -1 \\ 0 & 0 & 0 & -1 & -1 & 2 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = ?$$

# The constant vector is an eigenvector of L

The constant vector $x=[1,1,...,1]^T$ is an eigenvector of the Laplacian, and has eigenvalue $0$

$$Lx = \begin{bmatrix} 3 & -1 & -1 & 0 & -1 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 \\ -1 & -1 & 3 & -1 & 0 & 0 \\ 0 & 0 & -1 & 3 & -1 & -1 \\ -1 & 0 & 0 & -1 & 3 & -1 \\ 0 & 0 & 0 & -1 & -1 & 2 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = 0 \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

Does it need to be this specific graph? Why?
Does it need to be the vector [1, 1, ..., 1]? Why?

# If the graph is disconnected

- If the graph is disconnected into two components, the same argument as for the adjacency matrix applies, and

$$\lambda_1 = \lambda_2 = 0$$

- The multiplicity of eigenvalue 0 is equal to the number of connected components

Let's compute this quantity.
Is it: 1) a matrix, 2) a vector, 3) a number?

$$x^T L x$$

# Prove this!

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

$$L_{ij} = D_{ij} - A_{ij}$$

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

Assume that E only contains each edge in one direction

Think of this quantity as the "stress" produced by the assignment of node labels x

# Proof

$$
\begin{aligned}
x^T L x &= \sum_{i=1}^{n} \sum_{j=1}^{n} L_{ij} x_i x_j \\
&= \sum_{i=1}^{n} \sum_{j=1}^{n} (D_{ij} - A_{ij}) x_i x_j \\
&= \sum_{i=1}^{n} k_i x_i^2 - \sum_{(i,j)\in E} 2 x_i x_j \\
&= \sum_{(i,j)\in E} \left(x_i^2 + x_j^2\right) - \sum_{(i,j)\in E} 2 x_i x_j \\
&= \sum_{(i,j)\in E} \left(x_i^2 + x_j^2 - 2 x_i x_j\right) = \sum_{(i,j)\in E} (x_i - x_j)^2
\end{aligned}
$$

# Proof (detail)

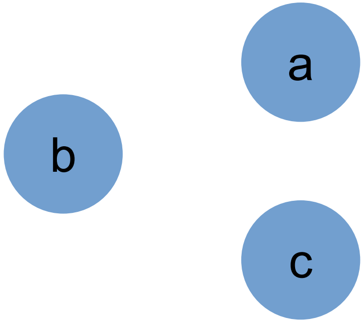
$$\sum_{i=1}^{n} k_i x_i^2 = \sum_{(i,j)\in E} \left(x_i^2 + x_j^2\right)$$

Node *u* appears in this sum $k_u$ times

The degree of node *u* is the number of times it is one of the ends of an edge in *E*

$$k_u = \left|\{(i,j) \in E : i = u \vee j = u\}\right|$$

# Example




$$E = \{(a,b),(b,c)\}$$
$$k_a = 1$$
$$k_b = 2$$
$$k_c = 1$$

$$\begin{aligned}\sum_{i=1}^{n} k_i x_i^2 &= k_a x_a^2 + k_b x_b^2 + k_c x_c^2 \\ &= x_a^2 + 2x_b^2 + x_c^2 \\ &= \left(x_a^2 + x_b^2\right) + \left(x_b^2 + x_c^2\right) \\ &= \sum_{(i,j)\in\{(a,b),(b,c)\}} \left(x_i^2 + x_j^2\right)\end{aligned}$$

# 1) All the eigenvalues of the Laplacian are non-negative

- If *v* is an eigenvector of *L* of eigenvalue $\lambda$:

$$\lambda v^T v = v^T L v = \sum_{(i,j)\in E} (v_i - v_j)^2 \geq 0$$

- This means all eigenvalues $\lambda$ are non-negative

# 2) Zero is always an eigenvalue of the Laplacian with eigenvector = the constant vector

- If $x$ is the eigenvector of eigenvalue 0, $Lx = 0$
- Then $$x^T L x = \sum_{(i,j)\in E} (x_i - x_j)^2 = 0$$

From this, we deduct that $x_i = x_j$ for any pair $i, j$
even if $i$ and $j$ are not directly connected by an edge. Why?

# The eigenvector *x* of $\lambda=0$ is the constant vector if the graph is connected

- If *x* is the eigenvector of eigenvalue 0, $Lx = 0$
- Then $$x^T L x = \sum_{(i,j)\in E} (x_i - x_j)^2 = 0$$
- Hence, for any pair of nodes *(i,j)* connected by an edge, $x_i = x_j$
- Given the graph is connected, there is a path between any two nodes ⇒

$x_i = x_j = x_k \ldots$ for **any** pair of nodes *(i,j),* **even the ones not connected by an edge**, $x_i = x_j$

- Hence *x* is a constant vector

# In summary, the Laplacian matrix L = D - A

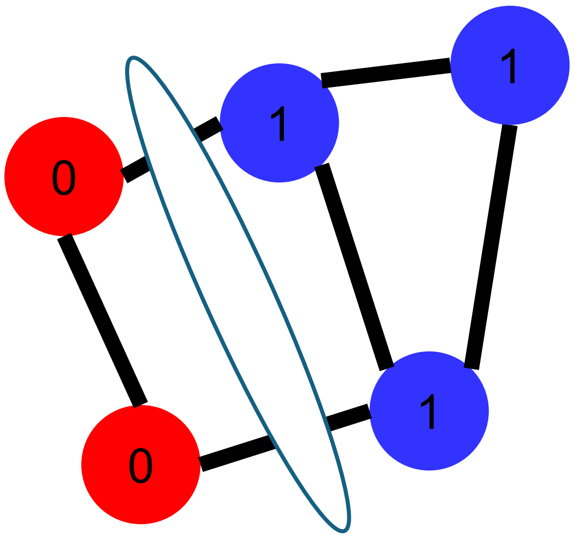
- Is symmetric, eigenvectors are orthogonal
- Has *N* eigenvalues that are non-negative
- *0* is always one eigenvalue $0 = \lambda_1 \leq \lambda_2 \leq ... \leq \lambda_N$
- The multiplicity of eigenvalue *0* equals the number of connected components of the graph

# The second smallest eigenvalue of the Laplacian

# $x^T L x$ and graph cuts

- Suppose *c(S, S')* is a cut of graph G
- Set

$$x_i = \begin{cases} 1 & \text{if } i \in S \\ 0 & \text{if } i \in S' \end{cases}$$




$$|c(S, S')| = 2$$

$$x^T L x = \sum_{(i,j)\in E} (x_i - x_j)^2 = \sum_{(i,j)\in c(S,S')} 1^2 = |c(S, S')|$$

# Rayleigh quotient

- For symmetric matrices, the second smallest eigenvalue is

$$\lambda_2 = \min_x \frac{x^T M x}{x^T x}$$

- If *x* is an eigenvector, $\frac{x^T M x}{x^T x}$ is its eigenvalue

https://en.wikipedia.org/wiki/Rayleigh_quotient

# Second eigenvector

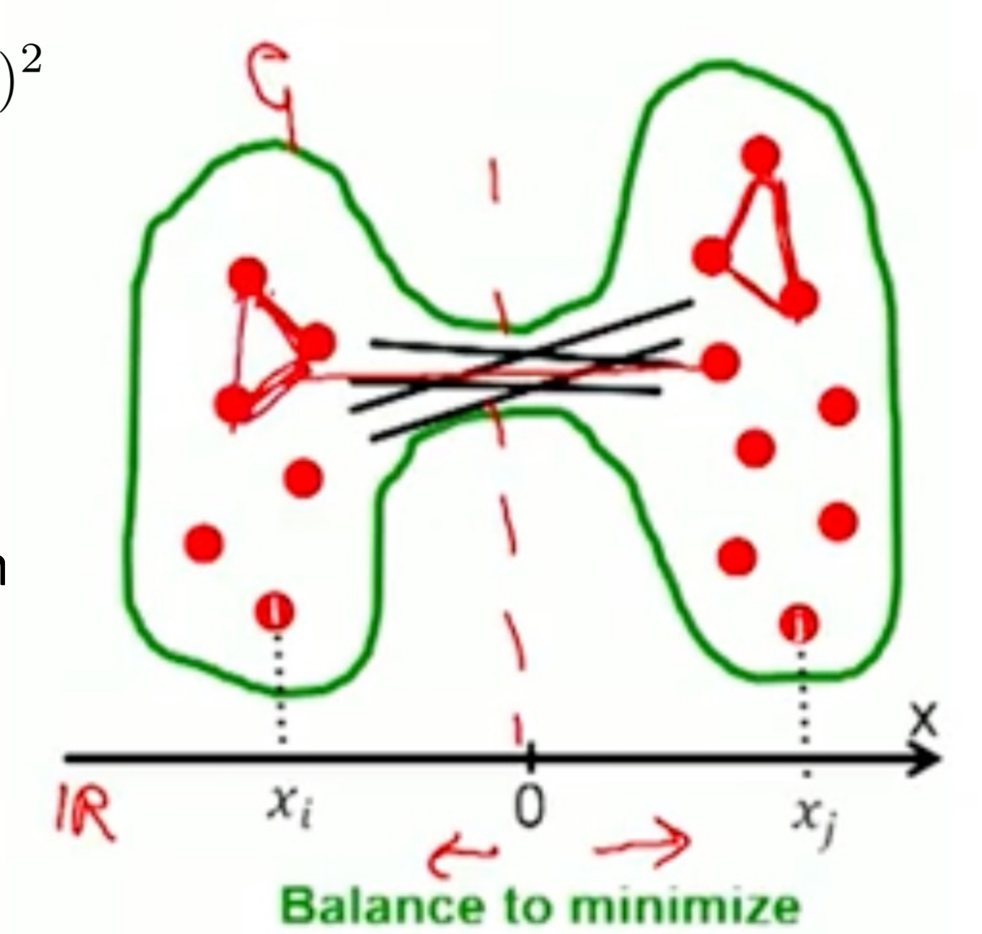
- Orthogonal to the first one: $x \cdot \vec{1} = 0 \Rightarrow \sum_i x_i = 0$
- Normal: $\sum_i x_i^2 = 1$

$$\lambda_2 = \min_x \frac{x^T L x}{x^T x} = \min_{x: \sum x_i = 0} \frac{x^T L x}{\sum x_i^2} = \min_{x: \sum x_i = 0 \wedge \sum x_i^2 = 1} \sum_{(i,j) \in E} (x_i - x_j)^2$$

# Second eigenvector

$$\lambda_2 = \min_{x:\sum x_i=0 \wedge \sum x_i^2=1} \sum_{(i,j)\in E} (x_i - x_j)^2$$

If the graph is connected
but almost partitioned
into two component,
the optimal *x* should have values
similar to each other in each partition



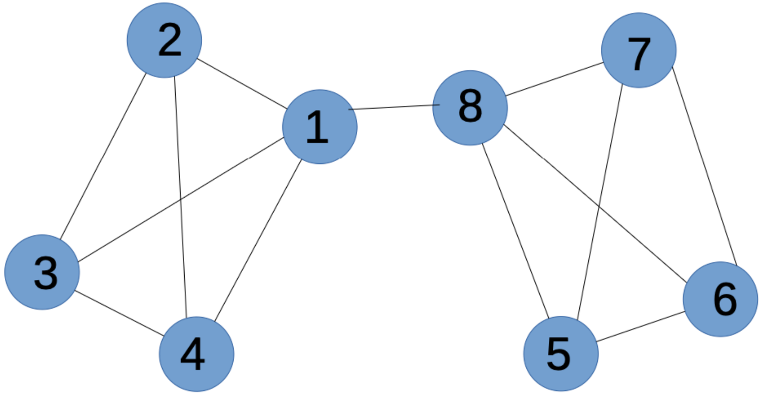

Nodes should be placed at both sides of 0 because $\sum x_i = 0$

# Second eigenvalue and eigenvector

$$\lambda_2 = \min_{x:\sum x_i=0 \wedge \sum x_i^2=1} \sum_{(i,j)\in E} (x_i - x_j)^2$$

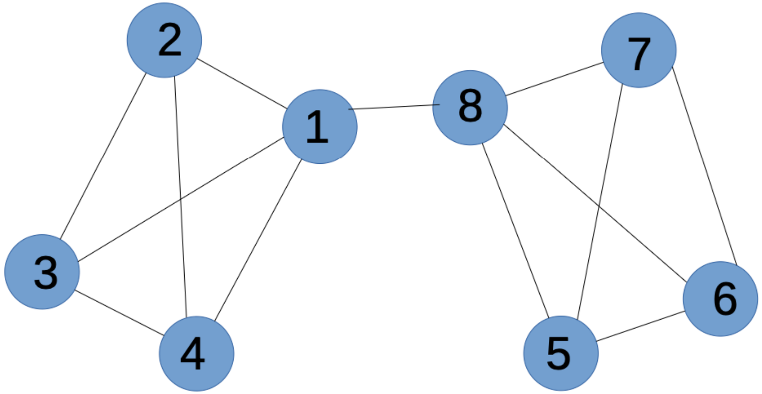
- The second eigenvalue tells us how well the graph can be partitioned into two:
- The smaller, the more disconnected the components
- Its eigenvector tells HOW to partition the graph into two:
- Eigenvector components assign each node to a community (positive/negative)

# Example Graph 1




$$L = \begin{bmatrix} 4 & -1 & -1 & -1 & 0 & 0 & 0 & -1 \\ -1 & 3 & -1 & -1 & 0 & 0 & 0 & 0 \\ -1 & -1 & 3 & -1 & 0 & 0 & 0 & 0 \\ -1 & -1 & -1 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & -1 & -1 & -1 \\ 0 & 0 & 0 & 0 & -1 & 3 & -1 & -1 \\ 0 & 0 & 0 & 0 & -1 & -1 & 3 & -1 \\ -1 & 0 & 0 & 0 & -1 & -1 & -1 & 4 \end{bmatrix}$$

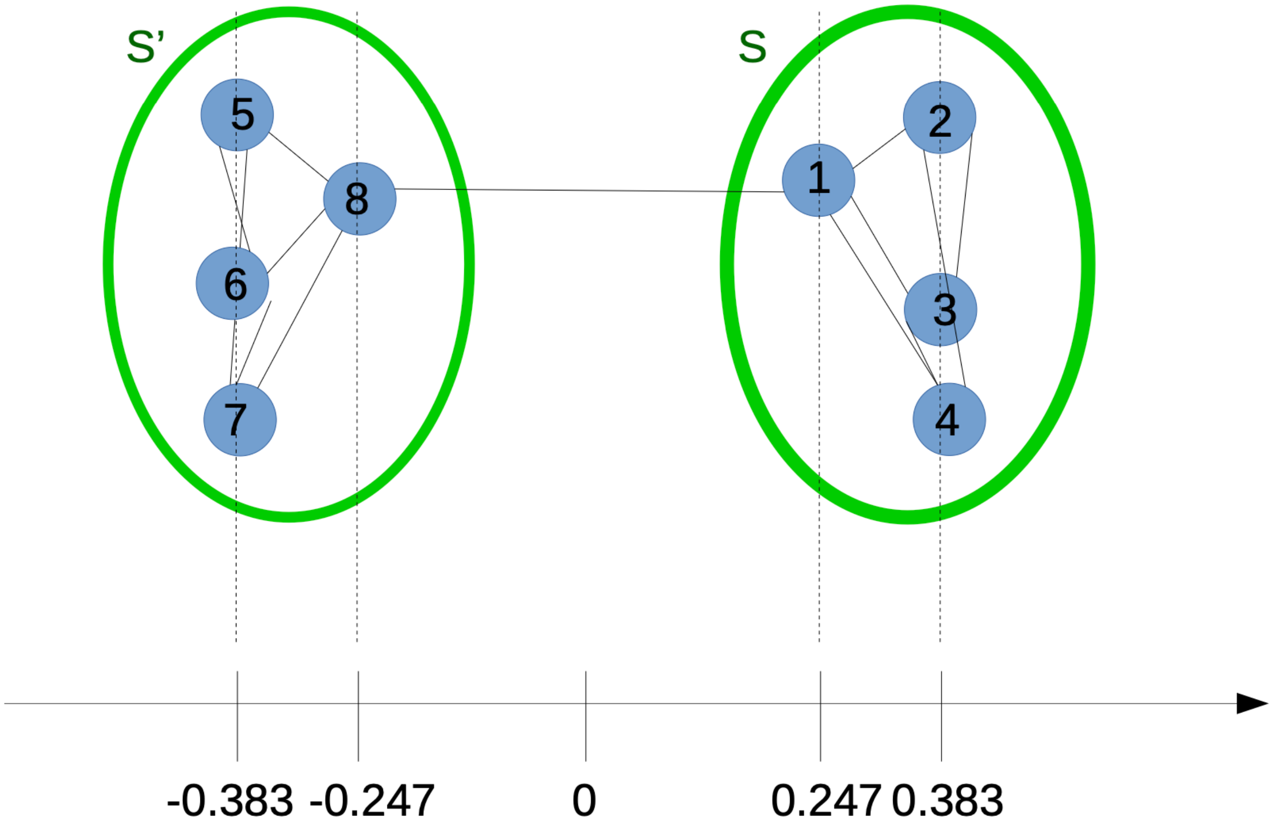
# Example Graph 1 (second eigenvalue of L)




$$\lambda_1 = 0$$
$$\lambda_2 = 0.354$$

$$L = \begin{bmatrix} 4 & -1 & -1 & -1 & 0 & 0 & 0 & -1 \\ -1 & 3 & -1 & -1 & 0 & 0 & 0 & 0 \\ -1 & -1 & 3 & -1 & 0 & 0 & 0 & 0 \\ -1 & -1 & -1 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & -1 & -1 & -1 \\ 0 & 0 & 0 & 0 & -1 & 3 & -1 & -1 \\ 0 & 0 & 0 & 0 & -1 & -1 & 3 & -1 \\ -1 & 0 & 0 & 0 & -1 & -1 & -1 & 4 \end{bmatrix} \qquad v_2 = \begin{bmatrix} 0.247 \\ 0.383 \\ 0.383 \\ 0.383 \\ -0.383 \\ -0.383 \\ -0.383 \\ -0.247 \end{bmatrix}$$

# Example Graph 1, communities

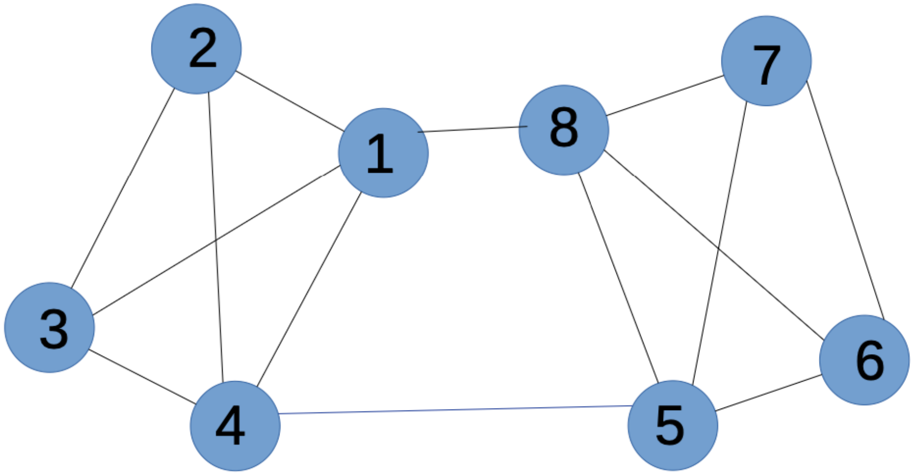



$$\lambda_1 = 0$$

$$\lambda_2 = 0.354$$

$$v_2 = \begin{bmatrix} 0.247 \\ 0.383 \\ 0.383 \\ 0.383 \\ -0.383 \\ -0.383 \\ -0.383 \\ -0.247 \end{bmatrix}$$

# Example Graph 2



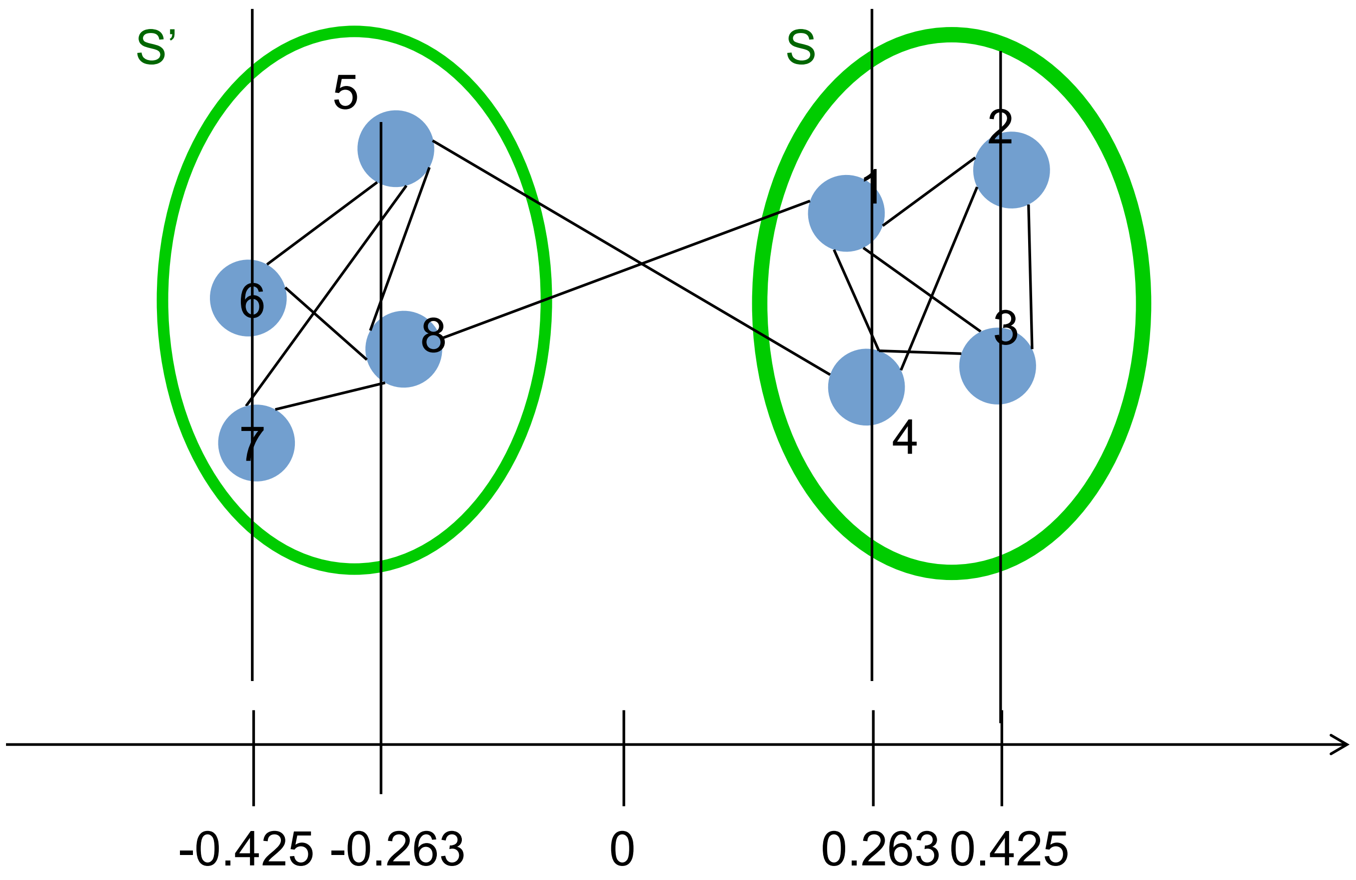

$$\lambda_1 = 0$$

$$\lambda_2 = 0.764$$

$$L = \begin{bmatrix} -1 & 3 & -1 & -1 & 0 & 0 & 0 & 0 \\ -1 & -1 & 3 & -1 & 0 & 0 & 0 & 0 \\ -1 & -1 & -1 & 4 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 4 & -1 & -1 & -1 \\ 0 & 0 & 0 & 0 & -1 & 3 & -1 & -1 \\ 0 & 0 & 0 & 0 & -1 & -1 & 3 & -1 \\ -1 & 0 & 0 & 0 & -1 & -1 & -1 & 4 \end{bmatrix}$$

$$v_2 = \begin{bmatrix} 0.263 \\ 0.425 \\ 0.425 \\ 0.263 \\ -0.263 \\ -0.425 \\ -0.425 \\ -0.263 \end{bmatrix}$$

# Example Graph 2, communities

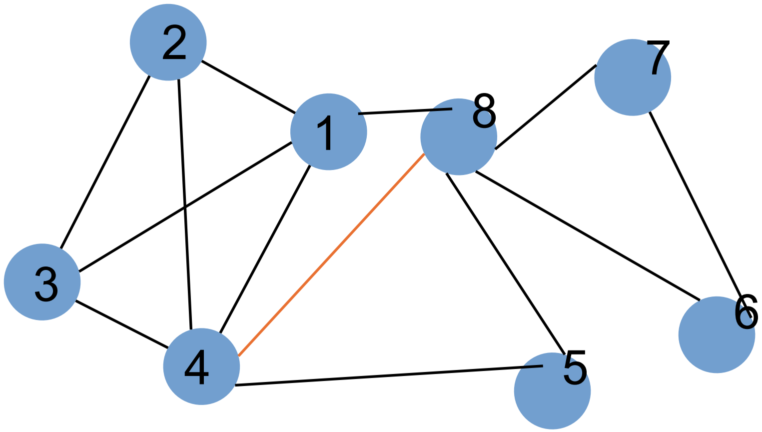



$$\lambda_1 = 0$$

$$\lambda_2 = 0.764$$

$$v_2 = \begin{bmatrix} 0.263 \\ 0.425 \\ 0.425 \\ 0.263 \\ -0.263 \\ -0.425 \\ -0.425 \\ -0.263 \end{bmatrix}$$

# Example Graph 3



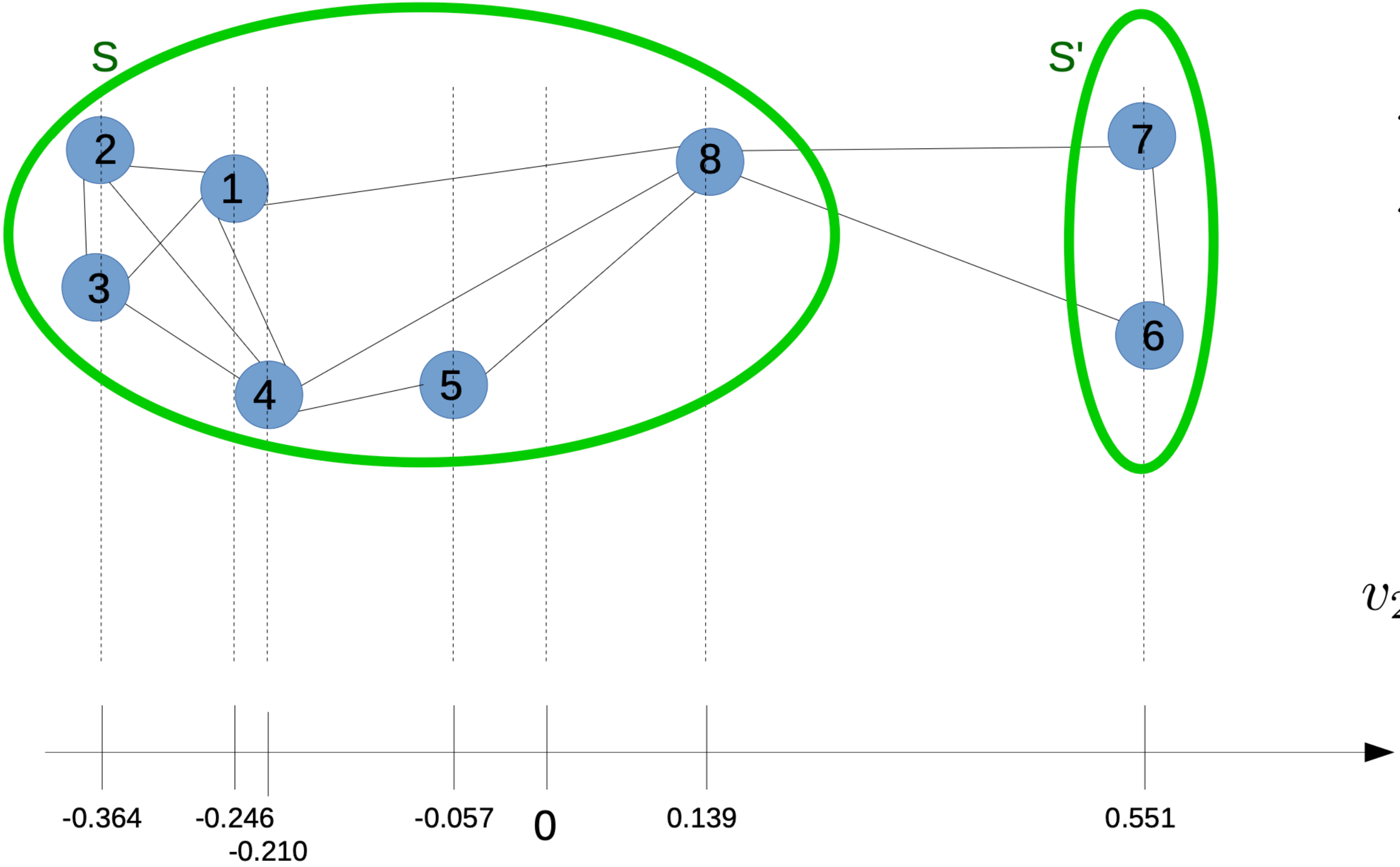

$$\lambda_1 = 0$$

$$\lambda_2 = 0.748$$

$$L = \begin{bmatrix} 4 & -1 & -1 & -1 & 0 & 0 & 0 & -1 \\ -1 & 3 & -1 & -1 & 0 & 0 & 0 & 0 \\ -1 & -1 & 3 & -1 & 0 & 0 & 0 & 0 \\ -1 & -1 & -1 & 5 & -1 & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 & 2 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 2 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 2 & -1 \\ -1 & 0 & 0 & -1 & -1 & -1 & -1 & 5 \end{bmatrix}$$

$$v_2 = \begin{bmatrix} -0.246 \\ -0.364 \\ -0.364 \\ -0.210 \\ -0.057 \\ 0.551 \\ 0.551 \\ 0.139 \end{bmatrix}$$

# Example Graph 3, projected (where to cut?)

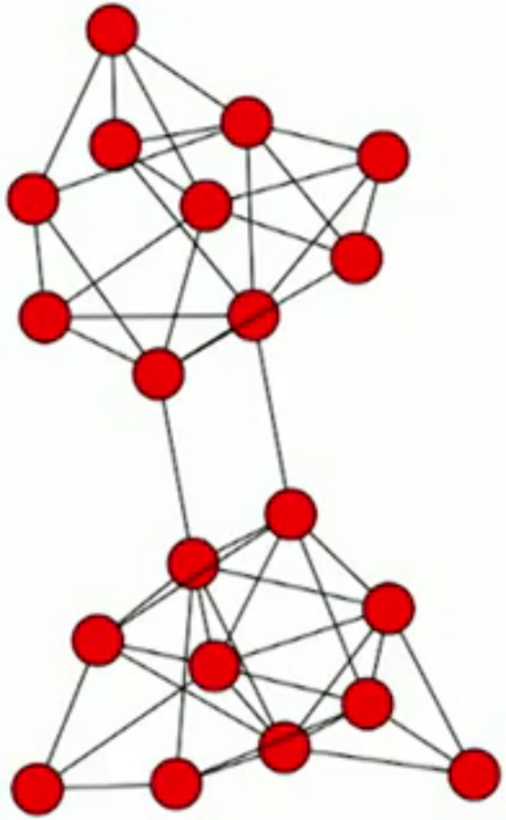


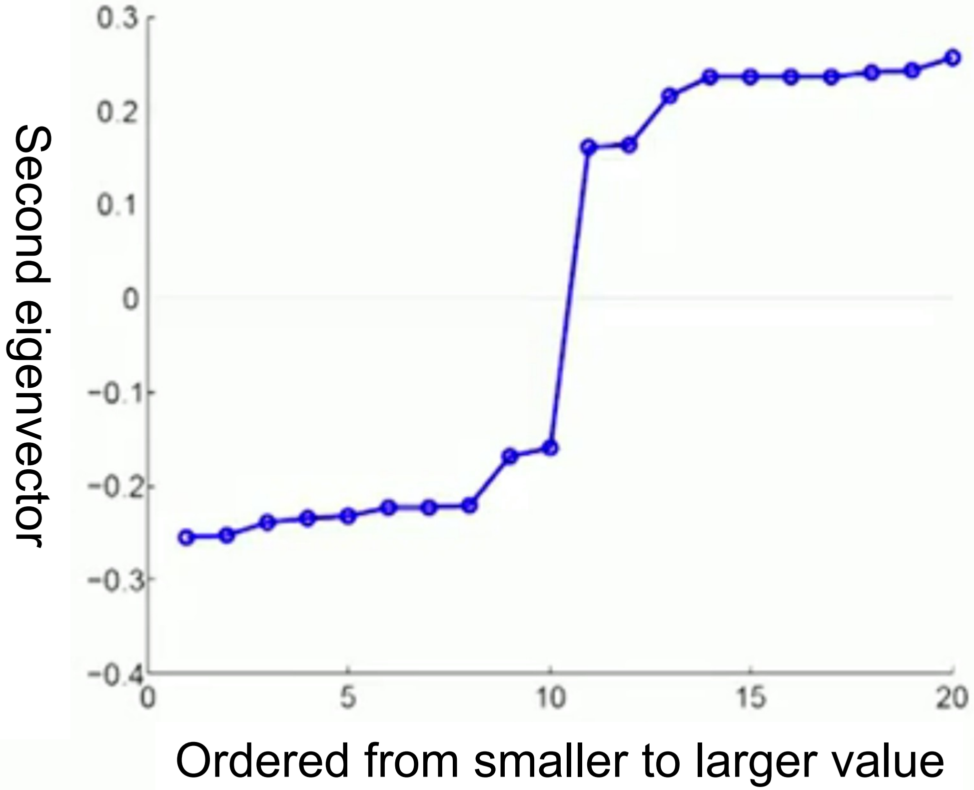

$$\lambda_1 = 0$$

$$\lambda_2 = 0.748$$

$$v_2 = \begin{bmatrix} -0.246 \\ -0.364 \\ -0.364 \\ -0.210 \\ -0.057 \\ 0.551 \\ 0.551 \\ 0.139 \end{bmatrix}$$

# A graph with two communities in $\mathbb{R}^1$

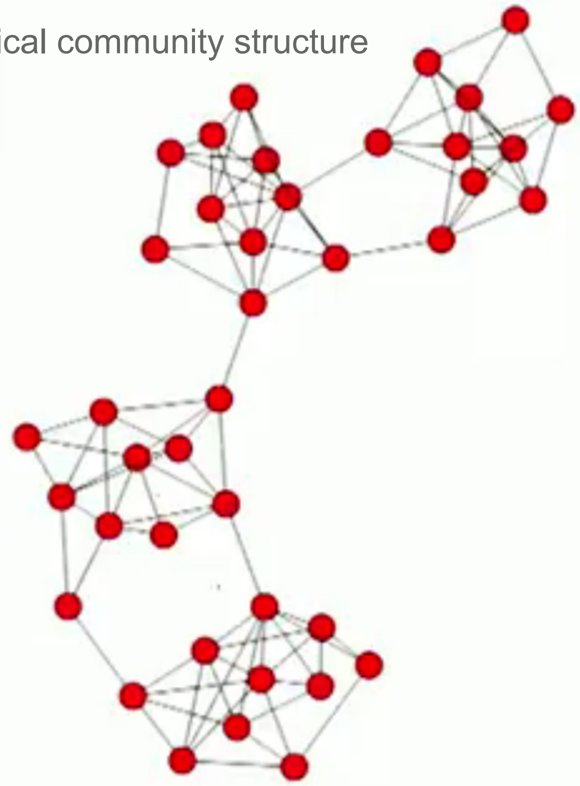


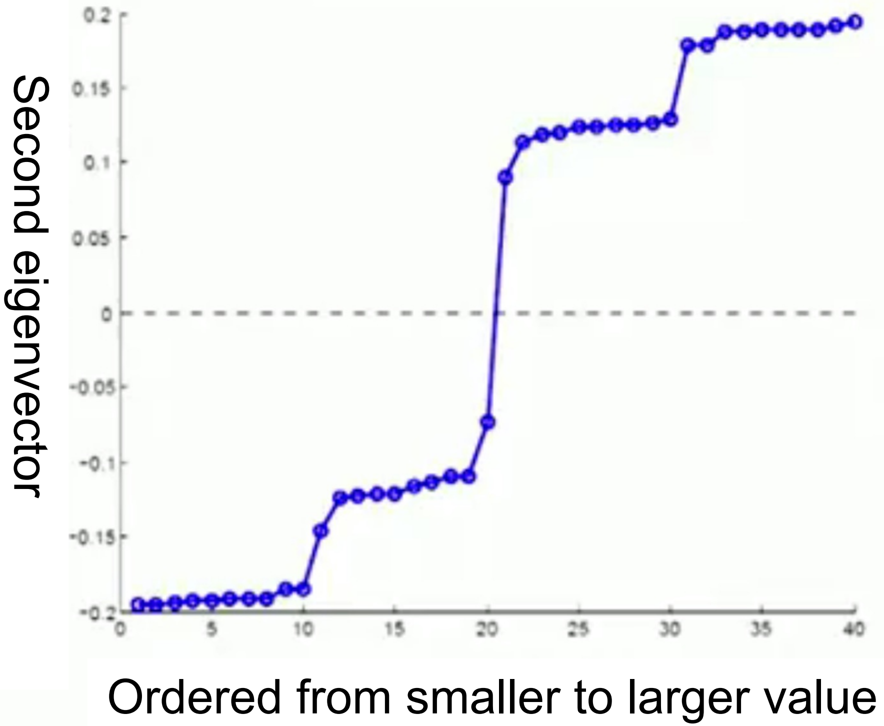



https://www.youtube.com/watch?v=jpTjj5PmcMM

# A graph with four communities i$\mathbb{R}^1$

Note the hierarchical community structure






https://www.youtube.com/watch?v=jpTjj5PmcMM

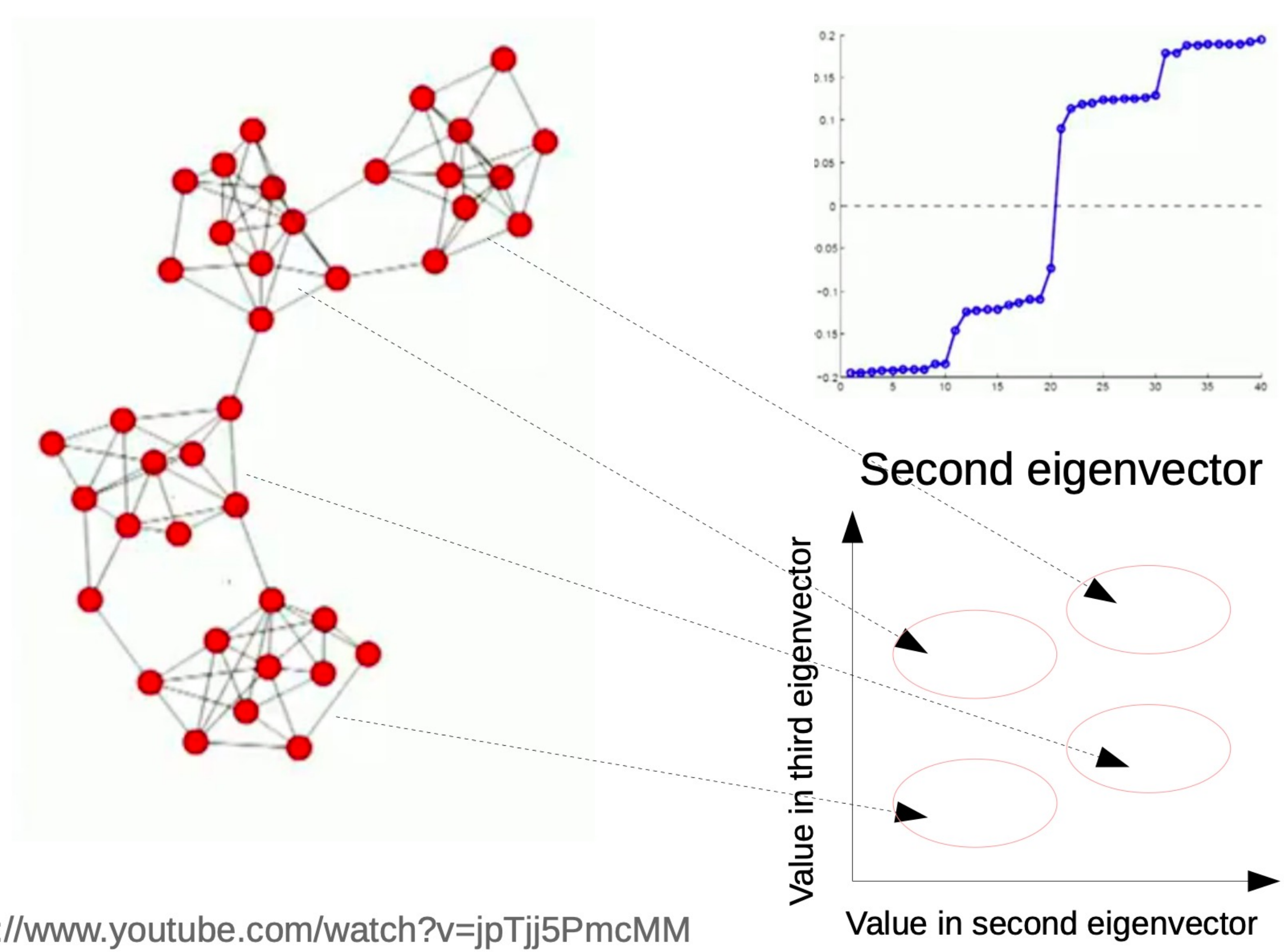
# Application: graph drawing

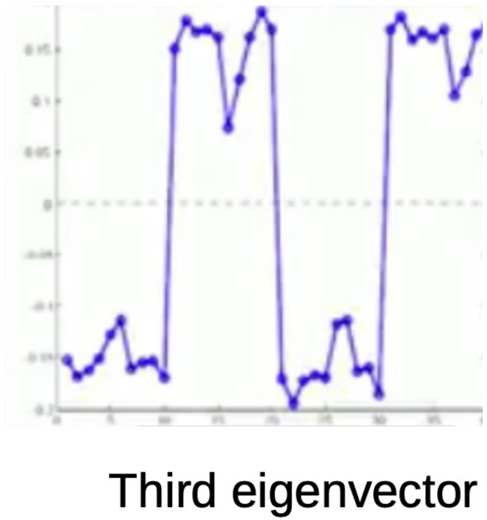
# Smallest eigenvalues and eigenvectors

$$\lambda_2 = \min_{x:\sum x_i = 0 \wedge \sum x_i^2 = 1} \sum_{(i,j)\in E} (x_i - x_j)^2$$

- Eigenvectors corresponding to the smallest eigenvalues minimize distances among neighbors!
- You can use these eigenvectors as the nodes coordinates
- The eigenvector of the first eigenvalue, equal to zero, is the constant vector: not useful for embedding

# A graph with four communities $\mathbb{R}^2$

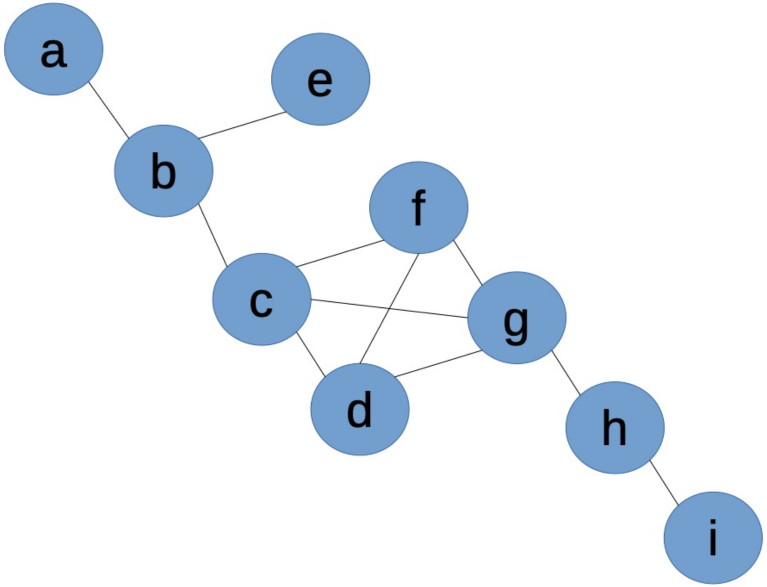


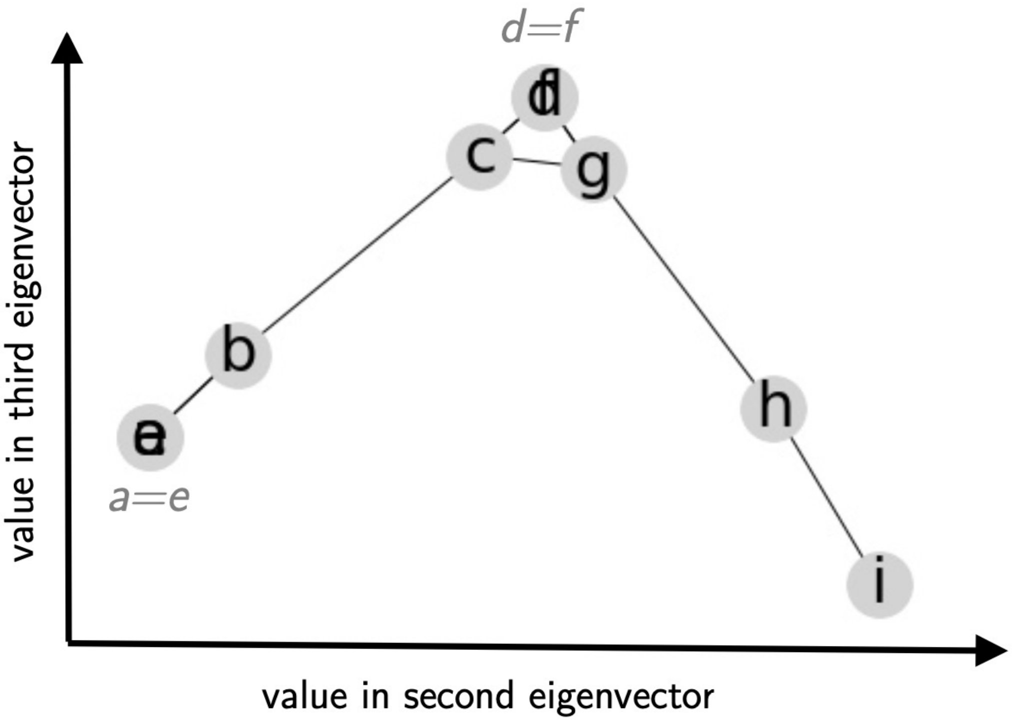

This can be used to draw the graph in $R^2$

https://www.youtube.com/watch?v=jpTjj5PmcMM

# The graph from the initial exercise



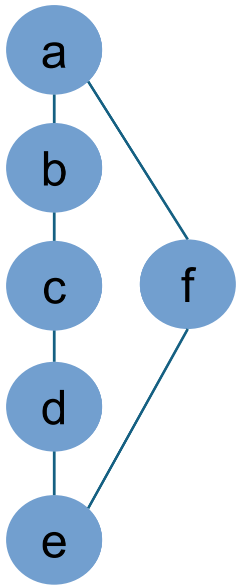

Input nodes and edges




Spectral embedding

# Exercise: spectral projection

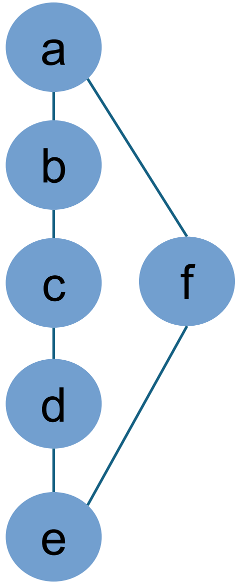
- Write the Laplacian
- Get the second and third eigenvector (e.g., “online eigenvector calculator”)
- Obtain projection



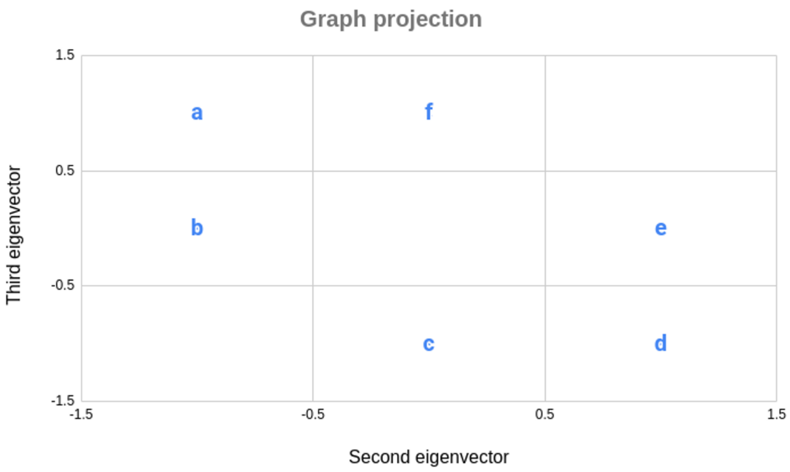

Link to spreadsheet: https://upfbarcelona.padlet.org/chato/shyq9m6f2g2dh1bw

# Answer: spectral projection

$$L = \begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 \end{pmatrix} - \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 2 & -1 & 0 & 0 & 0 & -1 \\ -1 & 2 & -1 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 \\ -1 & 0 & 0 & 0 & -1 & 2 \end{pmatrix}$$

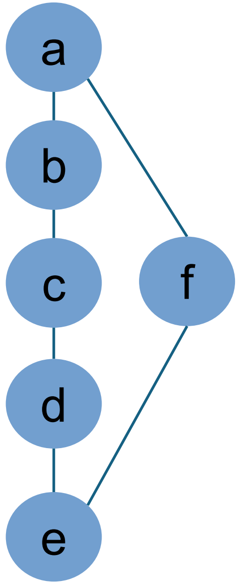


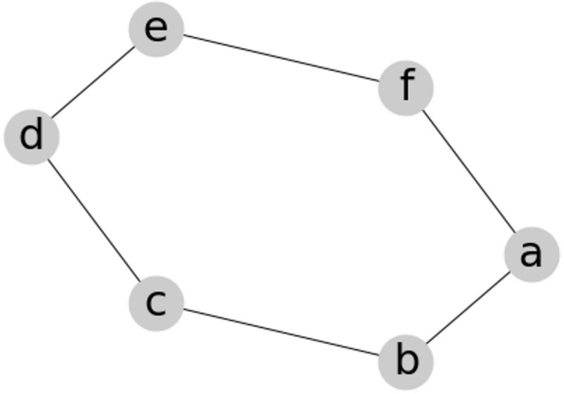

$$v_2 = \begin{pmatrix} -1 \\ -1 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} \qquad v_3 = \begin{pmatrix} 1 \\ 0 \\ -1 \\ -1 \\ 0 \\ 1 \end{pmatrix}$$



# Answer: spectral projection (Python)

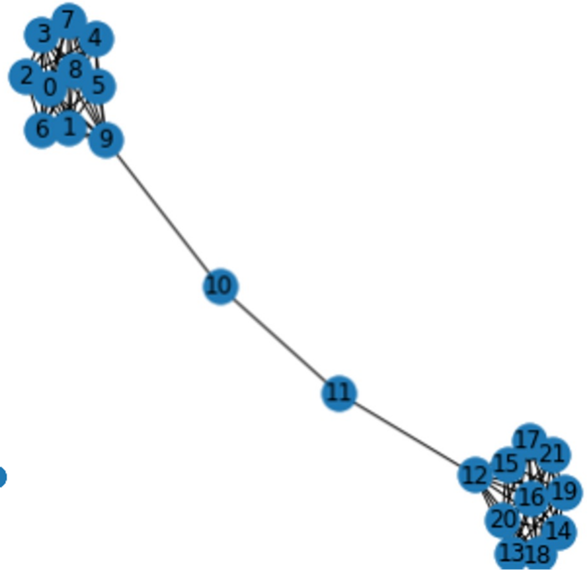
```
import networkx as nx

G  = nx.from_edgelist([('a', 'b'), ('b', 'c'),
('c', 'd'), ('d', 'e'), ('e', 'f'), ('f', 'a')])

nx.draw_spectral(G, with_labels=True,
font_size=30, node_size=1500, node_color='#ccc')
```



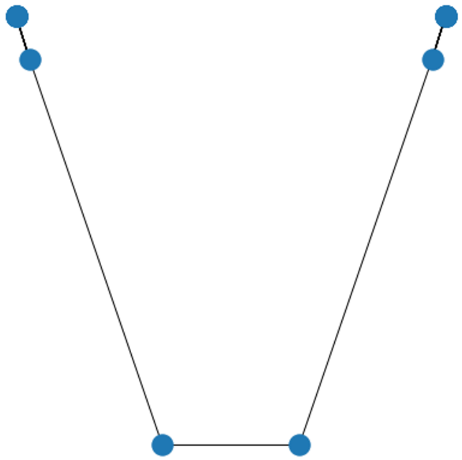



# A barbell graph in $R^2$ (code)

```
B = nx.barbell_graph(10,2)

plt.figure(figsize=(6,6))
nx.draw_networkx(B)
_ = plt.show()
```



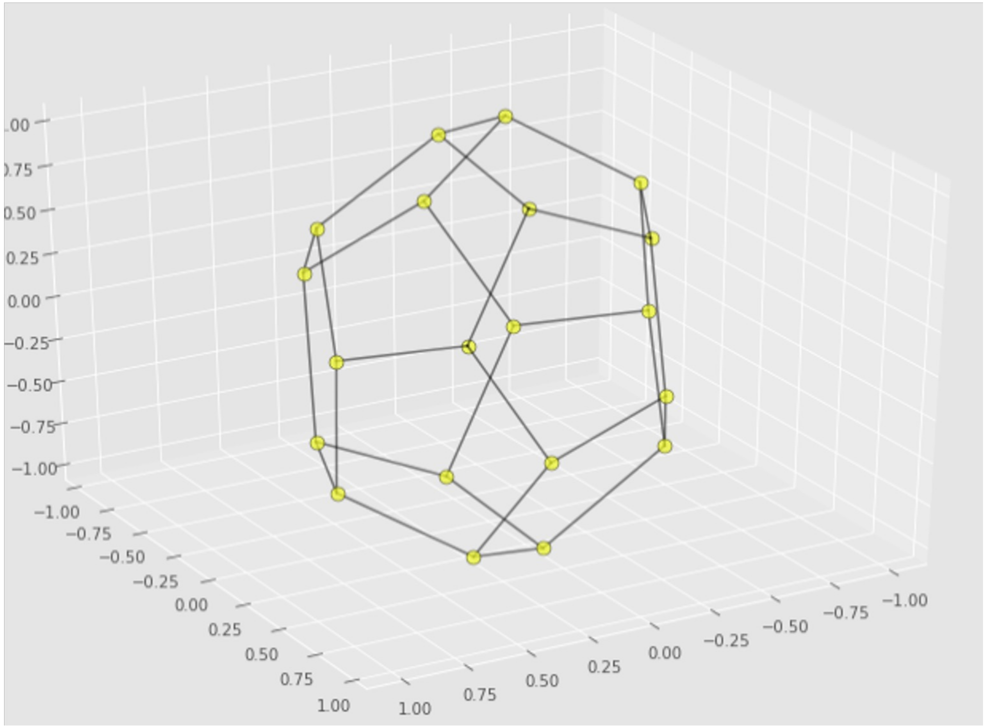

```
plt.figure(figsize=(6,6))
nx.draw_spectral(B)
_ = plt.show()
```



Graph Laplacian

# Dodecahedral graph in 3D

```
g = nx.dodecahedral_graph()
pos = nx.spectral_layout(g, dim=3)
network_plot_3D_alt(g, 60, pos)
```

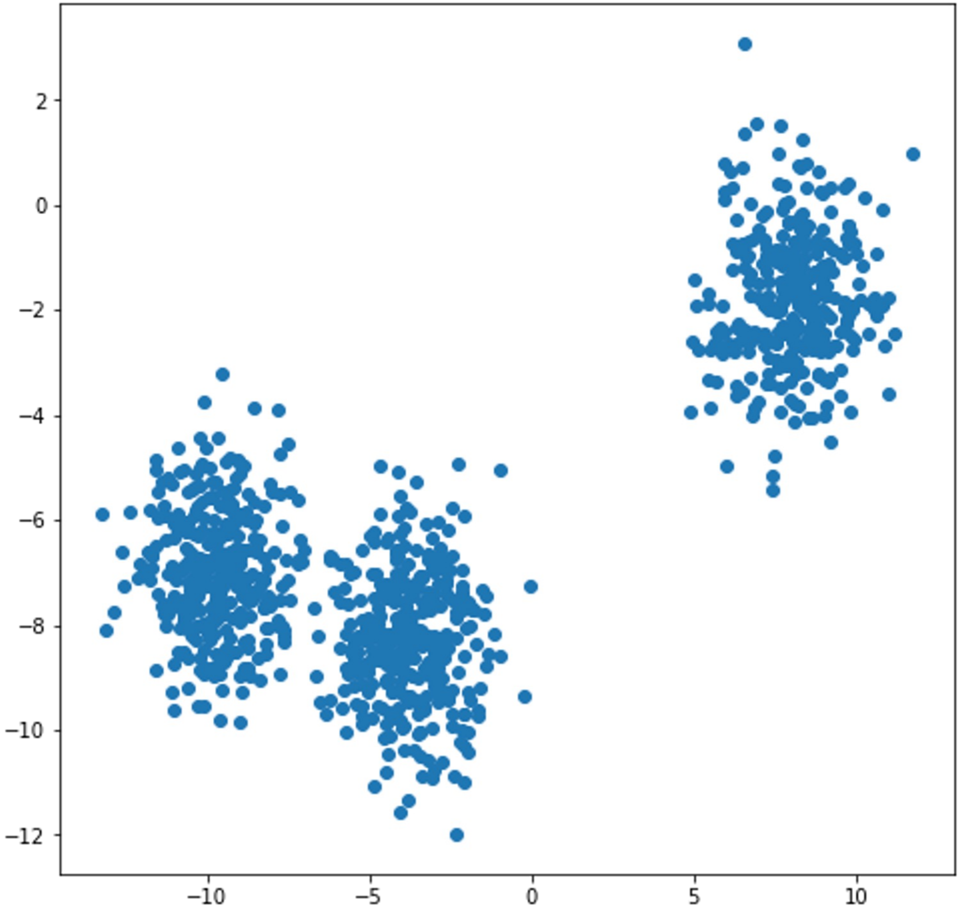



https://www.idtools.com.au/3d-network-graphs-python-mplot3d-toolkit/

# Application: spectral clustering

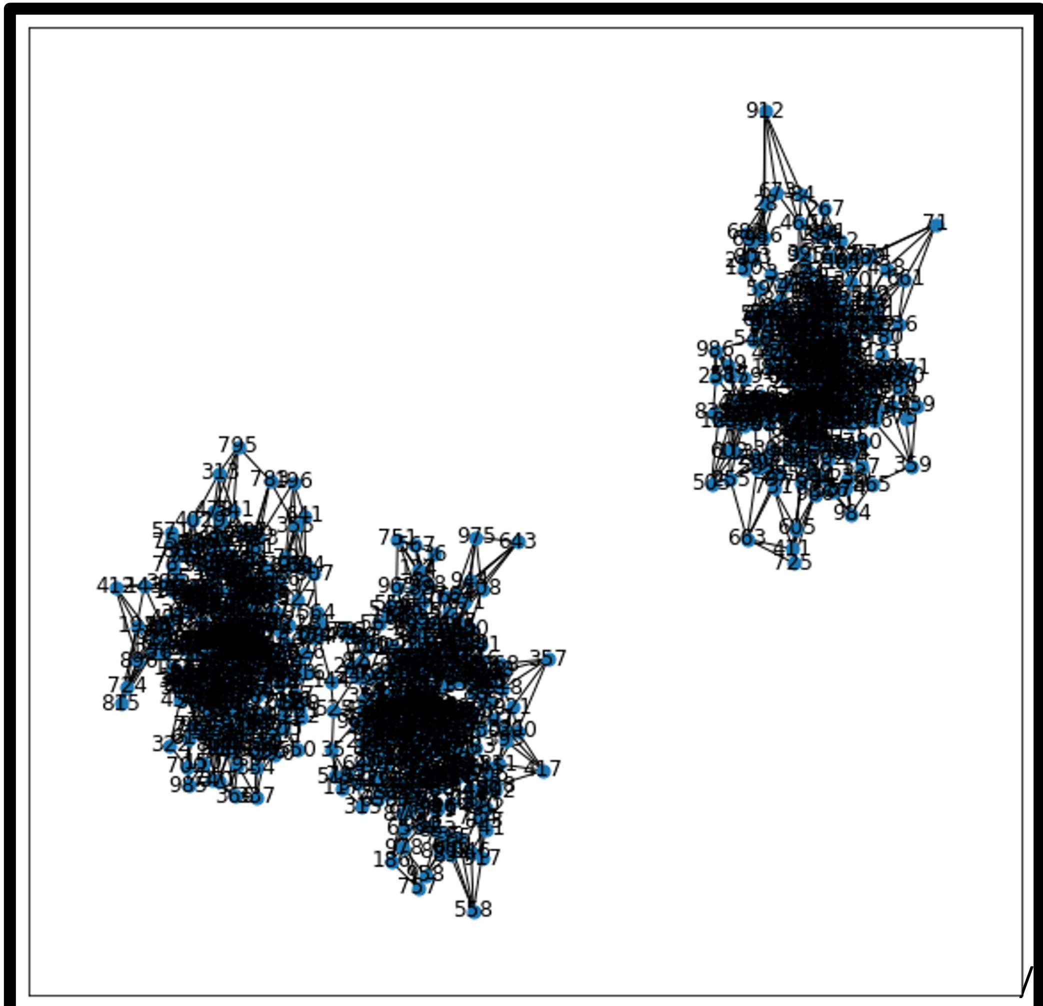
# Generating data

```
from sklearn.datasets import
   make_blobs

N = 1000

x, _ = make_blobs(
   n_samples=N,
   centers=3,
   cluster_std=1.2)


plt.figure(figsize=(8,8))

plt.scatter(x[:,0], x[:,1])

plt.show()
```



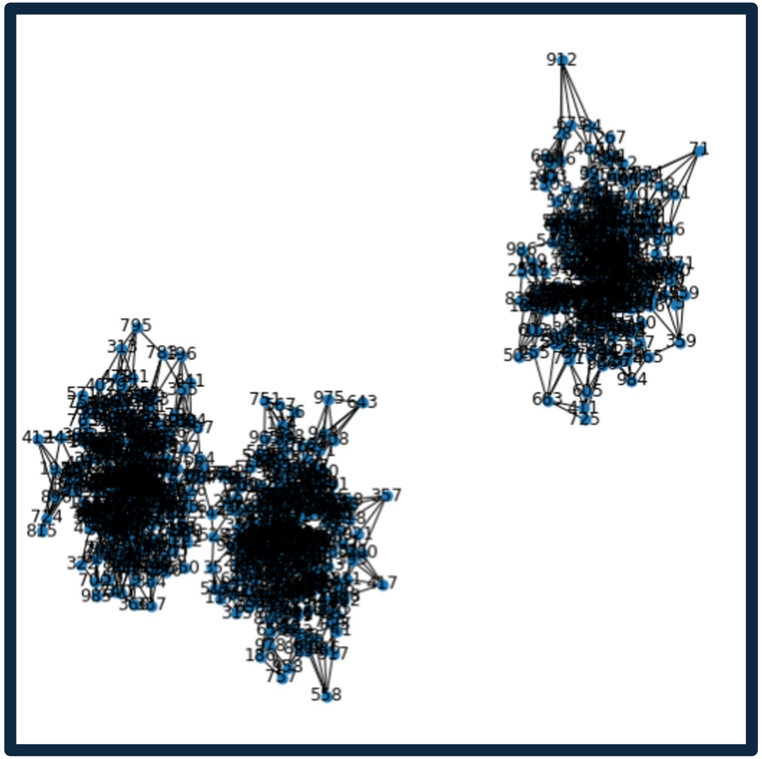
# Connect nodes to k=5 nearest neighbors

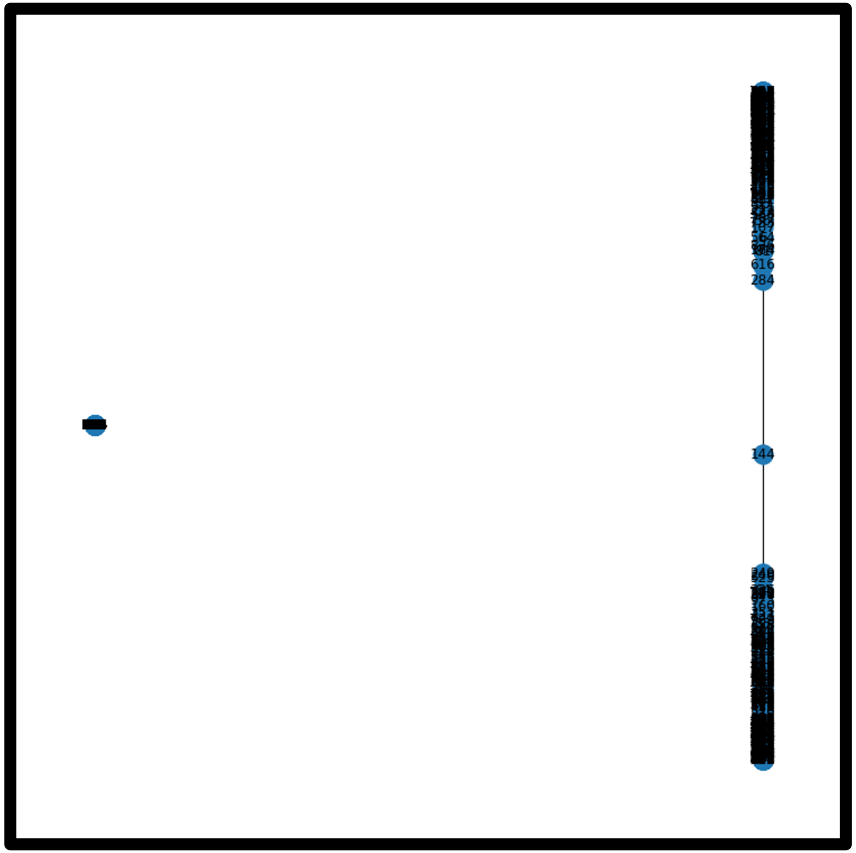
```
from sklearn.neighbors
   import NearestNeighbors

nbrs = NearestNeighbors(
    n_neighbors=6,         # includes self
    algorithm='ball_tree')
    .fit(x)

distances, neighbors =
    nbrs.kneighbors(x)

G = nx.Graph()

for neighbor_list in neighbors:

    source_node = neighbor_list[0]

    for target_index in range(1,
        len(neighbor_list)):

        target_node = neighbor_list[target_index]

        G.add_edge(source_node, target_node)
```



# Perform spectral embedding

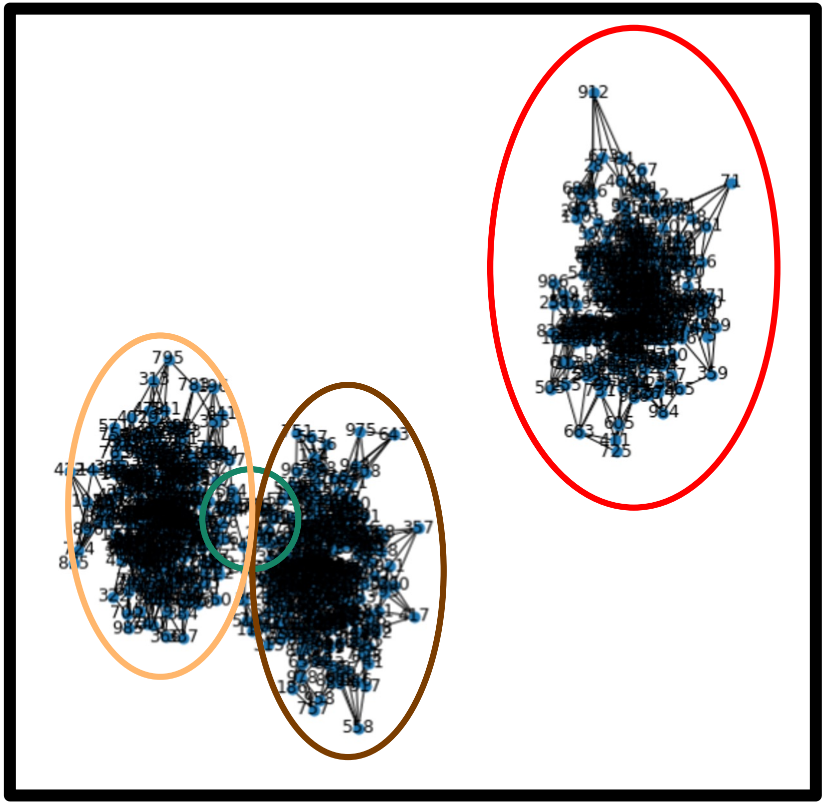
```
nx.draw_spectral(G, with_labels=True)
```

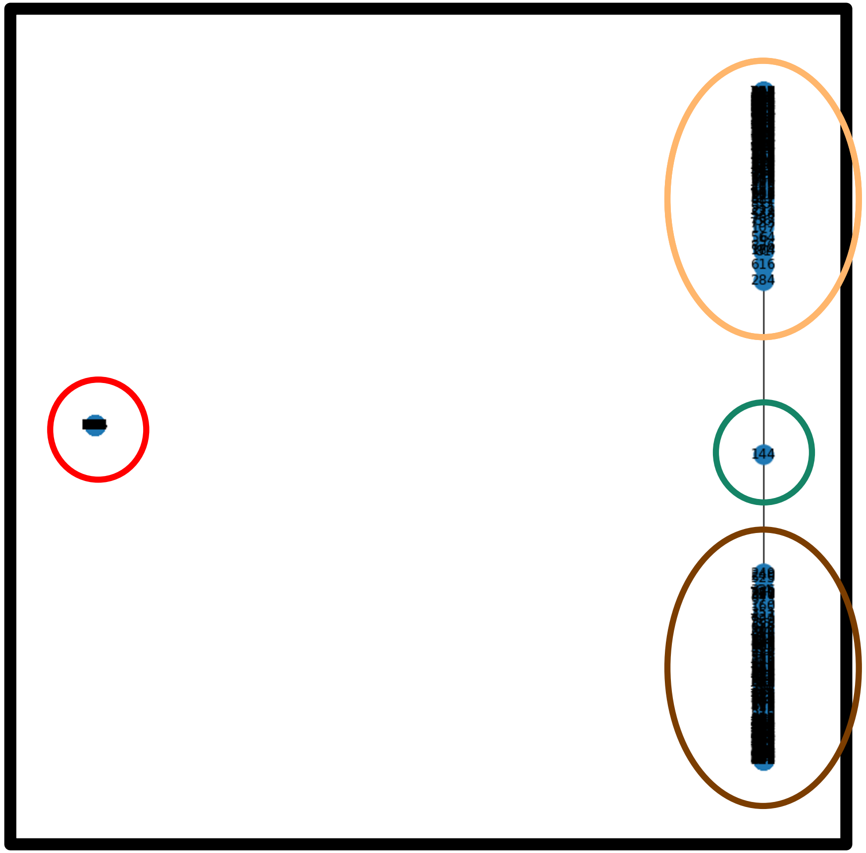




# Perform spectral embedding

```
nx.draw_spectral(G, with_labels=True)
```



# Summary

# Things to remember

- Graph Laplacian
- Laplacian and graph components
- Spectral graph embedding

# Sources


- **J. Leskovec (2016). Defining the graph laplacian** [video] https://www.youtube.com/watch?v=siCPjpUtE0A&t=2s
- E. Terzi (2013). Graph cuts — The part on spectral graph partitioning
- D. A. Spielman (2009): The Laplacian
- CS168: The Modern Algorithmic Toolbox
- Lectures #11: Spectral Graph Theory, I
- URLs cited in the footer of slides

# Exercises for this topic

- Mining of Massive Datasets (2014) by Leskovec et al.
  - Exercises 10.4.6