

Spectral Graph Embedding

Social Networks Analysis and Graph Algorithms

Prof. Carlos Castillo — <https://chato.cl/teach>



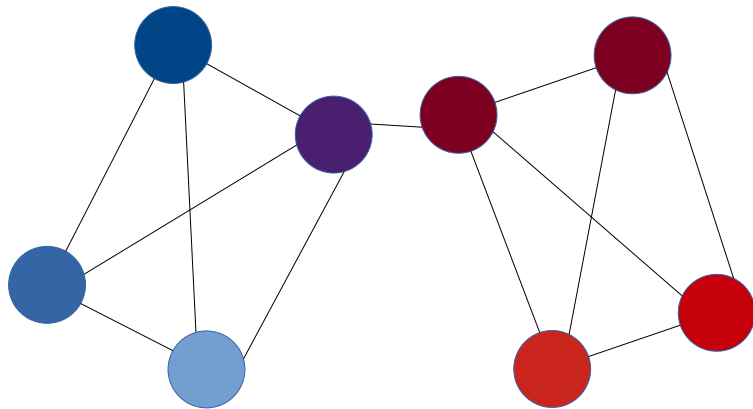
**Universitat
Pompeu Fabra**
Barcelona

Sources

- J. Leskovec (2016). **Defining the graph laplacian** [video]
- E. Terzi (2013). **Graph cuts** — The part on spectral graph partitioning
- D. A. Spielman (2009): **The Laplacian**
- URLs cited in the footer of slides

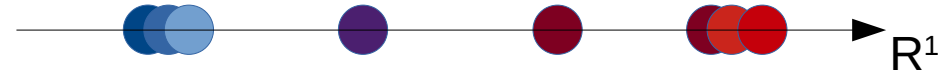
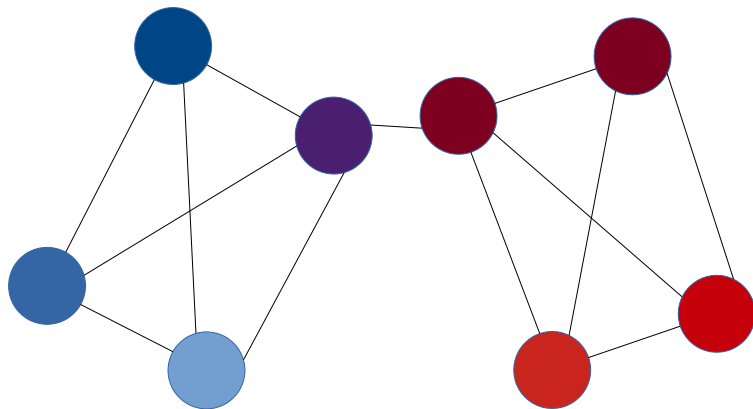
Graphs are nice, but ...

- They describe only local relationships
- We would like to understand a global structure
- We will try to transform a graph into a more familiar object: a cloud of points in \mathbb{R}^k



Graphs are nice, but ...

- They describe only local relationships
- We would like to understand a global structure
- We will try to transform a graph into a more familiar object: a cloud of points in \mathbb{R}^k



Distances should be somehow preserved

What is a graph embedding?

- A graph **embedding** (or graph **projection**) is a mapping from a graph to a vector space
- If the vector space is \mathbb{R}^2 you can think of an embedding as a way of **drawing** a graph on paper

Exercise: draw this graph

$$V = \{v1, v2, \dots, v8\}$$

$$E = \{ (v1, v2), (v2, v3), (v3, v4), (v4, v1), (v5, v6), (v6, v7), (v7, v8), \\ (v8, v5), (v1, v5), (v2, v6), (v3, v7), (v4, v8) \}$$

Draw this graph on paper, upload a photo



What constitutes a good drawing?



Padlet: <https://upfbarcelona.padlet.org/chato/3hwnctvqb7p1z6xx>

In a good graph embedding ...

- Pairs of nodes that are **connected** to each other should be **close**
- Pairs of nodes that are **not connected** should be **far**
- **Compromises will need to be made**

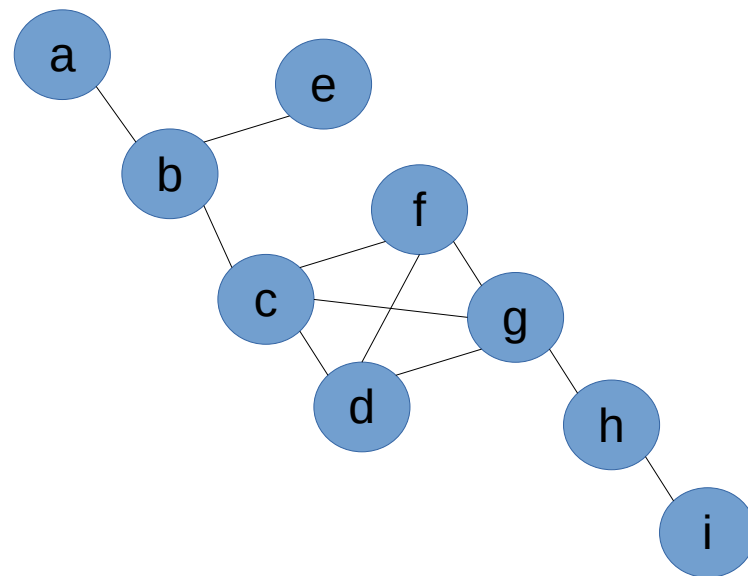
Random projections

Random graph projection (2D)

- Start a BFS from a random node, that has $x=1$, and nodes visited have ascending x
- Start a BFS from another random node, which has $y=1$, and nodes visited have ascending y
- Project node i to position (x_i, y_i)

Exercise: random projection

- Given this graph
- Pick a random node u
 - Distances from u are the x positions
- Pick a random node v
 - Distances from v are the y positions
- Draw the graph in an \mathbb{R}^2 plane



Padlet: <https://upfbarcelona.padlet.org/chato/9pd56scbpko5svdj>

Refresher about eigenvectors/eigenvalues

Eigenvectors and eigenvalues

- In general $Av = \lambda v$ means A has an eigenvector v of eigenvalue λ
- In **symmetric** matrices, **eigenvectors are orthogonal**

Suppose $\lambda_1 \neq \lambda_2$

$$\lambda_1 v_1^T v_2 = v_1^T A v_2 = v_1^T \lambda_2 v_2 = \lambda_2 v_1^T v_2$$

- This implies $v_1^T v_2 = 0$

In symmetric matrices

- The **multiplicity** of an eigenvalue λ is the dimension of the space of eigenvectors of eigenvalue λ
- Every $n \times n$ symmetric matrix has n eigenvalues counted with multiplicity
- Hence, it has an orthonormal basis of eigenvectors

Rayleigh quotient

In symmetric matrices M , the second eigenvalue is

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

Eigenvectors of the adjacency matrix

Properties of adjacency matrix

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

- How many **non-zeros** are in every **row** of A?

$$\begin{bmatrix} a_{11} & \dots & a_{1n} \\ a_{21} & \dots & a_{2n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \dots & a_{nn} \end{bmatrix}$$

Adjacency matrix of $G=(V,E)$

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

Can you write y_i using E ?

$$\begin{bmatrix} a_{11} & \dots & a_{1n} \\ a_{21} & \dots & a_{2n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \dots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

Adjacency matrix of $G=(V,E)$

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

- **What is Ax ?** Think of x as a set of labels/values:

$$\begin{bmatrix} a_{11} & \dots & a_{1n} \\ a_{21} & \dots & a_{2n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \dots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

$$y_i = \sum_{j:(j,i) \in E} x_j$$

Ax is a vector whose i^{th} coordinate contains the sum of the x_j who are in-neighbors of i


Spectral graph theory ...

- Studies the eigenvalues and eigenvectors of a graph matrix
 - Adjacency matrix $Ax = \lambda x$
 - Laplacian matrix (next)

- Suppose graph is d-regular: $k_i = d \ \forall i$

- What is the value of

- What does that imply?


$$\begin{bmatrix} a_{11} & \dots & a_{1n} \\ a_{21} & \dots & a_{2n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \dots & a_{nn} \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} = ?$$

An eigenvector of a d-regular graph

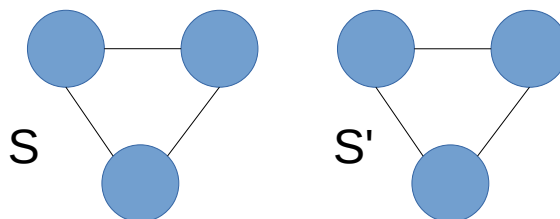
- Suppose graph is d-regular, i.e. all nodes have degree d:

$$\begin{bmatrix} a_{11} & \dots & a_{1n} \\ a_{21} & \dots & a_{2n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \dots & a_{nn} \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} = \begin{bmatrix} d \\ d \\ \vdots \\ d \end{bmatrix} = d \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$$

- Hence, $[1, 1, \dots, 1]^T$ is an eigenvector of eigenvalue d

Disconnected graphs

- Suppose the graph is regular **and disconnected**

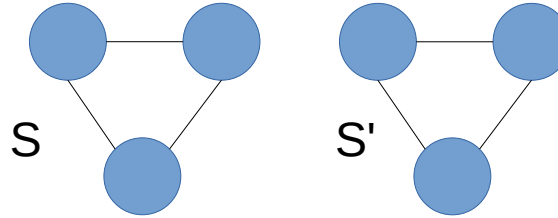


- Then its adjacency matrix has **block structure**:

$$A = \begin{bmatrix} S & 0 \\ 0 & S' \end{bmatrix}$$

Disconnected graphs

- Suppose the graph is regular **and disconnected**

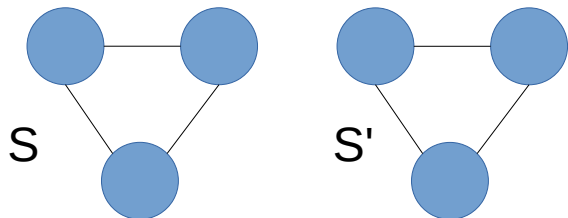


$$\text{Let } x_i^S = \begin{cases} 1 & \text{if } i \in S \\ 0 & \text{if } i \in S' \end{cases}$$

$$\begin{bmatrix} S & 0 \\ 0 & S' \end{bmatrix} \begin{bmatrix} 1 \\ \vdots \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = ?$$

Disconnected graphs

- Suppose the graph is regular **and disconnected**

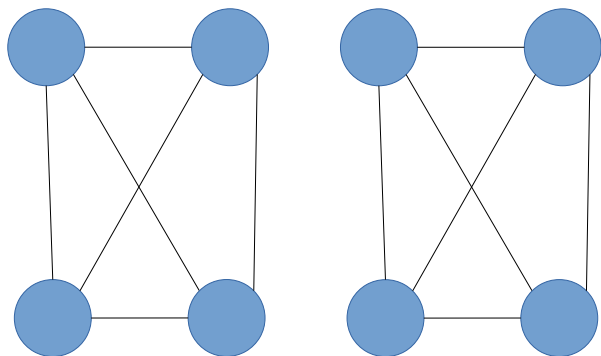


$$Ax^{S'} = dx^{S'}$$

- What is the multiplicity of eigenvalue d ?
- What happens if there are more than 2 connected components?

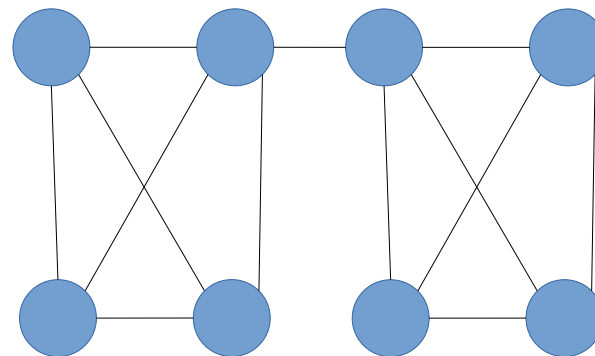
In general

Disconnected graph



$$\lambda_1 = \lambda_2$$

Almost disconnected graph



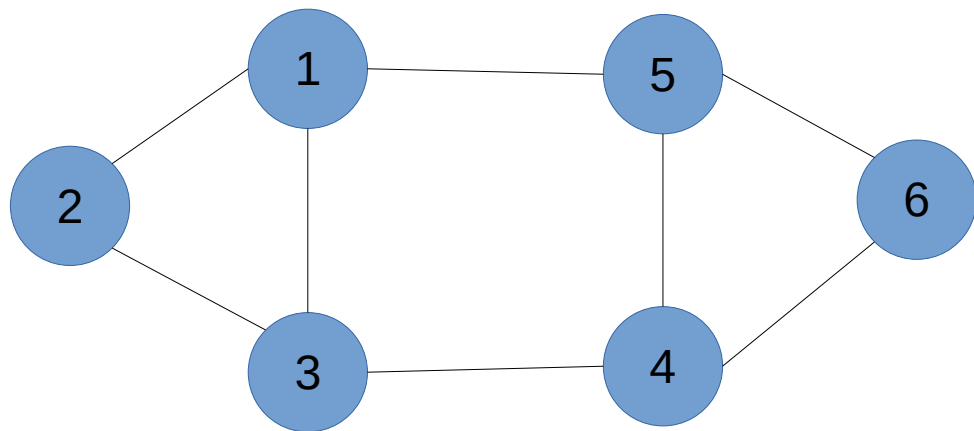
$$\lambda_1 \approx \lambda_2$$

Small “**eigengap**”

Graph Laplacian

Adjacency matrix

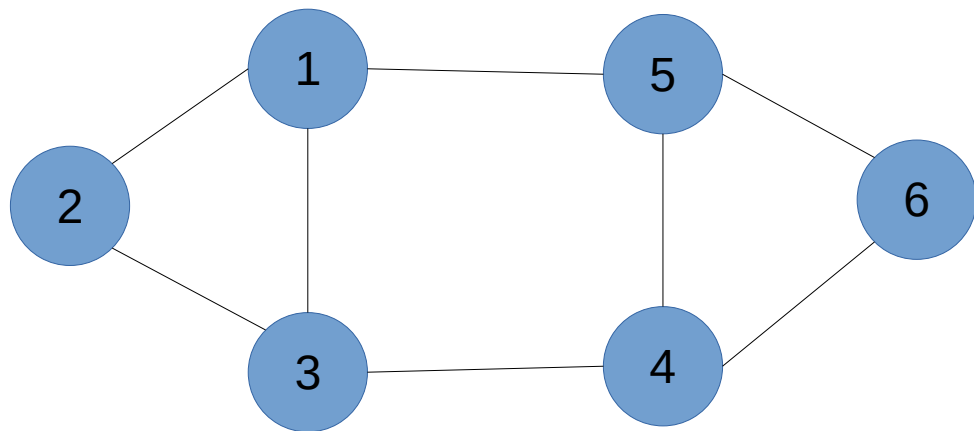
$$A_{ij} = \begin{cases} 1 & \text{if } (i, j) \in E \\ 0 & \text{otherwise} \end{cases}$$



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

Degree matrix

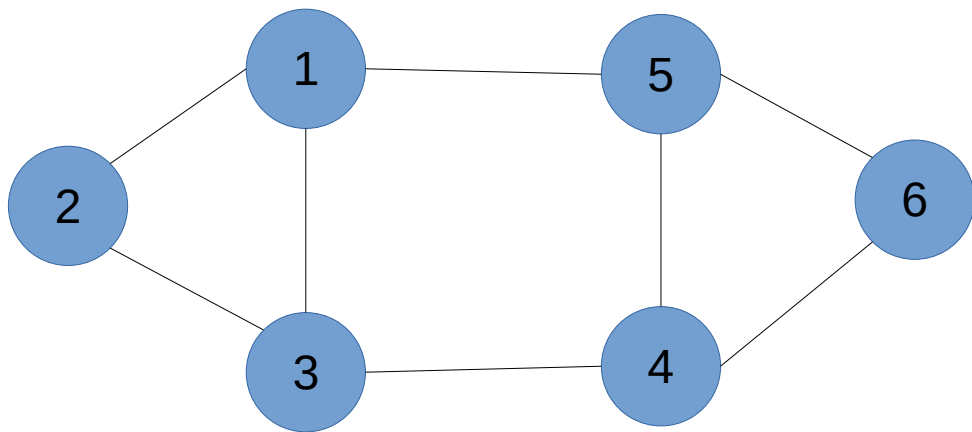
$$D_{ij} = \begin{cases} k_i & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$



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

Laplacian matrix

$$L = D - A$$



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

Given A is symmetric, L is symmetric.

They only differ in the diagonal.

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 $x=[1,1,\dots,1]^T$ 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 $[1, 1, \dots, 1]^T$? 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

$$x^T L x$$

Prove this!

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

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

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

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

One of the eigenvalues of L is 0

- If x is such that $x_i = x_j$ for all i, j :

$$x^T Lx = \sum_{(i,j) \in E} (x_i - x_j)^2 = 0 \Rightarrow Lx = 0$$

- This means 0 is an eigenvalue of L

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 $x^T Lx = \sum_{(i,j) \in E} (x_i - x_j)^2 = 0$

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 $x^T Lx = \sum_{(i,j) \in E} (x_i - x_j)^2 = 0$
- 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 \Rightarrow
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

All the eigenvalues of the Laplacian are non-negative

- If v is an eigenvector of L of eigenvalue λ :

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

- This means all eigenvalues λ are non-negative

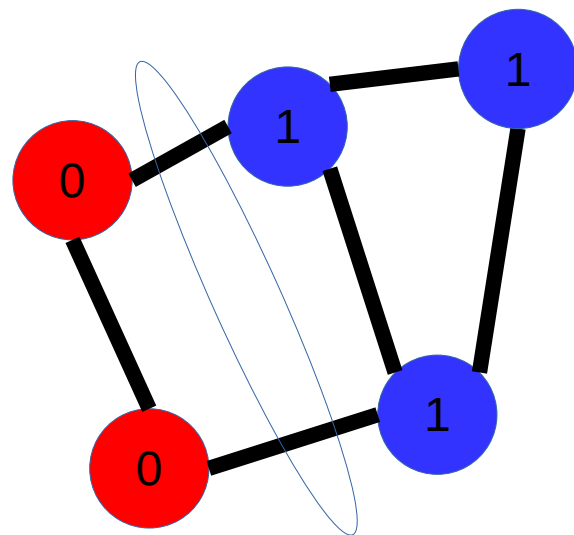
In summary, the Laplacian matrix $L = D - A$

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

The second eigenvector of the Laplacian in a connected graph

$x^T L x$ and graph cuts

- Suppose (S, S') is a cut of graph G
- Set $x_i = \begin{cases} 1 & \text{if } i \in S \\ 0 & \text{if } i \in S' \end{cases}$



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

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

Remember

- For symmetric matrices

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

Second eigenvector

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

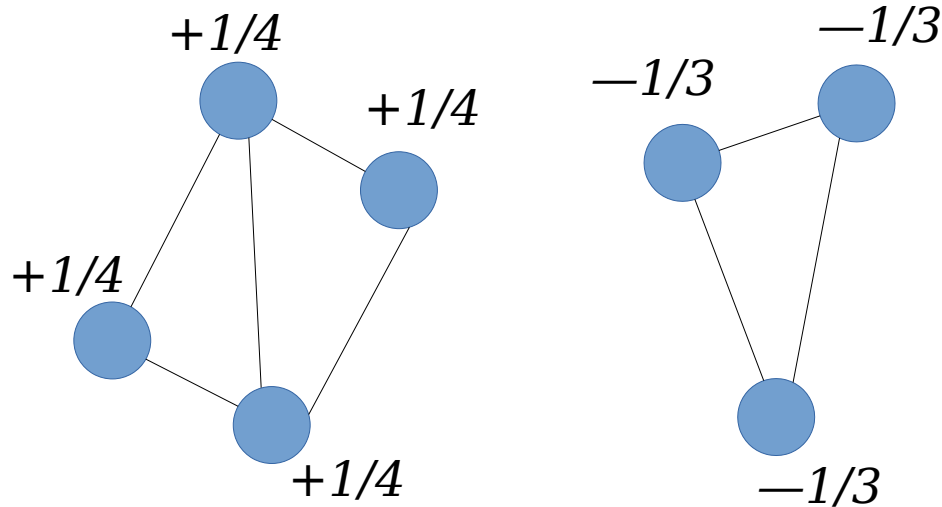
$$\lambda_2 = \min_x \frac{x^T L x}{x^T x} = \min_{x: \sum x_i = 0} \frac{x^T L x}{\sum x_i^2} = \min_{x: \sum x_i = 0} \sum_{(i,j) \in E} (x_i - x_j)^2$$

The second eigenvalue tells us how well the graph can be partitioned into two

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

If the graph is divided into two connected components of sizes N_1 and N_2 , the optimal x is:

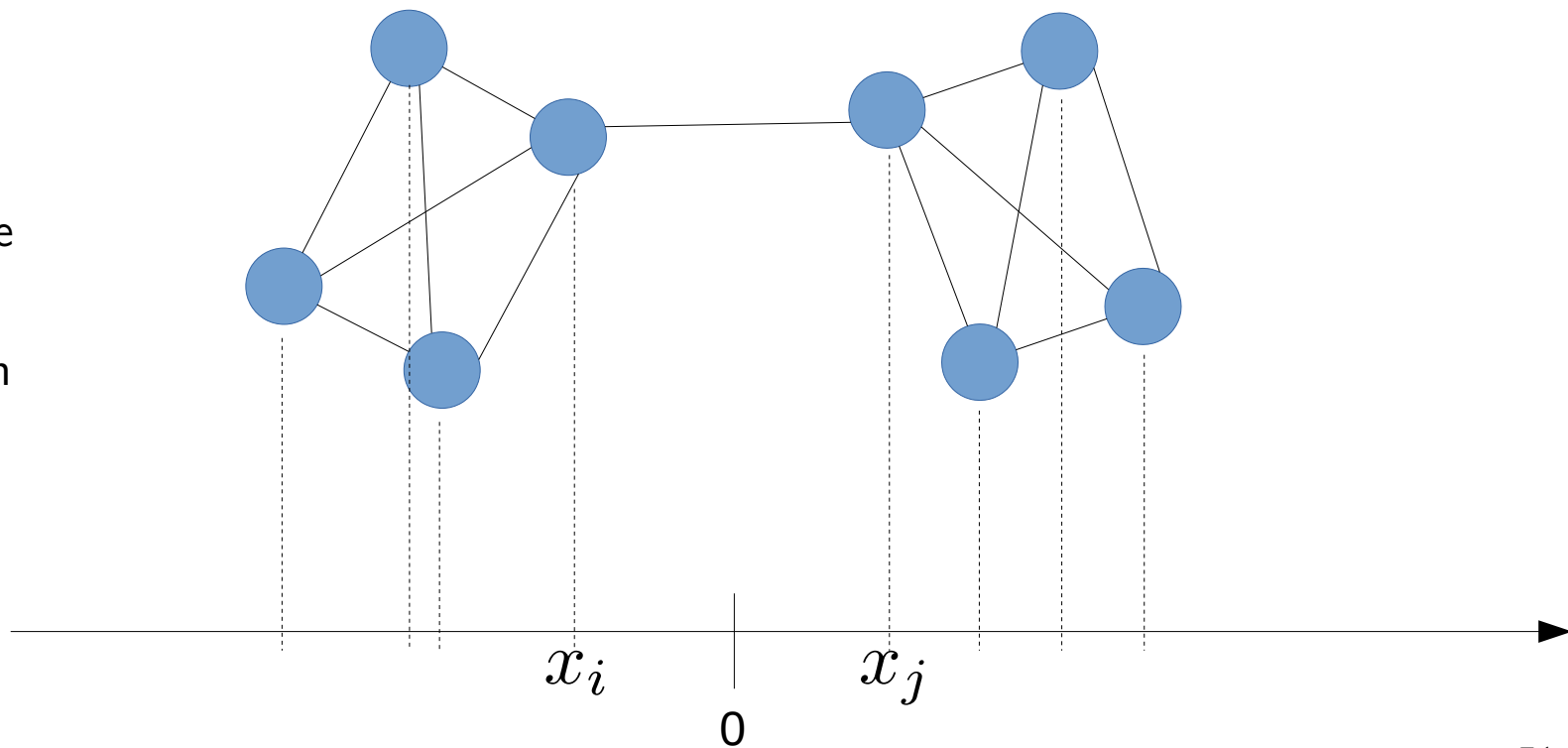
$x_i = +1/N_1$ for nodes in one component, and
 $x_i = -1/N_2$ for nodes in the second component



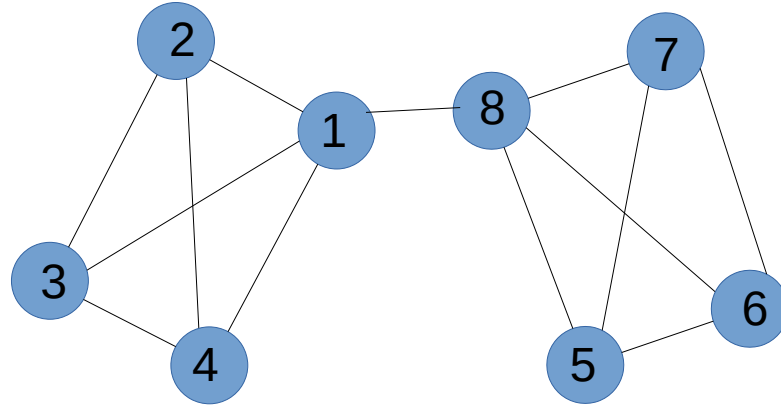
The second eigenvalue tells us how well the graph can be partitioned into two

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

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

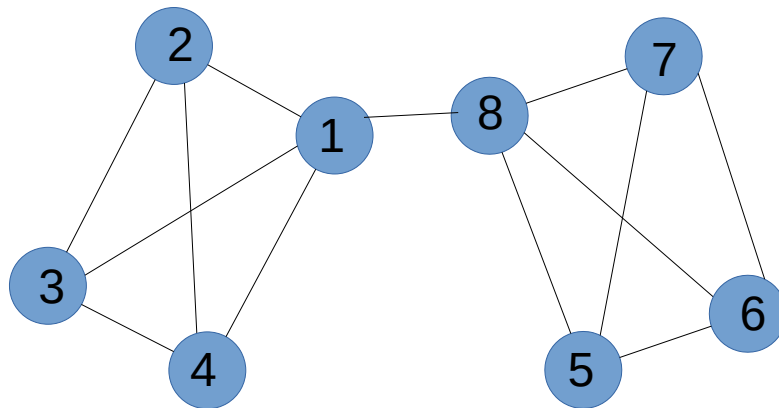


Example Graph 1



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

Example Graph 1 (second eigenvalue of L)



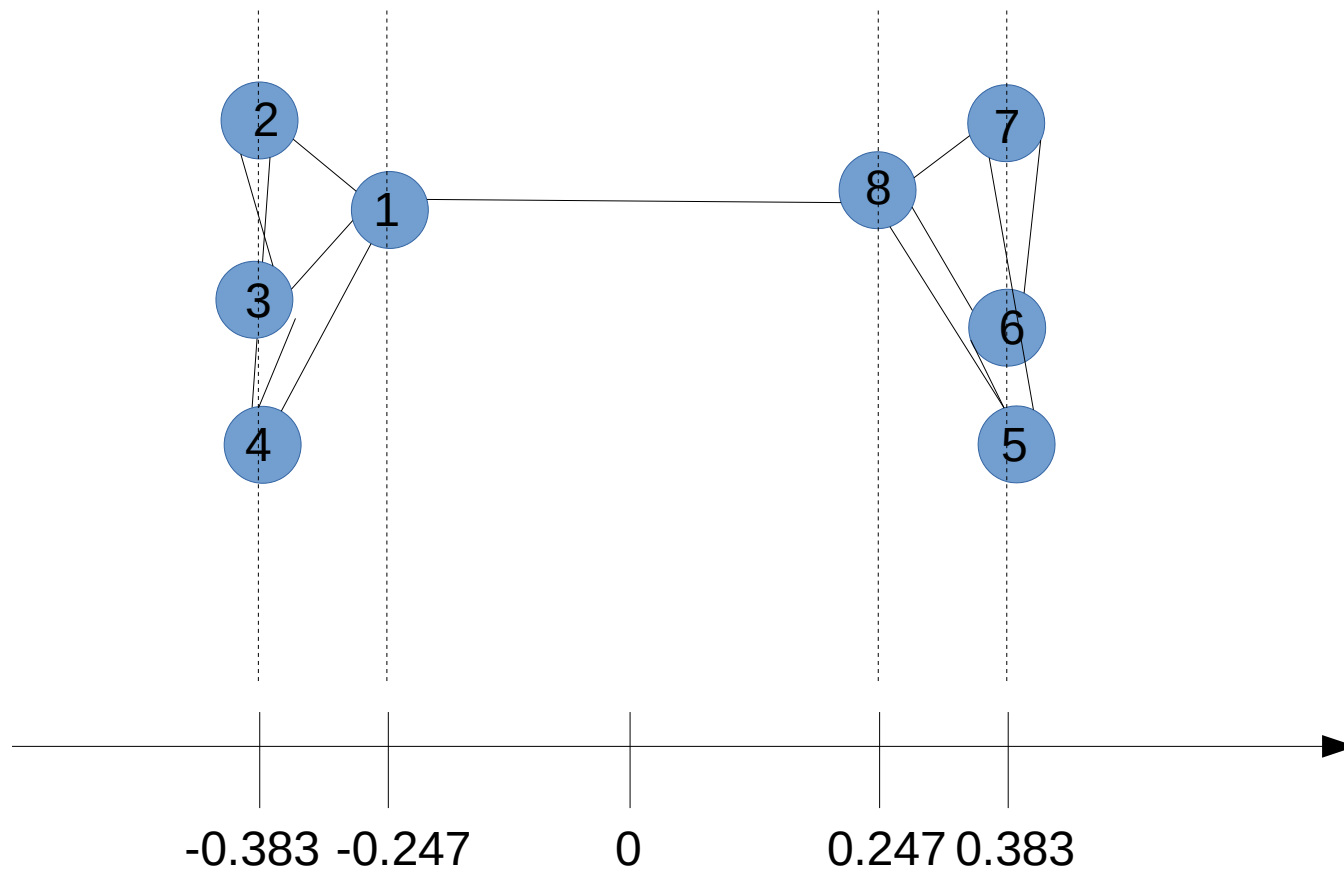
$$\lambda_1 = 0$$

$$\lambda_2 = 0.354$$

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

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

Example Graph 1, projected in \mathbb{R}^1

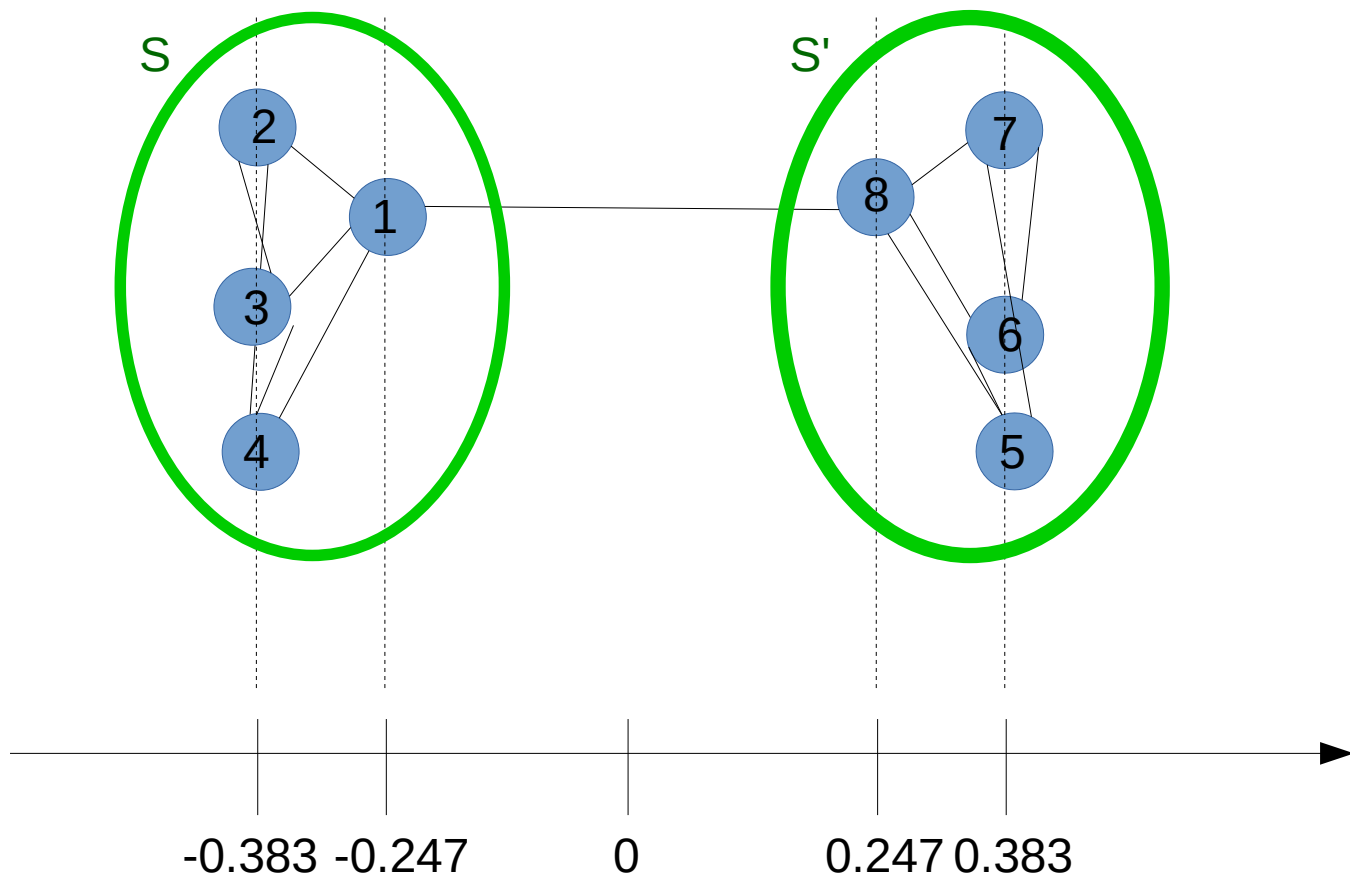


$$\lambda_1 = 0$$

$$\lambda_2 = 0.354$$

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

Example Graph 1, communities

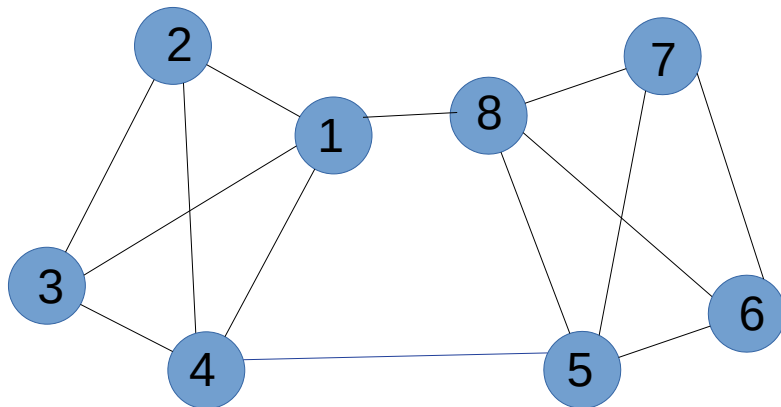


$$\lambda_1 = 0$$

$$\lambda_2 = 0.354$$

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

Example Graph 2



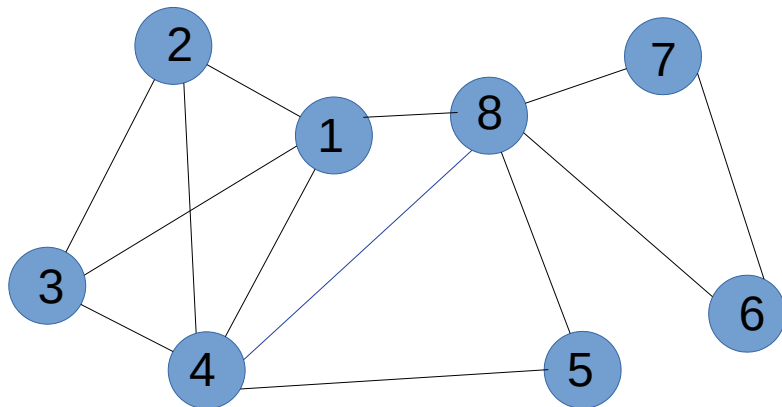
$$\lambda_1 = 0$$

$$\lambda_2 = 0.764$$

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

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

Example Graph 3



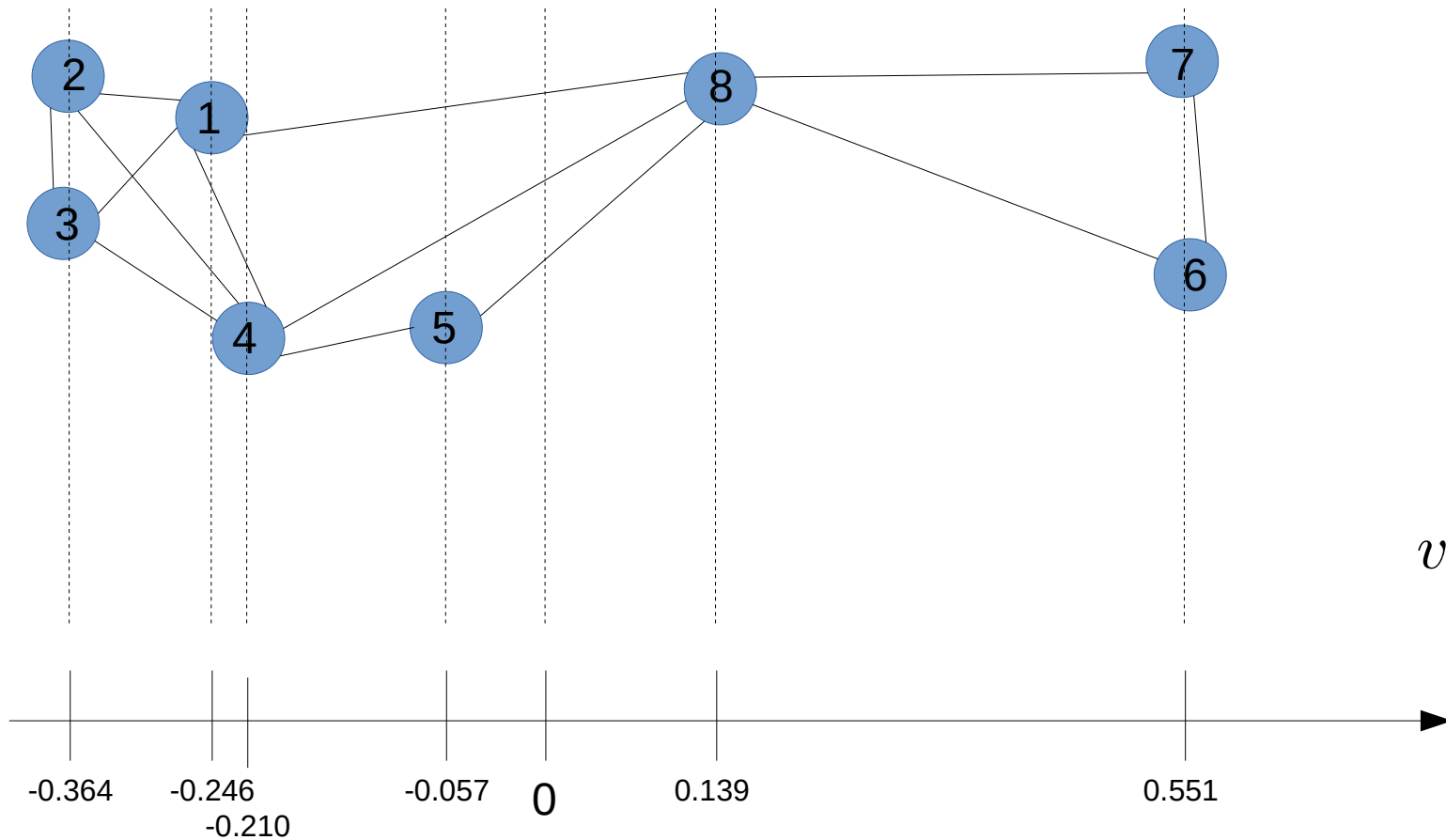
$$\lambda_1 = 0$$

$$\lambda_2 = 0.748$$

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

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

Example Graph 3, projected (where to cut?)

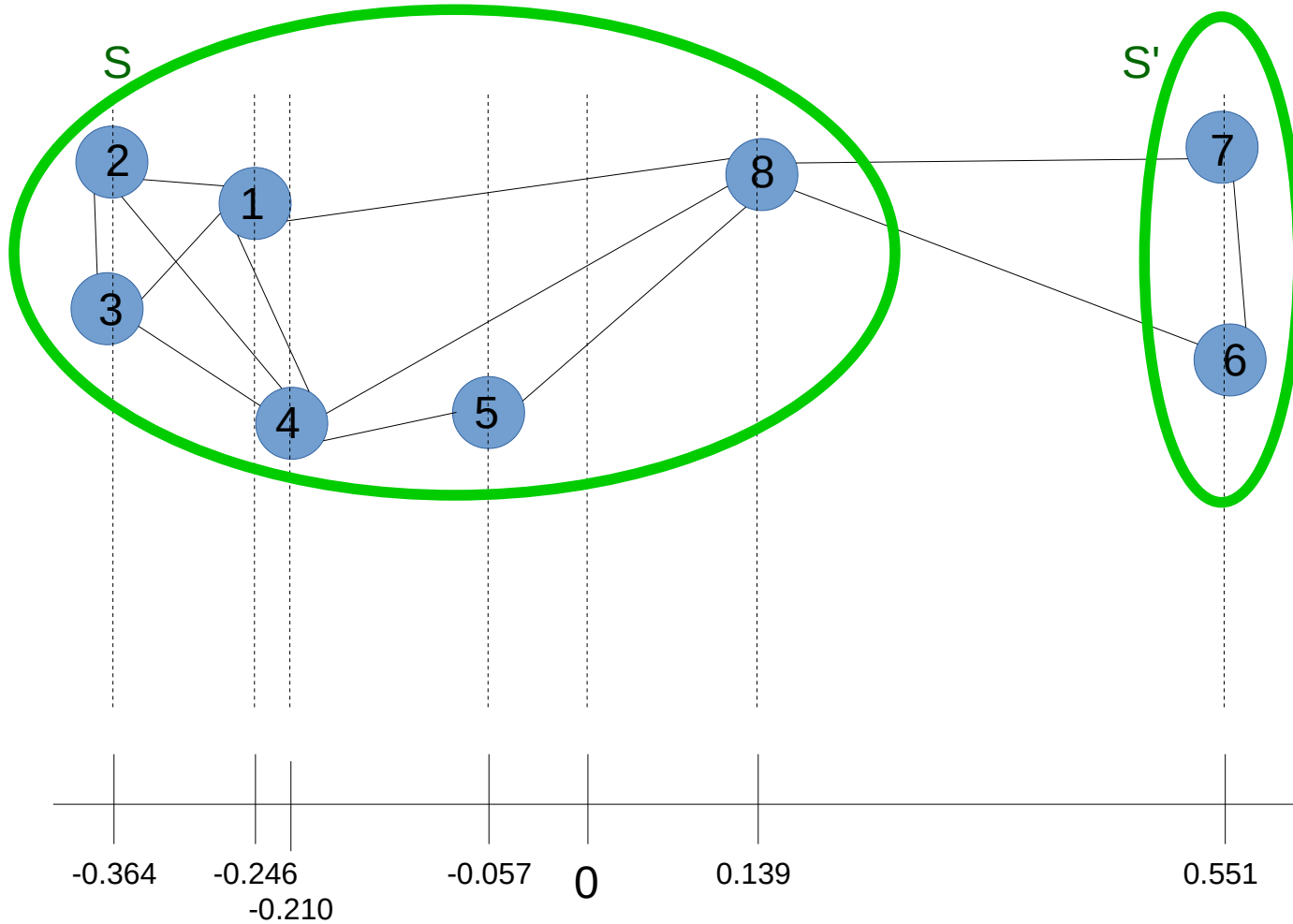


$$\lambda_1 = 0$$

$$\lambda_2 = 0.748$$

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

Example Graph 3, projected (where to cut?)

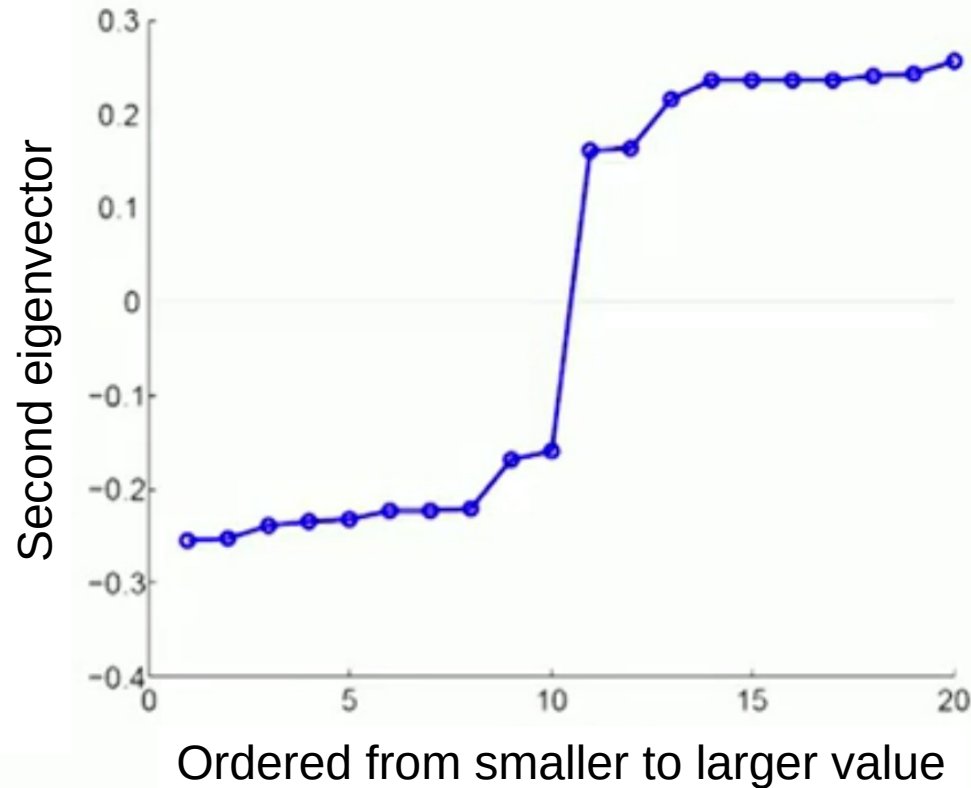
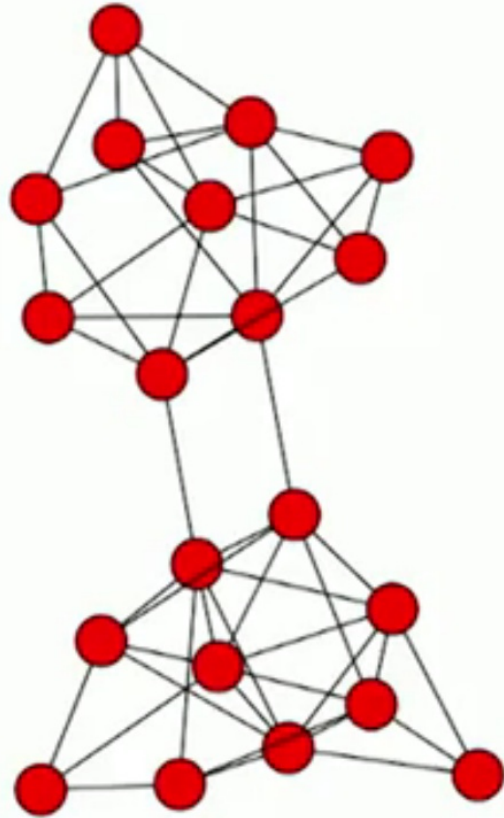


$$\lambda_1 = 0$$

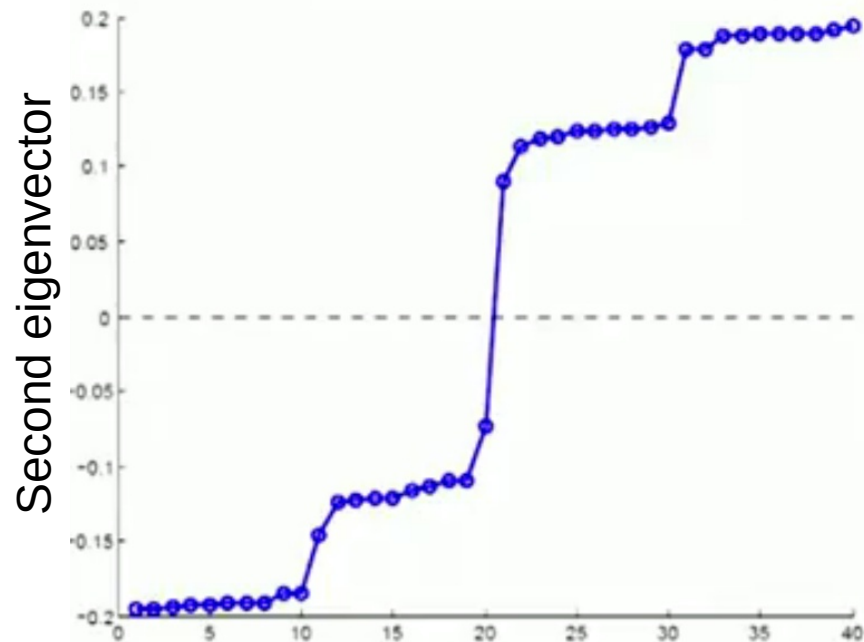
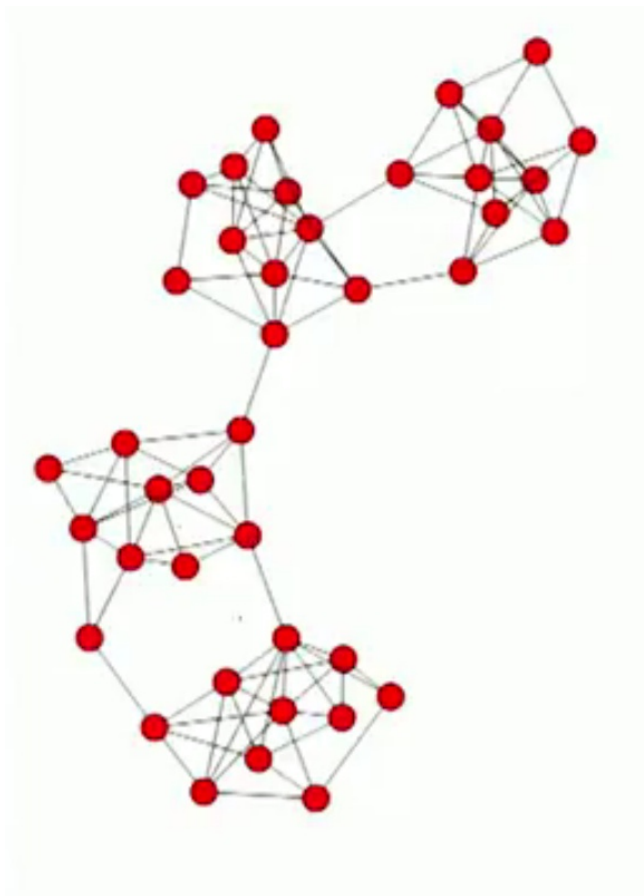
$$\lambda_2 = 0.748$$

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

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



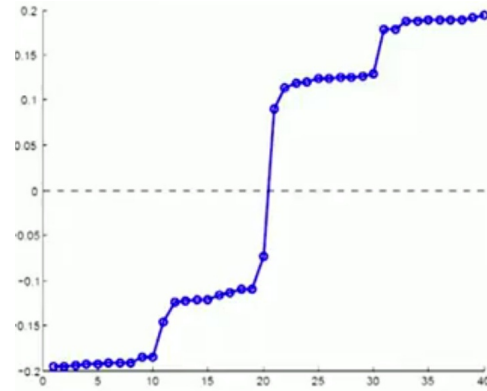
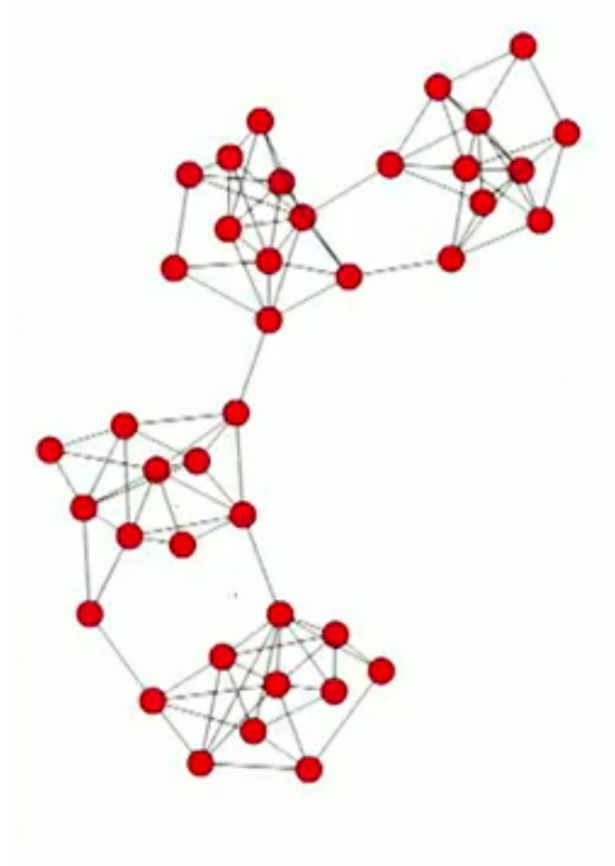
A graph with four communities in \mathbb{R}^1



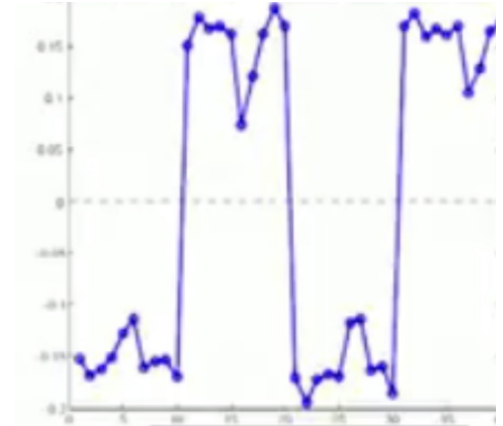
Ordered from smaller to larger value

Application: graph drawing

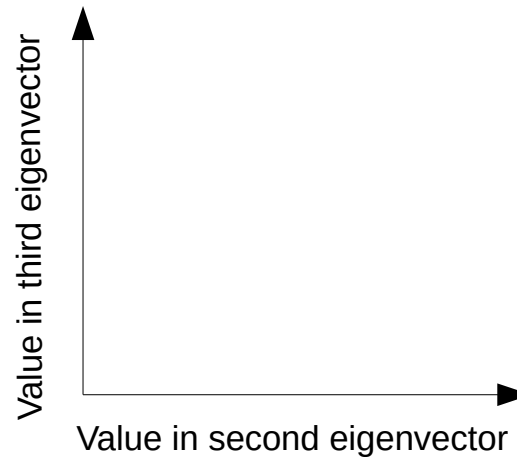
A graph with four communities in \mathbb{R}^2



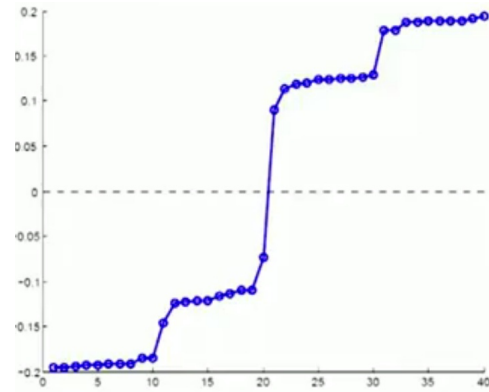
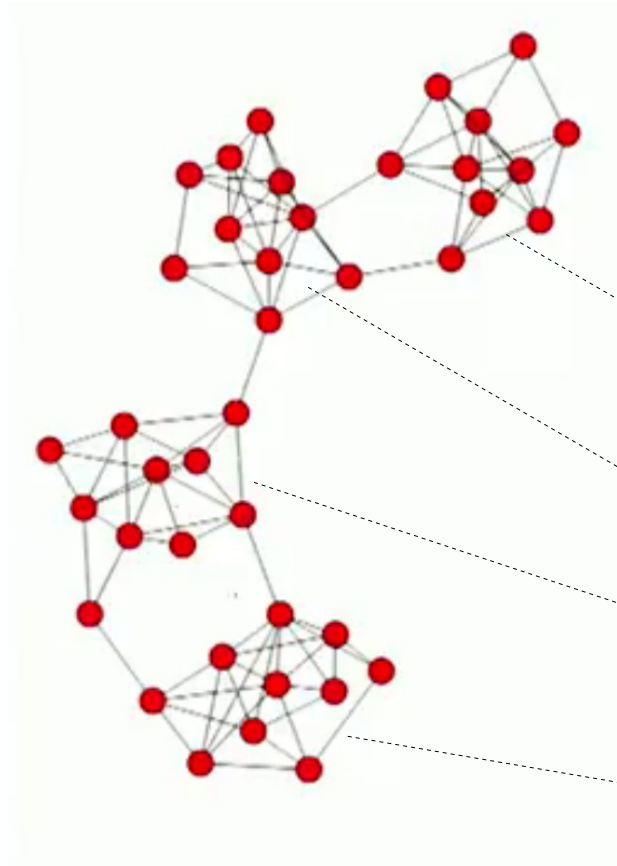
Second eigenvector



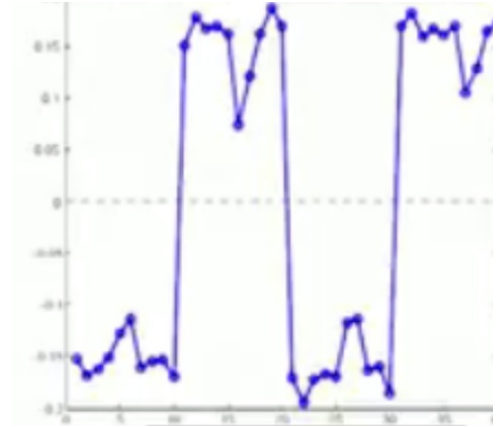
Third eigenvector



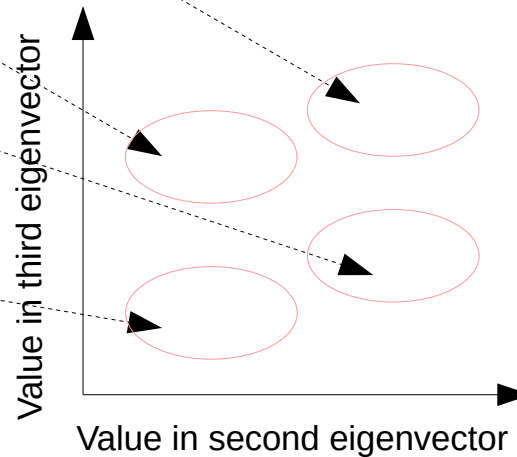
A graph with four communities in \mathbb{R}^2 (cont)



Second eigenvector

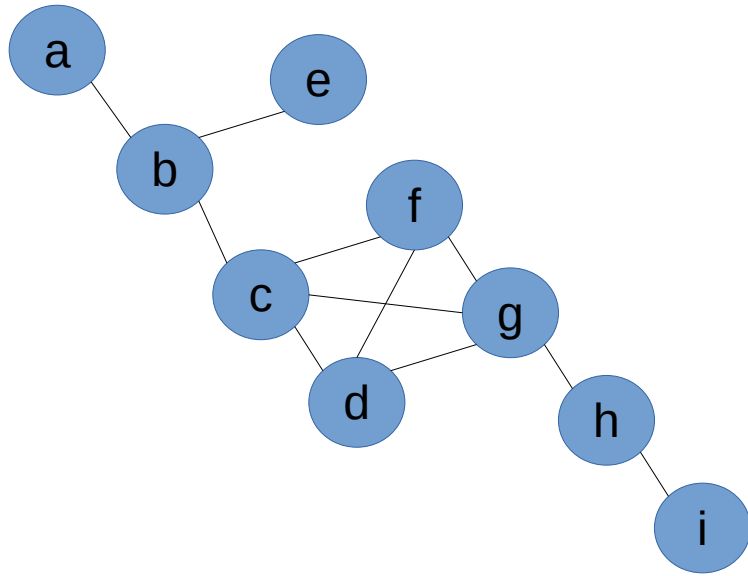


Third eigenvector

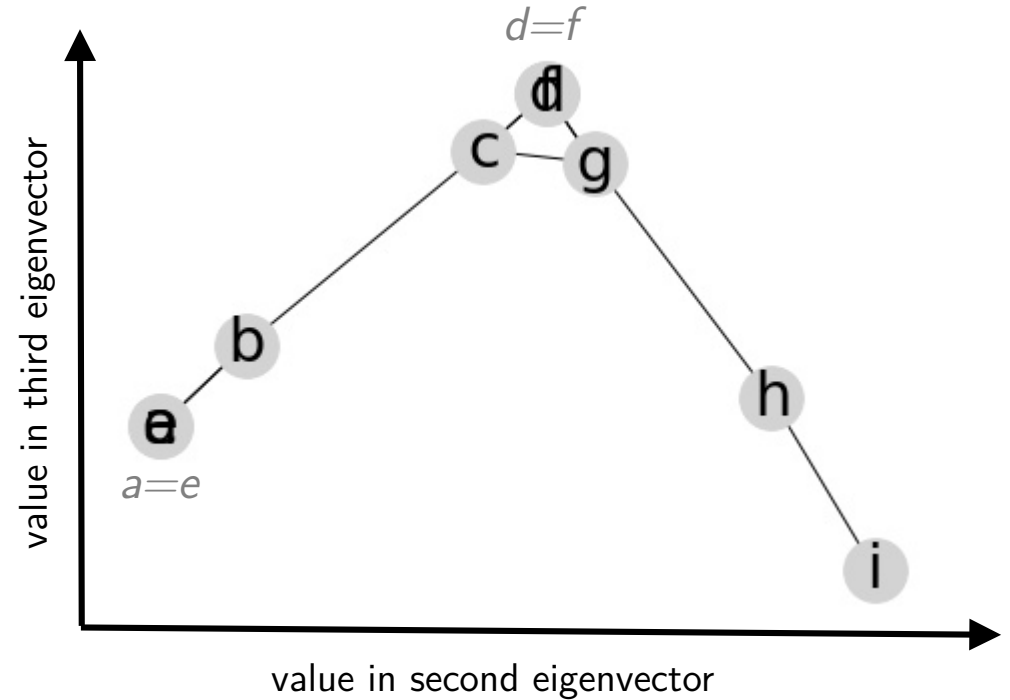


This can be used to draw the graph in \mathbb{R}^2

The graph from the initial exercise



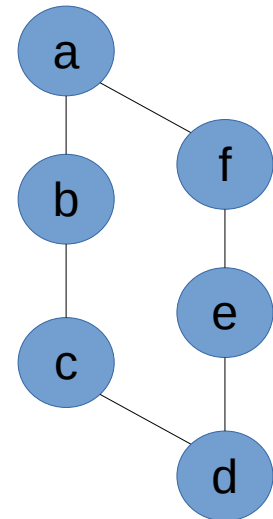
Input nodes and edges



Spectral embedding

Exercise: spectral projection

- Write the Laplacian
- Get the second and third eigenvector
- Obtain projection



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

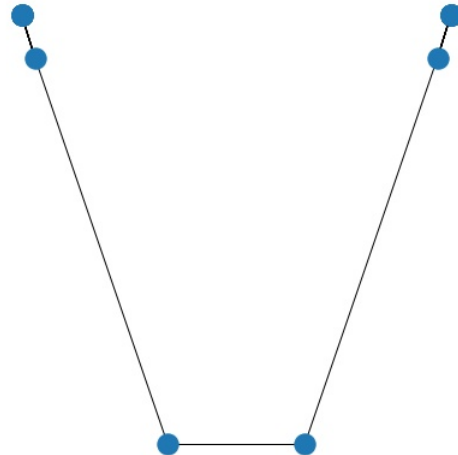
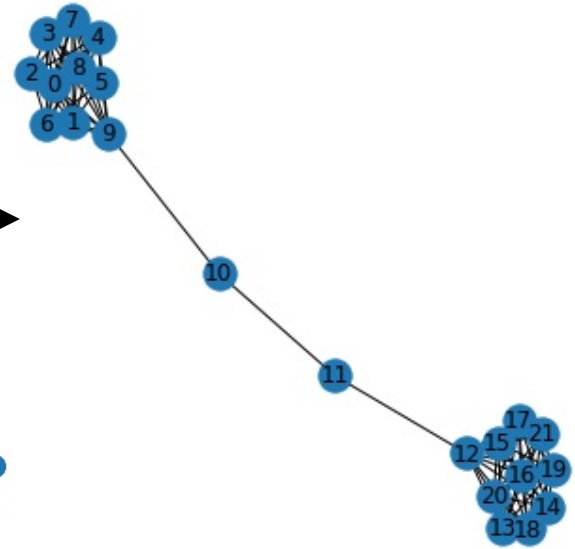


A barbell graph in R^2 (code)

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

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

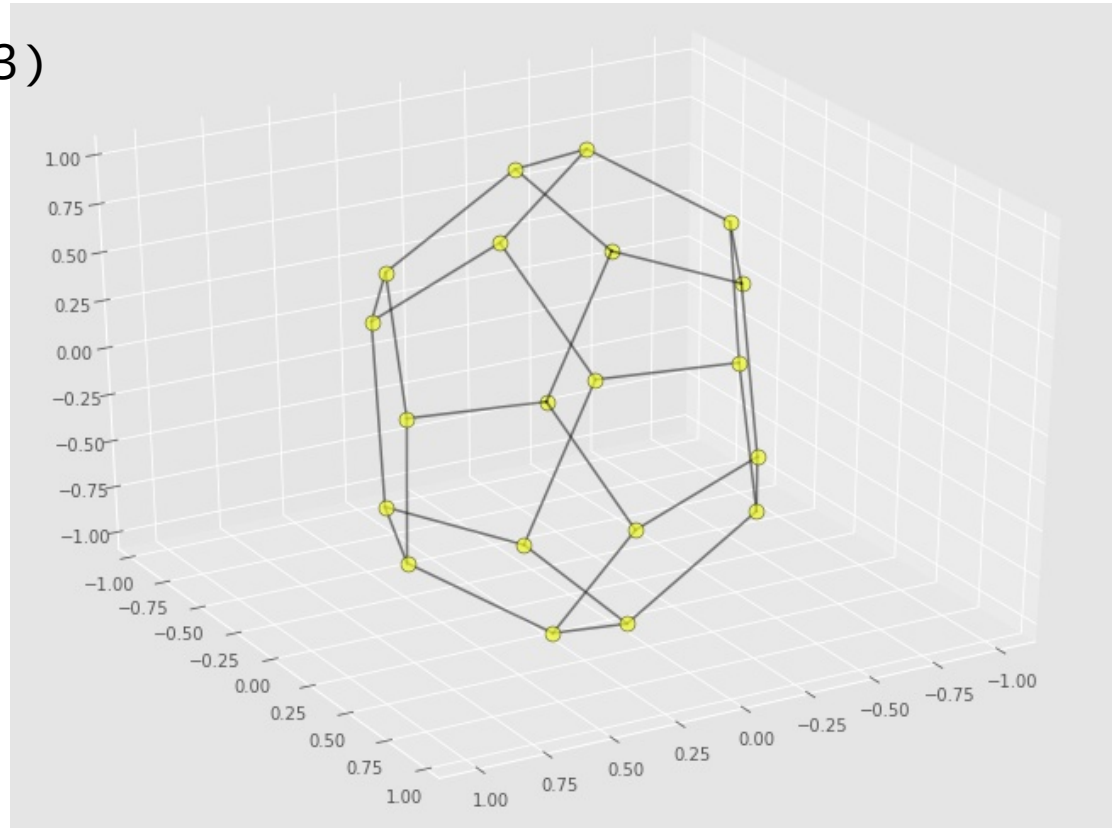
```
plt.figure(figsize=(6,6))  
nx.draw_spectral(B)  
_ = plt.show()
```



Graph Laplacian

Dodecahedral graph in 3D

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



Application: spectral clustering

Generating data

```
from sklearn.datasets import  
    make_blobs
```

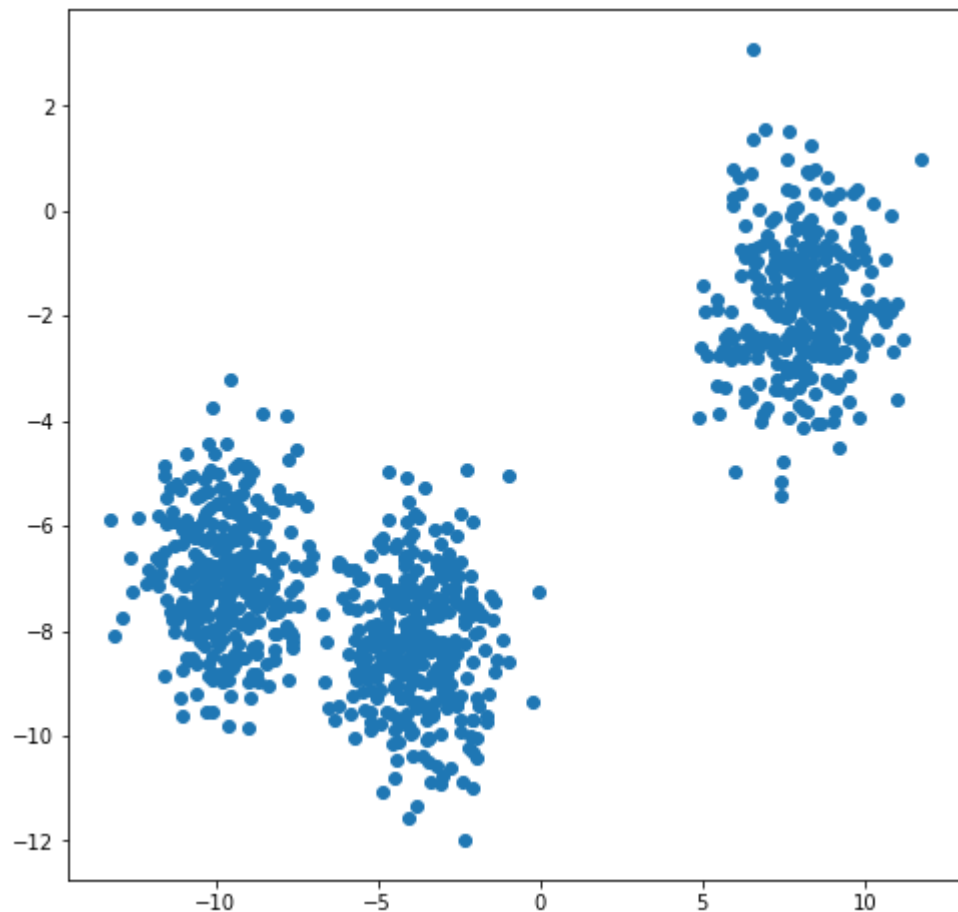
```
N = 1000
```

```
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

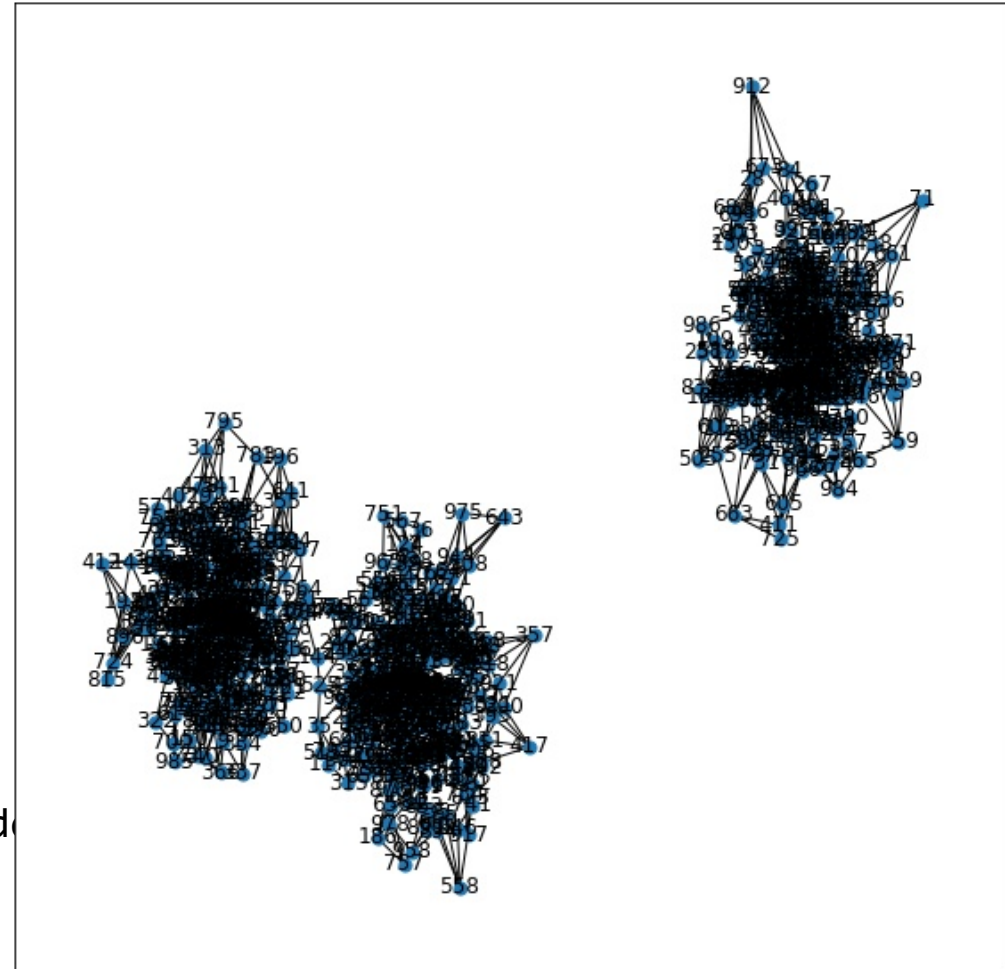
```
from sklearn.neighbors
    import NearestNeighbors

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

distances, neighbors =
    nbrs.kneighbors(x)

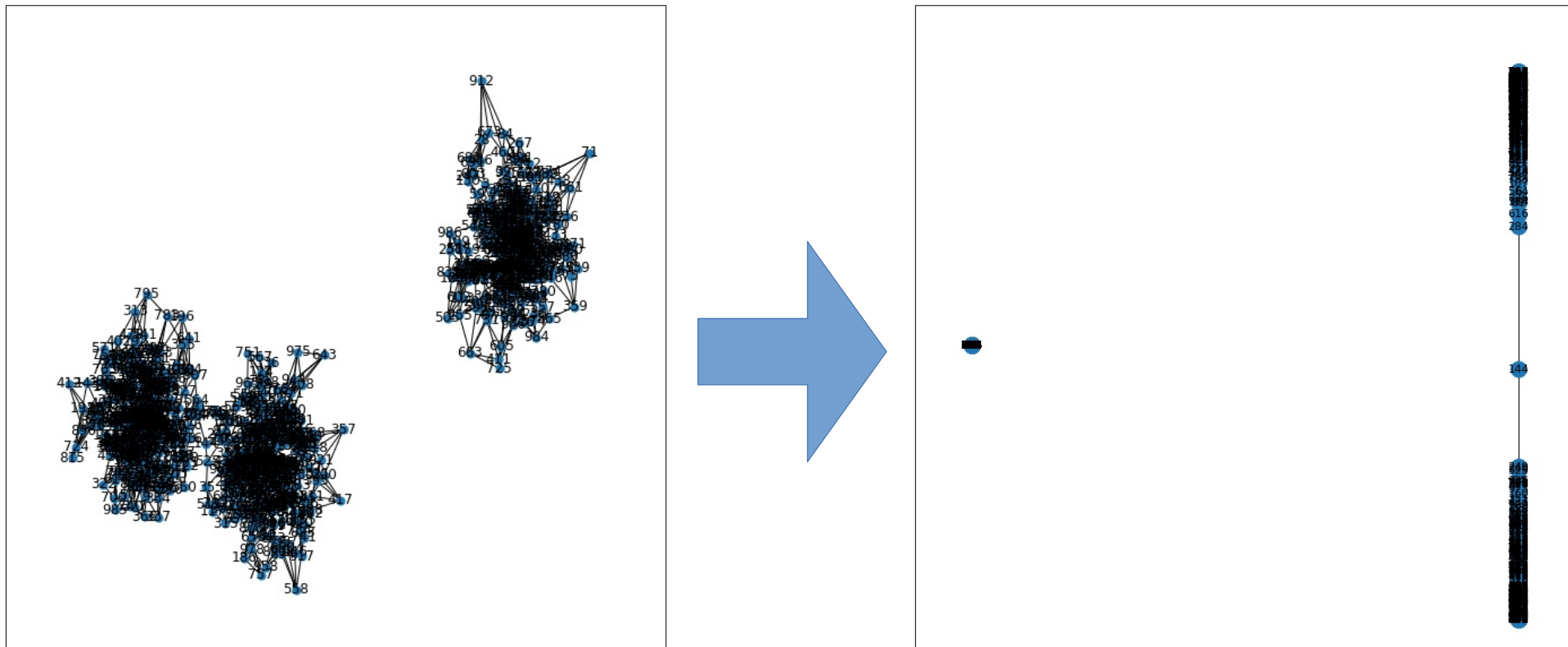
G = nx.Graph()

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)
```



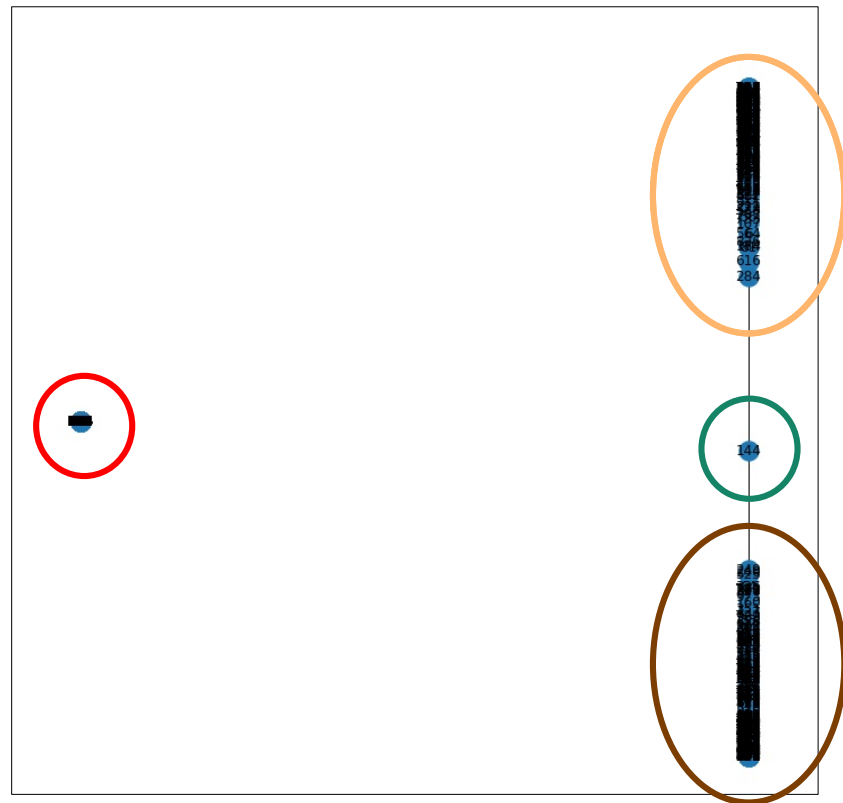
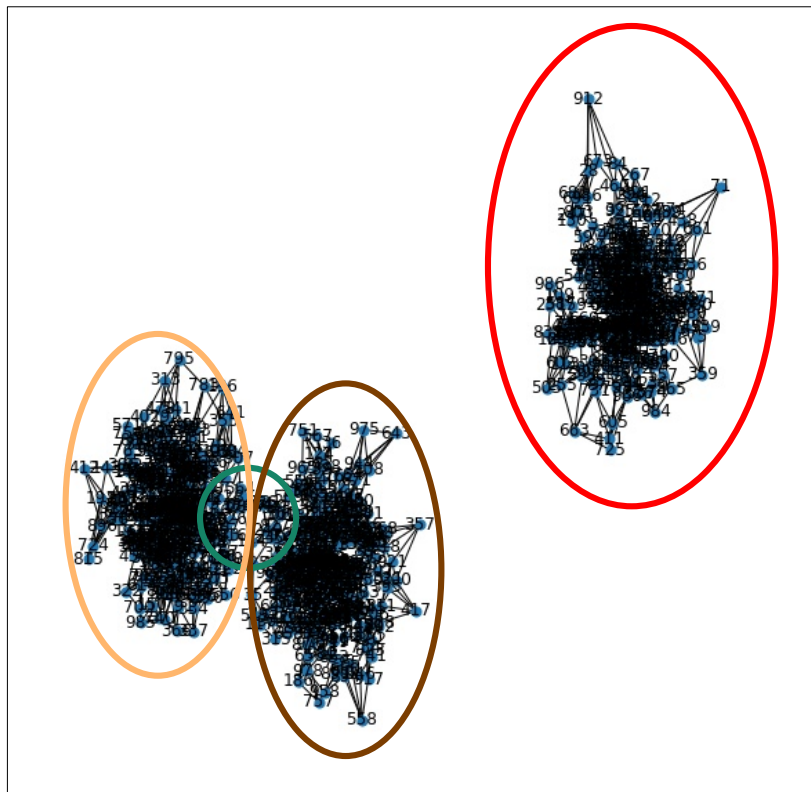
Perform spectral embedding

```
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

Exercises for this topic

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