Manifold Learning for Signal and Image Analysis
Lecture 4: Spectral Clustering

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

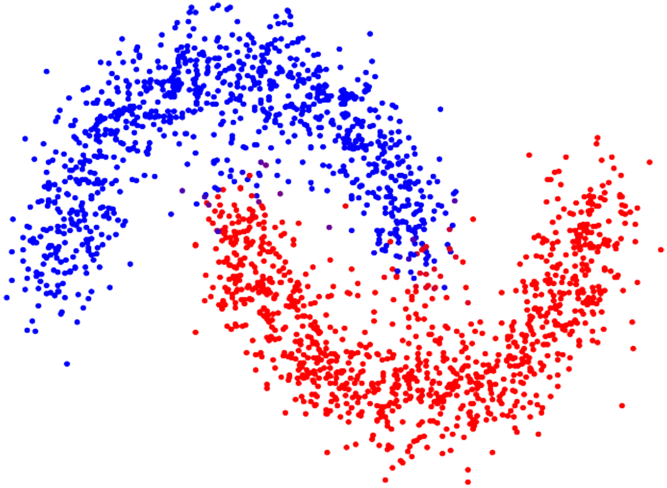
Outline of Lecture 4

- What is spectral clustering?
- We will use the material of Lecture 3
- We will discuss several spectral clustering algorithms
- Link between spectral clustering and graph partitioning
- Link between spectral clustering and random walks on graphs

Material for this lecture

- F. R. K. Chung. Spectral Graph Theory. 1997. (Chapter 1)
- M. Belkin and P. Niyogi. Laplacian Eigenmaps for Dimensionality Reduction and Data Representation. Neural Computation, 15, 1373–1396 (2003).
- U. von Luxburg. A Tutorial on Spectral Clustering. Statistics and Computing, 17(4), 395–416 (2007). (An excellent paper)
- Software: `http://open-specmatch.gforge.inria.fr/index.php`. Computes, among others, Laplacian embeddings of very large graphs.

Example



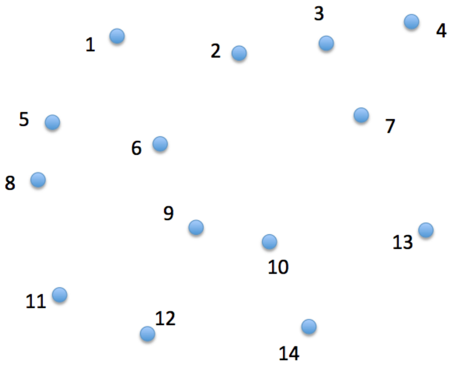
Which Clustering Method to Use?

- Techniques such as K-means or Gaussian mixtures will not work well because the clusters are neither spherical nor Gaussian.
- One needs to apply a non-linear transformation of the data such that “curved” clusters are transformed into “blobs”
- The general idea of spectral clustering is to build an undirected weigthed graph and to map the points (the graph's vertices) into the *spectral* space, spanned by the eigenvectors of the Laplacian matrix.

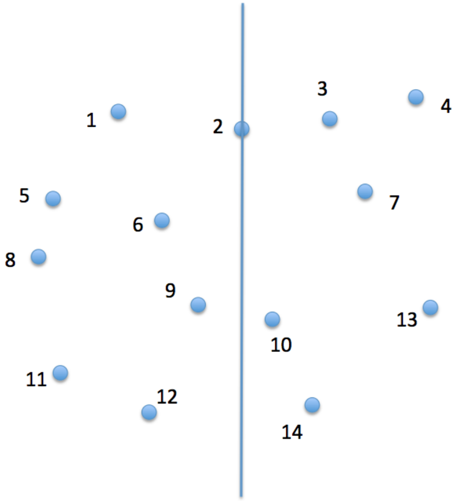
KD Trees

- KD-tree (K-dimensional tree) is a data structure that allows to organize a point cloud under the form of a binary tree.
- The basic idea is to recursively and alternatively project the points onto the x, y, z, x, y, z, etc., axes, to order the points along each axis and to split the set into two halves.
- This point-cloud organization facilitates and accelerates the search of nearest neighbors (at the price of kd-tree construction).
- A more elaborate method (requiring more pre-processing time) is to search for the principal direction and split the data using a plane orthogonal to this direction, and apply this strategy recursively.

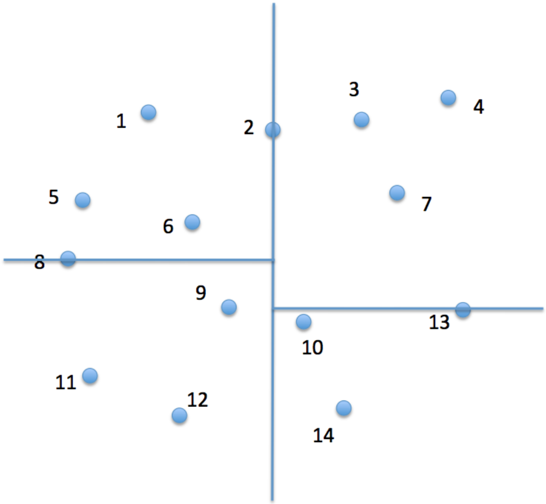
An Example of a 2D-tree (1)



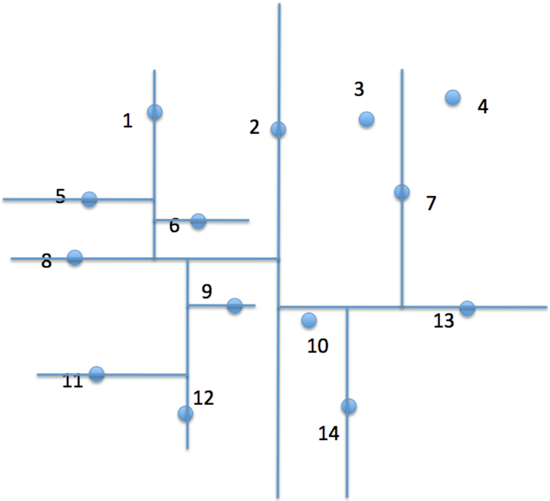
An Example of a 2D-tree (2)



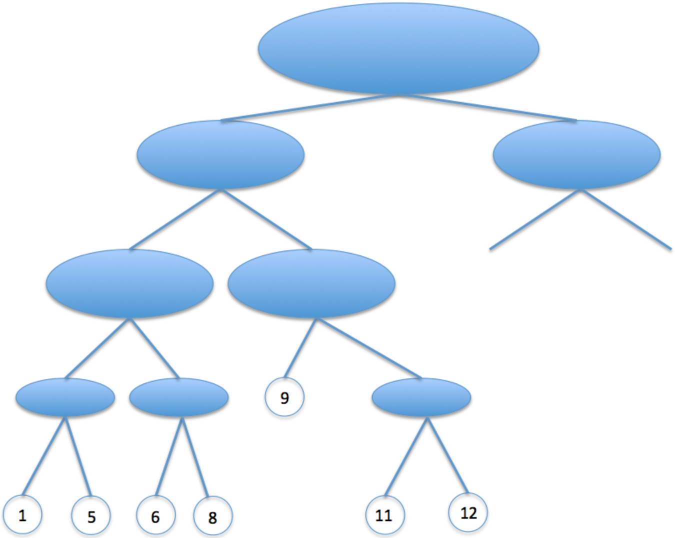
An Example of a 2D-tree (3)



An Example of a 2D-tree (4)



An Example of a 2D-tree (5)



K-means Clustering

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

$$J = \sum_{j=1}^{n} \sum_{k=1}^{m} r_{jk} \|\boldsymbol{x}_j - \boldsymbol{\mu}_k\|^2$$

The K-means Algorithm

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

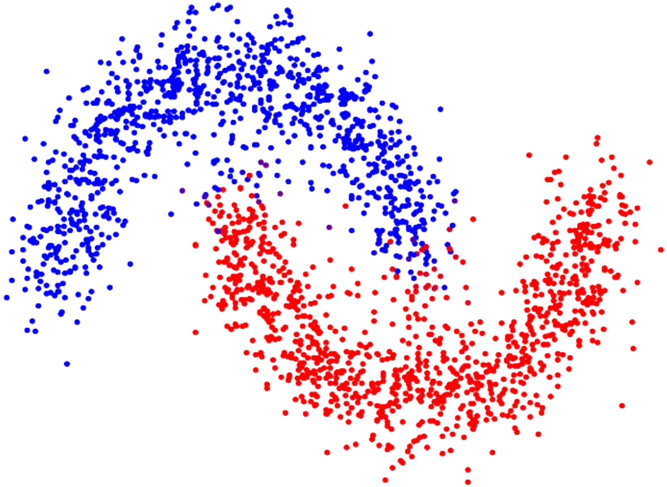
$$r_{jk} = \begin{cases} 1 & \text{if } k = \arg\min_l \|\boldsymbol{x}_j - \mu_l\|^2 \\ 0 & \text{otherwise} \end{cases}$$

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

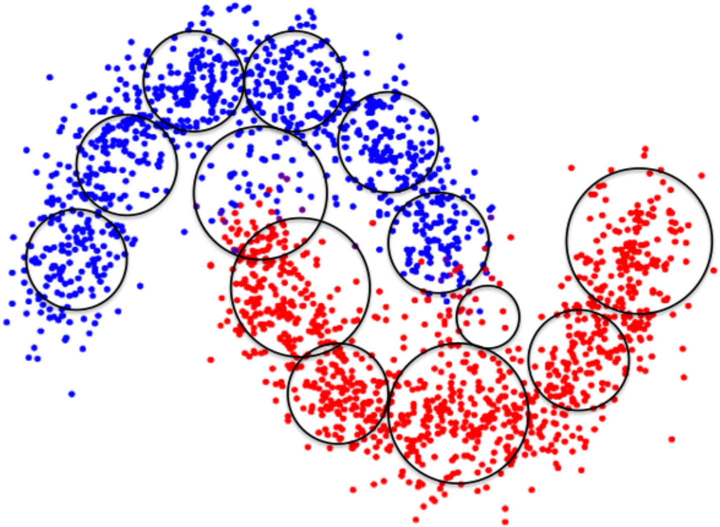
$$\boldsymbol{\mu}_k = \frac{\sum_{j=1}^n r_{jk} \boldsymbol{x}_j}{\sum_{j=1}^n r_{jk}}$$

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

How to Represent This Point Cloud?

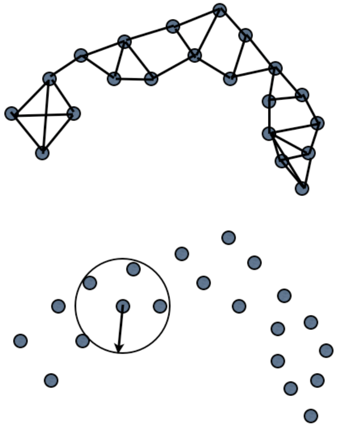


Spherical Clusters



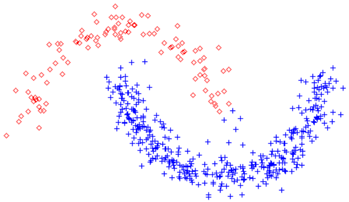
Building a Graph from a Point Cloud

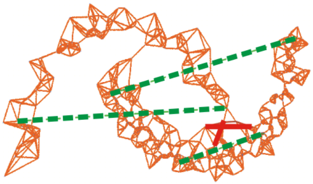
- K-nearest neighbor (KNN) rule
- ε-radius rule
- Other more sophisticated rules can be found in the literature, i.e., Lee and Verleysen. Nonlinear Dimensionality Reduction (Appendix E). Springer. 2007.



- Remark: The KD-tree data structure can be used to facilitate graph construction when the number of points is large.

An Example of Graph Building





The Graph Partitioning Problem

- We want to find a partition of the graph such that the edges between different groups have very low weight, while the edges within a group have high weight.
- **The mincut problem**:
  1. Edges between groups have very low weight, and
  2. Edges within a group have high weight.
  3. Choose a partition of the graph into k groups that mimimizes the following criterion:

$$\text{mincut}(A_1, \ldots, A_k) := \frac{1}{2} \sum_{i=1}^{k} W(A_i, \overline{A}_i)$$

- with

$$W(A, B) = \sum_{i \in A, j \in B} w_{ij}$$

RatioCut and NormalizedCut

- Often, the mincut solution isolates a vertex from the rest of the graph.
- Request that the groups are reasonably large.
- **Ratio cut** (Hagen & Kahng 1992) minimizes:

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

- Here $|A|$ refers to the number of vertices in group A.
- **Normalized cut**: (Shi & Malik 2000)

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

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 NormalizedCut criterion.

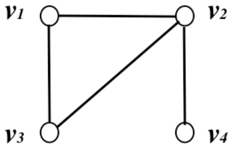
The Laplacian matrix of a graph

- $f : \mathcal{V} \longrightarrow \mathbb{R}$, i.e., $f(v_1), \dots, f(v_n)$.
- $(\mathbf{L}\boldsymbol{f})(v_i) = \sum_{v_j \sim v_i} (f(v_i) - f(v_j))$
- Connection between the Laplacian and the adjacency matrices:

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

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

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



Matrices of an undirected weighted graph

- We consider *undirected weighted graphs*; Each edge e_{ij} is weighted by $w_{ij} > 0$. We obtain:

$$\mathbf{\Omega} := \begin{cases} \Omega_{ij} = w_{ij} & \text{if there is an edge } e_{ij} \\ \Omega_{ij} = 0 & \text{if there is no edge} \\ \Omega_{ii} = 0 & \end{cases}$$

- The degree matrix: $\mathbf{D} = \sum_{i \sim j} w_{ij}$

The Laplacian on an undirected weighted graph

- $\mathbf{L} = \mathbf{D} - \mathbf{\Omega}$
- 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 $\leftrightarrow$ $w_{ij} \geq 0$.
- $\mathbf{L}$ has n non-negative, real-valued eigenvalues: $0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$.

The Laplacian in Practice

- A graph vertex v_i is associated with a point $\boldsymbol{x}_i \in \mathbb{R}^D$.
- The weight w_{ij} of an edge e_{ij} is defined by the Gaussian kernel:

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

- This defines a similarity function between two *nearby* points.

Other adjacency matrices

- The *normalized weighted adjacency matrix*

$$\mathbf{\Omega}_N = \mathbf{D}^{-1/2}\mathbf{\Omega}\mathbf{D}^{-1/2}$$

- The *transition* matrix of the Markov process associated with the graph:

$$\mathbf{\Omega}_R = \mathbf{D}^{-1}\mathbf{\Omega} = \mathbf{D}^{-1/2}\mathbf{\Omega}_N\mathbf{D}^{1/2}$$

Several Laplacian matrices

- The *unnormalized Laplacian* which is also referred to as the *combinatorial Laplacian* $\mathbf{L}_C$,
- the *normalized Laplacian* $\mathbf{L}_N$, and
- the *random-walk Laplacian* $\mathbf{L}_R$ also referred to as the *discrete Laplace operator*.

We have:

$$\begin{aligned}
\mathbf{L}_C &= \mathbf{D} - \mathbf{\Omega} \\
\mathbf{L}_N &= \mathbf{D}^{-1/2}\mathbf{L}_C\mathbf{D}^{-1/2} = \mathbf{I} - \mathbf{\Omega}_N \\
\mathbf{L}_R &= \mathbf{D}^{-1}\mathbf{L}_C = \mathbf{I} - \mathbf{\Omega}_R
\end{aligned}$$

Some spectral properties of the Laplacians

Laplacian	Null space	Eigenvalues	Eigenvectors
$\mathbf{L}_C = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$	$\boldsymbol{u}_1 = \mathbf{1}$	$0 = \lambda_1 < \lambda_2 \leq \ldots \leq \lambda_n \leq 2\max_i(d_i)$	$\boldsymbol{u}_{i>1}^\top \mathbf{1} = 0$, $\boldsymbol{u}_i^\top \boldsymbol{u}_j = \delta_{ij}$
$\mathbf{L}_N = \mathbf{W}\mathbf{\Gamma}\mathbf{W}^\top$	$\boldsymbol{w}_1 = \mathbf{D}^{1/2}\mathbf{1}$	$0 = \gamma_1 < \gamma_2 \leq \ldots \leq \gamma_n \leq 2$	$\boldsymbol{w}_{i>1}^\top \mathbf{D}^{1/2}\mathbf{1} = 0$, $\boldsymbol{w}_i^\top \boldsymbol{w}_j = \delta_{ij}$
$\mathbf{L}_R = \mathbf{T}\mathbf{\Gamma}\mathbf{T}^{-1}$ $\mathbf{T} = \mathbf{D}^{-1/2}\mathbf{W}$	$\boldsymbol{t}_1 = \mathbf{1}$	$0 = \gamma_1 < \gamma_2 \leq \ldots \leq \gamma_n \leq 2$	$\boldsymbol{t}_{i>1}^\top \mathbf{D}\mathbf{1} = 0$, $\boldsymbol{t}_i^\top \mathbf{D}\boldsymbol{t}_j = \delta_{ij}$

Spectral properties of adjacency matrices

From the relationship between the normalized Laplacian and adjacency matrix: $\mathbf{L}_N = \mathbf{I} - \mathbf{\Omega}_N$ one can see that their eigenvalues satisfy $\gamma = 1 - \delta$.

Adjacency matrix	Eigenvalues	Eigenvectors
$\mathbf{\Omega}_N = \mathbf{W}\mathbf{\Delta}\mathbf{W}^\top$, $\mathbf{\Delta} = \mathbf{I} - \mathbf{\Gamma}$	$-1 \leq \delta_n \leq \ldots \leq \delta_2 < \delta_1 = 1$	$\boldsymbol{w}_i^\top \boldsymbol{w}_j = \delta_{ij}$
$\mathbf{\Omega}_R = \mathbf{T}\mathbf{\Delta}\mathbf{T}^{-1}$	$-1 \leq \delta_n \leq \ldots \leq \delta_2 < \delta_1 = 1$	$\boldsymbol{t}_i^\top \mathbf{D}\boldsymbol{t}_j = \delta_{ij}$

The Laplacian of a graph with one connected component

- $\mathbf{L}\boldsymbol{u} = \lambda \boldsymbol{u}$.
- $\mathbf{L}\mathbf{1} = \mathbf{0}$, $\lambda_1 = 0$ is the smallest eigenvalue.
- The *one* vector: $\mathbf{1} = (1 \dots 1)^\top$.
- $0 = \boldsymbol{u}^\top \mathbf{L} \boldsymbol{u} = \sum_{i,j=1}^n w_{ij}(u(v_i) - u(v_j))^2$.
- If any two vertices are connected by a path, then $\boldsymbol{u} = (u(v_1), \dots, u(v_n))$ needs to be constant at all vertices such that the quadratic form vanishes. Therefore, a graph with one connected component has the constant vector $\boldsymbol{u}_1 = \mathbf{1}$ 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 $\mathbf{1}_{L_i} = (000011111 0000)^\top \in \mathbb{R}^n$
- These vectors are the *indicator vectors* of the graph's connected components.
- Notice that $\mathbf{1}_{L_1} + \ldots + \mathbf{1}_{L_k} = \mathbf{1}$

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 depends on the graph's structure and it is difficult to analyse.
- The Fiedler value is the *algebraic connectivity of a graph*, the further from 0, the more connected.
- The Fiedler 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

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

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

- 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:

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

Spectral embedding using the *unnormalized* Laplacian

- Compute the eigendecomposition $\mathbf{L}_C = \mathbf{D} - \mathbf{\Omega} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$.
- 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 $n \times k$ column-orthogonal matrix $\widetilde{\mathbf{U}} = [\boldsymbol{u}_2 \ldots \boldsymbol{u}_{k+1}]$:

$$\widetilde{\mathbf{U}} = \left[\begin{array}{ccc} \boldsymbol{u}_2(v_1) & \ldots & \boldsymbol{u}_{k+1}(v_1) \\ \vdots & & \vdots \\ \boldsymbol{u}_2(v_n) & \ldots & \boldsymbol{u}_{k+1}(v_n) \end{array} \right]$$

- Embedding: The i-row of this matrix correspond to the representation of vertex v_I in the $\mathbb{R}^k$ basis spanned by the orthonormal vector basis $\boldsymbol{u}_2, \ldots, \boldsymbol{u}_{k+1}$.
- Therefore: $\mathbf{Y} = [\boldsymbol{y}_1 \ldots \boldsymbol{y}_i \ldots \boldsymbol{y}_n] = \widetilde{\mathbf{U}}^\top$

Spectral embedding using the random-walk Laplacian

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

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

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

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

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

- Hence, vectors $\boldsymbol{w}_2, \dots, \boldsymbol{w}_{k+1}$ do not form an orthonormal basis.
- The embedding using the random-walk Laplacian:

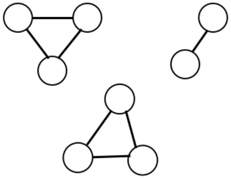
$$\mathbf{Y} = [\boldsymbol{y}_1 \dots \boldsymbol{y}_i \dots \boldsymbol{y}_n] = \widetilde{\mathbf{W}}^\top$$

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.

Spectral Clustering Analysis : The Ideal Case

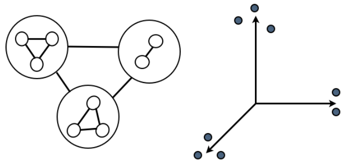


- $\lambda_1 = \lambda_2 = \lambda_3 = 0$
- $\boldsymbol{w}_1, \boldsymbol{w}_2, \boldsymbol{w}_3$ form an orthonormal basis.
- The connected components collapse to $(100), (010), (001)$.
- Clustering is trivial in this case.

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

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

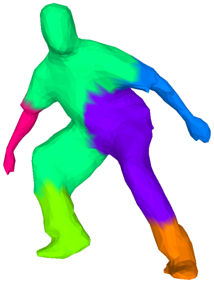
Spectral Clustering Analysis : The Perturbed Case



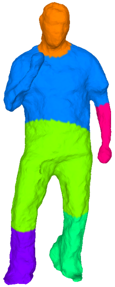
- See (von Luxburg '07) for a detailed analysis.
- The connected components are no longer *disconnected*, but they are only connected by few edges with low weight.

- 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.

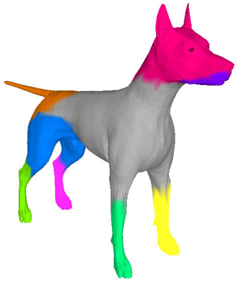
Mesh segmentation using spectral clustering



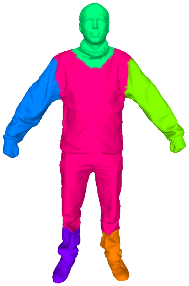
K=6



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
- The PERCEPTION group uses it for shape matching, shape segmentation, shape recognition, etc.