SPECTRAL CLUSTERING

Zitao Liu,Salim Malakouti

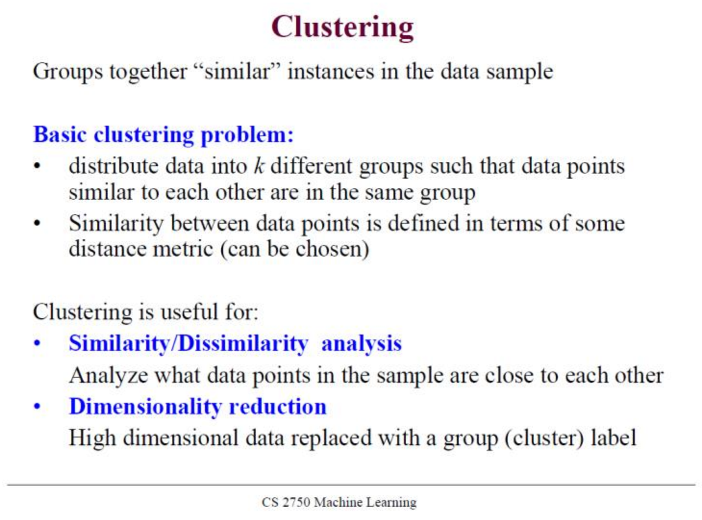
Agenda

- Brief Clustering Review
- Similarity Graph
- Graph Partitioning
- Graph Laplacian Properties
- Spectral Clustering Algorithm
- Graph Cut Point of View
- Random Walk Point of View
- Practical Details

Agenda

- Brief Clustering Review
- Similarity Graph
- Graph Partitioning
- Graph Laplacian Properties
- Spectral Clustering Algorithm
- Graph Cut Point of View
- Random Walk Point of View
- Practical Details

Clustering Review



Clustering

Groups together "similar" instances in the data sample

Basic clustering problem:

- distribute data into *k* different groups such that data points similar to each other are in the same group
- Similarity between data points is defined in terms of some distance metric (can be chosen)

Clustering is useful for:

- **Similarity/Dissimilarity analysis**
  Analyze what data points in the sample are close to each other
- **Dimensionality reduction**
  High dimensional data replaced with a group (cluster) label

CS 2750 Machine Learning

K-means Clustering

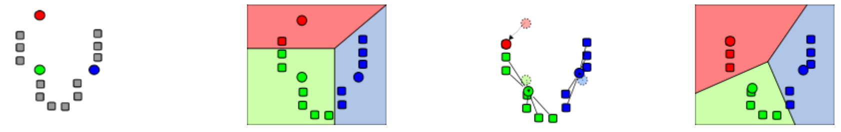
Given a set of observations ($\mathbf{x}_1$, $\mathbf{x}_2$, …, $\mathbf{x}_n$), where each observation is a *d*-dimensional real vector, *k*-means clustering aims to partition the *n* observations into *k* sets ($k \leq n$) $\mathbf{S} = \{S_1, S_2, …, S_k\}$ so as to minimize the within-cluster sum of squares (WCSS):

$$\underset{S}{\arg\min} \sum_{i=1}^{k} \sum_{x_j \in S_i} \left\| x_j - u_i \right\|^2$$

where $\boldsymbol{\mu}_i$ is the mean of points in S_i.

K-means Clustering

Standard Algorithm



1) k initial "means" (in this case k=3) are randomly selected from the data set.

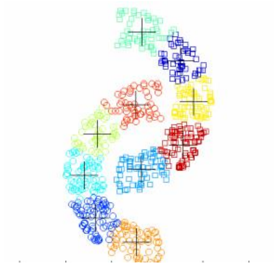
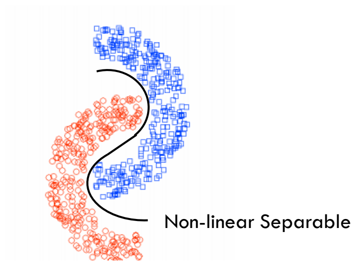
2) k clusters are created by associating every observation with the nearest mean.

3) The centroid of each of the k clusters becomes the new means.

4) Steps 2 and 3 are repeated until convergence has been reached.

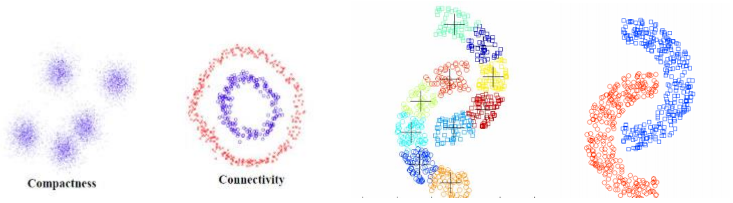
K-means Shortcomings

- K-means
  - Criteria: Compactness of clusters $$\arg\min_{S} \sum_{i=1}^{k} \sum_{x_j \in S_i} \left\| x_j - u_i \right\|^2$$
  - Major drawback:
    - It can't separate clusters that **are non-linearly separable**

K-means Shortcomings

- Spectral Clustering
  - Criteria: Connectivity



  - Instead of linear similarities use non-linear distance/similarity of data or geodesic distance of points
    - **Similarity Graph**

Agenda

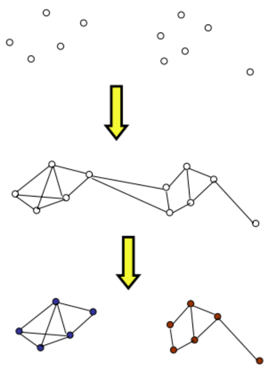
General Idea

First – graph representation of data
(largely, application dependent)

Second– graph partitioning:

Partition Properties:

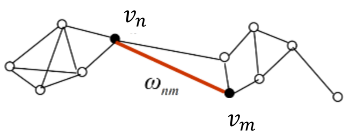
- Weakly connections in between components
- Strongly connections within components



Graph Notation

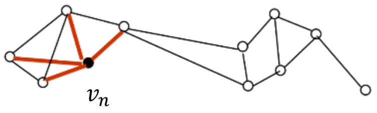
G=(*V*,*E*) :

- Vertex set $V = \{v_1, \dots, v_n\}$
- Weighted adjacency matrix:



$$\boldsymbol{W = (w_{ij})\ i, j = 1, \dots, n} \quad \boldsymbol{w_{ij} \geq 0} \text{ and } \boldsymbol{w_{ij} = w_{ji}}$$

- Degree $d_i = \sum_{j=1}^{n} w_{ij}$

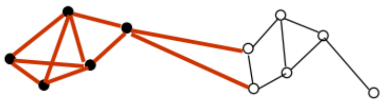


- **Degree matrix D** Diagonal matrix with the degrees $d_1, \dots, d_n$ on the diagonal.

Graph Notation

G=(*V*,*E*) :

- A is a **subset** of V
  - $\bar{A} = V \backslash A$
  - Indicator Vector $\mathbb{1}_A = (f_1, \dots, f_n)' \in \mathbb{R}^n \quad f_i \in \{0,1\}$
  - "Size" of a subset $A \subset V$

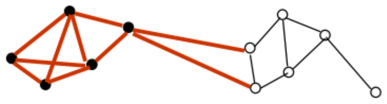


$$|A| := \text{the number of vertices in } A$$
$$vol(A) := \sum_{i \in A} d_i$$

  - Similarity/Connectivity of two subsets A and B

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

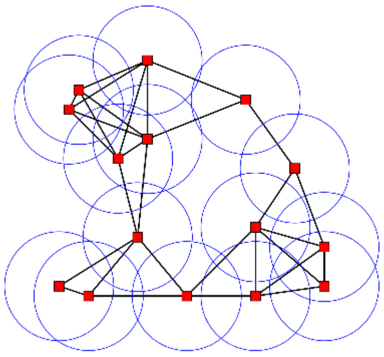
Graph Notation



- **Connected Subset** A subset A of a graph is connected if any two vertices in A can be joined by a path such that all intermediate points also lie in A.
- **Connected Component** it is connected and if there are no connections between vertices in A and $\bar{A}$.
- **Partition:** A set of nonempty sets $A_1, \ldots, A_k$ form a partition of the graph if $A_i \cap A_j = \emptyset$ and $A_1 \cup \cdots \cup A_k = V$.

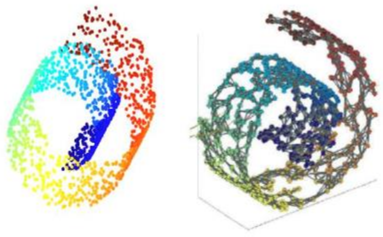
Similarity Graph Construction

- ***Goal:*** *Model local neighborhood relationships between data points*
- ε-neighborhood graph
  - Connect all points whose pairwise distances are smaller than ε
    - Considered unweighted graph



Similarity Graph Construction

- k-nearest neighbor graph
  - Connect vertex v_i with vertex v_j if v_j is among the k-nearest neighbors of v_i.
  - Is directed graph
    - Knn graph: obtained by loosing directions
    - Mutual knn graph: connect only if both vertices have the other in their knn



Similarity Graph Construction

- Fully connected graph
  - Connect all points with positive similarity with each other
  - Not modeling local neighborhoods
    - We have to use similarity function has to do it (Gaussian Similarity Function)

$$W_{ij} = e^{\frac{|x_i - x_j|^2}{2\sigma^2}}$$

- σ controls size of neighborhood

All of introduced graphs are regularly used in spectral clustering!

Agenda

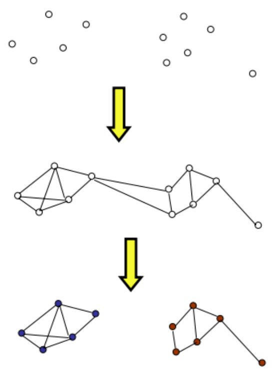
Graph Partitioning

First – graph representation of data
(largely, application dependent)

Second– graph partitioning:

Partition Properties:

- Weakly connections in between components
- Strongly connections within components



Graph Cut

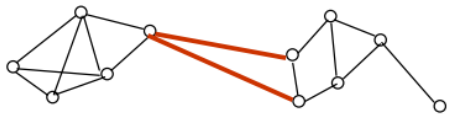
$G=(V,E)$:

- For two not necessarily disjoint set $A, B \subset V$, we define

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

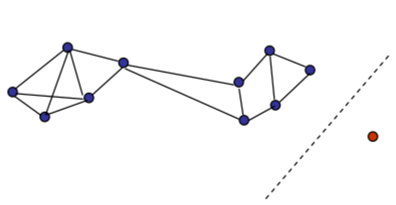
- Minicut: choosing a partition $A_1, A_2, \ldots, A_K$ which minimizes

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

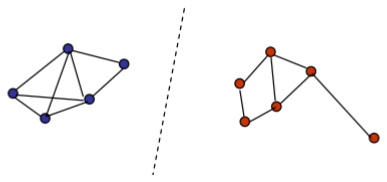


Graph Cut Shortcoming

- Sensitive to outliers



What we get



What we want

Solutions

- "Size" of a subset $A \subset V$ could be defined:

$$|A| := \text{the number of vertices in } A$$

Solutions:

$$vol(A) := \sum_{i \in A} d_i$$

- RatioCut(Hagen and Kahng, 1992)

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

- Ncut(Shi and Malik, 2000)

$$Ncut(A_1, \dots, A_k) := \frac{1}{2} \sum_{i=1}^{k} \frac{W(A_i, \overline{A_i})}{vol(A_i)} = \sum_{i=1}^{k} \frac{cut(A_i, \overline{A_i})}{vol(A_i)}$$

Problem

NP hard!!!: can't be solved in polynomial time

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

$$\min Ncut(A_1, \dots, A_k) := \frac{1}{2} \sum_{i=1}^{k} \frac{W(A_i, \overline{A_i})}{vol(A_i)} = \sum_{i=1}^{k} \frac{cut(A_i, \overline{A_i})}{vol(A_i)}$$

Solution: Approximation

Solving Simple Case

- Approximation RatioCut for **k=2**
  - Our goal is to solve the optimization problem:

$$\min_{A\subset V} RatioCut(A,\bar{A})$$

- We can also write:

$$\min_{A\subset V} RatioCut(A,\bar{A}) = \min_{A\subset V}\left(\frac{1}{|V|}|V|RatioCut(A,\bar{A})\right)$$

$$\boldsymbol{\min_{A\subset V}(|V|RatioCut(A,\bar{A}))}$$

Solving Simple Case

We know:

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

Hence, we can rewrite the objective function as:

$$|V|RatioCut(A,\bar{A}) = |V|\left(\frac{cut(A,\bar{A})}{|A|}+\frac{cut(A,\bar{A})}{|\bar{A}|}\right)$$

$$= cut(A,\bar{A})\left(\frac{|A|+|\bar{A}|}{|A|}+\frac{|A|+|\bar{A}|}{|\bar{A}|}\right)$$

$$= cut(A,\bar{A})\left(\frac{|\bar{A}|}{|A|}+\frac{|A|}{|\bar{A}|}+2\right)$$

Since we know:

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

$$= \frac{1}{2}\sum_{i\epsilon A, j\epsilon\bar{A}} w_{ij}\left(\sqrt{\frac{|\bar{A}|}{|A|}}+\sqrt{\frac{|A|}{|\bar{A}|}}\right)^2 + \sum_{i\epsilon\bar{A}, j\epsilon A} w_{ij}\left(-\sqrt{\frac{|\bar{A}|}{|A|}}-\sqrt{\frac{|A|}{|\bar{A}|}}\right)^2$$

Solving Simple Case

$$= \frac{1}{2} \sum_{i\epsilon A, j\epsilon \bar{A}} w_{ij}\left(\sqrt{\frac{|\bar{A}|}{|A|}} + \sqrt{\frac{|A|}{|\bar{A}|}}\right)^2 + \sum_{i\epsilon \bar{A}, j\epsilon A} w_{ij}\left(-\sqrt{\frac{|\bar{A}|}{|A|}} - \sqrt{\frac{|A|}{|\bar{A}|}}\right)^2$$

If we define f_i as:

$$f_i = \begin{cases} \sqrt{|\bar{A}|/|A|}, & \text{if } v_i \in A \\ -\sqrt{|A|/|\bar{A}|}, & \text{if } v_i \in \bar{A} \end{cases}$$

$$\sum_{j=1}^{n} \underbrace{\sum_{i=1}^{n} w_{ij}}_{d_j} f_j^2 = \sum_{j=1}^{n} d_j f_j^2$$

We can rewrite the objective function as:

$$= \frac{1}{2} \sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2$$

$$= \frac{1}{2}\left(\sum_{i,j=1}^{n} w_{ij} f_i^2 - 2 \sum_{i,j=1}^{n} w_{ij} f_i f_j + \sum_{i,j=1}^{n} w_{ij} f_j^2 \right)$$

$$= \frac{1}{2}\left(\sum_{i=1}^{n} d_i f_i^2 - 2 \sum_{i,j=1}^{n} f_i f_j w_{ij} + \sum_{j=1}^{n} d_j f_j^2 \right)$$

$$= \sum_{i=1}^{n} d_i f_i^2 - \sum_{i,j=1}^{n} f_i f_j w_{ij}$$

$$= f'Df - f'Wf$$

$$= f'Lf$$

D= degree matrix (n*n)

$W(n*n) = (w_{ij})\ i,j = 1,\dots,n \quad w_{ij} \geq 0$

L = D - W

Solving Simple Case

$$\min_{A \subset V} f'Lf \quad \text{if} \quad f_i = \begin{cases} \sqrt{|\bar{A}|/|A|}, & \text{if } v_i \in A \\ -\sqrt{|A|/|\bar{A}|}, & \text{if } v_i \in \bar{A} \end{cases}$$

We can say that f is orthogonal to the constant vector $\mathbb{1}$

$$f\mathbb{1} = \sum_{i=1}^{n} f_i = \sum_{i\epsilon A} \sqrt{\frac{|\bar{A}|}{|A|}} - \sum_{i\epsilon \bar{A}} \sqrt{\frac{|A|}{|\bar{A}|}} = |A|\sqrt{\frac{|\bar{A}|}{|A|}} - |\bar{A}|\sqrt{\frac{|A|}{|\bar{A}|}} = 0$$

f satisfies:

$$||f||^2 = \sum_{i=1}^{n} f_i^2 = |A|\frac{|\bar{A}|}{|A|} + |\bar{A}|\frac{|A|}{|\bar{A}|} = \text{n}$$

New Objective

$$\min_{A\subset V} f'Lf \ \ subject\ to\ \ f\perp\mathbb{1}, f_i\ as\ defined\ in\ (1), ||f|| = \sqrt{n}$$

$$f_i = \begin{cases} \sqrt{|\bar{A}|/|A|}, & \text{if } v_i \in A \\ -\sqrt{|A|/|\bar{A}|}, & \text{if } v_i \in \bar{A} \end{cases}$$

- A discrete optimization problem in which f is only allowed to have two particular values.
- There are no efficient algorithms for solving optimization problems with discrete values

NP hard!!! => RELXATION

Relaxed Objective

- Relaxation:
  - We remove constraints to create a relax version of the objective function which can be solved in polynomial time

$$\min_{A\subset V} f'Lf \ ,subject\ to\ f\perp\mathbb{1}\ \ ||f|| = \sqrt{n}$$

- The Relaxed version can be solved efficiency to get f that minimizes the objective function

Creating the Partition

- Approximation RatioCut for k=2

f is the vector in $\mathbb{R}$ that minimizes objective function

re-convert

Use the sign as indicator function

$$\begin{cases} v_i \in A & \text{if } f_i \geq 0 \\ v_i \in \overline{A} & \text{if } f_i < 0. \end{cases}$$

Only works for k = 2

f_i as points in R and do K-means

$$\begin{cases} v_i \in A & \text{if } f_i \in C \\ v_i \in \overline{A} & \text{if } f_i \in \overline{C}. \end{cases}$$

More General, works for any k

Eigenvectors and Eigenvalues

An **eigenvector** of a square matrix A is a non-zero vector v that, when the matrix multiplies v, yields the same as when some scalar multiplies v, the scalar multiplier often being denoted by λ. That is:

$$Av = \lambda v \quad => v^{-1}Av$$

λ is also called the eigenvalue of A

We also know that eigenvectors are orthogonal to each other

$$\min_{A \subset V} f'Lf \text{, subject to } f \perp \mathbb{1} \quad ||f|| = \sqrt{n}$$

f is the second eigenvector of L with eigenvalue equal to

$$\lambda = \text{RatioCut}(A, \bar{A})$$

$\mathbb{1}$ is the first one with eigenvalue 0.

Idea of Spectral Clustering

- We found K=2 eigenvectors of matrix L
  - L is Laplacian Matrix of Laplacian Graph
- We took data to a new dimensional system where data has linear structure
- We used K-Means to find best partition (clustering)

The fundamental idea of Spectral Clustering

Agenda

- Brief Clustering Review
- Similarity Graph
- Graph Laplacian
- Spectral Clustering Algorithm
- Graph Cut Point of View
- Random Walk Point of View
- Practical Details

Graph Laplacians

- Unnormalized Graph Laplacian

D= degree matrix where $d_i = \sum_{j=1}^{n} w_{ij}$

$$L = D - W \longrightarrow W = (w_{ij})\ i,j = 1,\dots,n \quad w_{ij} \geq 0$$

Properties of L

Proposition 1 The matrix L satisfies the following properties:

1. For every $f \in \mathbb{R}^n$ we have

$$f'Lf = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2$$

Graph Laplacians

- Unnormalized Graph Laplacian

$$L = D - W$$

$$f'Lf = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2$$

1. L is symmetric and positive semi-definite.
   - L is symmetric since W and D are
   - $f'Lf \geq=$ therfore positive semidefinite
2. The smallest eigenvalue of L is 0, the corresponding eigenvector is the constant one vector $\mathbb{1}$
3. L has n non-negative, real-valued eigenvalues $0 = \lambda_i \leq \lambda_2 \leq \cdots \leq \lambda_n$.

Graph Laplacians

- Unnormalized Graph Laplacian

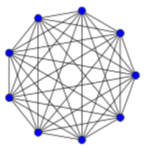
$$L = D - W$$

Proposition 2 (Number of connected components and the spectrum of *L*) Let G be an undirected graph with non-negative weights. The multiplicity k of the eigenvalue 0 of *L* equals the number of connected components $A_1, \ldots, A_k$ in the graph. The eigenspace of eigenvalue 0 is spanned by the indicator vectors $\mathbb{1}_{A_1}, \ldots, \mathbb{1}_{A_k}$ of those components.

Proof:

When *k* = 1

$Lf = 0f \quad (\lambda{=}0) \quad => \quad 0 = f'Lf = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2$



Only happens if $f_i = f_j$

Graph Laplacians

- Unnormalized Graph Laplacian

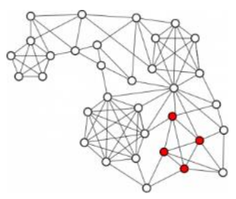
$$L = D - W$$

Proposition 2 (Number of connected components and the spectrum of *L*) Let G be an undirected graph with non-negative weights. The multiplicity k of the eigenvalue 0 of *L* equals the number of connected components $A_1, \ldots, A_k$ in the graph. The eigenspace of eigenvalue 0 is spanned by the indicator vectors $\mathbb{1}_{A_1}, \ldots, \mathbb{1}_{A_k}$ of those components.

Proof:

When $k > 1$, *L* can be written in a block form. the spectrum of *L* is given by the union of the spectra of L_i, and the corresponding eigenvectors of *L* are the eigenvectors of L_i, filled with 0 at the positions of the other blocks.

$$L = \begin{pmatrix} L_1 & & & \\ & L_2 & & \\ & & \ddots & \\ & & & L_k \end{pmatrix}$$



Graph Laplacians

- Normalized Graph Laplacian

$$L_{sym} := D^{-\frac{1}{2}} L D^{-\frac{1}{2}} = I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$$

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

We denote the first matrix by L_{sym} as it is a symmetric matrix, and the second one by L_{rw} as it is closely related to a random walk.

Graph Laplacians

Proposition 3 (Properties of L_{sym} and L_{rw}) The normalized Laplacians statisfy the following properties:

1. For every $f \in \mathbb{R}^n$ we have

$$f' L_{sym} f = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij}^2 \left(\frac{f_i}{\sqrt{d_i}} - \frac{f_j}{\sqrt{d_j}} \right)^2$$

$$L_{sym} := D^{-\frac{1}{2}} L D^{-\frac{1}{2}} = I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$$

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

2. *λ is an eigenvalue of L_{rw} with eigenvector u if and only if λ is an eigenvalue of L_{sym} with eigenvector $w = D^{1/2} u$.*

3. λ is an eigenvalue of L_{rw} with eigenvector u if and only if λ and u solve the generalized eigen problem $\boldsymbol{Lu} = \boldsymbol{\lambda D u}$. (~~$DD^{-1}$~~$Lu = \lambda Du$)

Graph Laplacians

$$f' L_{sym} f = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}^2 \left(\frac{f_i}{\sqrt{d_i}} - \frac{f_j}{\sqrt{d_j}}\right)^2$$

$$L_{sym} := D^{-\frac{1}{2}} L D^{-\frac{1}{2}} = I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$$
$$L_{rw} := D^{-1} L = I - D^{-1} W$$

4. 0 is an eigenvalue of L_{rw} with the constant one vector $\mathbb{1}$ as eigenvector. 0 is an eigenvalue of L_{sym} with eigenvector $D^{1/2}\mathbb{1}$. (former is obvious, latter derives from (2))
5. L_{sym} and L_{rw} are positive semi-definite and have *n* non-negative real-valued eigenvalues $0 = \lambda_i \leq \lambda_2 \leq \cdots \leq \lambda_n$.

Graph Laplacians

- Normalized Graph Laplacian

$$L_{sym} := D^{-\frac{1}{2}} L D^{-\frac{1}{2}} = I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$$
$$L_{rw} := D^{-1} L = I - D^{-1} W$$

Proposition 4 (Number of connected components and spectra of L_{sym} and L_{rw})
Let G be an undirected graph with non-negative weights. Then the multiplicity k of the eigenvalue 0 of both L_{sym} and L_{rw} equals the number of connected components $A_1, \ldots, A_k$ in the graph. For L_{rw} the eigenspace of 0 is spanned by the indicator vectors $\mathbb{1}_{A_i}$ of those components. For L_{sym}, the eigenspace of 0 is spanned by the vectors $D^{1/2}\mathbb{1}_{A_i}$.

Proof. The proof is analogous to the one of Proposition 2, using Proposition 3.

Agenda

- Brief Clustering Review
- Similarity Graph
- Graph Laplacian
- Spectral Clustering Algorithm
- Graph Cut Point of View
- Random Walk Point of View
- Practical Details

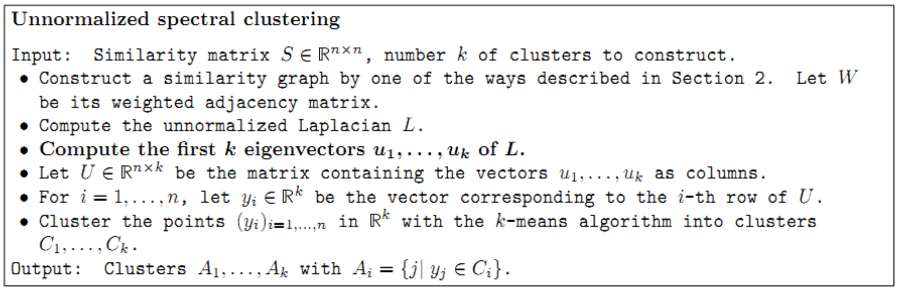
Algorithm

- Main trick is to change the representation of the abstract data points x_i to points $y_i \in \Re^k$
  - Unnormalized Spectral Clustering
  - Normalized Spectral Clustering 1
  - Normalized Spectral Clustering 2

Algorithm

- Unnormalized Graph Laplacian

$$L = D - W$$



Unnormalized spectral clustering

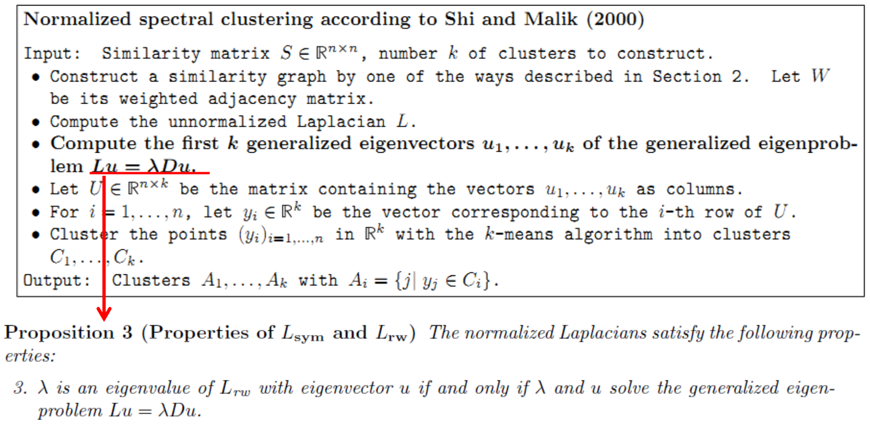
Input: Similarity matrix $S \in \mathbb{R}^{n \times n}$, number k of clusters to construct.
- Construct a similarity graph by one of the ways described in Section 2. Let W be its weighted adjacency matrix.
- Compute the unnormalized Laplacian L.
- **Compute the first k eigenvectors $u_1, \dots, u_k$ of L.**
- Let $U \in \mathbb{R}^{n \times k}$ be the matrix containing the vectors $u_1, \dots, u_k$ as columns.
- For $i = 1, \dots, n$, let $y_i \in \mathbb{R}^k$ be the vector corresponding to the i-th row of U.
- Cluster the points $(y_i)_{i=1,\dots,n}$ in $\mathbb{R}^k$ with the k-means algorithm into clusters $C_1, \dots, C_k$.

Output: Clusters $A_1, \dots, A_k$ with $A_i = \{j \mid y_j \in C_i\}$.

Algorithm

- Normalized Graph Laplacian

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



Normalized spectral clustering according to Shi and Malik (2000)

Input: Similarity matrix $S \in \mathbb{R}^{n \times n}$, number k of clusters to construct.
- Construct a similarity graph by one of the ways described in Section 2. Let W be its weighted adjacency matrix.
- Compute the unnormalized Laplacian L.
- **Compute the first k generalized eigenvectors $u_1, \dots, u_k$ of the generalized eigenproblem $\underline{Lu = \lambda Du}$.**
- Let $U \in \mathbb{R}^{n \times k}$ be the matrix containing the vectors $u_1, \dots, u_k$ as columns.
- For $i = 1, \dots, n$, let $y_i \in \mathbb{R}^k$ be the vector corresponding to the i-th row of U.
- Cluster the points $(y_i)_{i=1,\dots,n}$ in $\mathbb{R}^k$ with the k-means algorithm into clusters $C_1, \dots, C_k$.

Output: Clusters $A_1, \dots, A_k$ with $A_i = \{j \mid y_j \in C_i\}$.

Proposition 3 (Properties of L_{sym} and L_{rw}) *The normalized Laplacians satisfy the following properties:*

3. λ is an eigenvalue of L_{rw} with eigenvector u if and only if λ and u solve the generalized eigenproblem $Lu = \lambda Du$.

Algorithm

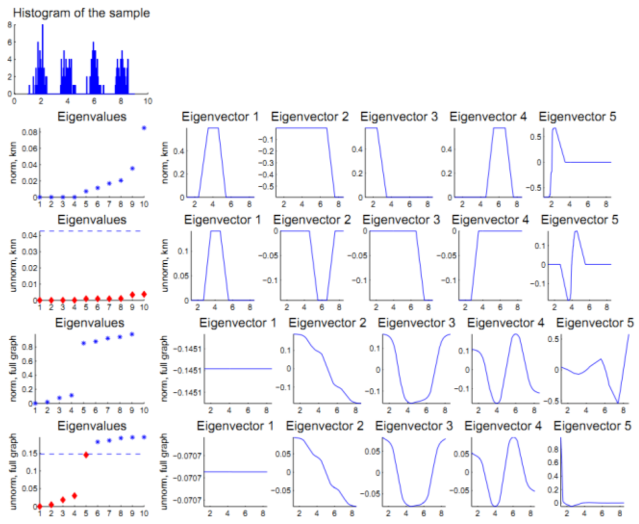
- Normalized Graph Laplacian

$$L_{sym} := D^{-\frac{1}{2}} L D^{-\frac{1}{2}} = I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$$

Normalized spectral clustering according to Ng, Jordan, and Weiss (2002)

```
Input: Similarity matrix $S \in \mathbb{R}^{n \times n}$, number $k$ of clusters to construct.
  • Construct a similarity graph by one of the ways described in Section 2. Let $W$
    be its weighted adjacency matrix.
  • Compute the normalized Laplacian $L_{\text{sym}}$.
  • Compute the first $k$ eigenvectors $u_1, \dots, u_k$ of $L_{\text{sym}}$.
  • Let $U \in \mathbb{R}^{n \times k}$ be the matrix containing the vectors $u_1, \dots, u_k$ as columns.
  • Form the matrix $T \in \mathbb{R}^{n \times k}$ from $U$ by normalizing the rows to norm 1,
    that is set $t_{ij} = u_{ij} / (\sum_k u_{ik}^2)^{1/2}$.
  • For $i = 1, \dots, n$, let $y_i \in \mathbb{R}^k$ be the vector corresponding to the $i$-th row of $T$.
  • Cluster the points $(y_i)_{i=1,\dots,n}$ with the $k$-means algorithm into clusters $C_1, \dots, C_k$.
Output: Clusters $A_1, \dots, A_k$ with $A_i = \{j \mid y_j \in C_i\}$.
```

Algorithm



Agenda

- Brief Clustering Review
- Similarity Graph
- Graph Laplacian
- Spectral Clustering Algorithm
- Graph Cut Point of View
- Random Walk Point of View
- Practical Details

Graph Cut

- Approximation RatioCut for k=2

$$\min_{A \subset V} \operatorname{RatioCut}(A, \overline{A}).$$

$$f_i = \begin{cases} \sqrt{|\bar{A}|/|A|}, & \text{if } v_i \in A \\ -\sqrt{|A|/|\bar{A}|}, & \text{if } v_i \in \bar{A} \end{cases}$$

$$\Updownarrow$$

$$\min_{A \subset V} f'Lf \ \textit{subject to} \ f \perp \mathbb{1}, f_i \text{ as defined in (1)}, ||f|| = \sqrt{n}$$

$$\Downarrow$$

Relaxation !!!

$\perp$

$$\min_{A \subset V} f'Lf \ , \textit{subject to } f \perp \mathbb{1} \ \ ||f|| = \sqrt{n}$$

$$\Downarrow$$

Rayleigh-Ritz Theorem

f is the eigenvector corresponding to the second smallest eigenvalue of *L* (the smallest eigenvalue of *L* is 0 with eigenvector $\mathbb{1}$)

Graph Cut Point of View

- It can be shown that:
  - Unnormalized Spectral Clustering algorithm is approximation of Ratio Cut Graph Partitioning
  - Normalized Spectral Clustering algorithms are approximations of Ncut Graph Partition

Abstract

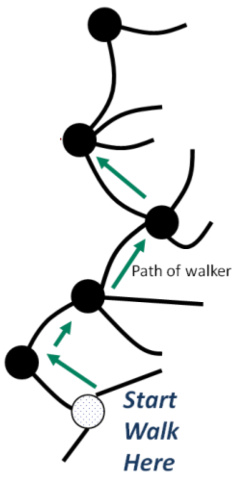
- Spectral Clustering gives an approximate solution to the solution of graph cut problem
- There is no guarantee on getting on the quality of the solution
- In general, efficient algorithms to approximate balanced graphs cut to a constant factor is also a NP-hard problem

Agenda

- Brief Clustering Review
- Similarity Graph
- Graph Laplacian
- Spectral Clustering Algorithm
- Graph Cut Point of View
- Random Walk Point of View
- Perturbation Theory Point of View
- Practical Details

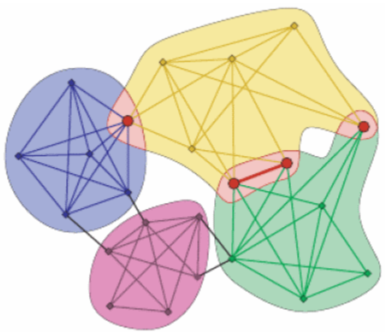
Random Walk

- A random walk on a graph is a stochastic process which randomly jumps from vertex to vertex.



Random Walk in Similarity Graph

- Random walk stays long within the same cluster and seldom jumps between clusters.
- A balanced partition with a low cut will also have the property that the random walk does not have many opportunities to jump between clusters.



Random walk

- **Transition probability** p_{ij} of jumping from v_i to v_j

$$p_{ij} = w_{ij}/d_i$$

- The **transition matrix** $P = (p_{ij})$ i,j = 1,…,n of random walk is defined by

$$P = D^{-1}W$$

- If the graph is connected and non-bipartite, the random walk always processes a unique stationary distribution $\pi = (\pi_1, \dots, \pi_n)'$, where

$$\pi_i = d_i/vol(V)$$

$$d_i = \sum_{j=1}^{n} w_{ij} \qquad vol(V) := \sum_{i \in V} d_i$$

Random walk

- Relationship between L_{rw} and *P*.

$$L_{rw} = I - P \qquad L_{rw} := D^{-1}L = I - D^{-1}W$$

- λ is an eigenvalue of L_{rw} with eigenvector *u* if and only if $1 - \lambda$ is an eigenvalue of *P* with eigenvector *u*.
- The largest eigenvectors of *P* and the smallest eigenvectors of L_{rw} can be used to describe cluster properties of the graph.

Random walk

- Random walks and Ncut

Proposition 5 (Ncut **via transition probabilities**) Let *G* be connected and non bi-partite. Assume that we run the random walk $(X_t)_{t \in N}$ starting with X_0 in the stationary distribution π. For disjoint subsets $A, B \subset V$, denote by $P(B|A) := P(X_1 \in B \,|X_0 \in A)$. Then:

$$Ncut(A, \bar{A}) = P(\bar{A}|A) + P(A|\bar{A}).$$

It tells us that when minimizing Ncut, we actually look for a cut through the graph such that **A random walk seldom transitions from A to $\bar{A}$ and vice versa.**

Random walk

- What is **commute distance**

The commute distance (resistance distance) c_{ij} between two vertices v_i and v_j is the **expected time it takes the random walk to travel from vertex v_i to vertex v_j and back.**

The commute distance between two vertices **decrease** if there are **many different short ways** to get from vertex v_i to vertex v_j.

- instead of only shortest path

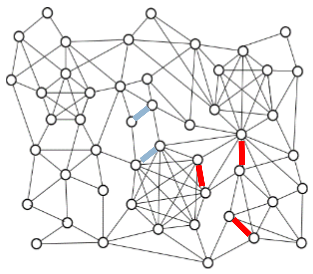
Random walk

The commute distance between two vertices **decrease** if there are **many different short ways** to get from vertex v_i to vertex v_j.

- instead of only shortest path

Points which are **connected by a short path** in the graph and **lie in the same high-density region** of the graph are considered closer to each other than points which are connected by a short path but lie in different high-density regions of the graph.

Well-suited for Clustering



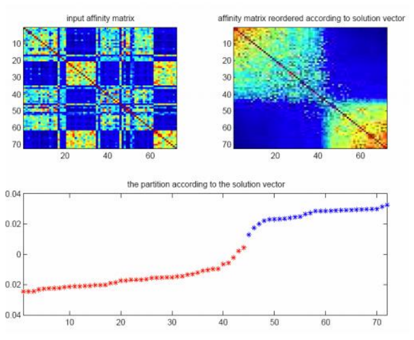
Agenda

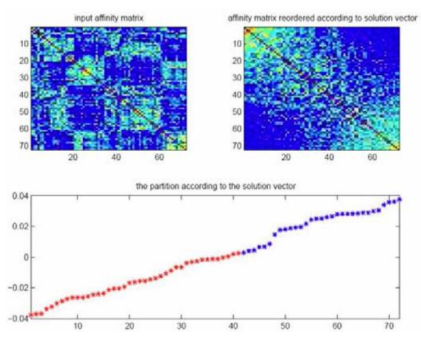
- Brief Clustering Review
- Similarity Graph
- Graph Laplacian
- Spectral Clustering Algorithm
- Graph Cut Point of View
- Random Walk Point of View
- Practical Details

Similarity function

- Constructing the similarity graph
  1. **Similarity Function Itself**
     - Make sure that points which are considered to be "very similar" by the similarity function are also closely related in the application the data comes from.
     - Global-range behavior of similarity function is not important
     - Domain specific

Similarity function

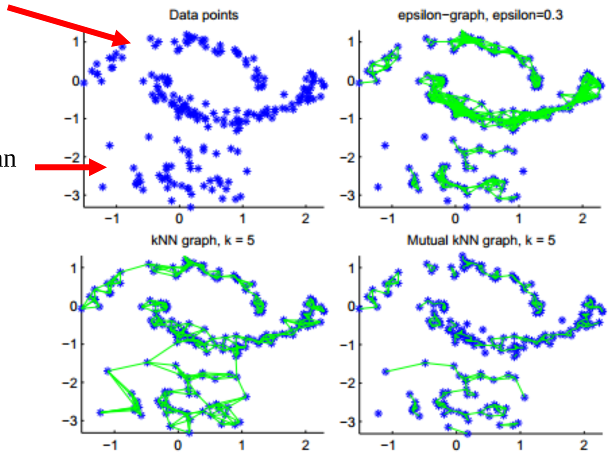




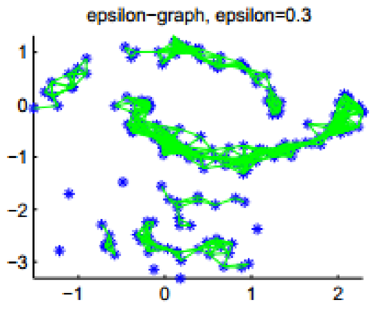
Similarity graph type

Moons

Gaussian

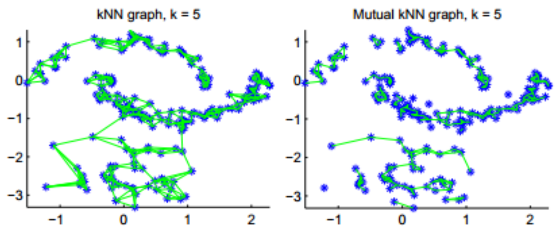


Similarity graph type



- Difficult to choose ε
  - Points on the moon are strongly connected
  - Points on Gaussian are barely connected
- Problem of "data on different scales"

Similarity graph type



- K-nearest neighbor graph
  - Can handle "data on different scales"
  - Can break into disconnected components if there are densely connected regions
- Mutual k-nearest neighbor
  - Can act in presence of data at different densities but doesn't mix them together
  - Well suited

Parameters of similarity graphs

- Significance of having connected similarity graph
  - In general, if the **similarity graph contains more connected** components than the number of clusters we ask the algorithm to detect, then spectral clustering will trivially return connected components as clusters
  - Unless one is perfectly sure that those connected components are the correct clusters, one should make sure that **the similarity graph is connected**, or only consists of **"few" connected components** and very **few or no isolated vertices.**
  - To prevent **outliers** to affect result
  - **Set parameters small enough to address local structure of data while insuring connectivity**

Parameters of similarity graphs

1. **Parameters of Similarity Graph**(k or ε)
   1. **KNN:**
      1. k in order of log(n);
   2. **Mutual KNN:**
      1. Has much fewer edges than knn
         1. k significantly larger than standard KNN;
      2. Maintain the aforementioned advantage of not connected regions of different density
         1. Keep it small enough
      3. Unfortunately, no general heuristics

Parameters of similarity graphs

1. **Parameters of Similarity Graph**(k or ε)
   1. **ε-neighborhood graph**:
      1. longest edge of MST;
      2. Order of $(\log(n)/n)^{d}$
      3. A significantly large $\boldsymbol{\varepsilon}$ will be selected when (too large):
         1. Data contains outlies
         2. Several tight components very far from each other
   2. **fully connected graph:**
      1. Select σ in a way that similarity is "not too small not too large"
      2. Select σ in order of the mean distance of a point to its k-th nearest neighbor. Or choose $k = \varepsilon$.

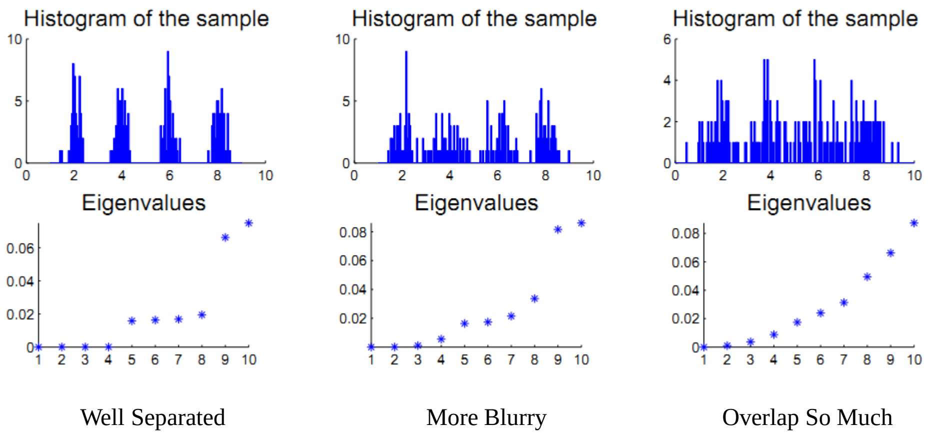
Computing eigenvectors

- **Computing Eigenvectors**
  1. How to compute the first eigenvectors efficiently for large L
  2. For sparse graphs from knn or ε
     1. Efficient methods exist to compute the first eigenvectors of sparse matrices. The speed of convergence of those algorithms depends on the size of the eigengap (also called spectral gap)
     2. Eigengap: $\gamma_k = |\lambda_k - \lambda_{k+1}|$
     3. The large the Eigengap the faster the algorithms

Determining k

- Well-justified criteria in model-based clustering settings
  - Based on log-likelihood of the data
- Methods specific to Spectral Clustering
  - Eigengap Heuristic
    - Used for all 3 graph laplacian
    - Choose K that $\lambda_1 \ldots \lambda_k$ are very small, but, λ_{k+1} is relatively large
    - based on **perturbation theory,** where we observe that in the ideal case of k completely disconnected clusters, the eigenvalue 0 has multiplicity k, and then there is a gap to the (k + 1)th eigenvalue λk+1 > 0

DETERMINING K



Well Separated More Blurry Overlap So Much

Eigengap Heuristic usually works well if the data contains very well pronounced clusters, but in ambiguous cases it also returns ambiguous results.

WHICH GRAPH LAPLACIAN

- The k-means step

  It is not necessary. People also use other techniques

- Which graph Laplacian should be used?

  When does it become important?

  If the distribution of degrees in graph is very regular and **most vertices have approximately the same degree**, then all the Laplacians are very **similar** to each other, and will work equally well for clustering. However, if the **degrees in the graph are very broadly distributed**, then the Laplacians **differ** considerably.

WHICH GRAPH LAPLACIAN

- Which graph Laplacian should be used?

  Why normalized is better than unnormalized spectral clustering?

Objective1:

1. We want to find a partition such that points in different clusters are dissimilar to each other, that is we want to minimize the between-cluster similarity. In the graph setting, this means to minimize $cut(A, \bar{A})$.

Both RatioCut and Ncut directly implement

Objective2:

2. We want to find a partition such that points in the same cluster are similar to each other, that is we want to maximize the within-cluster similarities $W(A, A)$, and $W(\bar{A}, \bar{A})$.

Only Ncut implements

Normalized spectral clustering implements both clustering objectives mentioned above, while unnormalized spectral clustering only implements the first obejctive.

WHICH GRAPH LAPLACIAN

- **Ncut:**
  - $W(A,A) = W(A,V) - W(A,\bar{A}) = vol(A) - cut(A,\bar{A})$
  - the within-cluster similarity is maximized if $cut(A,\bar{A})$is small and if vol(A) is large.
  - As this is exactly what we achieve by minimizing Ncut
  - Ncut criterion implements the second objective
- RatioCut:
  - Objective is to maximize |A| and $|\bar{A}|$ instead of vol(A) and vol($\bar{A}$)
  - |A| and $|\bar{A}|$ are not necessary related to the within-cluster similarity since within-cluster similarity depends on the edges and not the number of nodes
  - A graph with a lot of vertices with low weight edges to each other

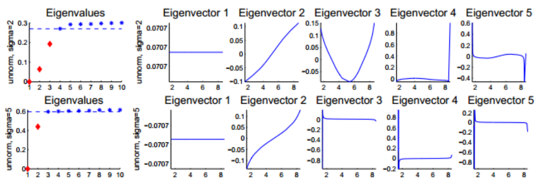
WHICH GRAPH LAPLACIAN

- **Second perspective: Consistency Issue**
  - $x_1 \ldots x_k$ have been sampled i.i.d according to some probability distribution P on some underlying data space X
  - **The most fundamental question is then the question of consistency:** if we draw more and more data points, do the clustering results of spectral clustering converge to a useful partition of the underlying space X ?

WHICH GRAPH LAPLACIAN

- Normalized Spectral clustering:
  - All consistency statements about normalized spectral clustering hold, for both Lsym and Lrw, under very mild conditions which are usually satisfied in real world applications

- Unnromalized:
  - All consistency statements about normalized spectral clustering hold, for both Lsym and Lrw, under very mild conditions which are usually satisfied in real world applications



WHICH GRAPH LAPLACIAN

- Which graph Laplacian should be used?

Why the eigenvectors of L_{rw} are better than those of L_{sym}?

1. Eigenvectors of L_{rw} are cluster indicator vectors $\mathbb{I}_{A_i}$, while the eigenvectors of L_{sym} are additionally multiplied with $D^{1/2}$, which might lead to undesired artifacts.

2. Using L_{sym} also does not have any computational advantages.

REFERENCE

- Ulrike Von Luxburg. A Tutorial on Spectral Clustering. Max Planck Institute for Biological Cybernetics Technical Report No. TR-149.
- Marina Meila and Jianbo Shi. A random walks view of spectral segmentation. In Tenth International Workshop on Artificial Intelligence and Statistics (AISTATS), 2001.
- A.Y. Ng, M.I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. Advances in Neural Information Processing Systems (NIPS),14, 2001.
- A. Azran and Z. Ghahramani. A new Approach to Data Driven Clustering. In International Conference on Machine Learning (ICML),11, 2006.
- A. Azran and Z. Ghahramani. Spectral Methods for Automatic Multiscale Data Clustering. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2006.
- Arik Azran. A Tutorial on Spectral Clustring. http://videolectures.net/mlcued08_azran_mcl/
- A. Singh, Spectral Clustering

Thanks