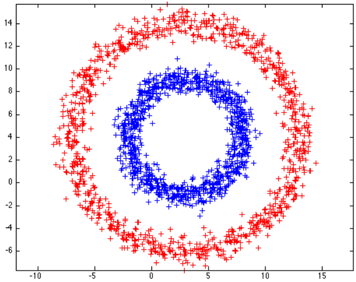
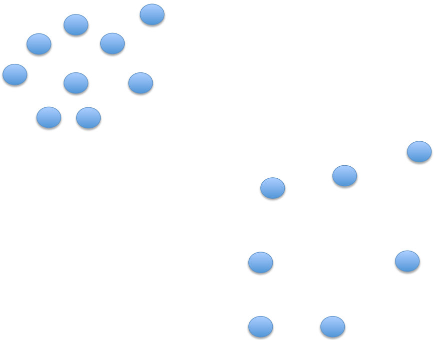
Spectral Clustering



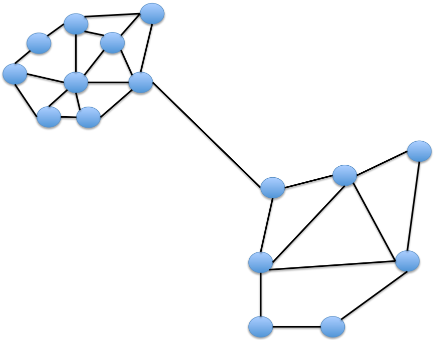
COPERNICUS MASTER
IN DIGITAL EARTH

General ideas



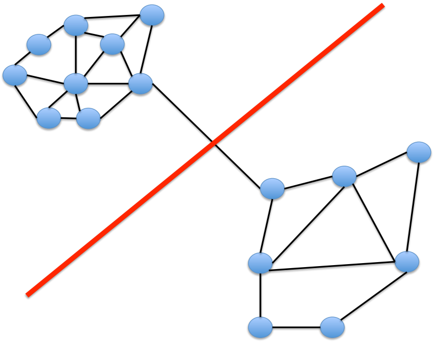
Dataset

General ideas



Construction of a graph

General ideas



Cut in the graph

Plan

Definition of a simple graph

A finite **graph** $G = \{V, E\}$ is defined as a set of :

- Vertices $V = \{v_1, v_2, ..., v_n\}$
- Edges $E = \{e_1, e_2, ..., e_n\}$
  - The number of vertices and edges is in general different
  - Vertices are composed of pairs $e = (x, y)$ of edges

Definition of a simple graph

A finite **graph** $G = \{V, E\}$ is defined as a set of :

- Vertices $V = \{v_1, v_2, ..., v_n\}$
- Edges $E = \{e_1, e_2, ..., e_n\}$
  - The number of vertices and edges is in general different
  - Vertices are composed of pairs $e = (x, y)$ of edges

Definition of a simple graph

A finite **graph** $G = \{V, E\}$ is defined as a set of :

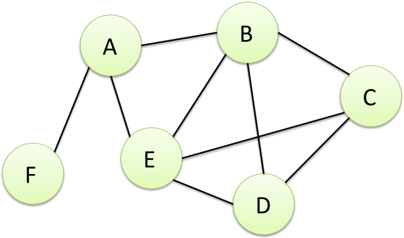
- Vertices $V = \{v_1, v_2, ..., v_n\}$
- Edges $E = \{e_1, e_2, ..., e_n\}$
  - The number of vertices and edges is in general different
  - Vertices are composed of pairs $e = (x, y)$ of edges

Definition of a simple graph

A finite **graph** $G = \{V, E\}$ is defined as a set of :

- Vertices $V = \{v_1, v_2, ..., v_n\}$
- Edges $E = \{e_1, e_2, ..., e_n\}$
  - The number of vertices and edges is in general different
  - Vertices are composed of pairs $e = (x, y)$ of edges

Example : How is defined the following graph ?

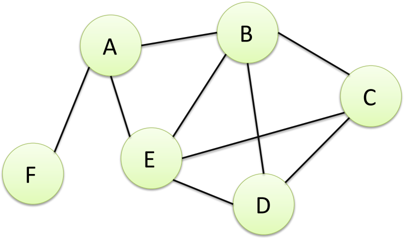


Definition of a simple graph

A finite **graph** $G = \{V, E\}$ is defined as a set of :

- Vertices $V = \{v_1, v_2, ..., v_n\}$
- Edges $E = \{e_1, e_2, ..., e_n\}$
  - The number of vertices and edges is in general different
  - Vertices are composed of pairs $e = (x, y)$ of edges

Example : How is defined the following graph ?



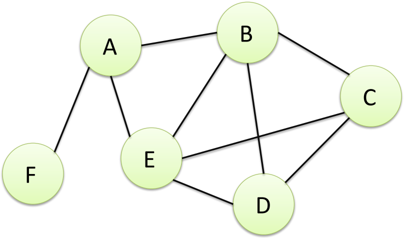
Caracterisation
$G = \{V, E\}$

Definition of a simple graph

A finite **graph** $G = \{V, E\}$ is defined as a set of :

- Vertices $V = \{v_1, v_2, ..., v_n\}$
- Edges $E = \{e_1, e_2, ..., e_n\}$
  - The number of vertices and edges is in general different
  - Vertices are composed of pairs $e = (x, y)$ of edges

Example : How is defined the following graph ?



Caracterisation

$G = \{V, E\}$

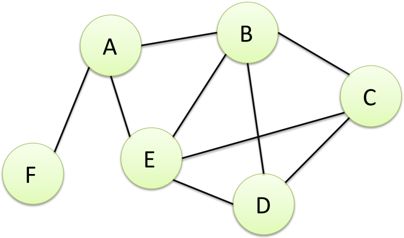
$V = \{A, B, C, D, E, F\}$

Definition of a simple graph

A finite **graph** $G = \{V, E\}$ is defined as a set of :

- Vertices $V = \{v_1, v_2, ..., v_n\}$
- Edges $E = \{e_1, e_2, ..., e_n\}$
  - The number of vertices and edges is in general different
  - Vertices are composed of pairs $e = (x, y)$ of edges

Example : How is defined the following graph ?



Caracterisation

$G = \{V, E\}$

$V = \{A, B, C, D, E, F\}$

$E =$

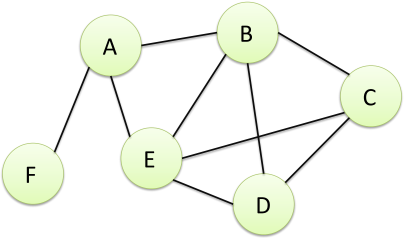
$\{(A, B), (A, E), (A, F), (B, E),$
$(B, D), (B, C), (C, E), (C, D)$
$, (D, E)\}$

Definition of a simple graph

A finite **graph** $G = \{V, E\}$ is defined as a set of :

- Vertices $V = \{v_1, v_2, ..., v_n\}$
- Edges $E = \{e_1, e_2, ..., e_n\}$
  - The number of vertices and edges is in general different
  - Vertices are composed of pairs $e = (x, y)$ of edges

Example : How is defined the following graph ?

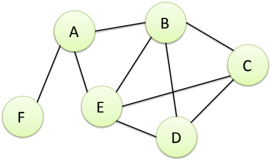


Important note

Here we focus only on simple graphes (vertices connect only 2 edges) but we can have multiple graphs (one edge can be connect with himself) and ordered (vertices can connect only in one direction – directed multigraph)

Characterisation

- Two edges (X, Y) are adjacents if there exists a vertice (X, Y) in E. They are neighbors
- One edge that joints X and Y is incident to X and Y
- The degree $d(X)$ of a vertex X is the number of incident edges to X



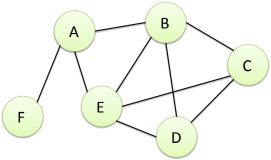
Example for B

- Degree $d(B) = 4$
- Incident edges are $(B, A), (B, E), (B, D), (B, C)$

Note : a vertex of 0 degree is isolated. A simple graph with n vertex has degrees $\in [0, n-1]$. If all vertices have a $n-1$-degree, the associated graph is complete.

Characterisation

- Two edges (X, Y) are adjacents if there exists a vertice (X, Y) in E. They are neighbors
- One edge that joints X and Y is incident to X and Y
- The degree $d(X)$ of a vertex X is the number of incident edges to X



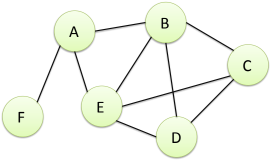
Example for B

- Degree $d(B) = 4$
- Incident edges are $(B, A), (B, E), (B, D), (B, C)$

Note : a vertex of 0 degree is isolated. A simple graph with n vertex has degrees $\in [0, n-1]$. If all vertices have a $n-1$-degree, the associated graph is complete.

Characterisation

- Two edges (X, Y) are adjacents if there exists a vertice (X, Y) in E. They are neighbors
- One edge that joints X and Y is incident to X and Y
- The degree $d(X)$ of a vertex X is the number of incident edges to X



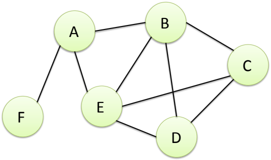
Example for B

- Degree $d(B) = 4$
- Incident edges are $(B, A), (B, E), (B, D), (B, C)$

Note : a vertex of 0 degree is isolated. A simple graph with n vertex has degrees $\in [0, n-1]$. If all vertices have a $n-1$-degree, the associated graph is complete.

Characterisation

- Two edges (X,Y) are adjacents if there exists a vertice (X,Y) in E. They are neighbors
- One edge that joints X and Y is incident to X and Y
- The degree $d(X)$ of a vertex X is the number of incident edges to X



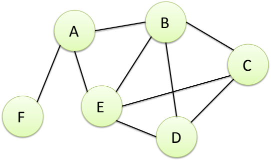
Example for B

- Degree $d(B) = 4$
- Incident edges are $(B,A),(B,E),(B,D),(B,C)$

Note : a vertex of 0 degree is isolated. A simple graph with n vertex has degrees $\in [0, n-1]$. If all vertices have a $n-1$-degree, the associated graph is complete.

Characterisation

- Two edges (X, Y) are adjacents if there exists a vertice (X, Y) in E. They are neighbors
- One edge that joints X and Y is incident to X and Y
- The degree $d(X)$ of a vertex X is the number of incident edges to X



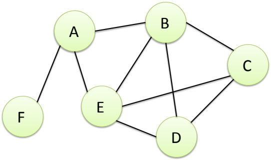
Example for B

- Degree $d(B) = 4$
- Incident edges are $(B, A), (B, E), (B, D), (B, C)$

Note : a vertex of 0 degree is isolated. A simple graph with n vertex has degrees $\in [0, n-1]$. If all vertices have a $n-1$-degree, the associated graph is complete.

Characterisation

- Two edges (X,Y) are adjacents if there exists a vertice (X,Y) in E. They are neighbors
- One edge that joints X and Y is incident to X and Y
- The degree $d(X)$ of a vertex X is the number of incident edges to X



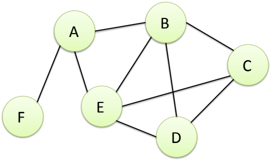
Example for B

- Degree $d(B) = 4$
- Incident edges are $(B,A),(B,E),(B,D),(B,C)$

Note : a vertex of 0 degree is isolated. A simple graph with n vertex has degrees $\in [0, n-1]$. If all vertices have a $n-1$-degree, the associated graph is complete.

Characterisation

- Two edges (X, Y) are adjacents if there exists a vertice (X, Y) in E. They are neighbors
- One edge that joints X and Y is incident to X and Y
- The degree $d(X)$ of a vertex X is the number of incident edges to X



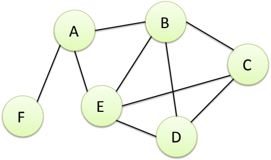
Example for B

- Degree $d(B) = 4$
- Incident edges are $(B, A), (B, E), (B, D), (B, C)$

Note : a vertex of 0 degree is isolated. A simple graph with n vertex has degrees $\in [0, n-1]$. If all vertices have a $n-1$-degree, the associated graph is complete.

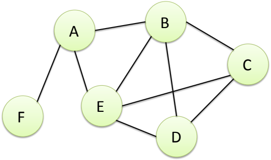
Characterization matrices

Let $G = \{V, E\}$ be a graph with n vertices. We call :



Characterization matrices

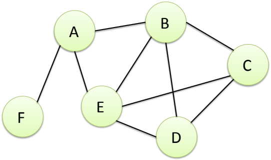
Let $G = \{V, E\}$ be a graph with n vertices. We call :



- The adjency matrix A, of dimension $(n \times n)$, the binary matrix where a_{ij} is 1 if one edge connects the i^{th} et j^{th} vertices of V.

Characterization matrices

Let $G = \{V, E\}$ be a graph with n vertices. We call :

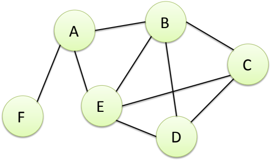


- The adjency matrix A, of dimension $(n \times n)$, the binary matrix where a_{ij} is 1 if one edge connects the i^{th} et j^{th} vertices of V. Ex :

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

Characterization matrices

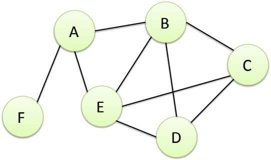
Let $G = \{V, E\}$ be a graph with n vertices. We call :



- The degree matrix D, of dimension $(n \times n)$, a diagonal matrix where d_{ii} is the number of incident edges of i^{th} vertex.

Characterization matrices

Let $G = \{V, E\}$ be a graph with n vertices. We call :

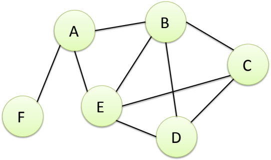


- The degree matrix D, of dimension $(n \times n)$, a diagonal matrix where d_{ii} is the number of incident edges of i^{th} vertex. Ex :

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

Characterization matrices

Let $G = \{V, E\}$ be a graph with n vertices. We call :



- The Laplacian matrix L, of dimension $(n \times n)$, such that $L = D - A$ (bilan)

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

Normalized Laplacian matrix

Définition : Normalized Laplacian matrix

$$L' = D^{-1/2} L D^{-1/2} = I - D^{-1/2} A D^{-1/2}$$

We get it from the graph

$$L' = \begin{cases} 1 & \text{on the diagonal if } \; deg(v_i) \neq 0 \\ -\dfrac{1}{\sqrt{d(i)d(j)}} & \text{if } i \neq j \; \text{ and } \; v_i \; \text{ and } \; v_j \; \text{ are adjacent} \\ 0 & \text{else} \end{cases}$$

Normalized Laplacian matrix

Définition : Normalized Laplacian matrix

$$L' = D^{-1/2} L D^{-1/2} = I - D^{-1/2} A D^{-1/2}$$

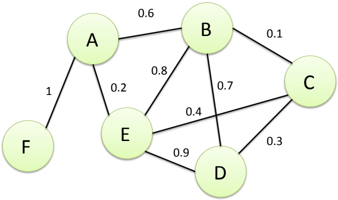
We get it from the graph

$$L' = \begin{cases} 1 & \text{on the diagonal if } \; deg(v_i) \neq 0 \\ -\dfrac{1}{\sqrt{d(i)d(j)}} & \text{if } i \neq j \; \text{ and } \; v_i \; \text{ and } \; v_j \; \text{ are adjacent} \\ 0 & \text{else} \end{cases}$$

In general

- Each edge connects 2 vertexes through an incidence function $\gamma(x_i, x_j)$

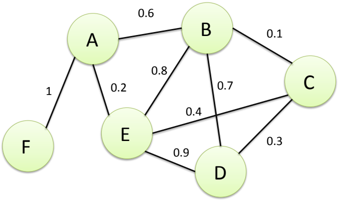
  $\Longrightarrow$ A graph is defined by $G = \{V, E, \gamma\}$ but we often omit γ and note $G = \{V, E\}$



In general

- Each edge connects 2 vertexes through an incidence function $\gamma(x_i, x_j)$

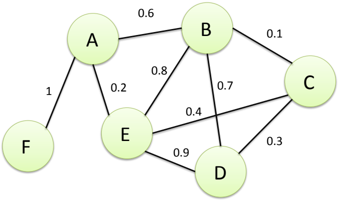
  $\Longrightarrow$ A graph is defined by $G = \{V, E, \gamma\}$ but we often omit γ and note $G = \{V, E\}$



In general

- Each edge connects 2 vertexes through an incidence function $\gamma(x_i, x_j)$

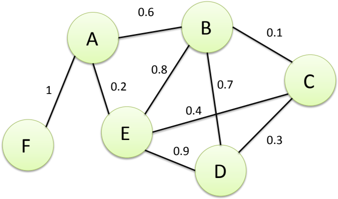
  $\Longrightarrow$ A graph is defined by $G = \{V, E, \gamma\}$ but we often omit γ and note $G = \{V, E\}$



- The adjency matric W is now the value a_{ij} that links the i^{th} and j^{th} vertexes of V.

In general

- Each edge connects 2 vertexes through an incidence function $\gamma(x_i, x_j)$
  $\Longrightarrow$ A graph is defined by $G = \{V, E, \gamma\}$ but we often omit γ and note $G = \{V, E\}$



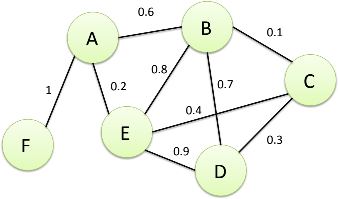
- The adjency matric W is now the value a_{ij} that links the i^{th} and j^{th} vertexes of V. Ex :

$$W = \begin{bmatrix} 0 & 0.6 & 0 & 0 & 0.2 & 1 \\ 0.6 & 0 & 0.1 & 0.7 & 0.8 & 0 \\ 0 & 0.1 & 0 & 0.3 & 0.4 & 0 \\ 0 & 0.7 & 0.3 & 0 & 0.9 & 0 \\ 0.2 & 0.8 & 0.4 & 0.9 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

In general

- Each edge connects 2 vertexes through an incidence function $\gamma(x_i, x_j)$

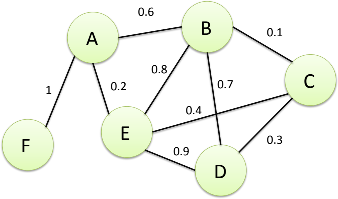
  $\Longrightarrow$ A graph is defined by $G = \{V, E, \gamma\}$ but we often omit γ and note $G = \{V, E\}$



- For degree matrix D : each element is the sum of weights

In general

- Each edge connects 2 vertexes through an incidence function $\gamma(x_i, x_j)$
  $\Longrightarrow$ A graph is defined by $G = \{V, E, \gamma\}$ but we often omit γ and note $G = \{V, E\}$

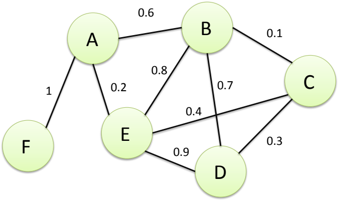


- For degree matrix D : each element is the sum of weights. Ex :

$$D = \begin{bmatrix} 1.8 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2.2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.8 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1.9 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2.3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

In general

- Each edge connects 2 vertexes through an incidence function $\gamma(x_i, x_j)$
  $\Longrightarrow$ A graph is defined by $G = \{V, E, \gamma\}$ but we often omit γ and note $G = \{V, E\}$



- The Laplacian matrix L, of dimension $(n \times n)$, is such that $L = D - W$

$$L = \begin{bmatrix} 1.8 & -0.6 & 0 & 0 & -0.2 & -1 \\ -0.6 & 2.2 & -0.1 & -0.7 & -0.8 & 0 \\ 0 & -0.1 & 0.8 & -0.3 & -0.4 & 0 \\ 0 & -0.7 & -0.3 & 1.9 & -0.9 & 0 \\ -0.2 & -0.8 & -0.4 & -0.9 & 2.3 & 0 \\ -1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Spectral theory on graphes

Main idea : analyze the spectra of associated matrices

$\Longrightarrow$ If one vertex is permuted, associated matrix change but not their spectra

With adjacency matrix, we get informations on :

- Cycles, regularity, ...

With Laplacian matrix, we get informations on :

- number of trees inside the graph, shapes (with eigenvalues, ...)

Many Applications

- Network traffic, space optimization, microbiology, image processing, data analysis, ...

Spectral theory on graphes

Main idea : analyze the spectra of associated matrices

$\Longrightarrow$ If one vertex is permuted, associated matrix change but not their spectra

With adjacency matrix, we get informations on :

- Cycles, regularity, ...

With Laplacian matrix, we get informations on :

- number of trees inside the graph, shapes (with eigenvalues, ...)

Many Applications

- Network traffic, space optimization, microbiology, image processing, data analysis, ...

Spectral theory on graphes

Main idea : analyze the spectra of associated matrices

$\Longrightarrow$ If one vertex is permuted, associated matrix change but not their spectra

With adjacency matrix, we get informations on :

- Cycles, regularity, ...

With Laplacian matrix, we get informations on :

- number of trees inside the graph, shapes (with eigenvalues, ...)

Many Applications

- Network traffic, space optimization, microbiology, image processing, data analysis, ...

Spectral theory on graphes

Main idea : analyze the spectra of associated matrices

$\Longrightarrow$ If one vertex is permuted, associated matrix change but not their spectra

With adjacency matrix, we get informations on :

- Cycles, regularity, ...

With Laplacian matrix, we get informations on :

- number of trees inside the graph, shapes (with eigenvalues, ...)

Many Applications

- Network traffic, space optimization, microbiology, image processing, data analysis, ...

Outline

Spectral clustering

Reminder : main families

- Centroid : create several clusters and evaluate their quality depending on some centroids (k-means, k-medoids, PAM, ...)
- Hierarchical : group in a hierarchical way data (AGNES, DIANA, ...)
- Density : rely on the adequacy of data with respect to a certain density (DBSCAN)
- Model-based : one model for each cluster
- Spectral : based on a graph-representation of data

Spectral clustering

Reminder : main families

- Centroid : create several clusters and evaluate their quality depending on some centroids (k-means, k-medoids, PAM, ...)
- Hierarchical : group in a hierarchical way data (AGNES, DIANA, ...)
- Density : rely on the adequacy of data with respect to a certain density (DBSCAN)
- Model-based : one model for each cluster
- Spectral : based on a graph-representation of data

Spectral clustering

Reminder : main families

- Centroid : create several clusters and evaluate their quality depending on some centroids (k-means, k-medoids, PAM, ...)
- Hierarchical : group in a hierarchical way data (AGNES, DIANA, ...)
- Density : rely on the adequacy of data with respect to a certain density (DBSCAN)
- Model-based : one model for each cluster
- Spectral : based on a graph-representation of data

$\Longrightarrow$ Graph partition

Spectral clustering : generalties

Interesting methods since

- Simple & fast implementation

Spectral clustering : generalties

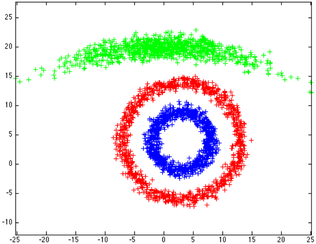
Interesting methods since

- Simple & fast implementation
- In general efficient in cpu-time

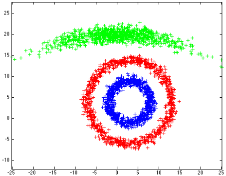
Spectral clustering : generalties

Interesting methods since

- Simple & fast implementation
- In general efficient in cpu-time
- No prior on clusters



General principle 1 – Graph construction (1/2)



From the dataset, construct a graph

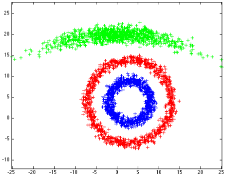
Several possibilities :

1. Complete graph : all vertices connected, all edges $\neq 0$
2. r-neighbour : vertices are connected only to "close" vertexes $< r$

   $\Longrightarrow$ Radius r to be defined
3. k-nn graph : vertices are connected to k nearest neighbours

   $\Longrightarrow$ Number k to be defined
4. One can combine r-neighbour and knn

General principle 1 – Graph construction (1/2)

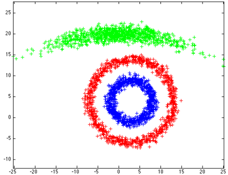


From the dataset, construct a graph

Several possibilities :

1. Complete graph : all vertices connected, all edges $\neq 0$
2. r-neighbour : vertices are connected only to "close" vertexes $< r$
   $\Longrightarrow$ Radius r to be defined
3. k-nn graph : vertices are connected to k nearest neighbours
   $\Longrightarrow$ Number k to be defined
4. One can combine r-neighbour and knn

General principle 1 – Graph construction (1/2)



From the dataset, construct a graph

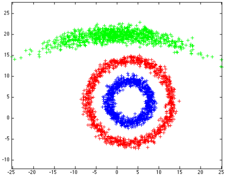
Several possibilities :

1. Complete graph : all vertices connected, all edges $\neq 0$
2. r-neighbour : vertices are connected only to "close" vertexes $< r$

   $\Longrightarrow$ Radius r to be defined
3. k-nn graph : vertices are connected to k nearest neighbours

   $\Longrightarrow$ Number k to be defined
4. One can combine r-neighbour and knn

General principle 1 – Graph construction (1/2)



From the dataset, construct a graph

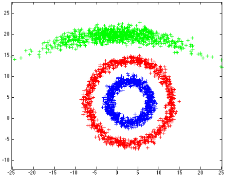
Several possibilities :

1. Complete graph : all vertices connected, all edges $\neq 0$
2. r-neighbour : vertices are connected only to "close" vertexes $< r$

   $\Longrightarrow$ Radius r to be defined
3. k-nn graph : vertices are connected to k nearest neighbours

   $\Longrightarrow$ Number k to be defined
4. One can combine r-neighbour and knn

General principle 1 – Graph construction (1/2)



From the dataset, construct a graph

Several possibilities :

1. Complete graph : all vertices connected, all edges $\neq 0$
2. r-neighbour : vertices are connected only to "close" vertexes $< r$
   $\Longrightarrow$ Radius r to be defined
3. k-nn graph : vertices are connected to k nearest neighbours
   $\Longrightarrow$ Number k to be defined
4. One can combine r-neighbour and knn

General principle 1 – Graph partition (2/2)

From data matrix X (of dimension $N \times P$: N points of dimension P), how to affect edge values ?

- Definition of a similarity matrix. This can be :
  - Covariance (between each pair of points x_i, x_j)
  - Cosinus of the angle formed by x_i, x_j
  - ...
  - Gaussian kernal

$$K(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)$$

Important note : for spatial data (images, ...), one can add spatial coordinates (x, y) to the data

General principle 1 – Graph partition (2/2)

From data matrix X (of dimension $N \times P$: N points of dimension P), how to affect edge values ?

- Definition of a similarity matrix. This can be :
  - Covariance (between each pair of points x_i, x_j)
  - Cosinus of the angle formed by x_i, x_j
  - ...
  - Gaussian kernal

$$K(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)$$

Important note : for spatial data (images, ...), one can add spatial coordinates (x, y) to the data

General principle 1 – Graph partition (2/2)

From data matrix X (of dimension $N \times P$: N points of dimension P), how to affect edge values ?

- Definition of a similarity matrix. This can be :
  - Covariance (between each pair of points x_i, x_j)
  - Cosinus of the angle formed by x_i, x_j
  - ...
  - Gaussian kernal

$$K(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)$$

Important note : for spatial data (images, ...), one can add spatial coordinates (x, y) to the data

General principle 1 – Graph partition (2/2)

From data matrix X (of dimension $N \times P$: N points of dimension P), how to affect edge values ?

- Definition of a similarity matrix. This can be :
  - Covariance (between each pair of points x_i, x_j)
  - Cosinus of the angle formed by x_i, x_j
  - ...
  - Gaussian kernal

$$K(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)$$

Important note : for spatial data (images, ...), one can add spatial coordinates (x, y) to the data

General principle 1 – Graph partition (2/2)

We aim at partitioning graph $G = (V, E, \gamma)$ into two disjoints ensembles A and B such as

$$\begin{cases} A \cup B = V \\ A \cap B = \emptyset \end{cases}$$

Several possibilities : MinCut, RatioCut, NCut, MinMaxCut, ... We define :

- The sum of connections between two clusters (A et B) :

$$Cut(A, B) = \sum_{x_i \in A, x_j \in B} \gamma(x_i, x_j) = \frac{1}{4} q^T (D - W) q$$

  with $q = \{-1, 1\}$
- The sum of connections inside one clustersr A :

$$Cut(A) = \sum_{x_i \in A, x_j \in A} \gamma(x_i, x_j)$$

- The total weights embed in cluster A :

$$Vol(A) = \sum_{x_i \in A} D(x_i) \text{ avec } D(x_i) \text{ (from the degree matrix)}$$

General principle 1 – Graph partition (2/2)

We aim at partitioning graph $G = (V, E, \gamma)$ into two disjoints ensembles A and B such as

$$\begin{cases} A \cup B = V \\ A \cap B = \emptyset \end{cases}$$

Several possibilities : MinCut, RatioCut, NCut, MinMaxCut, ... We define :

- The sum of connections between two clusters (A et B) :

$$Cut(A, B) = \sum_{x_i \in A, x_j \in B} \gamma(x_i, x_j) = \frac{1}{4} q^T (D - W) q$$

  with $q = \{-1, 1\}$
- The sum of connections inside one clustersr A :

$$Cut(A) = \sum_{x_i \in A, x_j \in A} \gamma(x_i, x_j)$$

- The total weights embed in cluster A :

$$Vol(A) = \sum_{x_i \in A} D(x_i) \text{ avec } D(x_i) \text{ (from the degree matrix)}$$

General principle 1 – Graph partition (2/2)

We aim at partitioning graph $G = (V, E, \gamma)$ into two disjoints ensembles A and B such as

$$\begin{cases} A \cup B = V \\ A \cap B = \emptyset \end{cases}$$

Several possibilities : MinCut, RatioCut, NCut, MinMaxCut, ... We define :

- The sum of connections between two clusters (A et B) :

$$Cut(A, B) = \sum_{x_i \in A, x_j \in B} \gamma(x_i, x_j) = \frac{1}{4} q^T (D - W) q$$

  with $q = \{-1, 1\}$

- The sum of connections inside one clustersr A :

$$Cut(A) = \sum_{x_i \in A, x_j \in A} \gamma(x_i, x_j)$$

- The total weights embed in cluster A :

$$Vol(A) = \sum_{x_i \in A} D(x_i) \text{ avec } D(x_i) \text{ (from the degree matrix)}$$

General principle 1 – Graph partition (2/2)

We aim at partitioning graph $G = (V, E, \gamma)$ into two disjoints ensembles A and B such as

$$\begin{cases} A \cup B = V \\ A \cap B = \emptyset \end{cases}$$

Several possibilities : MinCut, RatioCut, NCut, MinMaxCut, ... We define :

- The sum of connections between two clusters (A et B) :

$$Cut(A, B) = \sum_{x_i \in A, x_j \in B} \gamma(x_i, x_j) = \frac{1}{4} q^T (D - W) q$$

  with $q = \{-1, 1\}$

- The sum of connections inside one clustersr A :

$$Cut(A) = \sum_{x_i \in A, x_j \in A} \gamma(x_i, x_j)$$

- The total weights embed in cluster A :

$$Vol(A) = \sum_{x_i \in A} D(x_i) \text{ avec } D(x_i) \text{ (from the degree matrix)}$$

General principle 1 – Graph partition (2/2)

We aim at partitioning graph $G = (V, E, \gamma)$ into two disjoints ensembles A and B such as

$$\begin{cases} A \cup B = V \\ A \cap B = \emptyset \end{cases}$$

Several possibilities : MinCut, RatioCut, NCut, MinMaxCut, ... We define :

- The sum of connections between two clusters (A et B) :

$$Cut(A,B) = \sum_{x_i \in A, x_j \in B} \gamma(x_i, x_j) = \frac{1}{4} q^T (D - W) q$$

  with $q = \{-1, 1\}$
- The sum of connections inside one clustersr A :

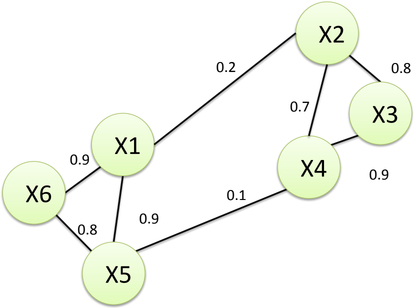
$$Cut(A) = \sum_{x_i \in A, x_j \in A} \gamma(x_i, x_j)$$

- The total weights embed in cluster A :

$$Vol(A) = \sum_{x_i \in A} D(x_i) \text{ avec } D(x_i) \text{ (from the degree matrix)}$$

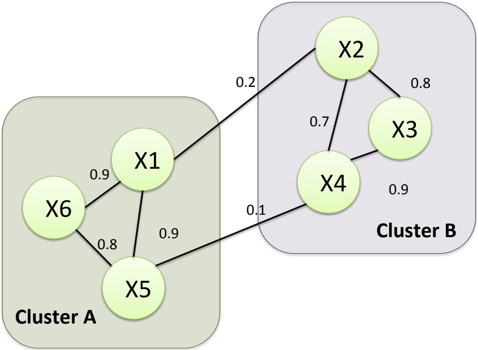
General principle 1 – Graph partition (2/2)

Example



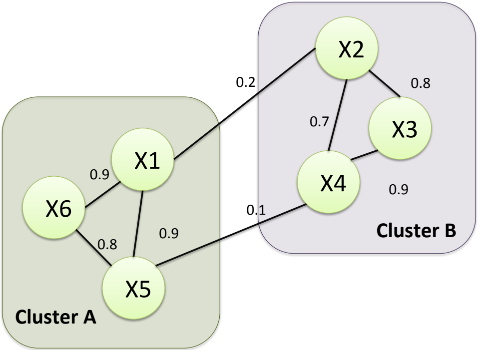
General principle 1 – Graph partition (2/2)

Example



General principle 1 – Graph partition (2/2)

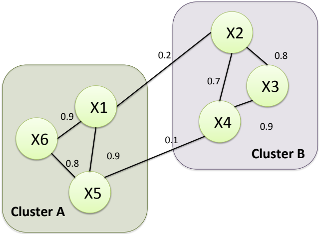
Example



Compute parameters associated with each cluster

General principle 1 – Graph partition (2/2)

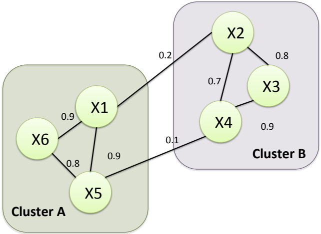
Example



- $Cut(A, B) = \sum_{x_i \in A, x_j \in B} \gamma(x_i, x_j)$
- $Cut(A) = \sum_{x_i \in A, x_j \in A} \gamma(x_i, x_j)$
- $Cut(B) = \sum_{x_i \in B, x_j \in B} \gamma(x_i, x_j)$
- $Vol(A) = \sum_{x_i \in A} D(x_i)$
- $Vol(B) = \sum_{x_i \in B} D(x_i)$

General principle 1 – Graph partition (2/2)

Example



- $Cut(A,B) = \sum_{x_i \in A, x_j \in B} \gamma(x_i, x_j) = 0.3$
- $Cut(A) = \sum_{x_i \in A, x_j \in A} \gamma(x_i, x_j) = 2.6$
- $Cut(B) = \sum_{x_i \in B, x_j \in B} \gamma(x_i, x_j) = 2.4$
- $Vol(A) = \sum_{x_i \in A} D(x_i) = 5.5$
- $Vol(B) = \sum_{x_i \in B} D(x_i) = 5.1$

Outline

General principle 2 – Graph cut

- Principles of MinCut (minimal cut) method is to search for two clusters A and B that have minimal connection sums ($Cut(A,B)$)
- Minimize a cost function :

$$\{A^*, B^*\} = \arg\min_{A,B} Cut(A,B)$$

- We define

$$q = \begin{cases} 1 \text{ if } i \in A \\ -1 \text{ if } i \in B \end{cases}$$

- this is equivalent to find q^* that minimize :

$$q^* = \arg\min_q q^T(D-W)q$$

General principle 2 – Graph cut

- Principles of MinCut (minimal cut) method is to search for two clusters A and B that have minimal connection sums $(Cut(A,B))$
- Minimize a cost function :

$$\{A^*, B^*\} = \arg\min_{A,B} Cut(A,B)$$

- We define

$$q = \begin{cases} 1 \text{ if } i \in A \\ -1 \text{ if } i \in B \end{cases}$$

- this is equivalent to find q^* that minimize :

$$q^* = \arg\min_q q^T (D - W) q$$

General principle 2 – Graph cut

- Principles of MinCut (minimal cut) method is to search for two clusters A and B that have minimal connection sums ($Cut(A, B)$)
- Minimize a cost function :

$$\{A^*, B^*\} = \arg\min_{A,B} Cut(A, B)$$

- We define

$$q = \begin{cases} 1 \text{ if } i \in A \\ -1 \text{ if } i \in B \end{cases}$$

- this is equivalent to find q^* that minimize :

$$q^* = \arg\min_{q} q^T (D - W) q$$

General principle 2 – Graph cut

- Principles of MinCut (minimal cut) method is to search for two clusters A and B that have minimal connection sums ($Cut(A,B)$)
- Minimize a cost function :

$$\{A^*, B^*\} = \arg\min_{A,B} Cut(A,B)$$

- We define

$$q = \begin{cases} 1 \text{ if } i \in A \\ -1 \text{ if } i \in B \end{cases}$$

- this is equivalent to find q^* that minimize :

$$q^* = \arg\min_{q} q^T (D - W) q$$

General principle 2 – Graph cut

- Reminder We search for q^* that minimize :

$$q^* = \arg\min_q q^T(D-W)q$$

with the constraint $q_i = \{1, -1\}$

- This is a discrete optimization problem
- A continuous formulation gives a trivial solution $(q=1)$ but not satisfactory (all points in the same cluster)
- In practice : eigen vector/value problem :

$$(D-W)q = \lambda q$$

General principle 2 – Graph cut

- Reminder We search for q^* that minimize :

$$q^* = \arg\min_q q^T(D - W)q$$

with the constraint $q_i = \{1, -1\}$

- This is a discrete optimization problem
- A continuous formulation gives a trivial solution ($q = 1$) but not satisfactory (all points in the same cluster)
- In practice : eigen vector/value problem :

$$(D - W)q = \lambda q$$

General principle 2 – Graph cut

- Reminder We search for q^* that minimize :

$$q^* = \arg\min_q q^T (D - W) q$$

with the constraint $q_i = \{1, -1\}$

- This is a discrete optimization problem
- A continuous formulation gives a trivial solution $(q = 1)$ but not satisfactory (all points in the same cluster)
- In practice : eigen vector/value problem :

$$(D - W) q = \lambda q$$

General principle 2 – Graph cut

- Reminder We search for q^* that minimize :

$$q^* = \arg\min_q q^T (D - W) q$$

with the constraint $q_i = \{1, -1\}$

- This is a discrete optimization problem
- A continuous formulation gives a trivial solution $(q = 1)$ but not satisfactory (all points in the same cluster)
- In practice : eigen vector/value problem :

$$(D - W) q = \lambda q$$

Analysis of Laplacian matrix

- Matrix $L = D - W$ symmetric (if non oriented graph)
- Immediate eigen vector : $q_1 = (1, 1, 1, 1, 1, 1, 1, ..., 1)^T$ (cf demo)

  $\Longrightarrow$ Not satisfactory since all points in the same cluster. Associated eigen value : $\lambda_1 = 0$

- Other eigen vectors give other possible partitions. The second eigen vector (in growing order), named Fiedler-vector, is used for the graph partition

  $\Longrightarrow$ The smaller λ_2, the more interesting the partition is

  $\Longrightarrow$ The assignation can be done with the sign of the eigenvector

cf example on matlab

$\Longrightarrow$ No guarantee to have homogeneous clusters

Analysis of Laplacian matrix

- Matrix $L = D - W$ symmetric (if non oriented graph)
- Immediate eigen vector : $q_1 = (1, 1, 1, 1, 1, 1, 1, ..., 1)^T$ (cf demo)

  $\Longrightarrow$ Not satisfactory since all points in the same cluster. Associated eigen value : $\lambda_1 = 0$
- Other eigen vectors give other possible partitions. The second eigen vector (in growing order), named Fiedler-vector, is used for the graph partition

  $\Longrightarrow$ The smaller λ_2, the more interesting the partition is

  $\Longrightarrow$ The assignation can be done with the sign of the eigenvector

cf example on matlab

$\Longrightarrow$ No guarantee to have homogeneous clusters

Analysis of Laplacian matrix

- Matrix $L = D - W$ symmetric (if non oriented graph)
- Immediate eigen vector : $q_1 = (1, 1, 1, 1, 1, 1, 1, ..., 1)^T$ (cf demo)

  $\Longrightarrow$ Not satisfactory since all points in the same cluster. Associated eigen value : $\lambda_1 = 0$
- Other eigen vectors give other possible partitions. The second eigen vector (in growing order), named Fiedler-vector, is used for the graph partition

  $\Longrightarrow$ The smaller λ_2, the more interesting the partition is

  $\Longrightarrow$ The assignation can be done with the sign of the eigenvector

cf example on matlab

$\Longrightarrow$ No guarantee to have homogeneous clusters

Analysis of Laplacian matrix

- Matrix $L = D - W$ symmetric (if non oriented graph)
- Immediate eigen vector : $q_1 = (1, 1, 1, 1, 1, 1, 1, ..., 1)^T$ (cf demo)

  $\Longrightarrow$ Not satisfactory since all points in the same cluster. Associated eigen value : $\lambda_1 = 0$
- Other eigen vectors give other possible partitions. The second eigen vector (in growing order), named Fiedler-vector, is used for the graph partition

  $\Longrightarrow$ The smaller λ_2, the more interesting the partition is

  $\Longrightarrow$ The assignation can be done with the sign of the eigenvector

cf example on matlab

$\Longrightarrow$ No guarantee to have homogeneous clusters

Analysis of Laplacian matrix

- Matrix $L = D - W$ symmetric (if non oriented graph)
- Immediate eigen vector : $q_1 = (1, 1, 1, 1, 1, 1, 1, ..., 1)^T$ (cf demo)

  $\Longrightarrow$ Not satisfactory since all points in the same cluster. Associated eigen value : $\lambda_1 = 0$
- Other eigen vectors give other possible partitions. The second eigen vector (in growing order), named Fiedler-vector, is used for the graph partition

  $\Longrightarrow$ The smaller λ_2, the more interesting the partition is

  $\Longrightarrow$ The assignation can be done with the sign of the eigenvector

cf example on matlab

$\Longrightarrow$ No guarantee to have homogeneous clusters

Analysis of Laplacian matrix

- Matrix $L = D - W$ symmetric (if non oriented graph)
- Immediate eigen vector : $q_1 = (1, 1, 1, 1, 1, 1, 1, ..., 1)^T$ (cf demo)

  $\Longrightarrow$ Not satisfactory since all points in the same cluster. Associated eigen value : $\lambda_1 = 0$
- Other eigen vectors give other possible partitions. The second eigen vector (in growing order), named Fiedler-vector, is used for the graph partition

  $\Longrightarrow$ The smaller λ_2, the more interesting the partition is

  $\Longrightarrow$ The assignation can be done with the sign of the eigenvector

cf example on matlab

$\Longrightarrow$ No guarantee to have homogeneous clusters

Analysis of Laplacian matrix

- Matrix $L = D - W$ symmetric (if non oriented graph)
- Immediate eigen vector : $q_1 = (1, 1, 1, 1, 1, 1, 1, ..., 1)^T$ (cf demo)

  $\Longrightarrow$ Not satisfactory since all points in the same cluster. Associated eigen value : $\lambda_1 = 0$
- Other eigen vectors give other possible partitions. The second eigen vector (in growing order), named Fiedler-vector, is used for the graph partition

  $\Longrightarrow$ The smaller λ_2, the more interesting the partition is

  $\Longrightarrow$ The assignation can be done with the sign of the eigenvector

cf example on matlab

$\Longrightarrow$ No guarantee to have homogeneous clusters

Minimal cut : coins

- No guarantee to have homogeneous clusters
  - $\Longrightarrow$ One can use the second eigenvector and cut it
- Multi-class partitioning
  - $\Longrightarrow$ One can use other eigenvectors
  - $\Longrightarrow$ One can use recursive cuts
  - $\Longrightarrow$ Packages : Scikit learn, Chaco, PaToH, Party, METIS
- Packages
  - $\Longrightarrow$ Scikit learn, Chaco, PaToH, Party, METIS, ...

Minimal cut : coins

- No guarantee to have homogeneous clusters
  - $\Longrightarrow$ One can use the second eigenvector and cut it
- Multi-class partitioning
  - $\Longrightarrow$ One can use other eigenvectors
  - $\Longrightarrow$ One can use recursive cuts
  - $\Longrightarrow$ Packages : Scikit learn, Chaco, PaToH, Party, METIS
- Packages
  - $\Longrightarrow$ Scikit learn, Chaco, PaToH, Party, METIS, ...

Minimal cut : coins

- No guarantee to have homogeneous clusters
  - $\Longrightarrow$ One can use the second eigenvector and cut it
- Multi-class partitioning
  - $\Longrightarrow$ One can use other eigenvectors
  - $\Longrightarrow$ One can use recursive cuts
  - $\Longrightarrow$ Packages : Scikit learn, Chaco, PaToH, Party, METIS
- Packages
  - $\Longrightarrow$ Scikit learn, Chaco, PaToH, Party, METIS, ...

Plan

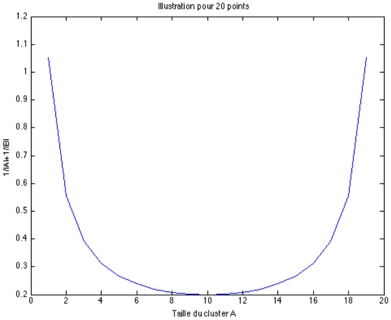
RatioCut method

- Idea : impose in the cost function homogeneous clusters

RatioCut method

- Idea : impose in the cost function homogeneous clusters
- Cost function :

$$\{A^*, B^*\} = \arg\min_{A,B} Cut(A,B)\left(\frac{1}{|A|} + \frac{1}{|B|}\right)^2$$



RatioCut method

- Idea : impose in the cost function homogeneous clusters
- One can replace cardinals by volumes. Cost-function :

$$\{A^*, B^*\} = \arg\min_{A,B} Cut(A, B) \left(\frac{1}{Vol(A)} + \frac{1}{Vol(B)} \right)^2$$

RatioCut method

- Similar system based on eigen vectors
- Until now, the cost function is composed of
  - Information about distances between clusters
  - Information about size of clusters
  - Need also internal informations inside clusters

RatioCut method

- Similar system based on eigen vectors
- Until now, the cost function is composed of
  - Information about distances between clusters
  - Information about size of clusters
  - Need also internal informations inside clusters

RatioCut method

- Similar system based on eigen vectors
- Until now, the cost function is composed of
  - Information about distances between clusters
  - Information about size of clusters
  - Need also internal informations inside clusters

Plan

MinMaxCut method

- Idea : compromise between clusters of homogeneous sizes, "far-away" from each others and hemegeneous
  1. Constraint 1 : minimize inter-connections :
  $$\{A^*, B^*\} = \arg\min_{A,B} Cut(A, B)$$
  2. Constraint 2 : maximize intra-connections :
  $$\max_{A} Cut(A) \text{ et } \max_{B} Cut(B)$$
- joint minimization of
$$\begin{cases} \{A^*, B^*\} = \arg\min_{A,B} Cut(A, B) \left(\frac{1}{Vol(A)} + \frac{1}{Vol(B)} \right) \\ \{A^*, B^*\} = \arg\min_{A,B} Cut(A, B) \left(\frac{1}{Cut(A)} + \frac{1}{Cut(B)} \right) \end{cases}$$
- This also leads to an eigen vector/value problem on Laplacian matrix L, with different constraints on q

MinMaxCut method

- Idea : compromise between clusters of homogeneous sizes, "far-away" from each others and hemegeneous
  1. Constraint 1 : minimize inter-connections :

$$\{A^*, B^*\} = \arg\min_{A,B} Cut(A,B)$$

  2. Constraint 2 : maximize intra-connections :

$$\max_{A} Cut(A) \text{ et } \max_{B} Cut(B)$$

- joint minimization of

$$\begin{cases} \{A^*, B^*\} = \arg\min_{A,B} Cut(A,B)\left(\dfrac{1}{Vol(A)} + \dfrac{1}{Vol(B)}\right) \\ \{A^*, B^*\} = \arg\min_{A,B} Cut(A,B)\left(\dfrac{1}{Cut(A)} + \dfrac{1}{Cut(B)}\right) \end{cases}$$

- This also leads to an eigen vector/value problem on Laplacian matrix L, with different constraints on q

MinMaxCut method

- Idea : compromise between clusters of homogeneous sizes, “far-away” from each others and hemegeneous
  1. Constraint 1 : minimize inter-connections :

$$\{A^*, B^*\} = \arg\min_{A,B} Cut(A,B)$$

  2. Constraint 2 : maximize intra-connections :

$$\max_A Cut(A) \text{ et } \max_B Cut(B)$$

- joint minimization of

$$\begin{cases} \{A^*, B^*\} = \arg\min_{A,B} Cut(A,B)\left(\frac{1}{Vol(A)} + \frac{1}{Vol(B)}\right) \\ \{A^*, B^*\} = \arg\min_{A,B} Cut(A,B)\left(\frac{1}{Cut(A)} + \frac{1}{Cut(B)}\right) \end{cases}$$

- This also leads to an eigen vector/value problem on Laplacian matrix L, with different constraints on q

MinMaxCut method

- Idea : compromise between clusters of homogeneous sizes, "far-away" from each others and hemegeneous
  1. Constraint 1 : minimize inter-connections :
  $$\{A^*, B^*\} = \arg\min_{A,B} Cut(A, B)$$
  2. Constraint 2 : maximize intra-connections :
  $$\max_A Cut(A) \text{ et } \max_B Cut(B)$$
- joint minimization of
$$\begin{cases} \{A^*, B^*\} = \arg\min_{A,B} Cut(A, B)\left(\frac{1}{Vol(A)} + \frac{1}{Vol(B)}\right) \\ \{A^*, B^*\} = \arg\min_{A,B} Cut(A, B)\left(\frac{1}{Cut(A)} + \frac{1}{Cut(B)}\right) \end{cases}$$
- This also leads to an eigen vector/value problem on Laplacian matrix L, with different constraints on q

MinMaxCut method

- Idea : compromise between clusters of homogeneous sizes, “far-away” from each others and hemegeneous
  1. Constraint 1 : minimize inter-connections :

$$\{A^*, B^*\} = \arg\min_{A,B} Cut(A,B)$$

  2. Constraint 2 : maximize intra-connections :

$$\max_{A} Cut(A) \text{ et } \max_{B} Cut(B)$$

- joint minimization of

$$\begin{cases} \{A^*, B^*\} = \arg\min_{A,B} Cut(A,B)\left(\dfrac{1}{Vol(A)} + \dfrac{1}{Vol(B)}\right) \\ \{A^*, B^*\} = \arg\min_{A,B} Cut(A,B)\left(\dfrac{1}{Cut(A)} + \dfrac{1}{Cut(B)}\right) \end{cases}$$

- This also leads to an eigen vector/value problem on Laplacian matrix L, with different constraints on q

Generalties

On methods

- If clusters are well separables, the 3 approaches are almost identical
- Otherwise, MinMaxCut is the most efficient

Extension to k classes

- MinCut :

$$A_I^* = \arg\min_{A_i} \sum_{i=1}^{k} Cut(A_i, \overline{A_i})$$

- RatioCut :

$$A_I^* = \arg\min_{A_i} \sum_{i=1}^{k} \frac{Cut(A_i, \overline{A_i})}{|A_i|}$$

- MinMaxCut

$$\begin{cases} A_I^* = \arg\min_{A_i} \sum_{i=1}^{k} \frac{Cut(A_i, \overline{A_i})}{Vol(A_i)} \\ A_i^* = \arg\min_{A_i} \sum_{i=1}^{k} \frac{Cut(A_i, \overline{A_i})}{Cut(A_i)} \end{cases}$$

Generalties

On methods

- If clusters are well separables, the 3 approaches are almost identical
- Otherwise, MinMaxCut is the most efficient

Extension to k classes

- MinCut :

$$A_I^* = \arg\min_{A_i} \sum_{i=1}^{k} Cut(A_i, \overline{A_i})$$

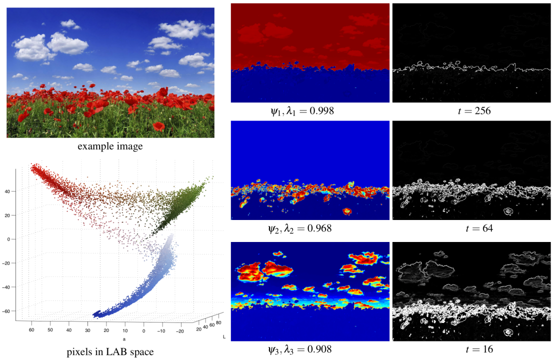
- RatioCut :

$$A_I^* = \arg\min_{A_i} \sum_{i=1}^{k} \frac{Cut(A_i, \overline{A_i})}{|A_i|}$$

- MinMaxCut

$$\begin{cases} A_I^* = \arg\min_{A_i} \sum_{i=1}^{k} \frac{Cut(A_i, \overline{A_i})}{Vol(A_i)} \\ A_i^* = \arg\min_{A_i} \sum_{i=1}^{k} \frac{Cut(A_i, \overline{A_i})}{Cut(A_i)} \end{cases}$$

Example in image processing



Source : *Diffusion Maps, Spectral Clustering and Eigenfunctions of Fokker-Planck Operators Boaz Nadler, Stephane Lafon, Ronald R. Coifman*

Outline

Clustering with Laplacian matrix

We can directly use the Laplacian matrix without cut

1. Step 1 : pre-processing
   - Graph construction and computation of adjacency, degree and Laplacian matrices
2. Step 2 : spectral analysis
   - Computation of eigenvectors and eigenvalues of Laplacian matrix
   - Change of variables in the eigen-vector space
3. Step 3 : clustering
   - Apply a simple clustering technique (k-means, ...) on this representation