Spectral Graph Theory

Dr Dorina Thanou

March 4, 2025

Spectral graph theory in a nutshell

- Another way to look at networks or graphs
  - We represent the graph as a connectivity matrix
  - We study the eigenvectors and the eigenvalues of that matrix

Quantum mechanics

$$i\hbar \frac{\partial}{\partial t}|\psi(t)\rangle = \hat{H}|\psi(t)\rangle$$

Schrödinger equation

- What makes eigenvalues interesting:
  - Eigenvalues are usually related to vibrations
  - Used by Shannon to determine the theoretical limit of information transmission
  - Useful for solving the Schrödinger equation
  - Define the natural frequencies of the bridge

Can we discover properties of the graph from the spectrum?

- Spectral graph theory: A topic studied from different perspectives
  - Theoretical computer science, machine learning, statistics
  - Differential geometry, mathematics, astronomy, chemistry, computer vision…

Outline

- Graph Laplacian operator
- Eigendecomposition of the graph Laplacian
  - What do the eigenvalues reveal about the graph?
  - What are the basic properties of the eigenvectors?
- Applications
  - Spectral embeddings
  - Spectral clustering
  - PageRank

Outline

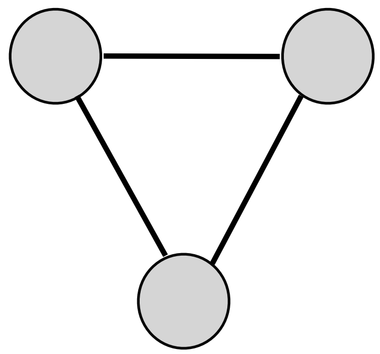
- **Graph Laplacian operator**

- Eigendecomposition of the graph Laplacian
  - What do the eigenvalues reveal about the graph?
  - What are the basic properties of the eigenvectors?

- Applications
  - Spectral embeddings
  - Spectral clustering
  - PageRank

Recap of classical graph matrices

- Undirected graph of N nodes, i.e., $|\mathcal{V}| = N$:



$$G = (\mathcal{V}, \mathcal{E}, W), \quad \mathcal{E} \subseteq \{(i,j) : i,j \in \mathcal{V}\}, \quad (i,j) = (j,i)$$

- Adjacency matrix or weight matrix :

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

$$\downarrow$$

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

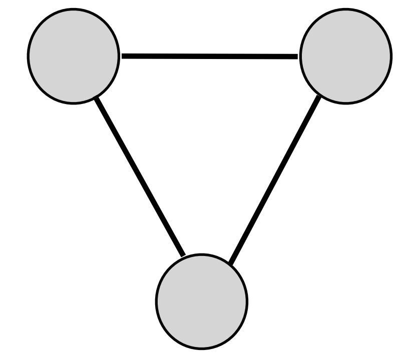
- If the graph is unweighted (often denoted as A) :

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

Recap of classical graph matrices

- Neighborhood of node i : Set of nodes connected to node i by an edge

$$\mathcal{N}_i = \{j : (i,j) \in \mathcal{E}\}$$



$$\downarrow$$

- Degree of a node i : It is the sum of the weights of the edges incident to node i

$$D_i = \sum_{j \in \mathcal{N}_i} W_{ij}$$

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

$$\downarrow$$

- Degree matrix: A diagonal matrix containing the degree of each node

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

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

The graph Laplacian matrix

- The combinatorial Laplacian is defined as:

$$L = D - W$$

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

$$\downarrow$$

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

  - Symmetric
  - Off-diagonal entries non-positive
  - Rows sum up to zero

- It is a positive semi-definite matrix:
  - For each function $f : \mathcal{V} \rightarrow \mathbb{R}$, where f_i is the value on the i^{th} node of the graph:

$$f^T L f = f^T (D - W) f = \sum_{i=1}^{N} D_{ii} f_i^2 - \sum_{i,j=1}^{N} f_i f_j W_{ij}$$

$$= \frac{1}{2} \sum_{i,j=1}^{N} W_{ij} (f_i - f_j)^2 \geq 0, \quad \forall f \in \mathbb{R}^N$$

Connection to continuous Laplacian

- Graph Laplacian: A discrete differential operator

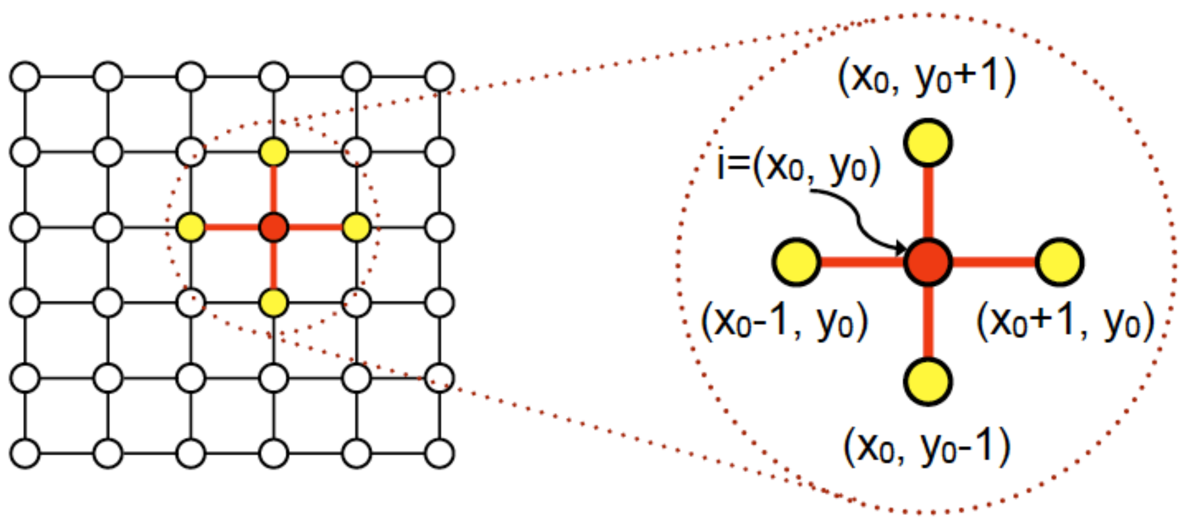
$$(Lf)(i) = \sum_{j \in \mathcal{N}_i} W_{i,j}(f_i - f_j)$$

- The Laplace operator:
  - A second-order differential operator: divergence of the gradient $\Delta f = \nabla^2 f$
  - The gradient is defined as: $\nabla f = (\frac{\partial f}{\partial x_1}, ..., \frac{\partial f}{\partial x_N})$
  - Finally, the Laplacian is: $\Delta f = \sum_{i=1}^{N} \frac{\partial^2 f}{\partial x_i^2}$

- The Laplacian matrix is the graph analogue to the Laplace operator on continuous functions!

Illustrative example

- Example: Unweighted grid graph

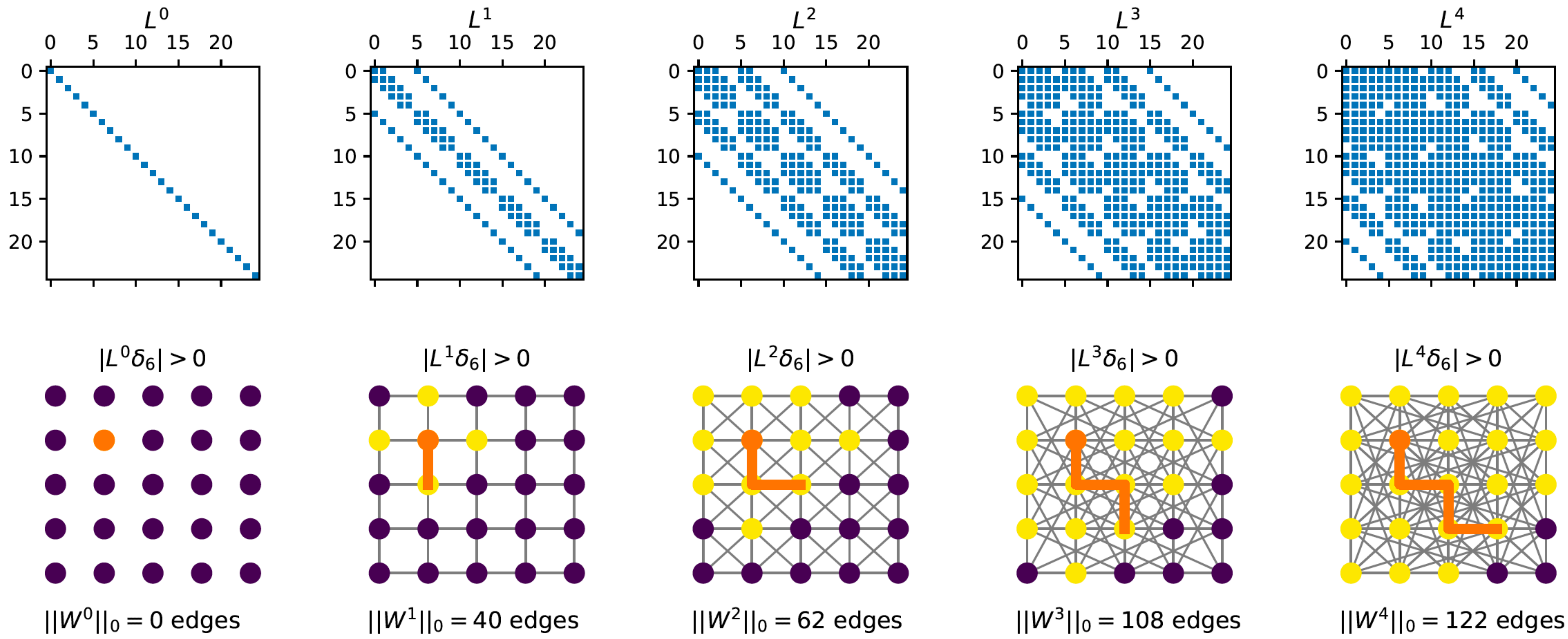


$$(Lf)(i) = \sum_{j \in \mathcal{N}_i} W_{i,j}(f_i - f_j)$$

$$-(Lf)(i) = [f(x_0+1, y_0) - f(x_0, y_0)] - [f(x_0, y_0) - f(x_0-1, y_0)]$$
$$+ [f(x_0, y_0+1) - f(x_0, y_0)] - [f(x_0, y_0) - f(x_0, y_0-1)]$$
$$\sim \frac{\partial^2 f}{\partial x^2}(x_0, y_0) + \frac{\partial^2 f}{\partial y^2}(x_0, y_0) = (\Delta f)(x_0, y_0)$$

Powers of the graph Laplacian

L^K defines the K-hop neighborhood: $d_G(v_i, v_j) > K \rightarrow (L^K)_{ij} = 0$



[Slide adapted from M. Defferrard]

Other Laplacian matrices

- Normalized Laplacian:
  - Symmetric matrix
  - Bounded spectrum (more in the following slides)

$$L_{sym} = D^{-1/2} L D^{-1/2} = I - D^{-1/2} W D^{-1/2}$$

- Random walk Laplacian:
  - Asymmetric matrix
  - Used often in dimensionality reduction techniques

$$L_{rw} = D^{-1} L = I - D^{-1} W$$

Random walk matrix

Outline

- Graph Laplacian operator

- **Eigendecomposition of the graph Laplacian**
  - **What do the eigenvalues reveal about the graph?**
  - What are the basic properties of the eigenvectors?

- Applications
  - Spectral embeddings
  - Spectral clustering
  - PageRank

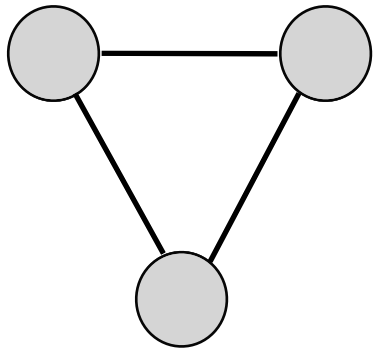
Spectral decomposition of the Laplacian matrix

- L has a complete set of orthonormal eigenvectors $L = \chi \Lambda \chi^T$

$$L = \underbrace{\begin{bmatrix} | & & | \\ \chi_1 & \dots & \chi_N \\ | & & | \end{bmatrix}}_{\chi} \underbrace{\begin{bmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_N \end{bmatrix}}_{\Lambda} \underbrace{\begin{bmatrix} - & \chi_1 & - \\ & \dots & \\ - & \chi_N & - \end{bmatrix}}_{\chi^T}$$

- Eigenvalues are usually sorted increasingly: $0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_N$
- In the case of the normalized Laplacian: $\lambda_N \leq 2$

Back to our toy example



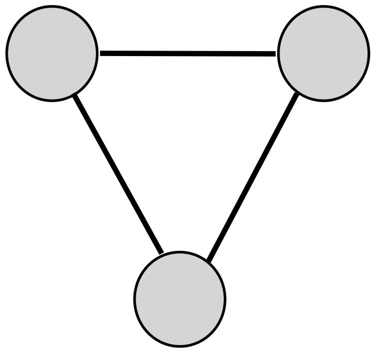
$$\rightarrow L = \begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix}$$

- From the spectral decomposition: $L\chi = \Lambda\chi$

- What is an eigenvector of L?

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

Back to our toy example



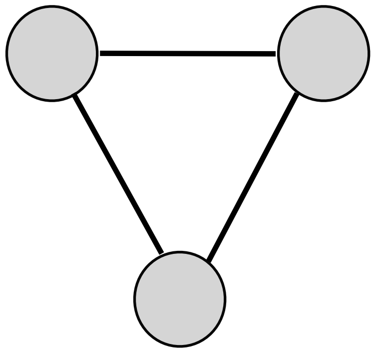
$$\rightarrow L = \begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix}$$

- From the spectral decomposition: $L\chi = \Lambda\chi$

- What is an eigenvector of L?

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

Back to our toy example

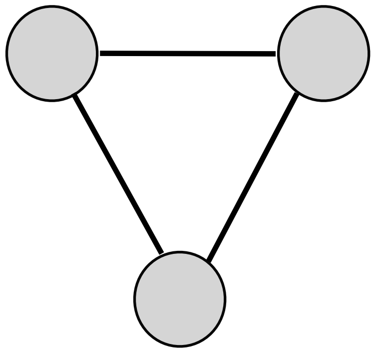


$$\rightarrow L = \begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix}$$

- From the spectral decomposition: $L\chi = \Lambda\chi$
- What is an eigenvector of L?

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

Back to our toy example



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

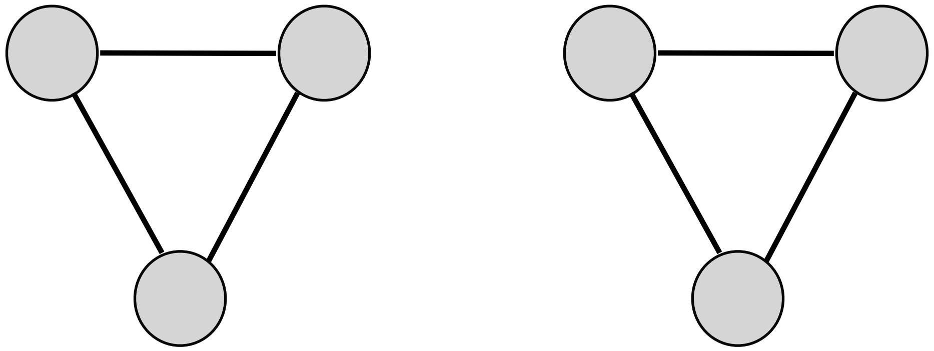
- From the spectral decomposition: $L\chi = \Lambda\chi$
- What is an eigenvector of L?

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

For any graph, $\chi_1 = [1, 1, \ldots, 1]^T$ is always an eigenvector with eigenvalue 0!

An extended toy example

- Consider a network of two disconnected components

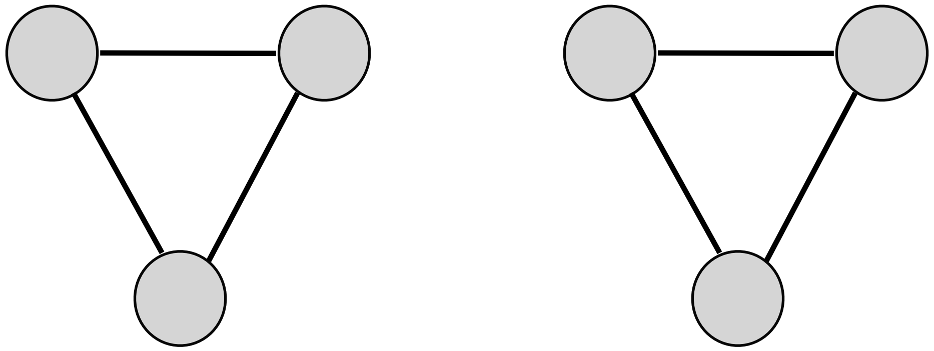


- How does the first eigenvector change?

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

An extended toy example

- Consider a network of two disconnected components

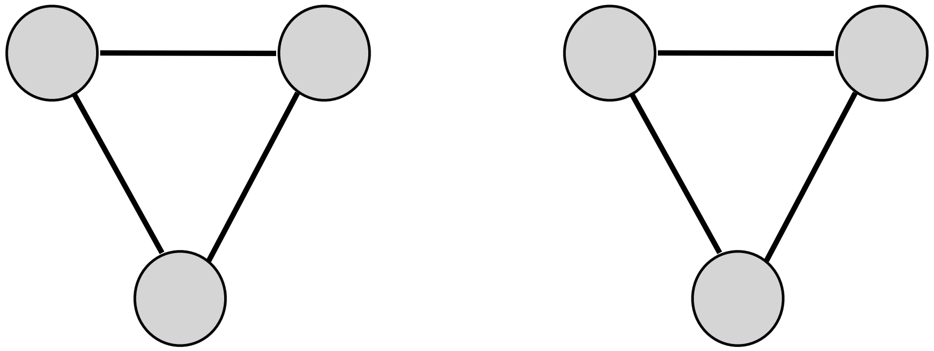


- How does the first eigenvector change?

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

An extended toy example

- Consider a network of two disconnected components



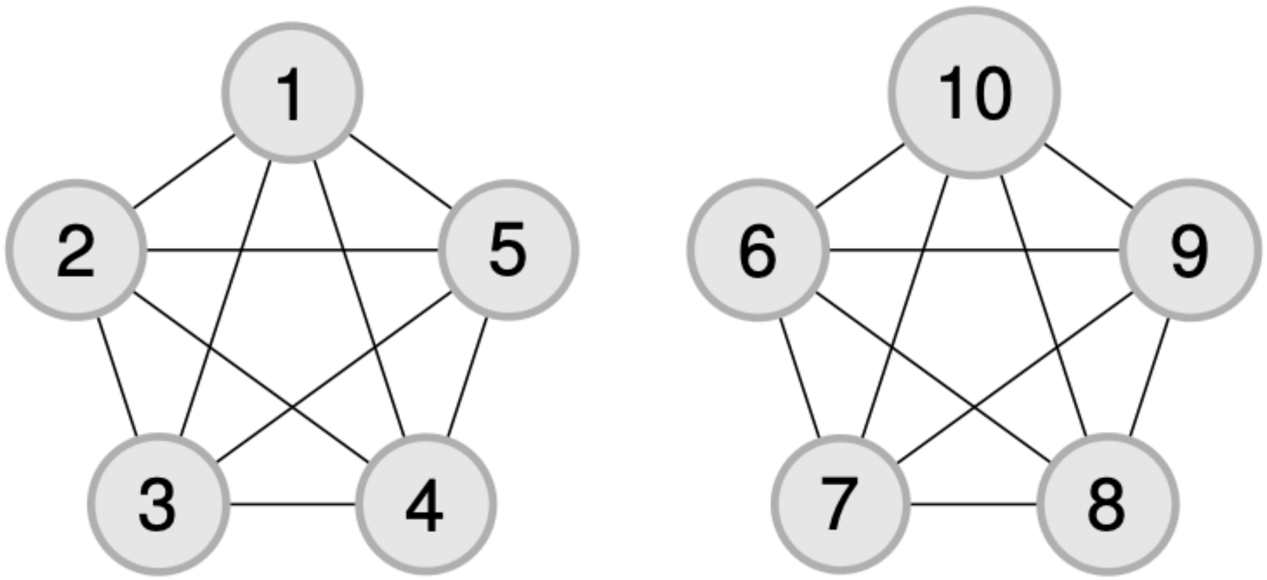
- How does the first eigenvector change?

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

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

Fiedler vector

- The second eigenvalue is $\lambda_2 > 0$ iff the graph is connected
- More connected graphs have higher values of λ_2

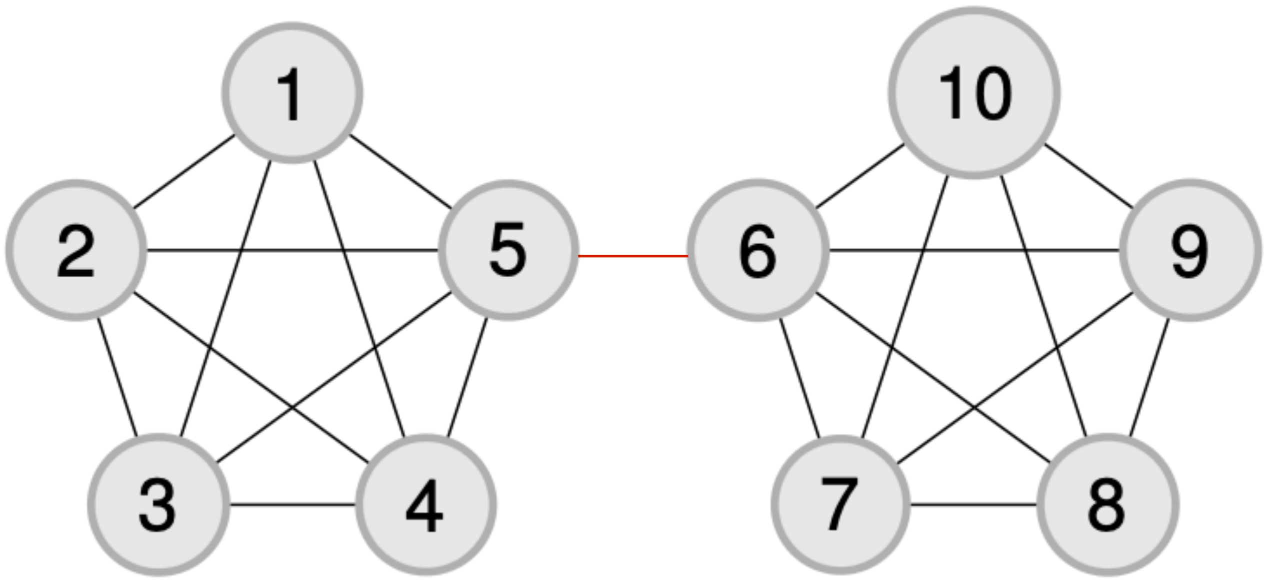


$$\lambda_2 = 0$$

- The eigenvalue λ_2 is called the algebraic connectivity
- The eigenvector corresponding to λ_2 is called the Fiedler vector

Fiedler vector

- The second eigenvalue is $\lambda_2 > 0$ iff the graph is connected
- More connected graphs have higher values of λ_2



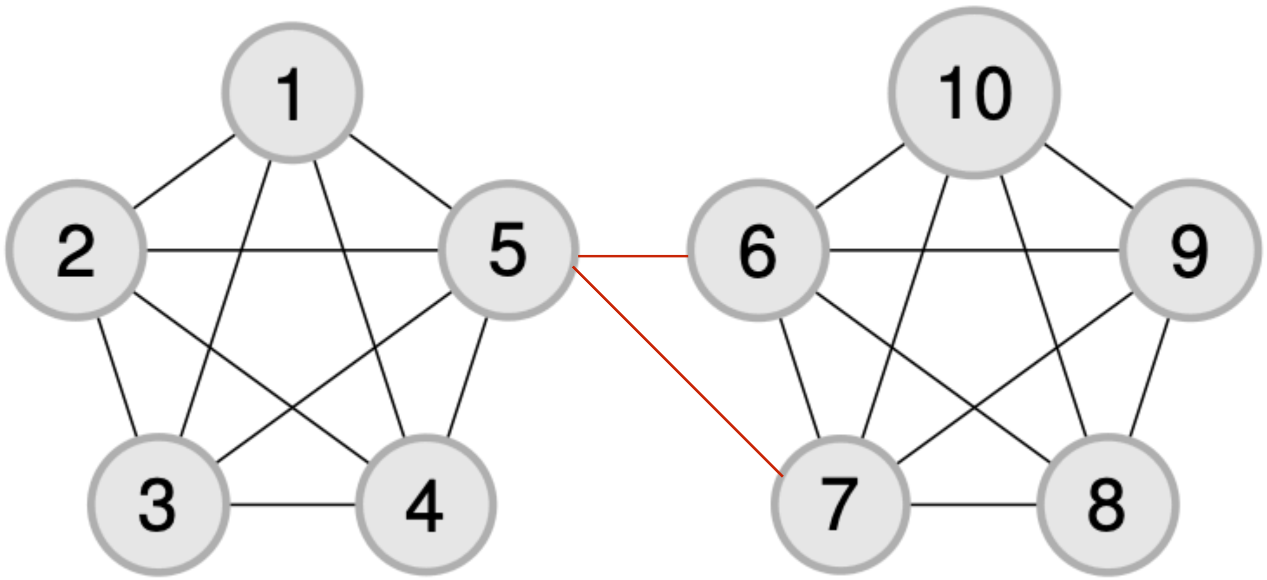
$\lambda_2 = 0$

$\lambda_2 \approx 0.298$

- The eigenvalue λ_2 is called the algebraic connectivity
- The eigenvector corresponding to λ_2 is called the Fiedler vector

Fiedler vector

- The second eigenvalue is $\lambda_2 > 0$ iff the graph is connected
- More connected graphs have higher values of λ_2



$\lambda_2 = 0$

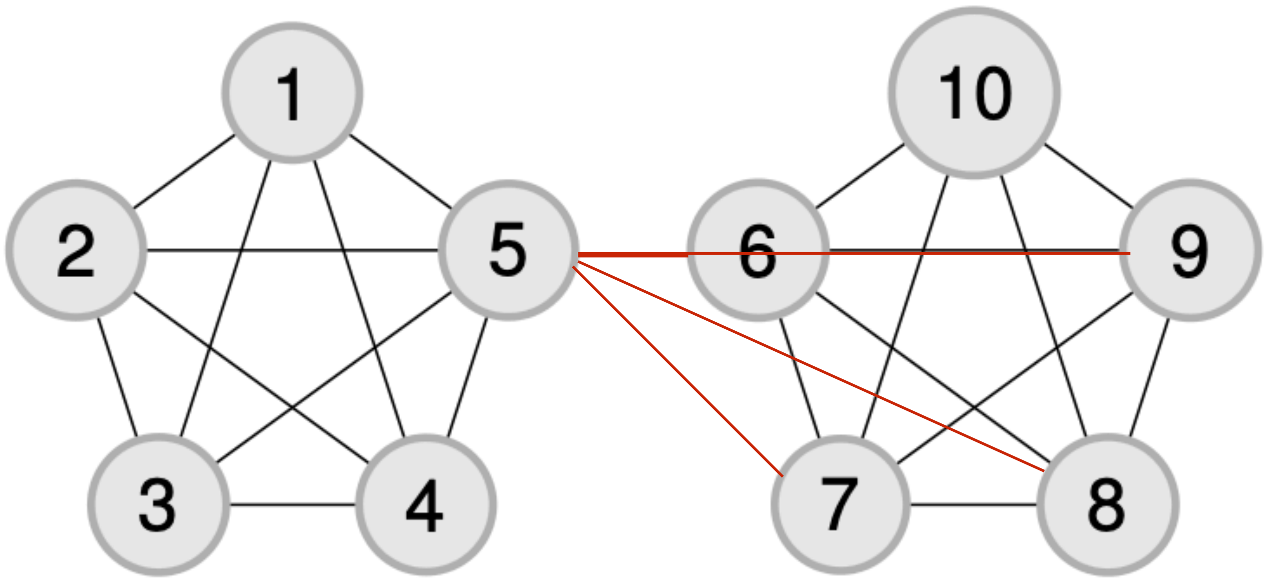
$\lambda_2 \approx 0.298$

$\lambda_2 \approx 0.535$

- The eigenvalue λ_2 is called the algebraic connectivity
- The eigenvector corresponding to λ_2 is called the Fiedler vector

Fiedler vector

- The second eigenvalue is $\lambda_2 > 0$ iff the graph is connected
- More connected graphs have higher values of λ_2



$\lambda_2 = 0$

$\lambda_2 \approx 0.298$

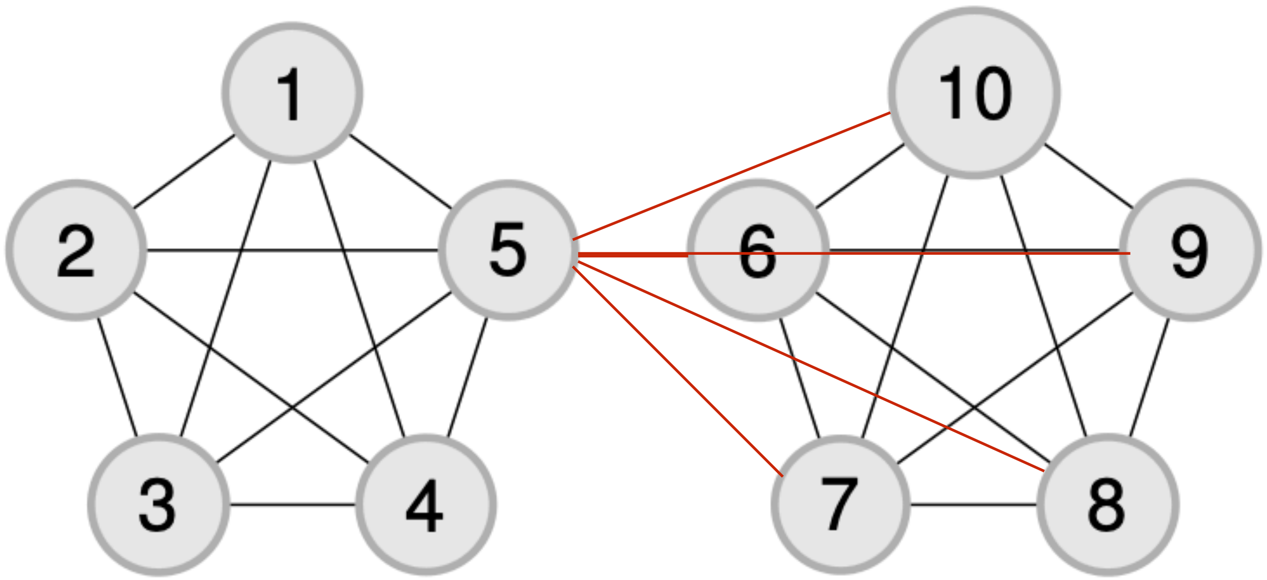
$\lambda_2 \approx 0.535$

$\lambda_2 \approx 0.876$

- The eigenvalue λ_2 is called the algebraic connectivity
- The eigenvector corresponding to λ_2 is called the Fiedler vector

Fiedler vector

- The second eigenvalue is $\lambda_2 > 0$ iff the graph is connected
- More connected graphs have higher values of λ_2



$\lambda_2 = 0$

$\lambda_2 \approx 0.298$

$\lambda_2 \approx 0.535$

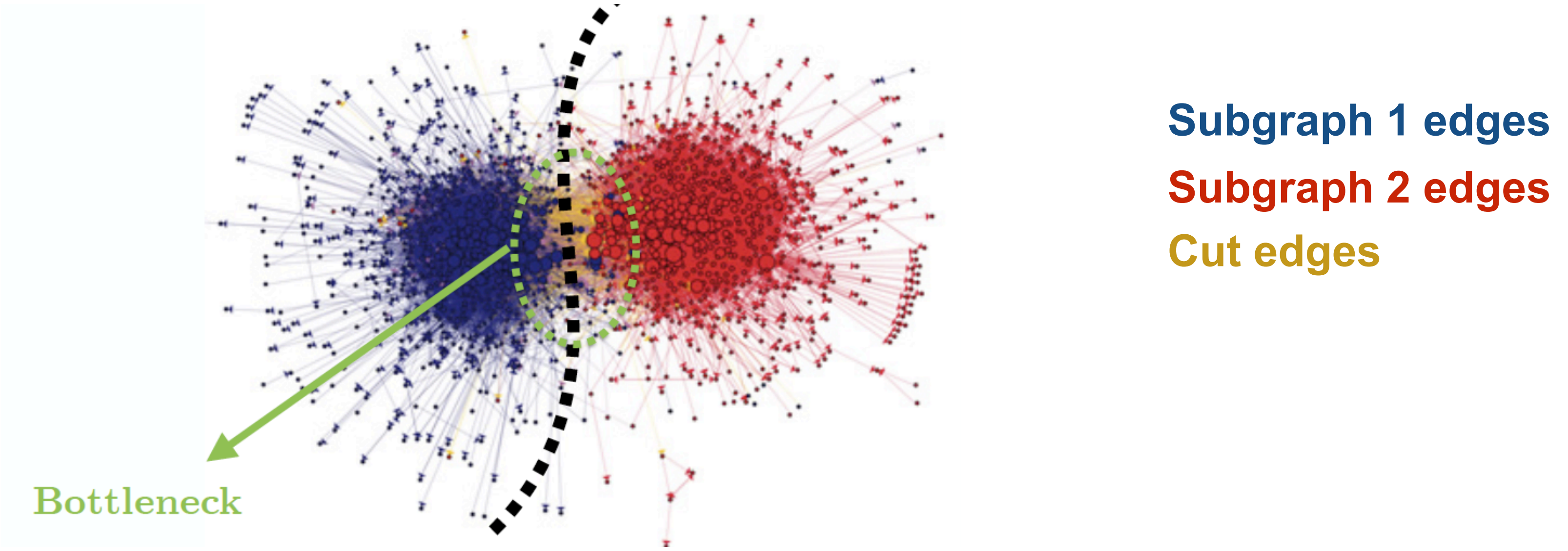
$\lambda_2 \approx 0.876$

$\lambda_2 \approx 1$

- The eigenvalue λ_2 is called the algebraic connectivity
- The eigenvector corresponding to λ_2 is called the Fiedler vector

Graph partitioning

- One of the fundamental problems when dealing with graphs
- It aims at cutting a weighted, undirected graph into two or more subgraphs, so that the total weight of the cut edges is as small as possible

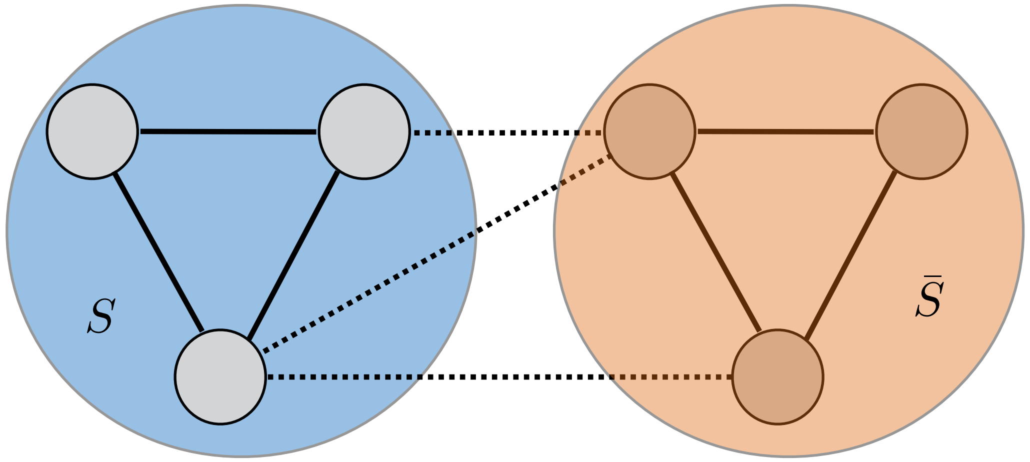


What can the spectrum tell us about partitioning the graph?

Cuts and bottlenecks

- **Cut:** Partition of the vertices into two disjoint sets
  - Let $S \subset V$, and $\bar{S} := V - S$ its complement
  - The cut induced by S is defined as $w(S, \bar{S}) := \sum_{i \in S, j \in \bar{S}} W_{ij}$
  - The volume of the set is

$$vol(S) := \sum_{i \in S} D_{ii}$$



- **Conductance** of a cut:

$$h_G(S) := \frac{w(S, \bar{S})}{\min\{vol(S), vol(\bar{S})\}}$$

- Conductance of a graph i.e., **Cheeger constant**
  - Small conductance means well-connected and partitionable subgraphs
  - Measures the presence of a bottleneck

$$h_G := \min_{S \subset V} h_G(S)$$

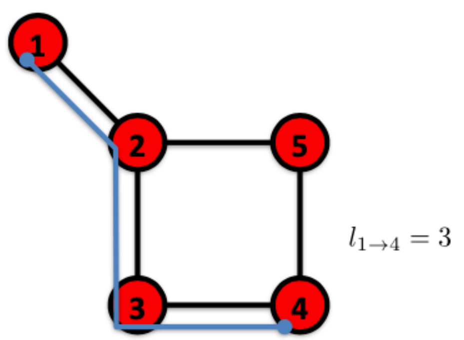
Bottlenecks and spectrum

- **Cheegers inequality:** relates the conductance of the graph with the eigenvalues of the normalized Laplacian

$$\frac{\lambda_2}{2} \leq h_G \leq \sqrt{2\lambda_2}$$

- Connection between diameter (maximum distance) and spectrum

$$d_{max}(G) \geq \frac{1}{\lambda_2 vol(G)}$$



- $\lambda_2 \to 0$: graph disconnected, large bottlenecks, large diameter
- $\lambda_2 \to 1$: graph fully connected, no bottlenecks, small diameter

Outline

- Graph Laplacian operator

- **Eigendecomposition of the graph Laplacian**
  - What do the eigenvalues reveal about the graph?
  - **What are the basic properties of the eigenvectors?**

- Applications
  - Spectral embeddings
  - Spectral clustering
  - PageRank

Rayleigh quotient

- For every function f which assigns a value to each vertex of the graph, the Rayleigh quotient of L is defined as

$$R_L(f) = \frac{f^T L f}{f^T f} = \frac{\sum_{i,j}^{N} W_{ij}(f_i - f_j)^2}{2 f^T f} \geq 0$$

- The Rayleigh quotient is maximized if f is an eigenvector of L corresponding to the largest eigenvalue
  - Hint on the proof: Set the gradient to the zero vector

$$\nabla \frac{f^T L f}{f^T f} = \frac{(f^T f)(2Lf) - (f^T L f)(2f)}{(f^T f)^2} = 0$$

$$L f = \frac{f^T L f}{f^T f} f$$

Eigenvalue!

A generalization of Rayleigh quotient

- Eigenvalues arise as a solution to natural optimization problems
- From the **Courant-Fischer theorem**, for any symmetric matrix L with increasing order of eigenvalues:

$$\chi_1 = \underset{f \in \mathbb{R}^N, \|f\|=1}{\operatorname{argmin}} f^T L f, \text{ and } \lambda_1 = \chi_1^T L \chi_1 = 0$$

$$\chi_2 = \underset{f \in \mathbb{R}^N, \|f\|=1, f \perp \chi_1}{\operatorname{argmin}} f^T L f, \text{ and } \lambda_2 = \chi_2^T L \chi_2$$

$$\vdots$$

$$\chi_N = \underset{f \in \mathbb{R}^N, \|f\|=1, f \perp \chi_1, \cdots, \chi_{N-1}}{\operatorname{argmin}} f^T L f, \text{ and } \lambda_N = \chi_N^T L \chi_N$$

- Proof can be found in chapter 1 of the book (see references)

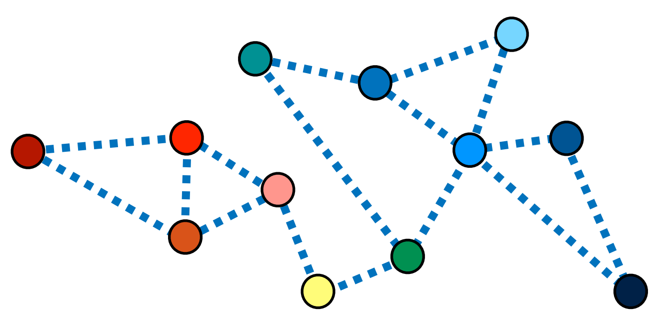
Connection with smoothness on the graph

- The smoothness of a function f on the graph is given by the graph Laplacian quadratic term

$$S_2(f) = \frac{1}{2}\sum_{i\in\mathcal{V}} \|\nabla_i f\|_2^2 = \sum_{i,j\in\mathcal{V}}^{N} W_{ij}(f_i - f_j)^2 = f^T L f$$

Proof in [6]

- $S_2(f)$ is small, i.e., the function f is smooth, when it has similar values at neighbouring vertices



$f^T L_1 f = 0.15$

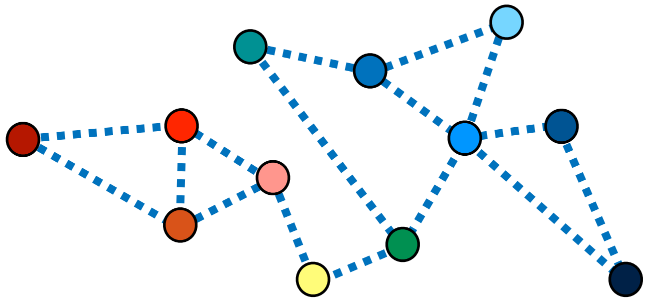
Connection with smoothness on the graph

- The smoothness of a function f on the graph is given by the graph Laplacian quadratic term

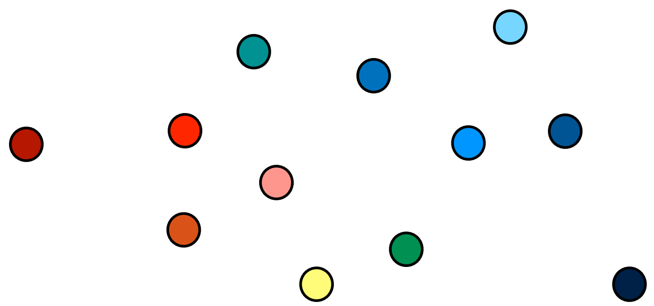
$$S_2(f) = \frac{1}{2}\sum_{i\in\mathcal{V}} \|\nabla_i f\|_2^2 = \sum_{i,j\in\mathcal{V}}^{N} W_{ij}(f_i - f_j)^2 = f^T L f$$

Proof in [6]

- $S_2(f)$ is small, i.e., the function f is smooth, when it has similar values at neighbouring vertices



$f^T L_1 f = 0.15$



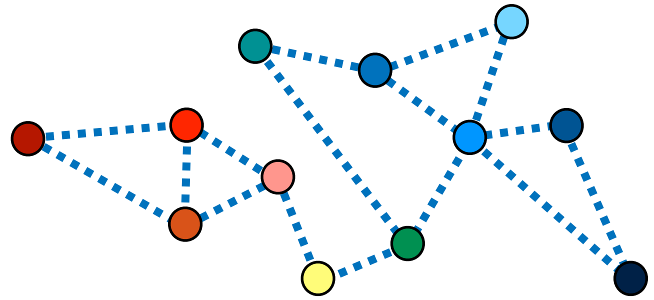
Connection with smoothness on the graph

- The smoothness of a function f on the graph is given by the graph Laplacian quadratic term

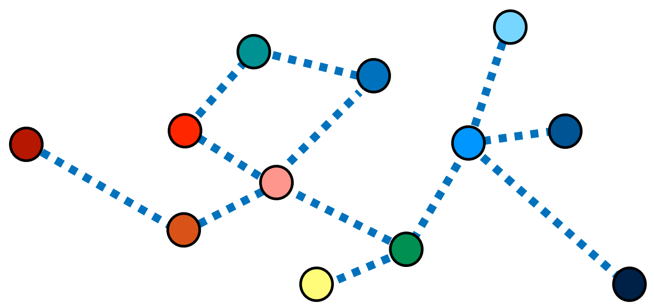
$$S_2(f) = \frac{1}{2}\sum_{i\in\mathcal{V}} \|\nabla_i f\|_2^2 = \sum_{i,j\in\mathcal{V}}^{N} W_{ij}(f_i - f_j)^2 = f^T L f$$

Proof in [6]

- $S_2(f)$ is small, i.e., the function f is smooth, when it has similar values at neighbouring vertices



$f^T L_1 f = 0.15$



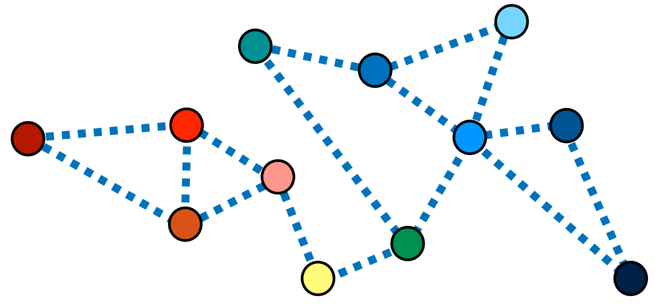
Connection with smoothness on the graph

- The smoothness of a function f on the graph is given by the graph Laplacian quadratic term

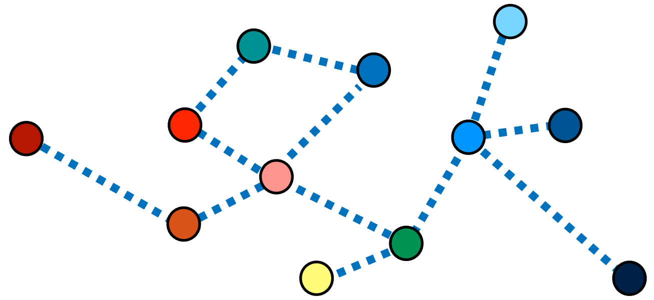
$$S_2(f) = \frac{1}{2}\sum_{i \in \mathcal{V}} \|\nabla_i f\|_2^2 = \sum_{i,j \in \mathcal{V}}^{N} W_{ij}(f_i - f_j)^2 = f^T L f$$

Proof in [6]

- $S_2(f)$ is small, i.e., the function f is smooth, when it has similar values at neighbouring vertices



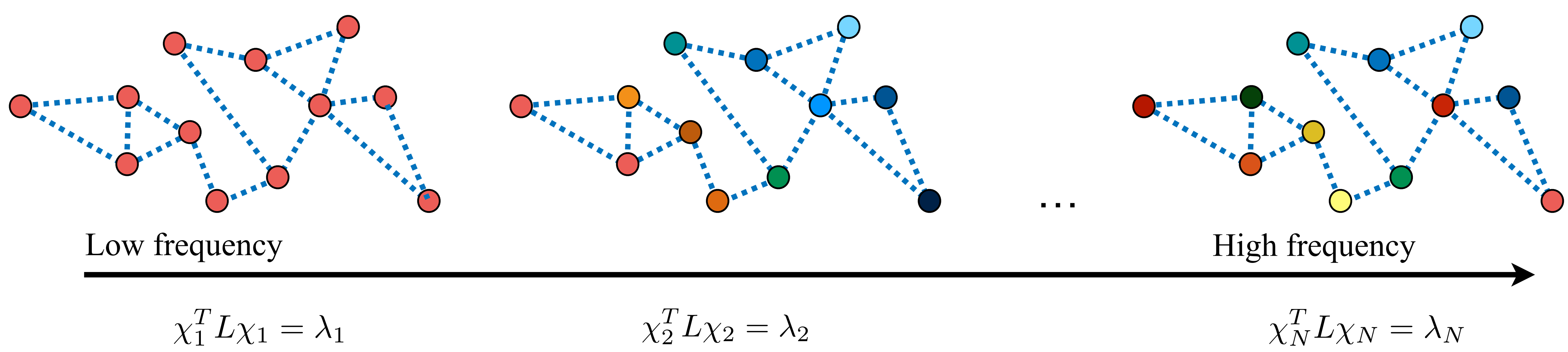
$f^T L_1 f = 0.15$



$f^T L_2 f = 1.8$

Eigenvectors as functions on the graph

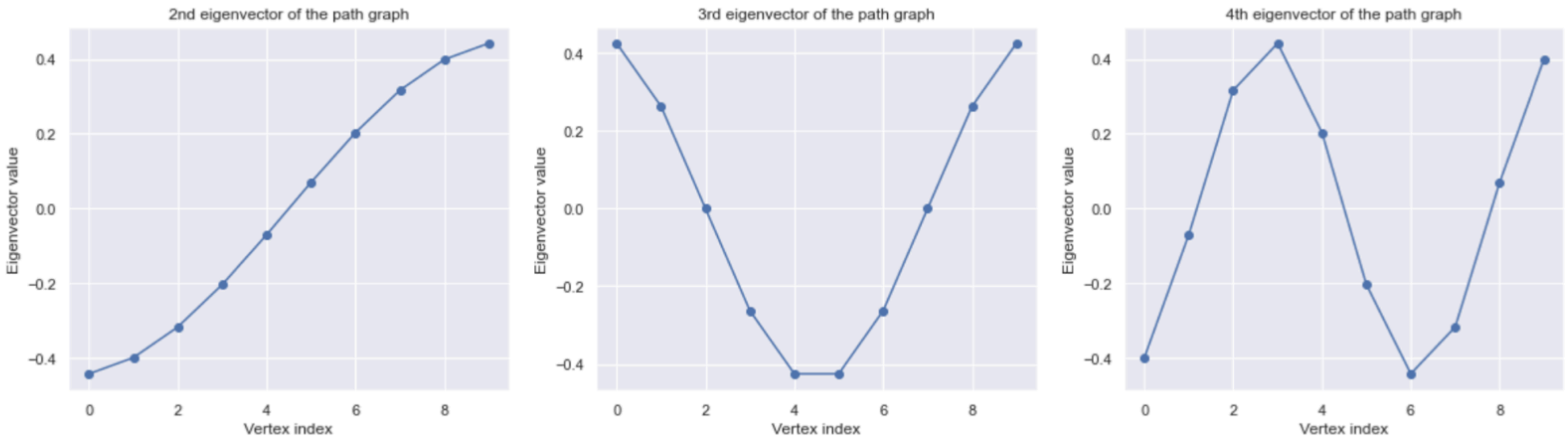
- From the generalization of the Rayleigh quotient and the global smoothness on the graph:
  - Eigenvectors form an orthonormal basis that goes from the most smooth to the least-smooth on the graph
  - Eigenvalues indicate how smooth eigenvectors are



Example: The path graph

- An example of 10 nodes:

- The corresponding eigenvectors:



Summary of the basic properties of the Laplacian matrix

- Consists of real, and non-negative eigenvalues
- It is positive semidefinite
- Some eigenvalues reveal information related to the connectivity of the graph
- Eigenvectors can be seen as functions on the graph with different levels of smoothness

Outline

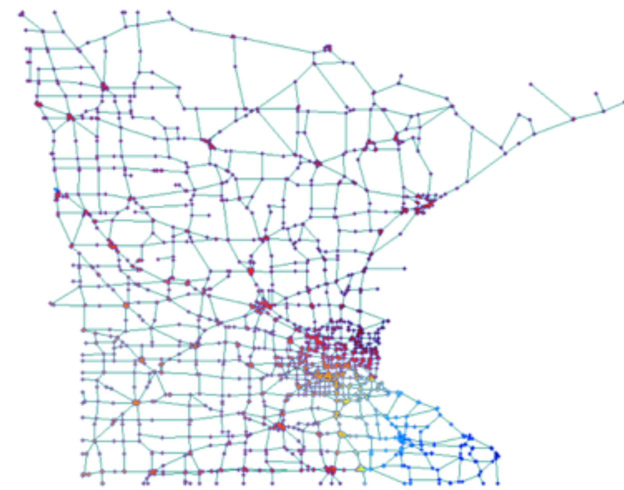
- Graph Laplacian operator

- Eigendecomposition of the graph Laplacian
  - What do the eigenvalues reveal about the graph?
  - What are the basic properties of the eigenvectors?

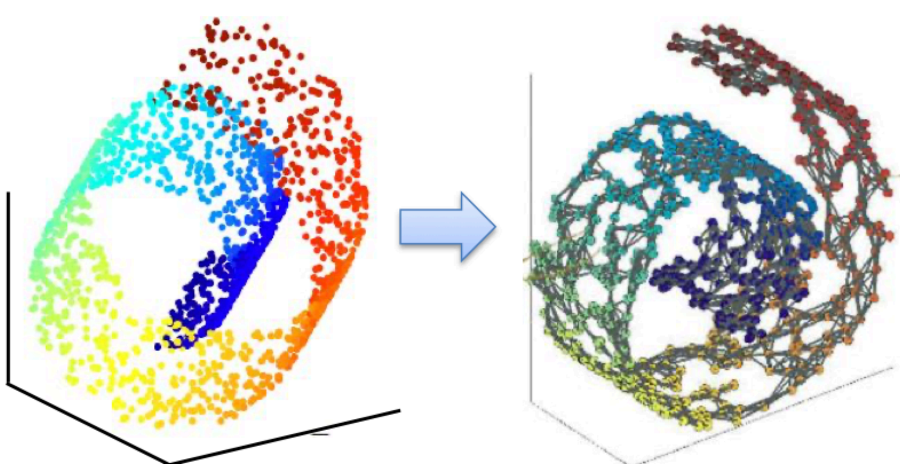
- **Applications**
  - Spectral embeddings
  - Spectral clustering
  - PageRank

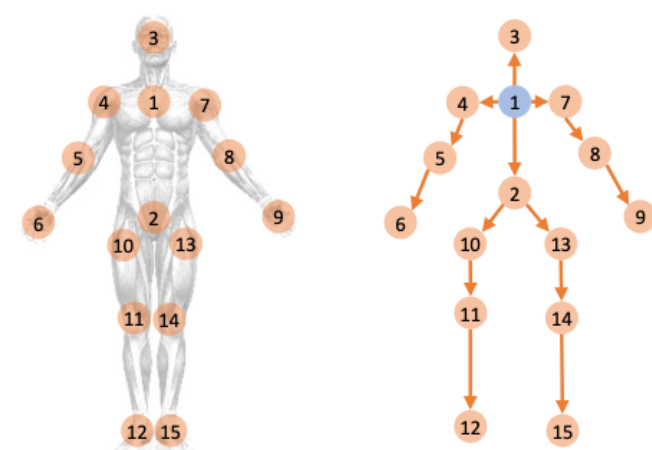
Spectral graph theory: A tool for analysing geometry

- Efficient data processing requires preserving underlying geometry
  - Often given in a network form …



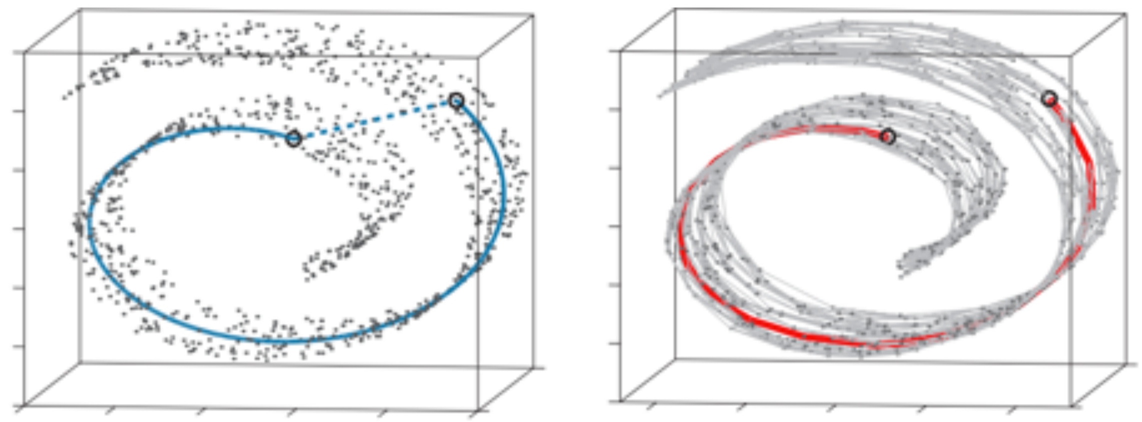
  - … or constructed from the data





Applications of spectral graph theory

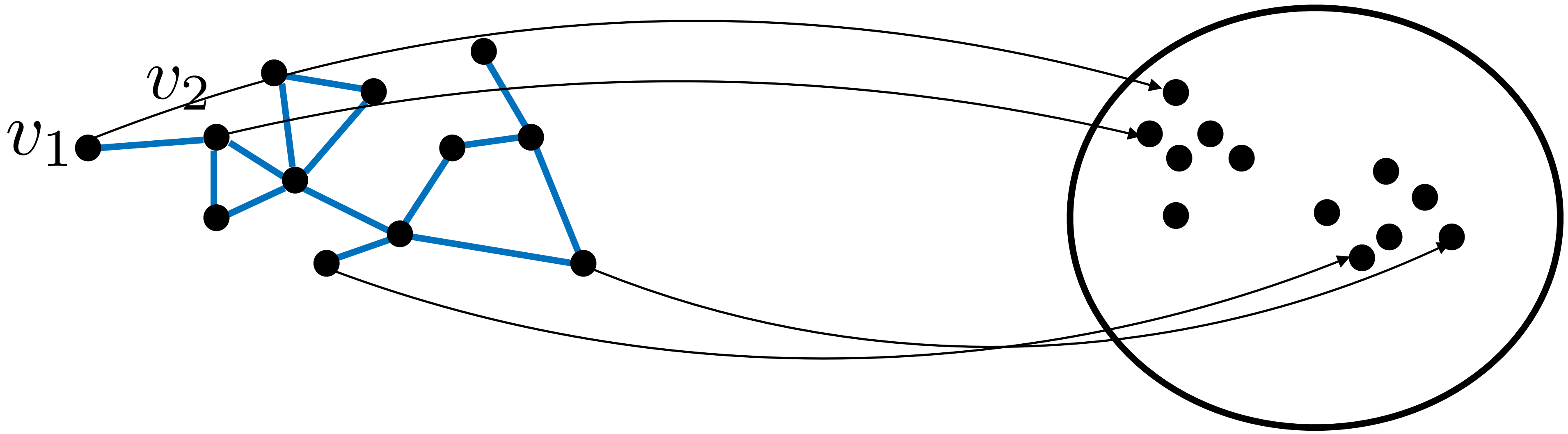
- How can we exploit the spectrum of the graph to design algorithms that capture the underlying geometry?



- Some of the well-known applications:
  - Spectral embeddings
  - Spectral clustering
  - Graph neural networks (more in the following lectures)
  - And many more…

Node embedding - reminder

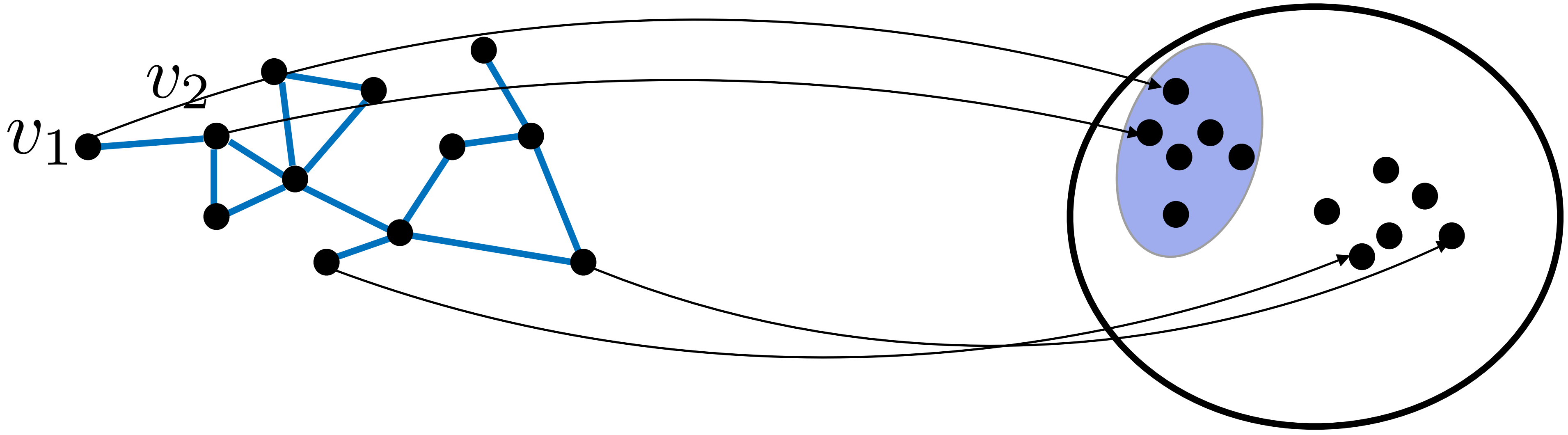
- Represent each node of the graph by a vector of low dimensions
  - Similarity in the embedding space takes intro account the complex graph structure



- An important step for further learning tasks (e.g., classification, clustering)
  - Discover relevant features
  - Data visualization and exploratory data analysis
  - Dimensionality reduction

Node embedding - reminder

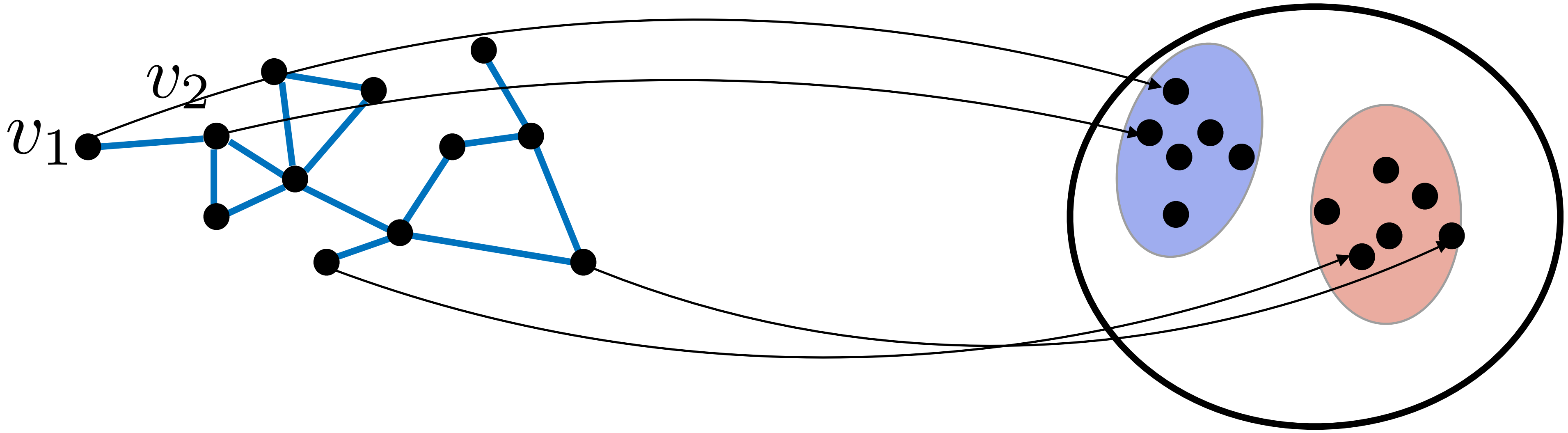
- Represent each node of the graph by a vector of low dimensions
  - Similarity in the embedding space takes intro account the complex graph structure



- An important step for further learning tasks (e.g., classification, clustering)
  - Discover relevant features
  - Data visualization and exploratory data analysis
  - Dimensionality reduction

Node embedding - reminder

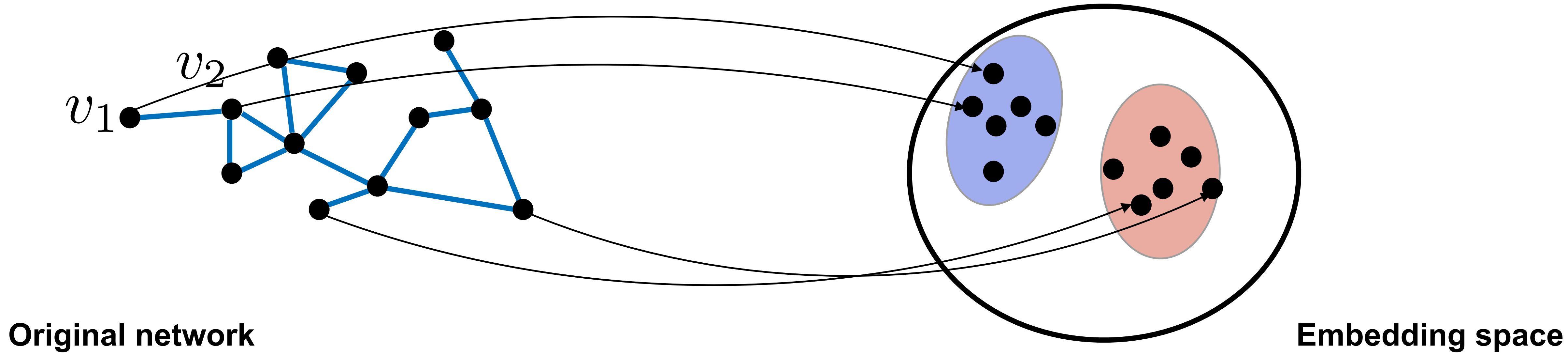
- Represent each node of the graph by a vector of low dimensions
  - Similarity in the embedding space takes intro account the complex graph structure



- An important step for further learning tasks (e.g., classification, clustering)
  - Discover relevant features
  - Data visualization and exploratory data analysis
  - Dimensionality reduction

Node embedding - reminder

- Represent each node of the graph by a vector of low dimensions
  - Similarity in the embedding space takes intro account the complex graph structure



- An important step for further learning tasks (e.g., classification, clustering)
  - Discover relevant features
  - Data visualization and exploratory data analysis
  - Dimensionality reduction

A spectral approach to node embedding

- Compute embeddings that minimize the expected square distance between nodes that are connected

Centered embeddings

Uncorrelated embedding coordinates

$$\min_{Y \in \mathbb{R}^{N \times K}: Y^T 1 = 0; Y^T Y = I_K} \sum_{(i,j) \in \mathcal{E}} W_{ij} \|Y_i - Y_j\|^2$$

$\Downarrow$ **Graph smoothness**

$$\min_{Y \in \mathbb{R}^{N \times K}: Y^T 1 = 0; Y^T Y = I_K} \operatorname{tr}(Y^T L Y)$$

$\Downarrow$ **Lagrangian**

$$\min_{Y \in \mathbb{R}^{N \times K}; Y^T 1 = 0} \operatorname{tr}(Y^T L Y - (Y^T Y - I_K)\Gamma)$$

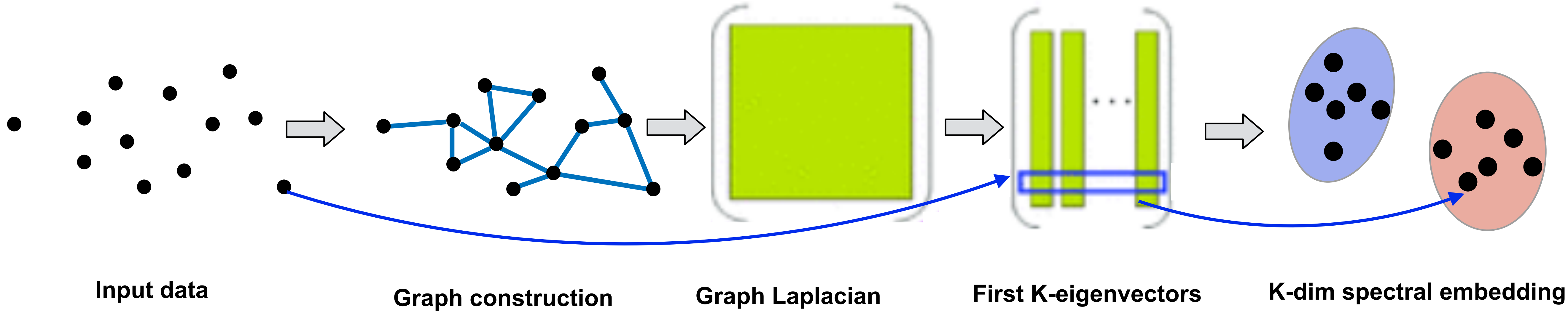
$\Downarrow$ **Gradient**

$$LY = Y\Gamma \quad \Rightarrow \quad u_i \rightarrow (\chi_2(i), ..., \chi_{K+1}(i))$$

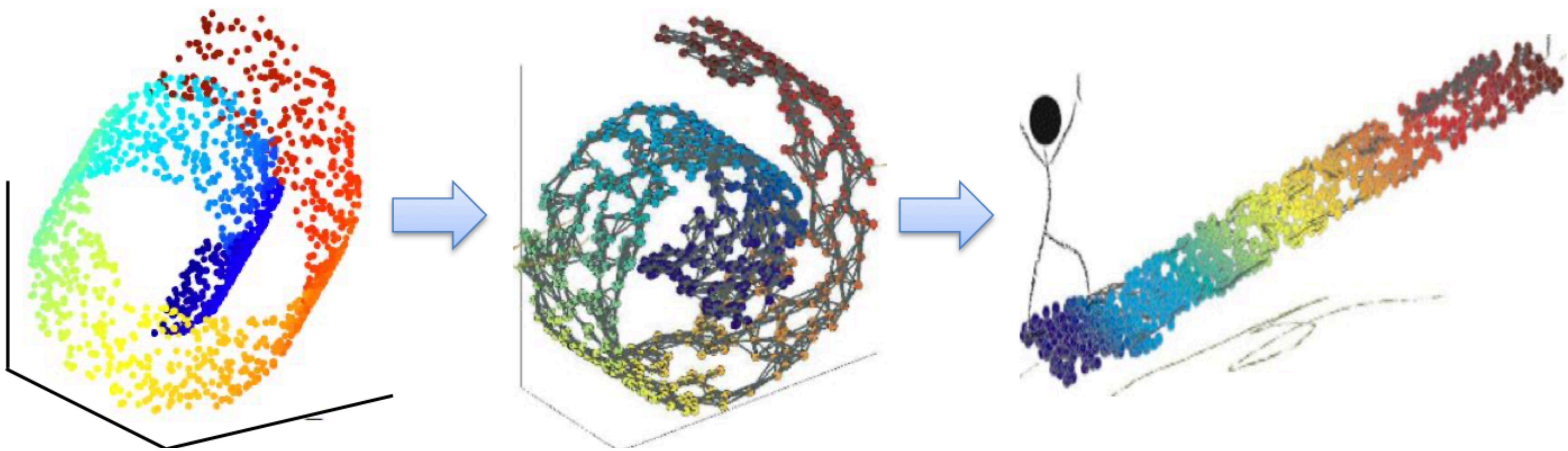
Laplacian Eigenmaps: K first non-trivial eigenvectors of the Laplacian!

More details in [4]

Spectral embedding in a nutshell



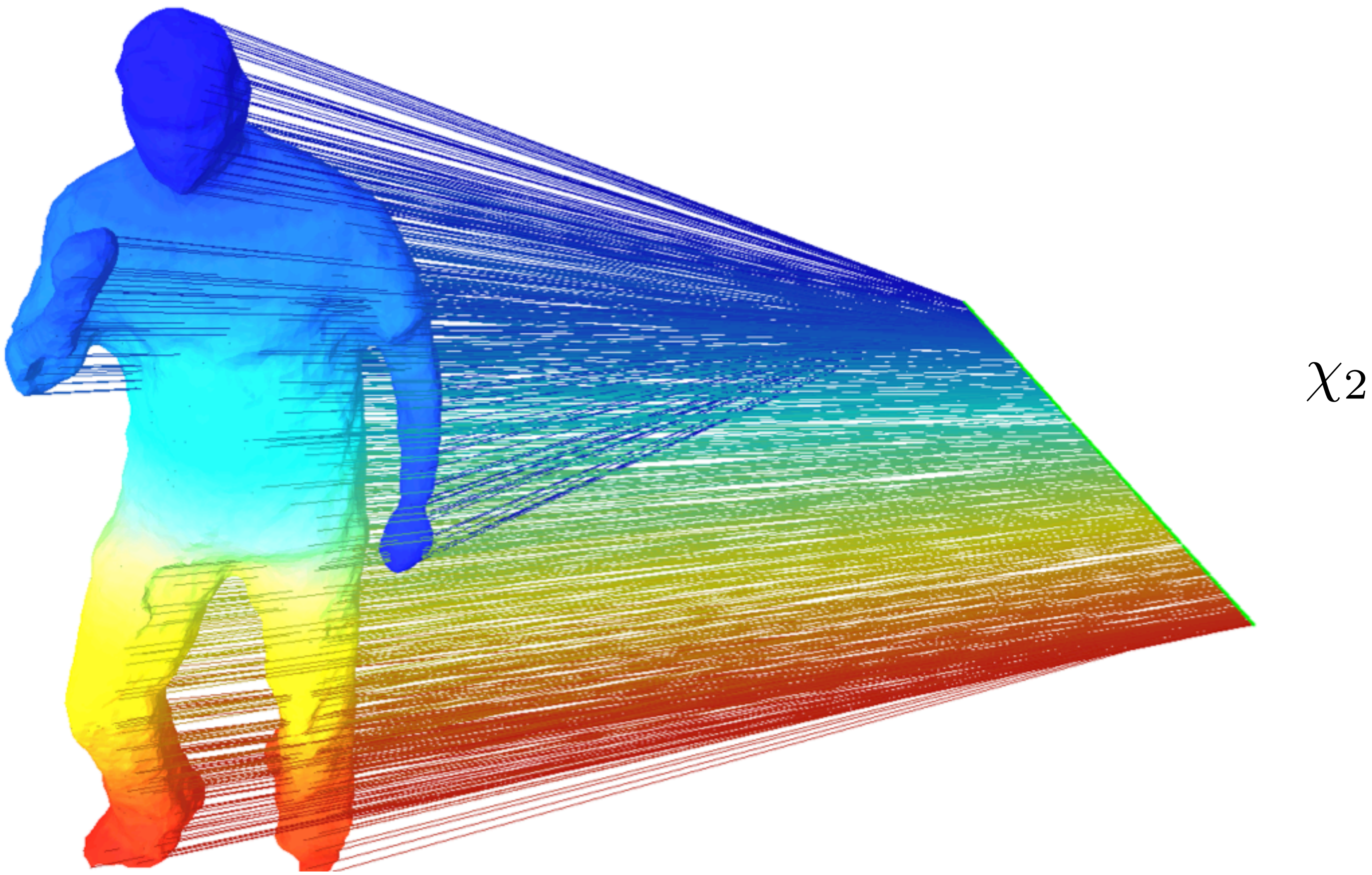
Example: Swiss roll



Swiss roll data | Graph construction | Spectral embedding

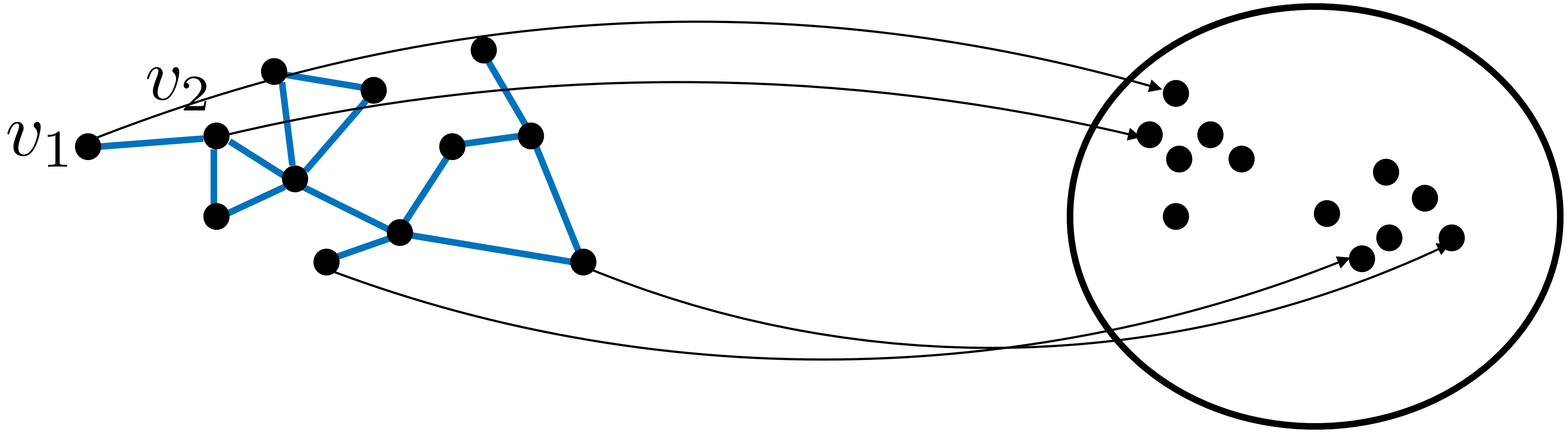
Example: 3D geometries

- Mapping a mesh/graph on the Fiedler vector of the normalized Laplacian



Clustering

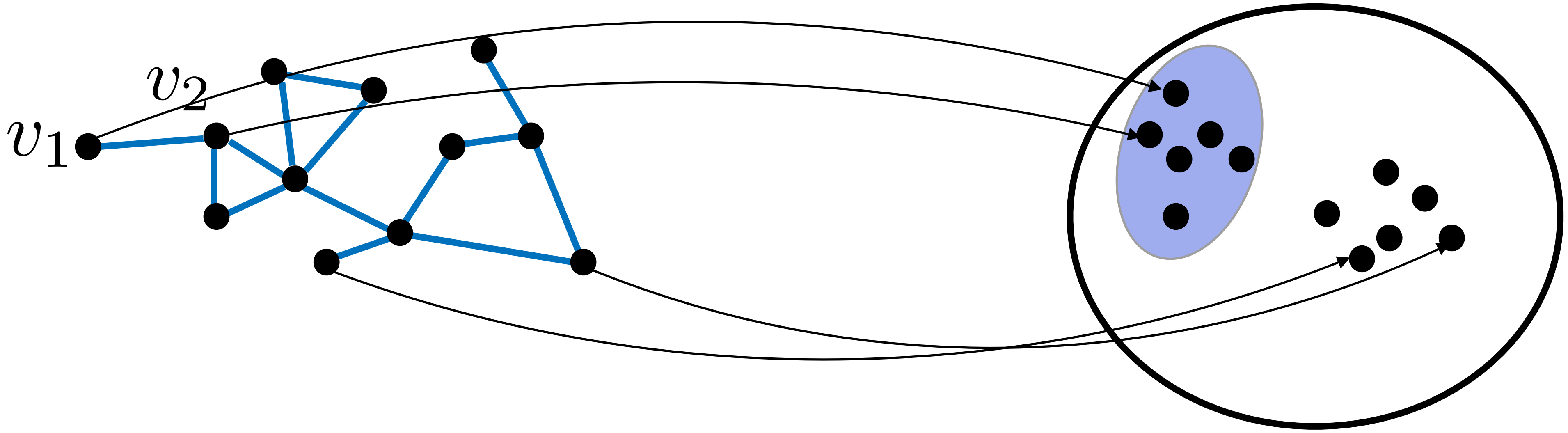
- Objective: Partition nodes of the graph into clusters
  - Points in the same cluster are similar to each other



- Requirement: Appropriate distance measure between nodes
  - Close relation to node embeddings

Clustering

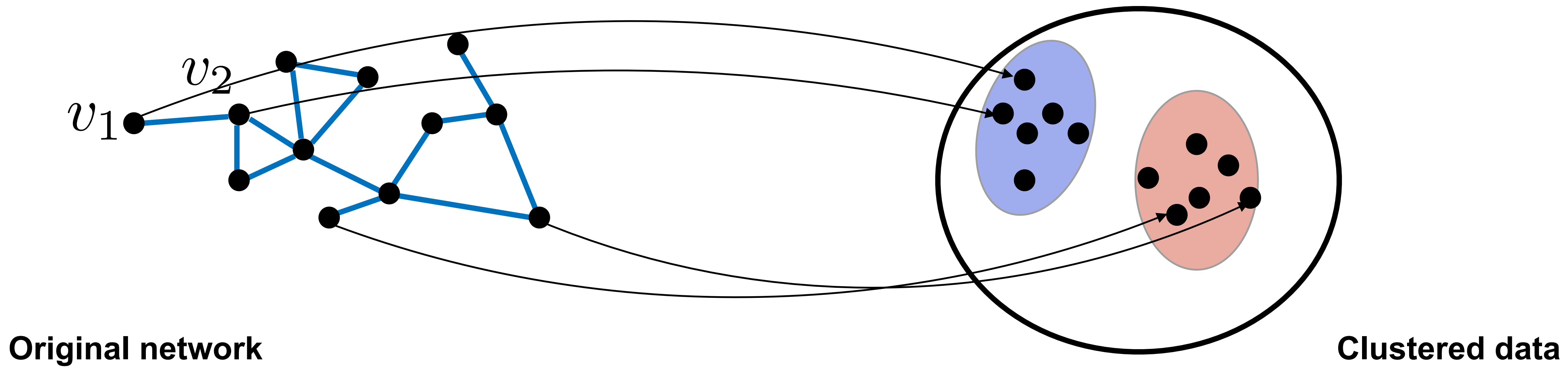
- Objective: Partition nodes of the graph into clusters
  - Points in the same cluster are similar to each other



- Requirement: Appropriate distance measure between nodes
  - Close relation to node embeddings

Clustering

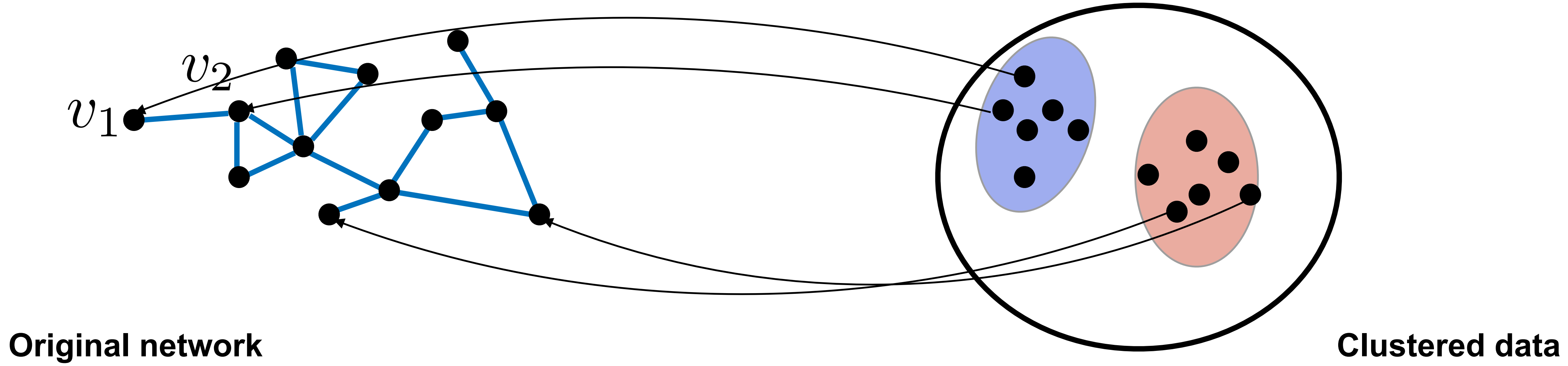
- Objective: Partition nodes of the graph into clusters
  - Points in the same cluster are similar to each other



- Requirement: Appropriate distance measure between nodes
  - Close relation to node embeddings

Clustering

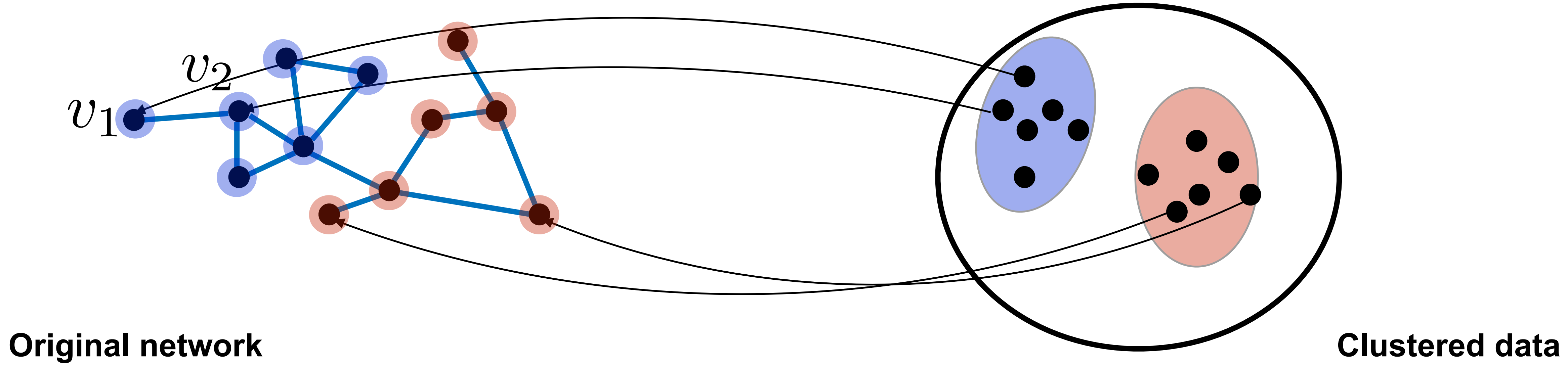
- Objective: Partition nodes of the graph into clusters
  - Points in the same cluster are similar to each other



- Requirement: Appropriate distance measure between nodes
  - Close relation to node embeddings

Clustering

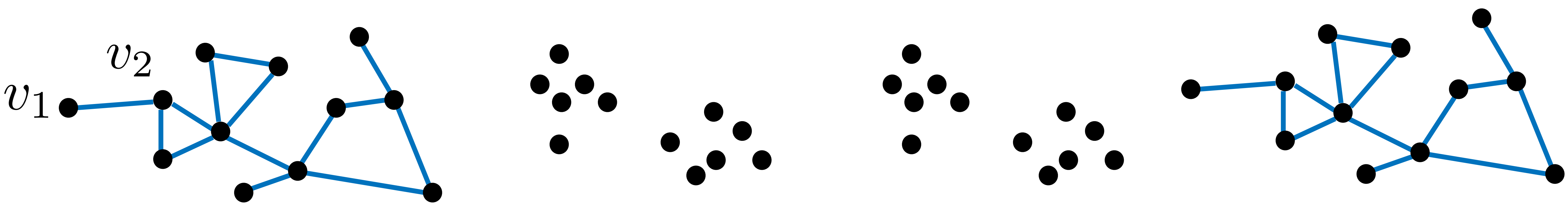
- Objective: Partition nodes of the graph into clusters
  - Points in the same cluster are similar to each other



- Requirement: Appropriate distance measure between nodes
  - Close relation to node embeddings

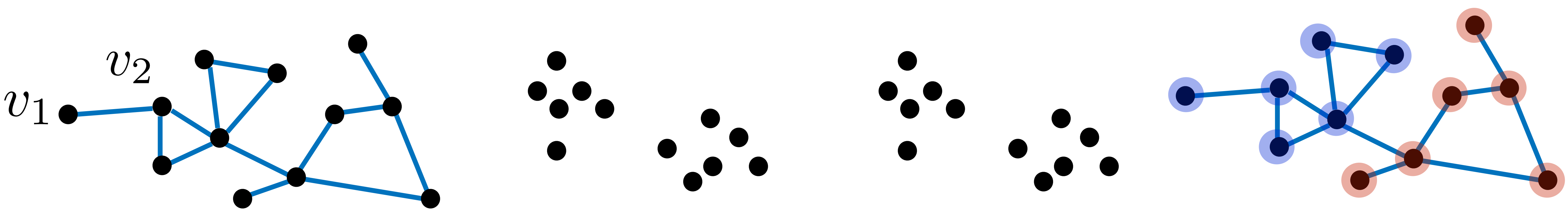
Spectral clustering in a nutshell

Graph (given or designed from the data) → **Spectral embedding** → **K-means** → **Clustered graph**



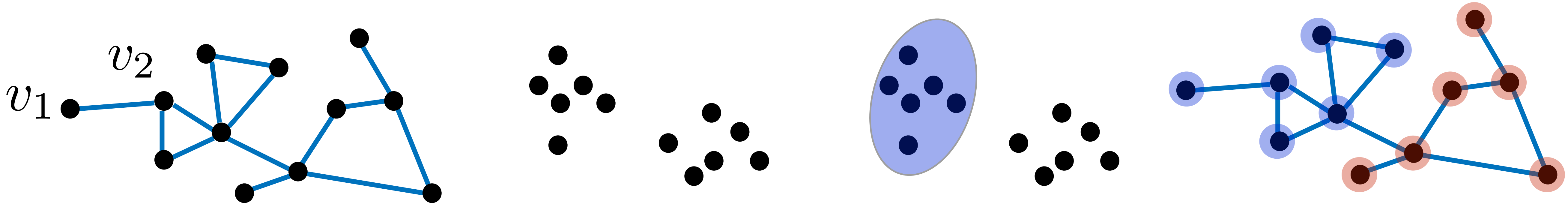
Spectral clustering in a nutshell

Graph (given or designed from the data) → **Spectral embedding** → **K-means** → **Clustered graph**

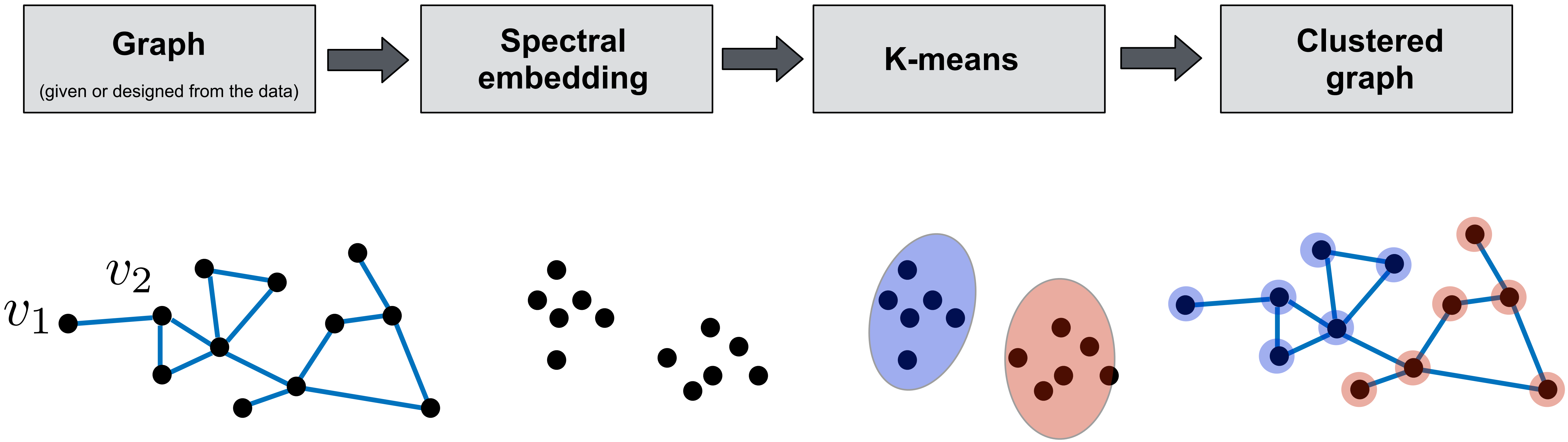


Spectral clustering in a nutshell

Graph
(given or designed from the data) → **Spectral embedding** → **K-means** → **Clustered graph**



Spectral clustering in a nutshell



Spectral clustering: How to define spectral embeddings?

Spectral embedding

- Define a representative connectivity matrix:
  - Combinatorial graph Laplacian $L = D - W$
  - Normalised graph Laplacian $L_{sym} = I - D^{-1/2} W D^{-1/2}$
  - Random walk Laplacian $L_{rw} = I - D^{-1} W$

- Compute the eigenvectors associated to the K smallest eigenvalues of that matrix

$$\widetilde{\chi} = [\chi_1, \chi_2, ..., \chi_K]$$

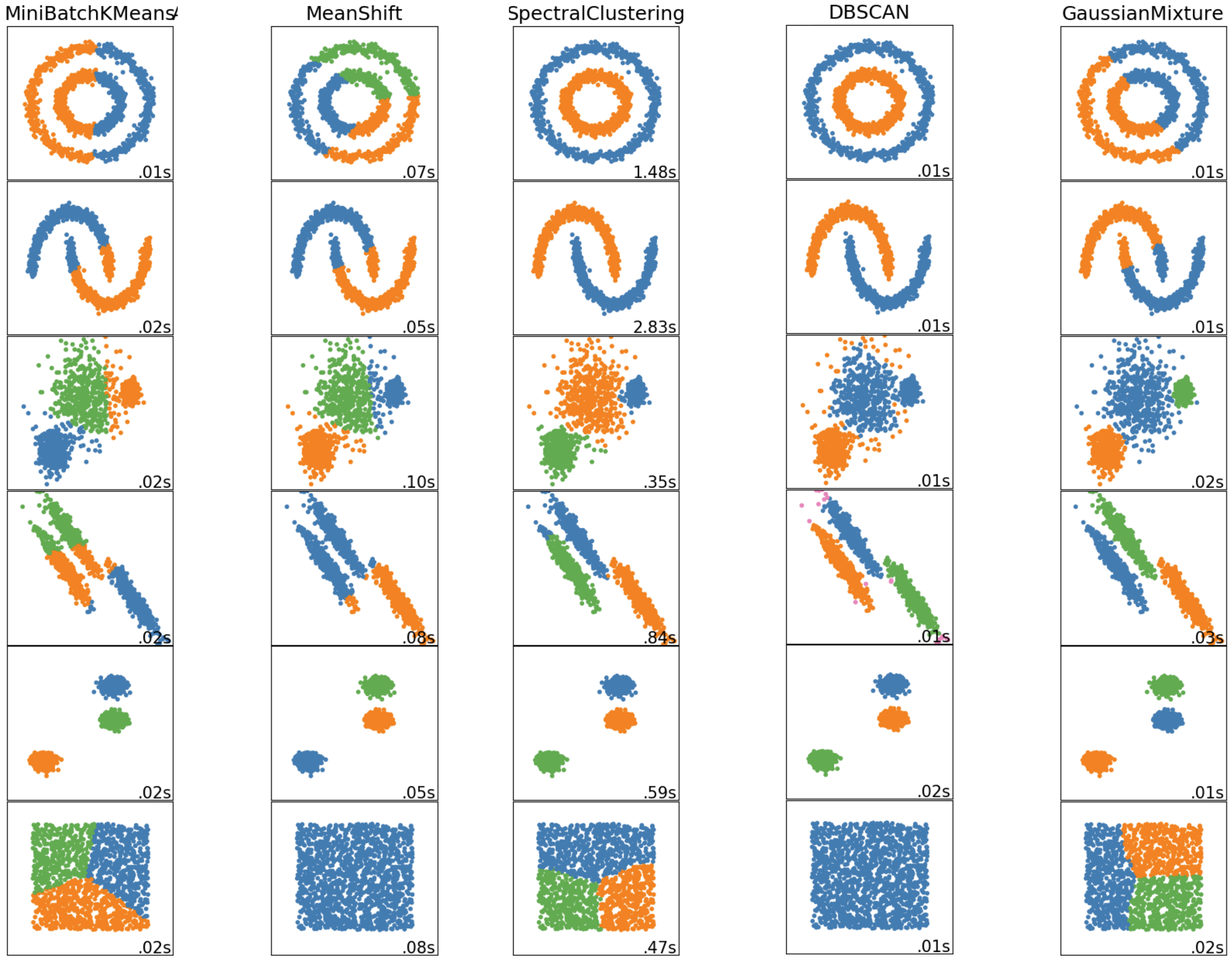
- Embed node i as follows:

i^{th} row of the embedding matrix $\widetilde{\chi}$

$$y_i = \frac{\widetilde{\chi}(i,:)^T}{q(\|\widetilde{\chi}(i,:)\|_2)}$$

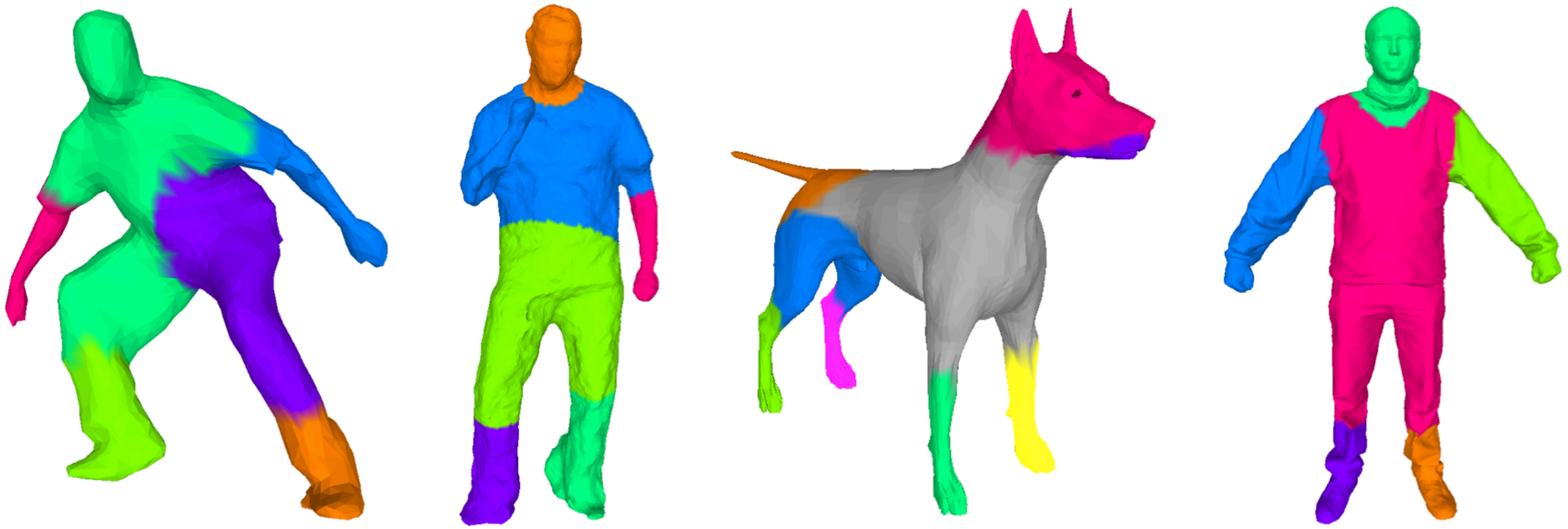
Normalization function

Illustrative example: Toy datasets in 2D



[Example from https://scikit-learn.org/stable/auto_examples/cluster/plot_cluster_comparison.html]

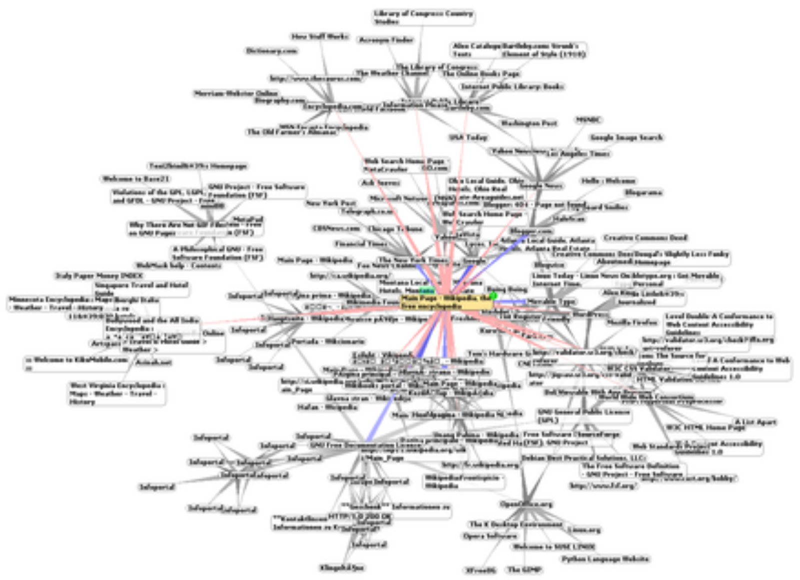
Shape segmentation



Spectral clustering on shapes lead to semantically meaningful clusters

Google PageRank

- One of the most popular algorithms for Internet search
  - Measures the most important web pages on the Internet that correspond to the user's search query



- PageRank consists of three steps:
  - User inputs query
  - Engine finds all relevant pages containing the query
  - Pages are ranked

- Approach:
  - Model the Web as a directed graph - 'web graph'
  - Nodes correspond to pages, and edges reflect directional links/recommendations
  - Rank pages using the web graph link structure: a page is important if it has many 'important' links

Google PageRank algorithm

- For a directed graph $G = (\mathcal{V}, \mathcal{E})$, the PageRank vector that we are looking for is defined as:

$$p = (1-\alpha)W_{rw}p + \frac{\alpha}{|\mathcal{V}|}\mathbf{1},\ p\mathbf{1}^T = \mathbf{1},\ W_{rw}(j,i) = \frac{1}{D_{ii}}$$

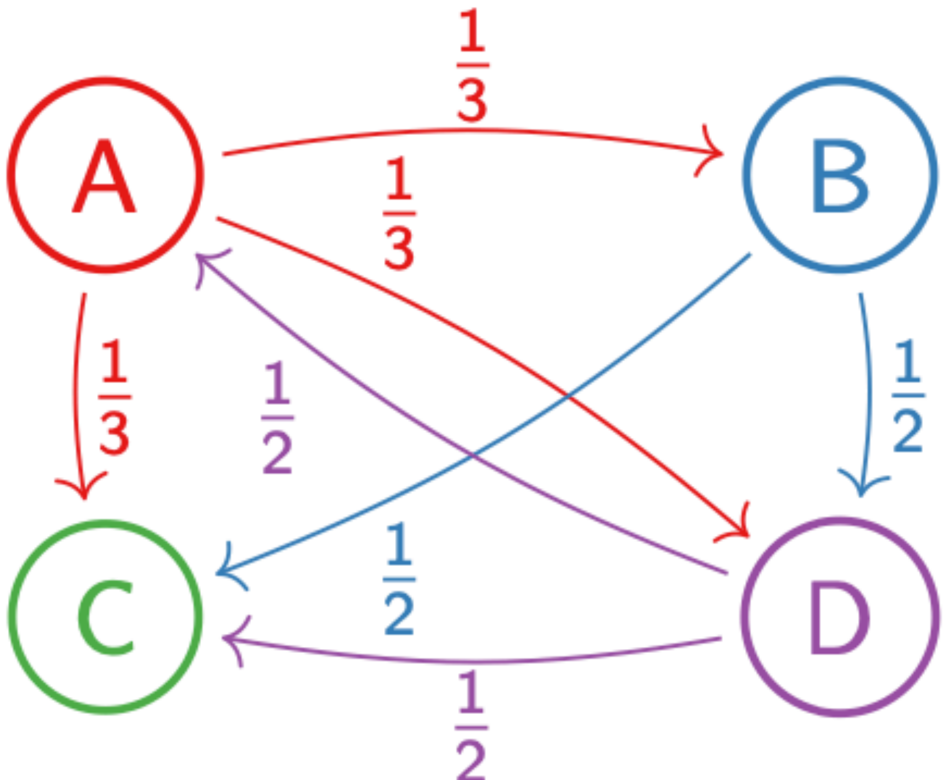
Outdegree

- The solution p is the eigenvector corresponding to the eigenvalue 1 of the matrix:

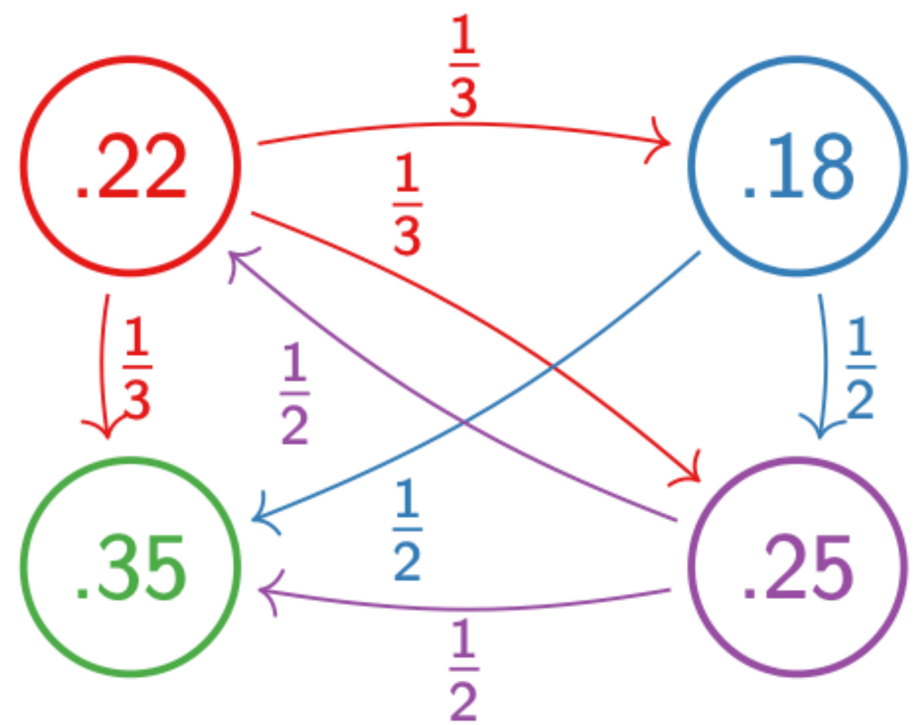
$$(1-\alpha)W_{rw} + \frac{\alpha}{|\mathcal{V}|}\mathbf{1}^T\mathbf{1}$$

  - Proof based on the Perron-Frobenious theorem for non-negative matrices [5]
- A stationary distribution of a randomized process “random surfer”
  - With probability $1-\alpha$ moves to a neighboring node
  - With probability α moves to a random node of the graph
  - The probability that a surfer visits a node is its PageRank

Example of PageRank



Initial graph



PageRank scores

$$W_{rw} = \begin{bmatrix} 0 & 0 & 1/4 & 1/2 \\ 1/3 & 0 & 1/4 & 0 \\ 1/3 & 1/2 & 1/4 & 1/2 \\ 1/3 & 1/2 & 1/4 & 0 \end{bmatrix} \Rightarrow (1-0.15)W_{rw} + \frac{0.15}{4} \Rightarrow p = \begin{bmatrix} 0.219 \\ 0.175 \\ 0.356 \\ 0.249 \end{bmatrix}$$

$$p = (1-\alpha)W_{rw}p + \frac{\alpha}{|\mathcal{V}|}\mathbf{1},\ p\mathbf{1}^T = \mathbf{1},\ W_{rw}(j,i) = \frac{1}{D_{ii}}$$

Summary

- Spectral graph theory:
  - A useful mathematical framework that reveals properties of the graph or network

- Spectrum tells us a lot about:
  - connectivity, bottlenecks, diameter…

- Eigenvectors are useful for defining a notion of smoothness on the graph
  - First eigenvectors of the graph Laplacian are smooth functions

- Many applications in different areas
  - Established frameworks: spectral clustering, spectral embeddings, PageRank
  - Emerging research topics: graph signal processing, graph neural networks (more in the following lectures…)

References

[1] Spectral and Algebraic Graph Theory, Daniel A. Spielman

[2] Spectral graph theory, Fan Chung

[3] A tutorial on spectral clustering, Ulrike von Luxburg *Statistics and Computing*, 2007

[4] Laplacian Eigenmaps for Dimensionality Reduction and Data Representation, Belkin et al., *Neural Comp.*, 2003

[5] How Google Ranks Web Pages, Brian White

[6] The Emerging Field of Signal Processing on Graphs, Shuman et al, 2013