A Short Tutorial on Graph Laplacians, Laplacian Embedding, and Spectral Clustering

Radu Horaud
INRIA Grenoble Rhone-Alpes, France
Radu.Horaud@inrialpes.fr
http://perception.inrialpes.fr/

Introduction

- The *spectral graph theory* studies the properties of graphs via the eigenvalues and eigenvectors of their associated graph matrices: the *adjacency matrix* and the *graph Laplacian* and its variants.
- Both matrices have been extremely well studied from an algebraic point of view.
- The Laplacian allows a natural link between discrete representations, such as graphs, and continuous representations, such as vector spaces and manifolds.
- The most important application of the Laplacian is *spectral clustering* that corresponds to a computationally tractable solution to the *graph partitionning problem*.
- Another application is *spectral matching* that solves for *graph matching*.

Applications of spectral graph theory

- *Spectral partitioning*: automatic circuit placement for VLSI (Alpert et al 1999), image segmentation (Shi & Malik 2000),
- *Text mining and web applications*: document classification based on semantic association of words (Lafon & Lee 2006), collaborative recommendation (Fouss et al. 2007), text categorization based on reader similarity (Kamvar et al. 2003).
- *Manifold analysis*: Manifold embedding, manifold learning, mesh segmentation, etc.

Basic graph notations and definitions

We consider *simple graphs* (no multiple edges or loops), $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$:

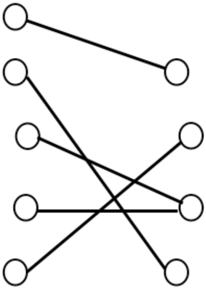
- $\mathcal{V}(\mathcal{G}) = \{v_1, \dots, v_n\}$ is called the *vertex set* with $n = |\mathcal{V}|$;
- $\mathcal{E}(\mathcal{G}) = \{e_{ij}\}$ is called the *edge set* with $m = |\mathcal{E}|$;
- An edge e_{ij} connects vertices v_i and v_j if they are adjacent or neighbors. One possible notation for adjacency is $v_i \sim v_j$;
- The number of neighbors of a node v is called the *degree* of v and is denoted by $d(v)$, $d(v_i) = \sum_{v_i \sim v_j} e_{ij}$. If all the nodes of a graph have the same degree, the graph is *regular*; The nodes of an *Eulerian* graph have even degree.
- A graph is *complete* if there is an edge between every pair of vertices.

Subgraph of a graph

- $\mathcal{H}$ is a *subgraph* of $\mathcal{G}$ if $\mathcal{V}(\mathcal{H}) \subseteq \mathcal{V}(\mathcal{G})$ and $\mathcal{E}(\mathcal{H}) \subseteq \mathcal{E}(\mathcal{G})$;
- a subgraph $\mathcal{H}$ is an *induced subgraph* of $\mathcal{G}$ if two vertices of $\mathcal{V}(\mathcal{H})$ are adjacent if and only if they are adjacent in $\mathcal{G}$.
- A *clique* is a complete subgraph of a graph.
- A *path* of k vertices is a sequence of k distinct vertices such that consecutive vertices are adjacent.
- A *cycle* is a connected subgraph where every vertex has exactly two neighbors.
- A graph containing no cycles is a *forest*. A connected forest is a *tree*.

A k-partite graph

- A graph is called *k-partite* if its set of vertices admits a partition into k classes such that the vertices of the same class are not adjacent.
- An example of a *bipartite* graph.

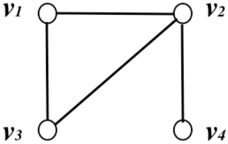


The adjacency matrix of a graph

- For a graph with n vertices, the entries of the $n \times n$ adjacency matrix are defined by:

$$\mathbf{A} := \begin{cases} A_{ij} = 1 & \text{if there is an edge } e_{ij} \\ A_{ij} = 0 & \text{if there is no edge} \\ A_{ii} = 0 & \end{cases}$$

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



Eigenvalues and eigenvectors

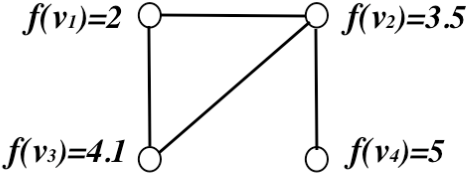
- $\mathbf{A}$ is a real-symmetric matrix: it has n real eigenvalues and its n real eigenvectors form an orthonormal basis.
- Let $\{\lambda_1, \dots, \lambda_i, \dots, \lambda_r\}$ be the set of *distinct* eigenvalues.
- The eigenspace S_i contains the eigenvectors associated with λ_i:

$$S_i = \{\boldsymbol{x} \in \mathbb{R}^n | \mathbf{A}\boldsymbol{x} = \lambda_i \boldsymbol{x}\}$$

- For real-symmetric matrices, the algebraic multiplicity is equal to the geometric multiplicity, for all the eigenvalues.
- The dimension of S_i (geometric multiplicity) is equal to the multiplicity of λ_i.
- If $\lambda_i \neq \lambda_j$ then S_i and S_j are mutually orthogonal.

Real-valued functions on graphs

- We consider real-valued functions on the set of the graph's vertices, $\boldsymbol{f} : \mathcal{V} \longrightarrow \mathbb{R}$. Such a function assigns a real number to each graph node.
- $\boldsymbol{f}$ is a vector indexed by the graph's vertices, hence $\boldsymbol{f} \in \mathbb{R}^n$.
- Notation: $\boldsymbol{f} = (f(v_1), \dots, f(v_n)) = (f(1), \dots, f(n))$.
- The eigenvectors of the adjacency matrix, $\mathbf{A}\boldsymbol{x} = \lambda \boldsymbol{x}$, can be viewed as *eigenfunctions*.



Matrix $\mathbf{A}$ as an operator and quadratic form

- The adjacency matrix can be viewed as an operator

$$\boldsymbol{g} = \mathbf{A}\boldsymbol{f}; g(i) = \sum_{i \sim j} f(j)$$

- It can also be viewed as a quadratic form:

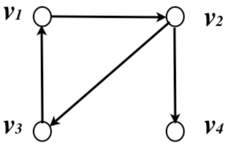
$$\boldsymbol{f}^\top \mathbf{A}\boldsymbol{f} = \sum_{e_{ij}} f(i)f(j)$$

The incidence matrix of a graph

- Let each edge in the graph have an arbitrary but fixed orientation;
- The incidence matrix of a graph is a $|\mathcal{E}| \times |\mathcal{V}|$ $(m \times n)$ matrix defined as follows:

$$\bigtriangledown := \begin{cases} \bigtriangledown_{ev} = -1 & \text{if } v \text{ is the initial vertex of edge } e \\ \bigtriangledown_{ev} = 1 & \text{if } v \text{ is the terminal vertex of edge } e \\ \bigtriangledown_{ev} = 0 & \text{if } v \text{ is not in } e \end{cases}$$

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



The incidence matrix: A discrete differential operator

- The mapping $\boldsymbol{f} \longrightarrow \bigtriangledown \boldsymbol{f}$ is known as the *co-boundary mapping* of the graph.
- $(\bigtriangledown \boldsymbol{f})(e_{ij}) = f(v_j) - f(v_i)$

$$\begin{pmatrix} f(2) - f(1) \\ f(1) - f(3) \\ f(3) - f(2) \\ f(4) - f(2) \end{pmatrix} = \begin{bmatrix} -1 & 1 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & -1 & 0 & +1 \end{bmatrix} \begin{pmatrix} f(1) \\ f(2) \\ f(3) \\ f(4) \end{pmatrix}$$

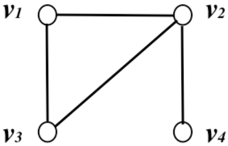
The Laplacian matrix of a graph

- $\mathbf{L} = \nabla^\top \nabla$
- $(\mathbf{L}\boldsymbol{f})(v_i) = \sum_{v_j \sim v_i} (f(v_i) - f(v_j))$
- Connection between the Laplacian and the adjacency matrices:

$$\mathbf{L} = \mathbf{D} - \mathbf{A}$$

- The degree matrix: $\mathbf{D} := D_{ii} = d(v_i)$.

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



The Laplacian matrix of an undirected weighted graph

- We consider *undirected weighted graphs*: Each edge e_{ij} is weighted by $w_{ij} > 0$.
- The Laplacian as an operator:

$$(\mathbf{L}\boldsymbol{f})(v_i) = \sum_{v_j \sim v_i} w_{ij}(f(v_i) - f(v_j))$$

- As a quadratic form:

$$\boldsymbol{f}^\top \mathbf{L}\boldsymbol{f} = \frac{1}{2}\sum_{e_{ij}} w_{ij}(f(v_i) - f(v_j))^2$$

- $\mathbf{L}$ is symmetric and positive semi-definite.
- $\mathbf{L}$ has n non-negative, real-valued eigenvalues: $0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$.

The Laplacian of a 3D discrete surface (mesh)

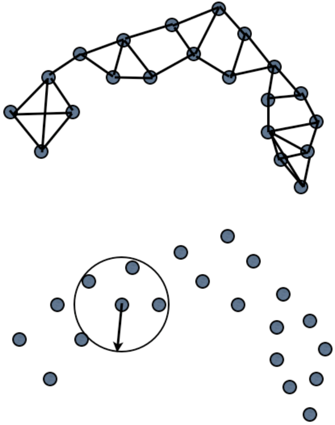
- A graph vertex v_i is associated with a 3D point $\boldsymbol{v}_i$.
- The weight of an edge e_{ij} is defined by the Gaussian kernel:

$$w_{ij} = \exp\left(-\|\boldsymbol{v}_i - \boldsymbol{v}_j\|^2/\sigma^2\right)$$

- $0 \leq w_{\min} \leq w_{ij} \leq w_{\max} \leq 1$
- Hence, the geometric structure of the mesh is encoded in the weights.
- Other weighting functions were proposed in the literature.

The Laplacian of a cloud of points

- 3-nearest neighbor graph
- ε-radius graph
- KNN may guarantee that the graph is connected (depends on the implementation)
- ε-radius does not guarantee that the graph has one connected component



The Laplacian of a graph with one connected component

- $\mathbf{L}\boldsymbol{u} = \lambda\boldsymbol{u}$.
- $\mathbf{L}\mathbf{1}_n = \mathbf{0}$, $\lambda_1 = 0$ is the smallest eigenvalue.
- The *one* vector: $\mathbf{1}_n = (1 \dots 1)^\top$.
- $0 = \boldsymbol{u}^\top\mathbf{L}\boldsymbol{u} = \sum_{i,j=1}^n w_{ij}(u(i) - u(j))^2$.
- If any two vertices are connected by a path, then $\boldsymbol{u} = (u(1), \dots, u(n))$ needs to be constant at all vertices such that the quadratic form vanishes. Therefore, a graph with one connected component has the constant vector $\boldsymbol{u}_1 = \mathbf{1}_n$ as the only eigenvector with eigenvalue 0.

A graph with $k > 1$ connected components

- Each connected component has an associated Laplacian. Therefore, we can write matrix $\mathbf{L}$ as a *block diagonal matrix*:

$$\mathbf{L} = \begin{bmatrix} \mathbf{L}_1 & & \\ & \ddots & \\ & & \mathbf{L}_k \end{bmatrix}$$

- The spectrum of $\mathbf{L}$ is given by the union of the spectra of $\mathbf{L}_i$.
- Each block corresponds to a connected component, hence each matrix $\mathbf{L}_i$ has an eigenvalue 0 with multiplicity 1.
- The spectrum of $\mathbf{L}$ is given by the union of the spectra of $\mathbf{L}_i$.
- The eigenvalue $\lambda_1 = 0$ has multiplicity k.

The eigenspace of $\lambda_1 = 0$ with multiplicity k

- The eigenspace corresponding to $\lambda_1 = \ldots = \lambda_k = 0$ is spanned by the k mutually orthogonal vectors:

$$\begin{gathered} \boldsymbol{u}_1 = \mathbf{1}_{L_1} \\ \ldots \\ \boldsymbol{u}_k = \mathbf{1}_{L_k} \end{gathered}$$

- with $\mathbf{1}_{L_i} = (0000111110000)^\top \in \mathbb{R}^n$
- These vectors are the *indicator vectors* of the graph's connected components.
- Notice that $\mathbf{1}_{L_1} + \ldots + \mathbf{1}_{L_k} = \mathbf{1}_n$

The Fiedler vector of the graph Laplacian

- The first non-null eigenvalue λ_{k+1} is called the Fiedler value.
- The corresponding eigenvector $\boldsymbol{u}_{k+1}$ is called the Fiedler vector.
- The multiplicity of the Fiedler eigenvalue is always equal to 1.
- The Fiedler value is the *algebraic connectivity of a graph*, the further from 0, the more connected.
- The Fidler vector has been extensively used for *spectral bi-partioning*
- Theoretical results are summarized in Spielman & Teng 2007: `http://cs-www.cs.yale.edu/homes/spielman/`

Eigenvectors of the Laplacian of connected graphs

- $\boldsymbol{u}_1 = \mathbf{1}_n, \mathbf{L}\mathbf{1}_n = \mathbf{0}$.
- $\boldsymbol{u}_2$ is the *the Fiedler vector* with multiplicity 1.
- The eigenvectors form an orthonormal basis: $\boldsymbol{u}_i^\top \boldsymbol{u}_j = \delta_{ij}$.
- For any eigenvector $\boldsymbol{u}_i = (\boldsymbol{u}_i(v_1) \dots \boldsymbol{u}_i(v_n))^\top,\ 2 \leq i \leq n$:

$$\boldsymbol{u}_i^\top \mathbf{1}_n = 0$$

- Hence the components of $\boldsymbol{u}_i,\ 2 \leq i \leq n$ satisfy:

$$\sum_{j=1}^{n} \boldsymbol{u}_i(v_j) = 0$$

- Each component is bounded by:

$$-1 < \boldsymbol{u}_i(v_j) < 1$$

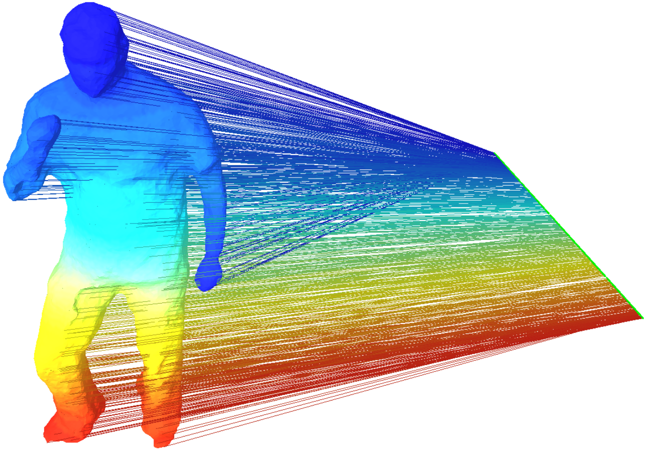
Laplacian embedding: Mapping a graph on a line

- Map a weighted graph onto a line such that connected nodes stay as close as possible, i.e., minimize $\sum_{i,j=1}^{n} w_{ij}(f(v_i) - f(v_j))^2$, or:

$$\arg\min_{\boldsymbol{f}} \boldsymbol{f}^\top \mathbf{L} \boldsymbol{f} \text{ with: } \boldsymbol{f}^\top \boldsymbol{f} = 1 \text{ and } \boldsymbol{f}^\top \mathbf{1} = 0$$

- The solution is the eigenvector associated with the smallest nonzero eigenvalue of the eigenvalue problem: $\mathbf{L}\boldsymbol{f} = \lambda \boldsymbol{f}$, namely the Fiedler vector $\boldsymbol{u}_2$.
- For more details on this minimization see Golub & Van Loan *Matrix Computations*, chapter 8 (The symmetric eigenvalue problem).

Example of mapping a graph on the Fiedler vector



Laplacian embedding

- Embed the graph in a k-dimensional Euclidean space. The embedding is given by the $n \times k$ matrix $\mathbf{F} = [\boldsymbol{f}_1 \boldsymbol{f}_2 \ldots \boldsymbol{f}_k]$ where the i-th row of this matrix – $\boldsymbol{f}^{(i)}$ – corresponds to the Euclidean coordinates of the i-th graph node v_i.
- We need to minimize (Belkin & Niyogi '03):

$$\arg \min_{\boldsymbol{f}_1 \ldots \boldsymbol{f}_k} \sum_{i,j=1}^{n} w_{ij} \| \boldsymbol{f}^{(i)} - \boldsymbol{f}^{(j)} \|^2 \text{ with: } \mathbf{F}^\top \mathbf{F} = \mathbf{I}.$$

- The solution is provided by the matrix of eigenvectors corresponding to the k lowest nonzero eigenvalues of the eigenvalue problem $\mathbf{L}\boldsymbol{f} = \lambda \boldsymbol{f}$.

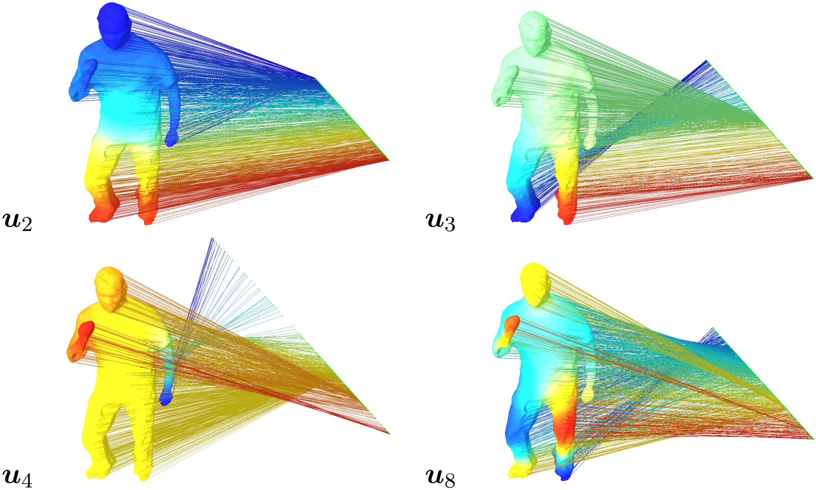
Spectral embedding using the *unnormalized* Laplacian

- Compute the eigendecomposition $\mathbf{L} = \mathbf{D} - \mathbf{A}$.
- Select the k smallest non-null eigenvalues $\lambda_2 \leq \ldots \leq \lambda_{k+1}$
- $\lambda_{k+2} - \lambda_{k+1} =$ **eigengap**.
- We obtain the $n \times k$ matrix $\mathbf{U} = [\boldsymbol{u}_2 \ldots \boldsymbol{u}_{k+1}]$:

$$\mathbf{U} = \begin{bmatrix} \boldsymbol{u}_2(v_1) & \ldots & \boldsymbol{u}_{k+1}(v_1) \\ \vdots & & \vdots \\ \boldsymbol{u}_2(v_n) & \ldots & \boldsymbol{u}_{k+1}(v_n) \end{bmatrix}$$

- $\boldsymbol{u}_i^\top \boldsymbol{u}_j = \delta_{ij}$ (orthonormal vectors), hence $\mathbf{U}^\top \mathbf{U} = \mathbf{I}_k$.
- Column i $(2 \leq i \leq k+1)$ of this matrix is a mapping on the eigenvector $\boldsymbol{u}_i$.

Examples of one-dimensional mappings



Euclidean L-embedding of the graph's vertices

- (Euclidean) $\mathbf{L}$-embedding of a graph:

$$\mathbf{X} = \mathbf{\Lambda}_k^{-\frac{1}{2}}\mathbf{U}^\top = [\boldsymbol{x}_1 \ \dots \ \boldsymbol{x}_j \ \dots \ \boldsymbol{x}_n]$$

The coordinates of a vertex v_j are:

$$\boldsymbol{x}_j = \begin{pmatrix} \frac{\boldsymbol{u}_2(v_j)}{\sqrt{\lambda_2}} \\ \vdots \\ \frac{\boldsymbol{u}_{k+1}(v_j)}{\sqrt{\lambda_{k+1}}} \end{pmatrix}$$

Justification for choosing the L-embedding

Both

- the *commute-time distance* (CTD) and
- the *principal-component analysis* of a graph (graph PCA)

are two important concepts; They allow to reason "statistically" on a graph. They are both associated with the *unnormalized* Laplacian matrix.

The commute-time distance

- The CTD is a well known quantity in Markov chains;
- It is the average number of (weighted) edges that it takes, starting at vertex v_i, to randomly reach vertex v_j for the first time and go back;
- The CTD decreases as the number of connections between the two nodes increases;
- It captures the connectivity structure of a small graph volume rather than a single path between the two vertices – such as the shortest-path geodesic distance.
- The CTD can be computed in closed form:

$$\mathsf{CTD}^2(v_i, v_j) = \mathsf{vol}(\mathcal{G})\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2$$

The graph PCA

- The mean (remember that $\sum_{j=1}^{n} \boldsymbol{u}_i(v_j) = 0$):

$$\overline{\boldsymbol{x}} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{x}_j = \boldsymbol{\Lambda}_k^{-\frac{1}{2}} \begin{pmatrix} \sum_{j=1}^{n} \boldsymbol{u}_2(v_j) \\ \vdots \\ \sum_{j=1}^{n} \boldsymbol{u}_{k+1}(v_j) \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}$$

- The covariance matrix:

$$\mathbf{S} = \frac{1}{n}\sum_{j=1}^{n} \boldsymbol{x}_j \boldsymbol{x}_j^\top = \frac{1}{n}\mathbf{X}\mathbf{X}^\top = \frac{1}{n}\boldsymbol{\Lambda}_k^{-\frac{1}{2}}\mathbf{U}^\top\mathbf{U}\boldsymbol{\Lambda}_k^{-\frac{1}{2}} = \frac{1}{n}\boldsymbol{\Lambda}_k^{-1}$$

- The vectors $\boldsymbol{u}_2, \ldots, \boldsymbol{u}_{k+1}$ are the directions of *maximum variance* of the graph embedding, with $\lambda_2^{-1} \geq \ldots \geq \lambda_{k+1}^{-1}$.

Other Laplacian matrices

- The normalized graph Laplacian (symmetric and semi-definite positive):

$$\mathbf{L}_n = \mathbf{D}^{-\frac{1}{2}}\mathbf{L}\mathbf{D}^{-\frac{1}{2}} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$$

- The transition matrix (allows an analogy with Markov chains):

$$\mathbf{L}_t = \mathbf{D}^{-1}\mathbf{A}$$

- The random-walk graph Laplacian:

$$\mathbf{L}_r = \mathbf{D}^{-1}\mathbf{L} = \mathbf{I} - \mathbf{L}_t$$

- These matrices are similar:

$$\mathbf{L}_r = \mathbf{D}^{-\frac{1}{2}}\mathbf{D}^{-\frac{1}{2}}\mathbf{L}\mathbf{D}^{-\frac{1}{2}}\mathbf{D}^{\frac{1}{2}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{L}_n\mathbf{D}^{\frac{1}{2}}$$

Eigenvalues and eigenvectors of $\mathbf{L}_n$ and $\mathbf{L}_r$

- $\mathbf{L}_r \boldsymbol{w} = \lambda \boldsymbol{w} \iff \mathbf{L}\boldsymbol{w} = \lambda \mathbf{D}\boldsymbol{w}$, hence:

$$\mathbf{L}_r : \quad \lambda_1 = 0; \quad \boldsymbol{w}_1 = \mathbf{1}$$

- $\mathbf{L}_n \boldsymbol{v} = \lambda \boldsymbol{v}$. By virtue of the similarity transformation between the two matrices:

$$\mathbf{L}_n : \quad \lambda_1 = 0 \quad \boldsymbol{v}_1 = \mathbf{D}^{\frac{1}{2}}\mathbf{1}$$

- More generally, the two matrices have the same eigenvalues:

$$0 = \lambda_1 \leq \ldots \leq \lambda_i \ldots \leq \lambda_n$$

- Their eigenvectors are related by:

$$\boldsymbol{v}_i = \mathbf{D}^{\frac{1}{2}}\boldsymbol{w}_i, \ \forall i = 1 \ldots n$$

Spectral embedding using the random-walk Laplacian $\mathbf{L}_r$

- The $n \times k$ matrix contains the first k eigenvectors of $\mathbf{L}_r$:

$$\mathbf{W} = \left[\begin{array}{ccc} \boldsymbol{w}_2 & \dots & \boldsymbol{w}_{k+1} \end{array} \right]$$

- It is straightforward to obtain the following expressions, where $\boldsymbol{d}$ and $\mathbf{D}$ are the degree-vector and the degree-matrix:

$$\boldsymbol{w}_i^\top \boldsymbol{d} = 0, \ \forall i, 2 \leq i \leq n$$

$$\mathbf{W}^\top \mathbf{D} \mathbf{W} = \mathbf{I}_k$$

- The isometric embedding using the random-walk Laplacian:

$$\mathbf{Y} = \mathbf{W}^\top = \left[\begin{array}{ccc} \boldsymbol{y}_1 & \dots & \boldsymbol{y}_n \end{array} \right]$$

The normalized additive Laplacian

- Some authors use the following matrix:

$$\mathbf{L}_a = \frac{1}{d_{\max}} (\mathbf{A} + d_{\max}\mathbf{I} - \mathbf{D})$$

- This matrix is closely related to $\mathbf{L}$:

$$\mathbf{L}_a = \frac{1}{d_{\max}} (d_{\max}\mathbf{I} - \mathbf{L})$$

- and we have:

$$\mathbf{L}_a \boldsymbol{u} = \mu \boldsymbol{u} \iff \mathbf{L}\boldsymbol{u} = \lambda \boldsymbol{u},\ \mu = 1 - \frac{\lambda}{d_{\max}}$$

The graph partitioning problem

- **The graph-cut problem**: Partition the graph such that:
  1. Edges between groups have very low weight, and
  2. Edges within a group have high weight.

$$\mathrm{cut}(A_1,\dots,A_k) := \frac{1}{2}\sum_{i=1}^{k} W(A_i,\overline{A}_i) \text{ with } W(A,B) = \sum_{i\in A, j\in B} w_{ij}$$

- **Ratio cut**: (Hagen & Kahng 1992)

$$\mathrm{RatioCut}(A_1,\dots,A_k) := \frac{1}{2}\sum_{i=1}^{k} \frac{W(A_i,\overline{A}_i)}{|A_i|}$$

- **Normalized cut**: (Shi & Malik 2000)

$$\mathrm{NCut}(A_1,\dots,A_k) := \frac{1}{2}\sum_{i=1}^{k} \frac{W(A_i,\overline{A}_i)}{\mathrm{vol}(A_i)}$$

What is spectral clustering?

- Both ratio-cut and normalized-cut minimizations are NP-hard problems
- Spectral clustering is a way to solve relaxed versions of these problems:
  1. The smallest non-null eigenvectors of the *unnormalized Laplacian* approximate the RatioCut minimization criterion, and
  2. The smallest non-null eigenvectors of the *random-walk Laplacian* approximate the NCut criterion.

Spectral clustering using the random-walk Laplacian

- For details see (von Luxburg '07)
- Input: Laplacian $\mathbf{L}_r$ and the number k of clusters to compute.
- Output: Cluster $C_1, \ldots, C_k$.

1. Compute $\mathbf{W}$ formed with the first k eigenvectors of the random-walk Laplacian.
2. Determine the spectral embedding $\mathbf{Y} = \mathbf{W}^\top$
3. Cluster the columns $\boldsymbol{y}_j, j = 1, \ldots, n$ into k clusters using the K-means algorithm.

K-means clustering

See Bishop'2006 (pages 424–428) for more details.

- What is a cluster: a group of points whose inter-point distance are small compared to distances to points outside the cluster.
- Cluster centers: $\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_k$.
- Goal: find an assignment of points to clusters as well as a set of vectors $\boldsymbol{\mu}_i$.
- Notations: For each point $\boldsymbol{y}_j$ there is a *binary indicator variable* $r_{ji} \in \{0, 1\}$.
- Objective: minimize the following *distorsion measure*:

$$J = \sum_{j=1}^{n} \sum_{i=1}^{k} r_{ji} \|\boldsymbol{y}_j - \boldsymbol{\mu}_i\|^2$$

The K-means algorithm

1. Initialization: Choose initial values for $\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_k$.
2. First step: Assign the j-th point to the closest cluster center:

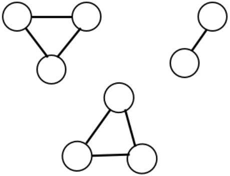
$$r_{ji} = \begin{cases} 1 & \text{if } i = \arg\min_l \|\boldsymbol{y}_j - \mu_l\|^2 \\ 0 & \text{otherwise} \end{cases}$$

3. Second Step: Minimize J to estimate the cluster centers:

$$\boldsymbol{\mu}_i = \frac{\sum_{j=1}^n r_{ji} \boldsymbol{y}_j}{\sum_{j=1}^n r_{ji}}$$

4. Convergence: Repeat until no more change in the assignments.

Spectral Clustering Analysis : The Ideal Case

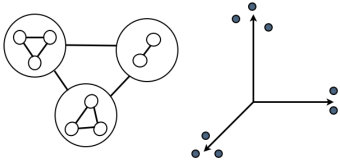


- $\lambda_1 = \lambda_2 = \lambda_3 = 0$
- $\boldsymbol{w}_1, \boldsymbol{w}_2, \boldsymbol{w}_3$ form an orthonormal basis.
- The connected components collapse to $(100), (010), (001)$.
- Clustering is trivial in this case.

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

$$\mathbf{Y} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

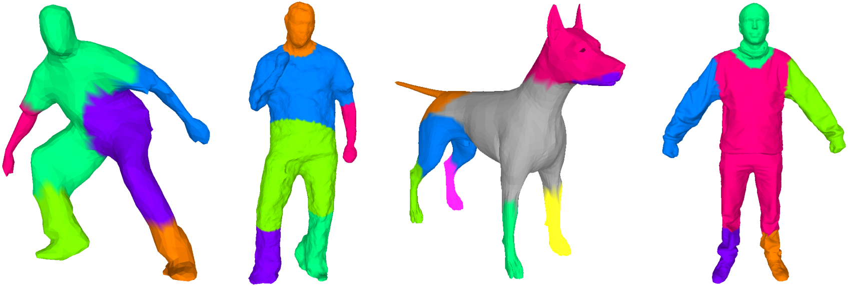
Spectral Clustering Analysis : The Perturbed Case



- See (von Luxburg '07) for a detailed analysis.
- The connected components are no longer *disconnected*, but they are only connected by few edges with low weight.

- The Laplacian is a perturbed version of the ideal case.
- Choosing the first k nonzero eigenvalues is easier the larger the eigengap between λ_{k+1} and λ_{k+2}.
- The fact that the first k eigenvectors of the perturbed case are approximately piecewise constant depends on $|\lambda_{k+2} - \lambda_{k+1}|$.
- Choosing k is a crucial issue.

Mesh segmentation using spectral clustering



K=6 K=6 K=9 K=6

Conclusions

- Spectral graph embedding based on the graph Laplacian is a very powerful tool;
- Allows links between graphs and Riemannian manifolds
- There are strong links with Markov chains and random walks
- It allows clustering (or segmentation) under some conditions
- We (PERCEPTION group) use it for shape matching, shape segmentation, shape recognition, etc.