Spectral Clustering: An Application to fMRI Datasets
(Based on a paper by Shen and Meyer, NeuroImage 2008)

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

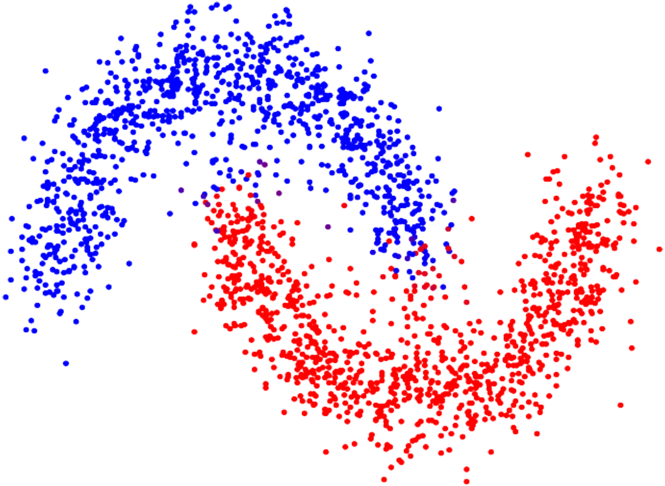
Outline

- A short introduction to spectral embedding of graphs and to spectral clustering
- Spectral clustering applied to fMRI data

Material

- X. Shen and F. Meyer. Low-Dimensional Embedding of fMRI Datasets. NeuroImage 41 (2008). http://ecee.colorado.edu/~fmeyer/Pub/neuroimage08.pdf
- F. Meyer and G. Stephens. Locality and Low-dimensions in the Prediction of Natural Experience from fMRI. In NIPS 2008.
- François Meyer's (formerly at IRISA) publications: http://ecee.colorado.edu/~fmeyer/publications.html
- Additional material: Course on *Data Analysis and Manifold Learning*. http://perception.inrialpes.fr/people/Horaud/Courses/DAML_2011.html

An 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 to the data such that “curved” clusters are transformed into “blobs”
- The general idea of spectral clustering:
  1. Build an undirected weigthed graph and its Laplacian matrix
  2. Map the graph’s vertices into the *spectral* space, spanned by the eigenvectors of the Laplacian matrix.
  3. Perform K-means in the spectral space

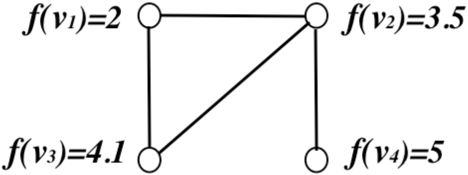
Basic Graph Notations and Definitions

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

- $\mathcal{V}(\mathcal{G}) = \{v_1, \ldots, v_n\}$ 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} with a positive weight ω_{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 degree of a node v_i is defined by d_i, $d_i = \sum_{v_i \sim v_j} \omega_{ij}$.

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: $\boldsymbol{f} = (f(v_1), \ldots, f(v_n)) = (f_1, \ldots, f_n)$.



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: $\mathbf{D} = \text{Diag}[d_i]$

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: $0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$.

Spectral Decomposition

- Mapping a function onto itself: $\mathbf{L}\boldsymbol{u} = \lambda \boldsymbol{u}$ (Eigenvalue/eigenvector pairs).
- Spectral decomposition: $\mathbf{L} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^\top$ with $\mathbf{U}\mathbf{U}^\top = \mathbf{I}$.
- Let $\mathbf{U}$ be:

$$\mathbf{U} = \begin{bmatrix} 1 & u_{12} & \dots & u_{1k} & \dots & u_{1n} \\ \vdots & & \vdots & & \vdots & \\ 1 & u_{n2} & \dots & u_{nk} & \dots & u_{nn} \end{bmatrix} \quad (1)$$

- Each column of $\mathbf{U}$, $\boldsymbol{u}_k = (u_{1k} \dots u_{ik} \dots u_{nk})^\top$, $2 \leq k \leq n$ is an eigenvector such that $\boldsymbol{u}_k^\top \mathbb{1} = 0$
- By omitting the first eigenvalue/eigenvector pair $\lambda_1 = 0 / \boldsymbol{u}_1 = \mathbb{1}$, we have:

$$\mathbf{L} = \sum_{k=2}^{n} \lambda_k \boldsymbol{u}_k \boldsymbol{u}_k^\top \quad (2)$$

Commute-time Embedding

- The Moore-Penrose pseudo-inverse of the Laplacian (we simply omit the zero eigenvalue):

$$\mathbf{L}^{\dagger} = \sum_{k=2}^{n} \frac{1}{\lambda_k} \boldsymbol{u}_k \boldsymbol{u}_k^{\top} \tag{3}$$

- Spectral decomposition:

$$\mathbf{L}^{\dagger} = \mathbf{U}\boldsymbol{\Lambda}^{-1}\mathbf{U}^{\top} \text{ with } \boldsymbol{\Lambda}^{-1} = \mathrm{Diag}[\lambda_2^{-1} \dots \lambda_k^{-1} \dots \lambda_n^{-1}]$$

$$\begin{aligned} \mathbf{L}^{\dagger} &= \left(\boldsymbol{\Lambda}^{-\frac{1}{2}}\mathbf{U}^{\top}\right)^{\top} \left(\boldsymbol{\Lambda}^{-\frac{1}{2}}\mathbf{U}^{\top}\right) \\ &= \mathbf{X}^{\top}\mathbf{X} \end{aligned}$$

Properties of the Commute-time Embedding

$$\mathbf{X} = \mathbf{\Lambda}^{-\frac{1}{2}} \mathbf{U}^\top = \left[\begin{array}{ccc} \boldsymbol{x}_1 & \dots & \boldsymbol{x}_i & \dots \boldsymbol{x}_n \end{array} \right]$$

$$\begin{aligned}
\boldsymbol{x}_i &= \left(\lambda_2^{-1/2} u_{i2} \dots \lambda_n^{-1/2} u_{in} \right)^\top \\
\|\boldsymbol{x}_i\|^2 &= \sum_{k=2}^{n} \lambda_k^{-1} u_{ik}^2 \\
\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2 &= \sum_{k=2}^{n} \lambda_k^{-1} (u_{ik} - u_{jk})^2 \\
\mathbf{X}\mathbb{1} &= 0 \\
\mathbf{\Sigma}_X &= \frac{1}{n} \mathbf{X}\mathbf{X}^\top = \frac{1}{n} \mathsf{Diag}[\lambda_2^{-1}, \dots, \lambda_n^{-1}]
\end{aligned}$$

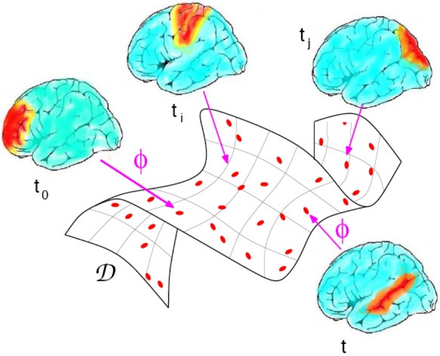
Spectral Clustering

- Input: Laplacian $\mathbf{L}$ and the number K of *principal* eigenvalue/eigenvector pairs
- Output: Cluster $C_1, \ldots, C_k$.

1. Compute $\mathbf{X}$ using the first K eigenvalue/eigenvector pairs.
2. Cluster the columns $\boldsymbol{x}_i, i = 1, \ldots, n$ of $\mathbf{X}$ into K clusters using the K-means algorithm (or your preferred clustering method).

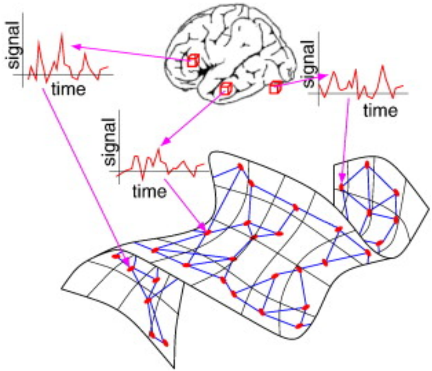
Low-dimensional embedding of fMRI

- fMRI provides a large scale measurement of neuronal activity



fMRI data

- Each voxel v_i generates a time series $\boldsymbol{x}_i = \begin{pmatrix} x_i(1) & \dots & x_i(T) \end{pmatrix}^\top \in \mathbb{R}^T$



A network of functionally correlated voxels

- A connectivity graph is formed with the standard approach:

$$W_{ij} = \begin{cases} \exp\left(-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2/\sigma^2\right) & \text{if } v_i \sim v_j \\ 0 & \text{otherwise} \end{cases}$$

- The Euclidean distance between time series:

$$\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2 = \sum_{t=1}^{T} \left(x_i(t) - x_j(t)\right)^2$$

- Choice for σ:

$$\sigma = 2 \min_{i<j} \|\boldsymbol{x}_i - \boldsymbol{x}_j\|$$

- Choice for n_n (nearest neighbor): user defined and varies from 7 to ... 100.

Matrices

- Diagonal degree matrix: $\mathbf{D}(i,i) = \sum_j W_{i,j}$
- Transition matrix: $\mathbf{P} = \mathbf{D}^{-1}\mathbf{W}$
- Transition probabilities:

$$P_{i,j} = \mathbf{P}(i,j) = \frac{W_{i,j}}{\sum_j W_{i,j}}$$

- It is a row-stochastic matrix: $\sum_j P_{i,j} = 1$

Random walk and commute-time

- Consider a random walk on the graph denoted by Z_n: if the walk is at v_i it jumps to one of its neighbors v_j with probability $P_{i,j}$.
- if v_i and v_j are in the same functional area and v_i and v_k are in different functional areas, we expect that $P_{i,j} \gg P_{i,k}$.
- The *average hitting time* measures the number of steps that it takes for a random walk starting in v_i to hit v_j for the first time:

$$H(v_i, v_j) = E_i[T_j] \text{ with } T_j = \min\{n \geq 0; Z_n = j\}$$

- The hitting time is not symmetric, use the commute time instead:

$$\kappa(v_i, v_j) = H(v_i, v_j) + H(v_j, v_i) = E_i[T_j] + E_j[T_i]$$

The commute time distance in the spectral domain

- Let $(\lambda_k, \phi_k)_{k=1}^N$ be the eigenvalue-eigenvector pairs of matrix $\mathbf{P}$ that can be easily computed because $\mathbf{D}^{1/2}\mathbf{P}\mathbf{D}^{-1/2}$ is a real symmetric matrix. Moreover (see Lecture #3) we have:

$$-1 \leq \lambda_N \leq \ldots \leq \lambda_k \leq \ldots \leq \lambda_1 = 1$$

- The commute time distance is:

$$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)^2 = \sum_{k=2}^{n} \frac{1}{1-\lambda_k} \left(\frac{\phi_k(i)}{\sqrt{\pi_i}} - \frac{\phi_k(j)}{\sqrt{\pi_j}} \right)$$

- $\boldsymbol{\pi} = (\pi_1 \ldots \pi_i \ldots \pi_N)^\top$ is the eigenvector $\mathbf{P}^\top \boldsymbol{\pi} = \boldsymbol{\pi}$ with:

$$\pi_i = \frac{d_i}{\sum_{i,j} W_{i,j}}$$

Embedding

- The initial time series can now be embedded using the mapping: $\boldsymbol{x}_i \longrightarrow \Psi(\boldsymbol{x}_i)$, i.e., $\mathbb{R}^T \rightarrow \mathbb{R}^K$:

$$\Psi(\boldsymbol{x}_i) = \frac{1}{\sqrt{\pi_i}} \left(\frac{\boldsymbol{\phi}_k(i)}{\sqrt{1-\lambda_2}} \dots \frac{\boldsymbol{\phi}_k(i)}{\sqrt{1-\lambda_k}} \dots \frac{\boldsymbol{\phi}_k(i)}{\sqrt{1-\lambda_n}} \right)^\top$$

- This is strictly equivalent to the commute-time embedding based on the normalized graph Laplacian (see Lecture #3).

Choosing the dimension

- Remind that each voxel in the brain corresponds to a graph node and there is a time series at each voxel:

$$\mathbf{X} = \begin{bmatrix} x_1(1) & \dots & x_1(t) & \dots & x_1(T) \\ \vdots & & \vdots & & \vdots \\ x_i(1) & \dots & x_i(t) & \dots & x_i(T) \\ \vdots & & \vdots & & \vdots \\ x_N(1) & \dots & x_N(t) & \dots & x_N(T) \end{bmatrix}$$

- Each column $\boldsymbol{x}(t)$ in this matrix is a scalar function defined over the graphs' vertices. It can be decomposed using the eigenvectors:

$$\boldsymbol{x}(t) = \sum_{k=2}^{n} \langle \boldsymbol{x}(t), \boldsymbol{\phi}_k \rangle \boldsymbol{\phi}_k + \boldsymbol{r}(t)$$

Choosing the dimension

- This can be written as:

$$\widehat{\boldsymbol{x}}(t) = \sum_{k=2}^{n} \langle \boldsymbol{x}(t), \boldsymbol{\phi}_k \rangle \boldsymbol{\phi}_k = \sum_{k=2}^{n} x_k(t) \boldsymbol{\phi}_k$$

  Therefore, each entry i (at each brain location or graph vertex) of this approximated vector is:

$$\widehat{\boldsymbol{x}}_i(t) = \sum_{k=2}^{n} x_k(t) \phi_k(i)$$

- The discrepancy between the initial observations and their approximate representation is:

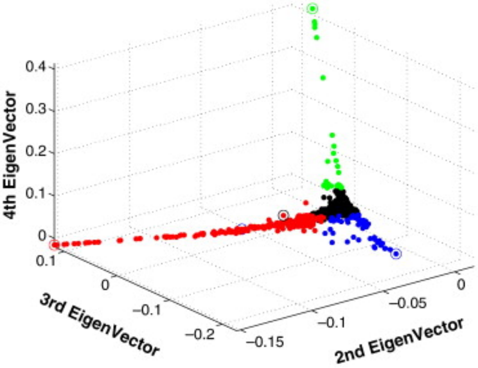
$$\varepsilon_i(K) = \frac{\sum_{t=1}^{T} (\boldsymbol{x}_i(t) - \widehat{\boldsymbol{x}}_i(t))^2}{\sum_{t=1}^{T} \boldsymbol{x}_i^2(t)}$$

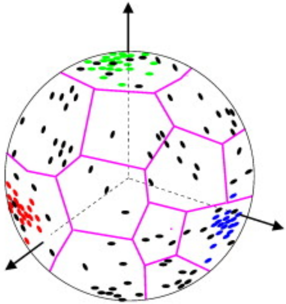
Choosing the dimension

- The authors suggest to average $\varepsilon_i(K)$ over the voxels lying in a "functional" region, and to take the max over all these average values.
- It is not clear what is a functional region, since this is what it is searched for and why the average is selected.

Segmentation using K-means

- The authors notice that the embedding has a star-like shape.





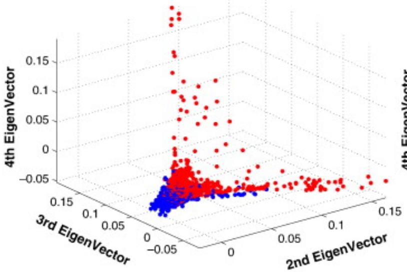
Segmentation using K-means

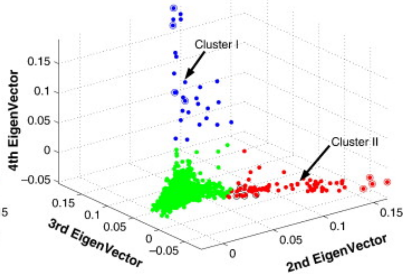
- The "arms" correspond to:
  - activated time series or
  - strong physiological artifacts
- The center blob corresponds to "background activity"
- The embedded data are projected on a hyper-sphere of dimension K.
- The background is spread over the sphere.
- The K-means algorithm is applied to the spherical data

Event related dataset

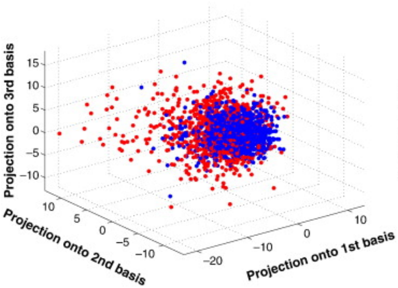
- Study of age-related changes in functional anatomy
- 2050 time series.

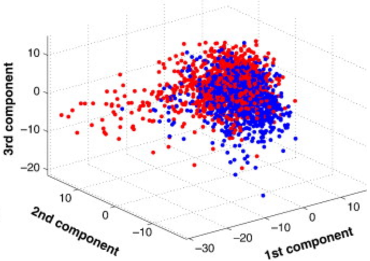
Spectral embedding/clustering results



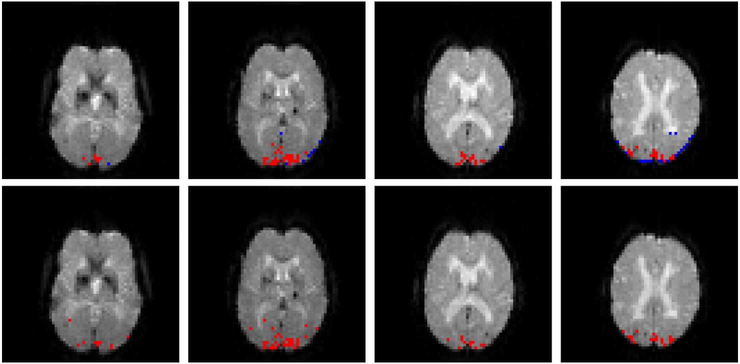


Results obtained with PCA and ISOMAP





Activation maps



Discussion

- The paper uses an extremely well studied method in machine learning.
- The method could also be applied to other types of brain data, such as EEG for discovering *crossmodal bindings*
- A more general approach would be to consider *graph kernels* or *diffusion kernels* and to apply kernel methods to this type of data.