A Tutorial on Spectral Clustering

Ulrike von Luxburg, (updated) 2007
Max Planck Institute for Biological Cybernetics
(arxiv.org/abs/0711.0189)

Slides: github.com/BrianCechmanek/cs886/

Presenter: Brian Cechmanek
Date: March 7 2018,
University of Waterloo
CS886

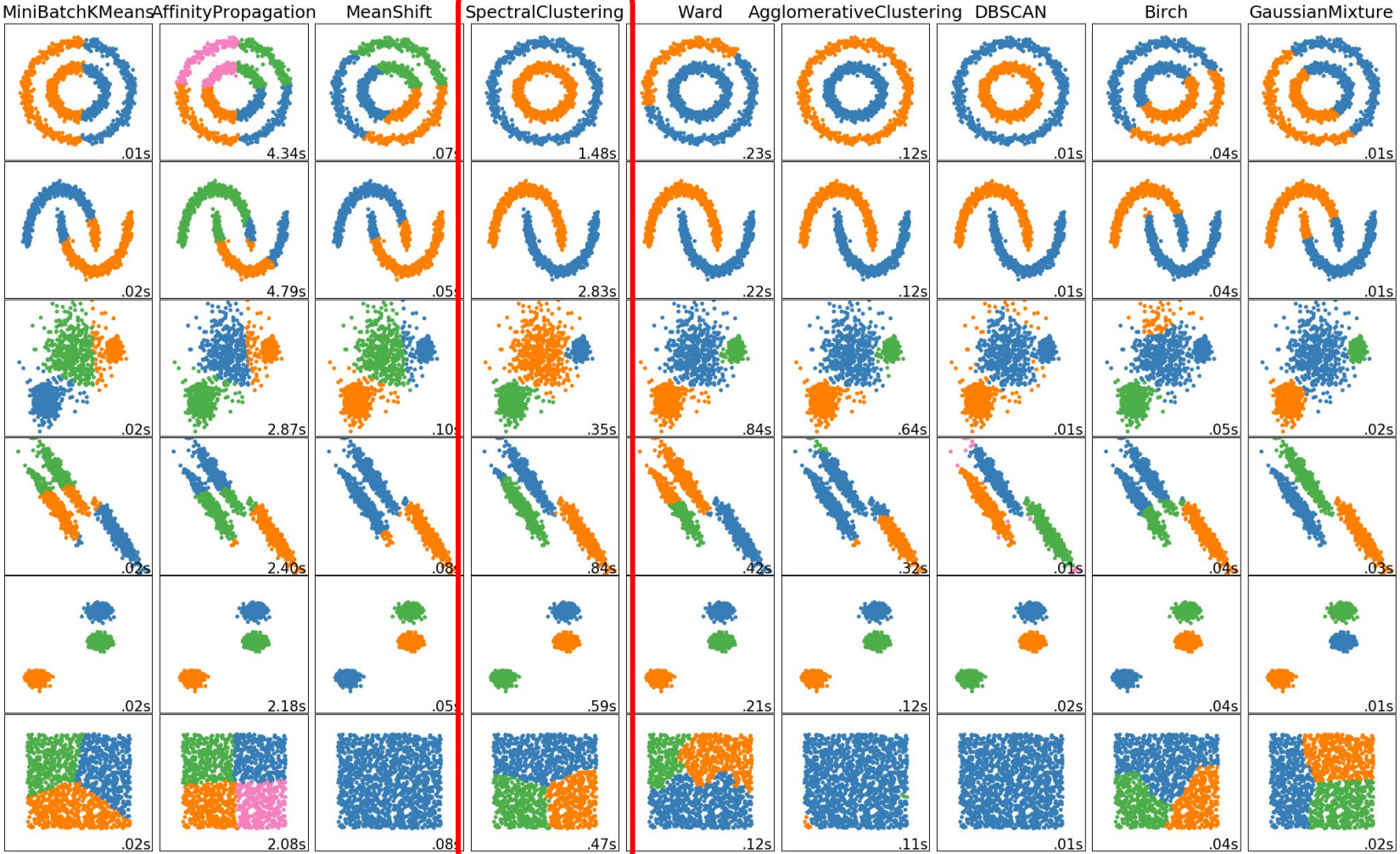
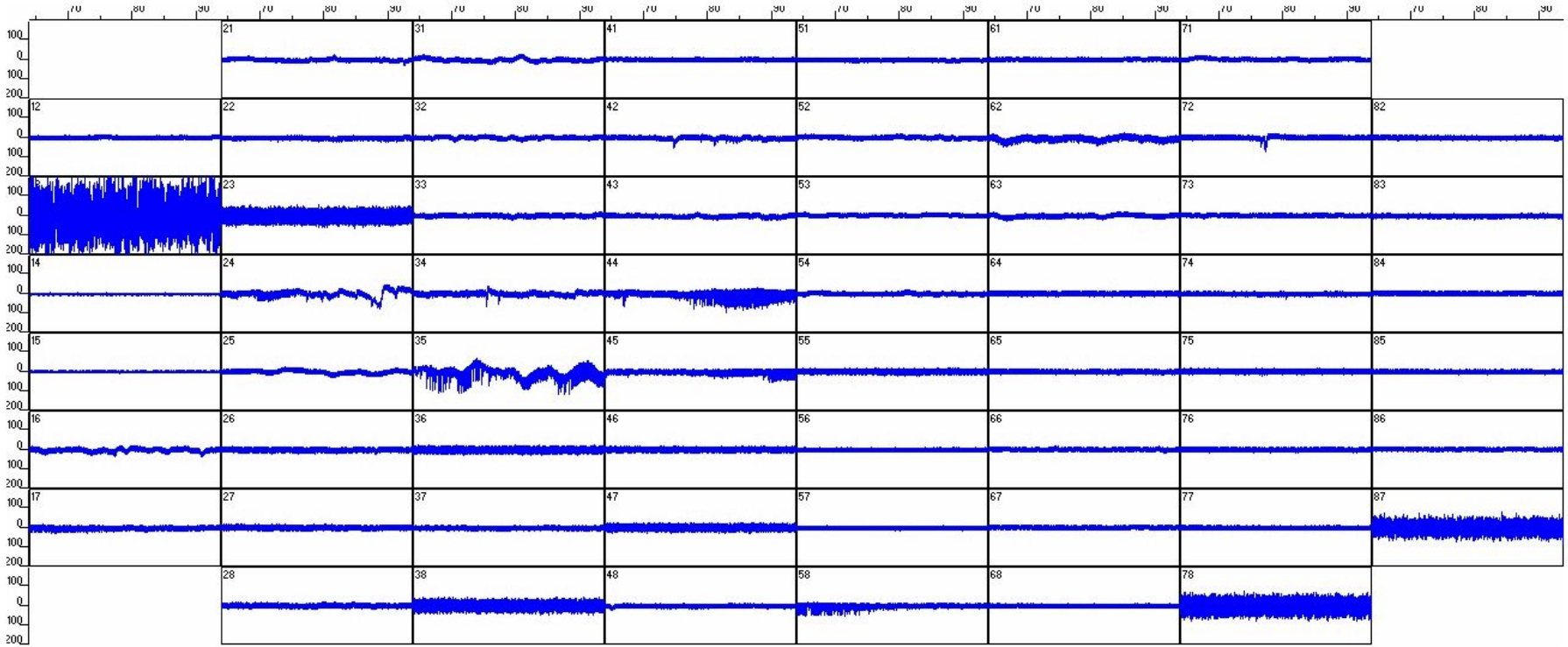


Image: http://scikit-learn.org

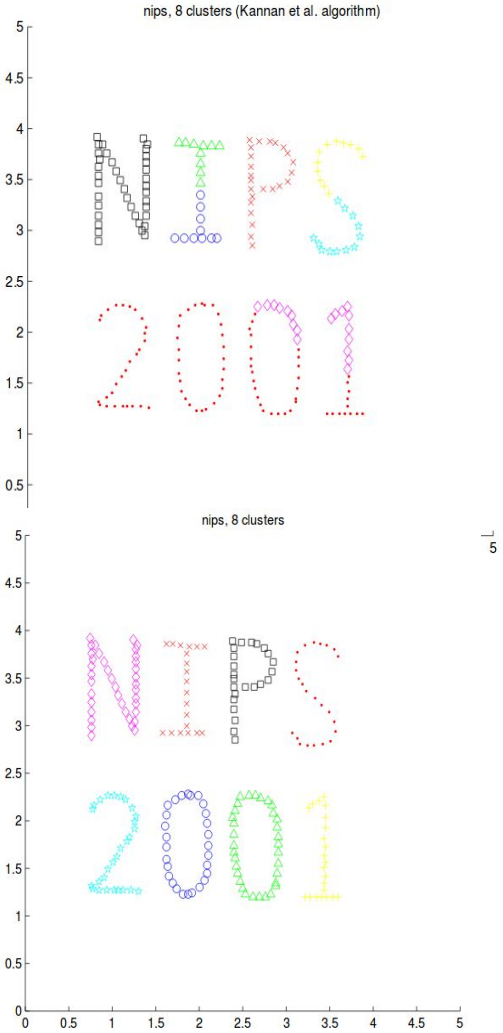
Motivation (mine)

- Non-intuitive feature extraction
- Onerous labelling process
- Non-intuitive grouping
- {0,1} classification vs k clusters?



Motivation

- Often outperforms traditional approaches
- Simple to implement
- Efficient computation by linear algebraic methods
- No assumptions on shapes of clusters



Outline (paper)

Goal: Derive algorithm from scratch, then discuss intuition

1. Similarity graphs
2. Graph Laplacians
3. Spectral Clustering algorithms
4. Algorithm intuition and POVs:
    a. Graph partitioning approach
    b. Random Walk perspective
    c. Perturbation theory
5. Practical issues and extensions
6. Additional literature

Algorithm preview:

Given [x_1, … x_n], and a notion of similarity

1. Construct a weighted undirected graph G(V,E)
2. Construct a similarity graph *S* from G
3. Map *S* to lower-dimensionality via graph Laplacian, eigenvalues, eigenvectors
4. Cluster the lower-dimension map

Similarity Graphs

Given a set of data points $x_1,..., x_n$

- and some notion of similarity $s_{ij} \geq 0$
- similarity graph G = (V;E), where v_i represents a data point x_i, and E = s(ij)

Goal: find a partition of the graph such

- that the edges between different groups have very low weights
    - Degree of vertex:

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

- The *degree matrix* is then *D*, with degrees $d_1,...d_n$ on the diagonal

Similarity Graphs (Cont)

- Given A ⊂ V, denote it's complement V \ A by Ā
- define the indicator vector $\mathbb{I}_A = (f_1,...,f_n)' \in \mathbb{R}^n$ as the vector with entries $f_i = 1$ if $v_i \in A$ and $f_i = 0$ otherwise
- $I \in A$ for $\{i \mid v_i \in A\}$ (convenient shorthand)
- For two not necessarily disjoint sets $A, B \subset V$:

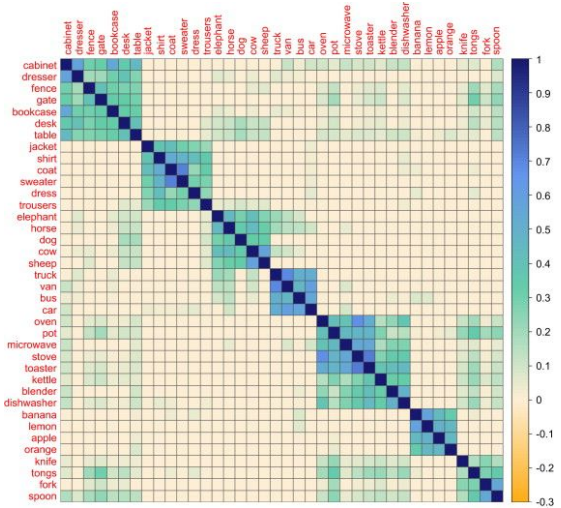
$$W(A,B) := \sum_{i \in A, j \in B} w_{ij}.$$

- Define: "size" of a subset $A \subset V$ as either:

|A| := the number of vertices in A

OR

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



Source: researchgate.net

Types of Similarity Graphs

Goal: construct a similarity graph to model the “local neighbourhood” relationships between the data points

1. ε-neighbourhood graph: connect v_i, v_j if pairwise distance ≤ ε .
2. KNN graph: connect v_i, v_j if v_j is among knn of v_i.
3. Fully connected graph: connect all points with $s_{ij} \geq 0$ and weight by s_{ij}.
    a. “this construction is only useful if the similarity function itself models local neighborhoods”

Example: $$s(x_i, x_j) = exp(-||x_i - x_j||^2/(2\sigma^2))$$

Here, σ controls the width of the neighbourhoods.
Note: σ plays a similar role as ε in ε-neighbourhood graphs.

- [Author] effects of similarity graph type is lacking research

Graph Laplacians

- Given: G, an undirected graph, with the weight matrix *W*, where:

$$w_{ij} = w_{ji} \geq 0$$

- And it's adjacency matrix *A*

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

$$D - A = L$$

$$\begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} - \begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 2 & -1 & 0 & 0 & -1 & 0 \\ -1 & 3 & -1 & 0 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 3 & -1 & -1 \\ -1 & -1 & 0 & -1 & 3 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \end{pmatrix}$$

- Do not assume normalization

Source: wikipedia (laplacian matrix)

Graph Laplacians: unnormalized (properties of *L*)

Proposition 1

1. For every vector **f** $\in \mathbb{R}^n$ we have: $f'Lf = \frac{1}{2}\sum_{i,j=1}^{n} w_{i,j}(f_i - f_j)^2$

2. *L* is symmetric and positive semi-definite ($\forall i : \lambda_i \geq 0$)
3. The smallest eigenvalue of *L* is 0, the corresponding eigenvector is the constant one vector $\mathbb{I}$.
4. *L* has n non-negative, real-valued eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq,\ldots,\leq \lambda_n$.

Proof (1):

$$f'Lf = f'Df - f'Wf = \sum_{i=1}^{n} d_i f_i^2 - \sum_{i,j=1}^{n} f_i f_j w_{i,j}$$

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

Graph Laplacians: unnormalized (Cont)

Proposition 2 (number of connected components and spectrum of *L*)

- The multiplicity *k* of the eigenvalue 0 of *L* equals the number of connected components $A_1,...,A_k$ in the graph.
- The eigenspace of eigenvalue 0 is spanned by the indicator vectors $\mathbb{I}A_1,...,\ \mathbb{I}Ak$ of those components.

Proof: with *k*=1 (graph is connected), assume *f* is an eigenvector with eigenvalue 0. Then:

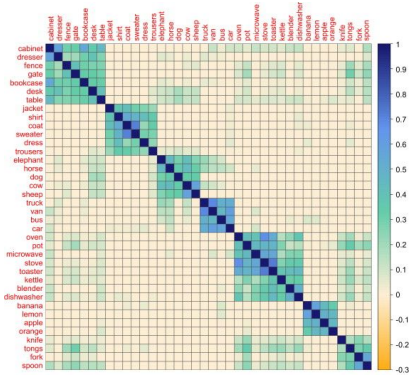
$$0 = f' L f = \sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2$$

- Thus, if two vertices v_i and v_j are connected (i.e., $w_{ij} > 0$), then $\mathbf{f_i = f_j}$.

Graph Laplacians: unnormalized (Cont)

- For *k* connected components (ordered according to their connected component), the adjacency matrix *W* has a block diagonal form, and thus *L*:

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



- Each block is a proper Laplacian of its own: the spectrum of $L = L_i \cap \ldots, L_n$.
- Every L_i has eigenvalue 0,... thus *L* has as many eigenvalues 0 as there are connected components
- The smallest non-zero eigenvalue of *L* is called the spectral gap
- The 2^{nd}-smallest value of *L* approximates the sparsest cut of a graph (later…)

Graph Laplacians: Normalized

- Recall: the *degree matrix* is *D, then...*

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

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

Proposition 3 (properties of L_{sym} and L_{rw})
1. For every **f** $\in \mathbb{R}^n$ we have:

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

Graph Laplacians: Normalized (Cont)

Proposition 3 (properties of L_{sym} and L_{rw})

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 $Lu = \lambda Du$.

4. 0 is an eigenvalue of L_{rw} with the constant one vector 𝟙 as eigenvector.
0 is an eigenvalue of *Lsym* with eigenvector $D^{1/2}$𝟙 .

5. L_{sym} and L_{rw} are positive semi-definite and have n non-negative real-valued eigenvalues $0 = \lambda_1 \leq ,.., \leq \lambda_n$.

Proofs: (omitted for time)

Graph Laplacians: Normalized (Cont)

- The elements of L_{sym} are given by:

$$L_{i,j}^{\text{sym}} := \begin{cases} 1 & \text{if } i = j \text{ and } \deg(v_i) \neq 0 \\ -\frac{1}{\sqrt{\deg(v_i)\deg(v_j)}} & \text{if } i \neq j \text{ and } v_i \text{ is adjacent to } v_j \\ 0 & \text{otherwise.} \end{cases}$$

- The elements of L_{rw} are given by:

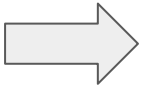
$$L_{i,j}^{\text{rw}} := \begin{cases} 1 & \text{if } i = j \text{ and } \deg(v_i) \neq 0 \\ -\frac{1}{\deg(v_i)} & \text{if } i \neq j \text{ and } v_i \text{ is adjacent to } v_j \\ 0 & \text{otherwise.} \end{cases}$$

Source: wikipedia (laplacian matrix)

Eigenvalues, eigenvectors: Map to lower-dimensionality

L(unnorm)

$$\begin{pmatrix} 2 & -1 & 0 & 0 & -1 & 0 \\ -1 & 3 & -1 & 0 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 3 & -1 & -1 \\ -1 & -1 & 0 & -1 & 3 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \end{pmatrix}$$



Eigenvalues	Eigenvectors
λ_0 = 0.7215863905035547	0.414, -0.505, 0.288, -0.0323, 0.567, 0.4084
λ_1 = 1.682569392019022	0.309, 0.040, 0.289, -0.469, -0.658, 0.408
λ_2 = 2.9999999999999996	0.069, 0.759, 0.289, 0.356, 0.205, 0.408
λ_3 = 4.891219848710292	-0.221, 0.201, -0.577, -0.561, 0.308, 0.408
λ_4 = 3.7046243687671274	0.221, -0.201, -0.577, 0.562, -0.308, -0.408
λ_5 = -1.8859412321929863e-16	-0.794, -0.294, 0.289, 0.144, -0.114, 0.408

- Does anyone see the (lazy) mistake here?

Spectral Clustering Algorithms:

3 to discuss

1. Unnormalized
2. Normalized according to Shi and Malik (2000)
3. Normalized according to Ng, Jordan & Weiss (2002)

Spectral Clustering Algorithms:

Unnormalized (approximates minimization of RatioCut)

Input: $S \in \mathbb{R}^{nxn}$, # clusters k

1. Construct Similarity Graph, let W be its weighted adjacency matrix
2. Compute L (unnormalized)
3. Compute first k eigenvectors $u_1,...,u_k$ of L.
4. Let $U \in \mathbb{R}^{nxk}$ contain vectors $u_1,...,u_k$ as columns.
5. For $i = 1,...,n$, let $y_i \in \mathbb{R}^k$ be the vector corresponding to the i-th row of U.
6. Cluster the points $(y_i)i=1,...,n$ in $\mathbb{R}^k$ with k-means, into clusters $C_1,...C_k$

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

Spectral Clustering Algorithms:

Normalized (Shi & Malik, 2000) (approximates minimization of Ncut)

Input: $S \in \mathbb{R}^{nxn}$, # clusters k

1. Construct Similarity Graph, let W be its weighted adjacency matrix
2. Compute L (unnormalized *)
3. Compute first k eigenvectors $u_1,..,u_k$ of L of the generalized eigenproblem $Lu = \lambda Du$ *
4. Let $U \in \mathbb{R}^{nxk}$ contain vectors $u_1,...,u_k$ as columns
5. For i = 1,...,n, let $y_i \in \mathbb{R}^k$ be the vector corresponding to the i-th row of U
6. Cluster the points (y_i)i=1,...,n in $\mathbb{R}^k$ with k-means, into clusters $C_1,...C_k$

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

* Corresponds to the eigenvectors of Lrw, which are normalized

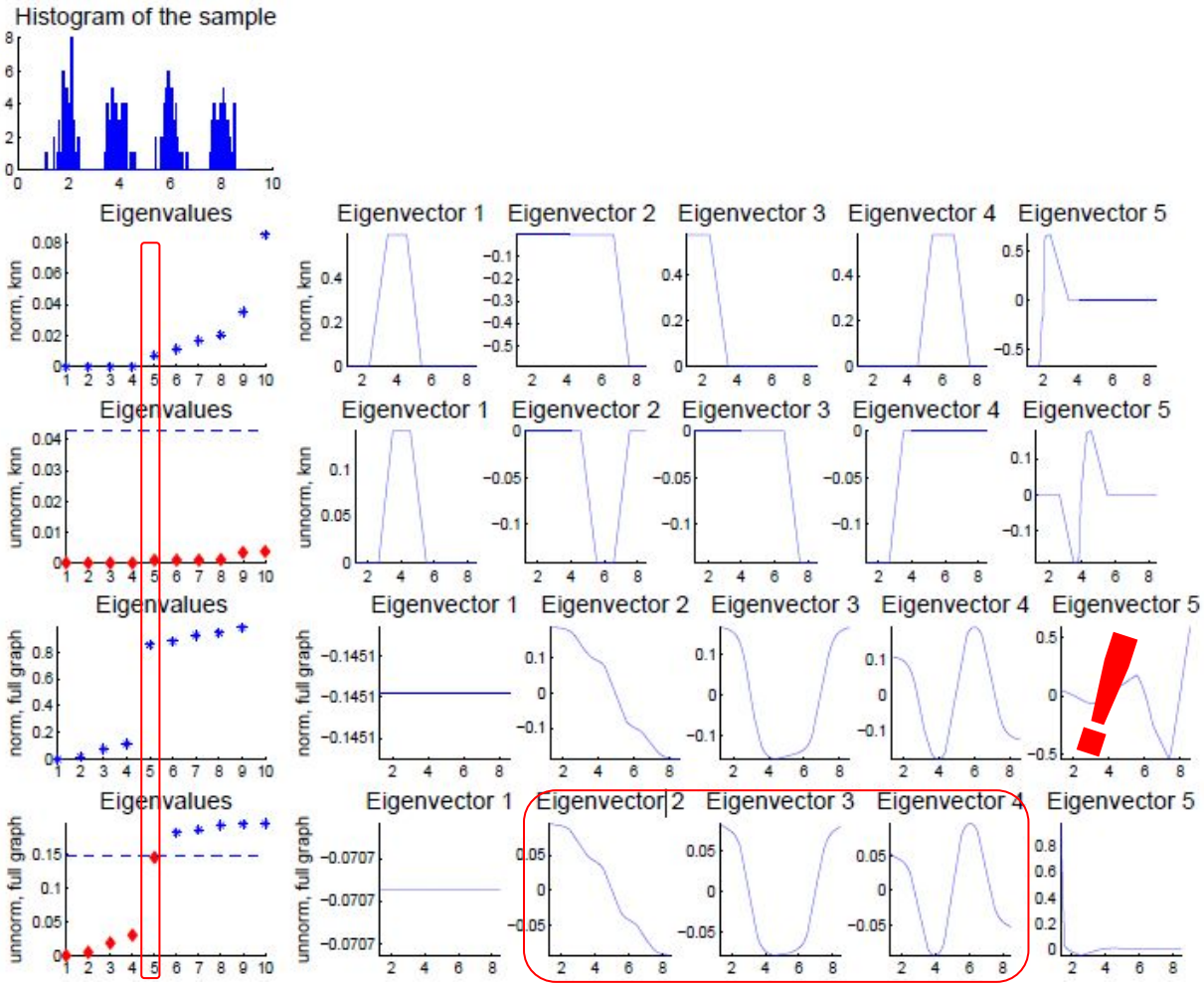
Spectral Clustering Algorithms:

Normalized (Ng, Jordan & Weiss, 2002)

Input: $S \in \mathbb{R}^{nxn}$, # clusters k

1. Construct Similarity Graph, let W be its weighted adjacency matrix
2. Compute L_{sym} (normalized)
3. Compute first k eigenvectors $u_1,..,u_k$ of L_{sym}
4. Let $U \in \mathbb{R}^{nxk}$ contain vectors $u_1,...,u_k$ as columns
5. Form the matrix $T \in \mathbb{R}^{nxk}$ from U by normalizing the rows to norm 1 that is set $t_{ij} = u_{ij} / (\sum_k u_{ik}^2)^{1/2}$
6. For i = 1,...,n, let $y_i \in \mathbb{R}^k$ be the vector corresponding to the i-th row of T
7. Cluster the points (y_i)i=1,...,n in $\mathbb{R}^k$ with k-means, into clusters $C_1,...C_k$

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


Histogram of the sample
Eigenvalues
norm, knn
Eigenvector 1
Eigenvector 2
Eigenvector 3
Eigenvector 4
Eigenvector 5
Eigenvalues
unnorm, knn
Eigenvector 1
Eigenvector 2
Eigenvector 3
Eigenvector 4
Eigenvector 5
Eigenvalues
norm, full graph
Eigenvector 1
Eigenvector 2
Eigenvector 3
Eigenvector 4
Eigenvector 5
Eigenvalues
unnorm, full graph
Eigenvector 1
Eigenvector 2
Eigenvector 3
Eigenvector 4
Eigenvector 5

Outline (paper)

- ~~Derive algorithm from scratch~~

1. ~~Similarity graphs~~
2. ~~Graph Laplacians~~
3. ~~Spectral Clustering algorithms~~
4. Algorithm intuition and POVs:
    a. Graph partitioning approach
    b. Random Walk perspective
    c. Perturbation theory
5. Practical issues and extensions
6. Additional literature

Intuition: Graph Cut (Partition) POV

Show that spectral clustering approximates [these] graph partitioning problems

- Given: *S* & *W* (similarity graph, adjacency matrix)

1. Goal: solve (min)cut(A_1,..., A_k) where A_i is “reasonably large”:

$$cut(A_1, \dots A_k) := \frac{1}{2} \sum_{i=1}^{k} W(A_i, \bar{A}_i)$$

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

3. $$Ncut(A_1, \dots A_k) := \frac{1}{2} \sum_{i=1}^{k} W(A_i, \bar{A}_i)/vol(A_i) = \sum_{i=1}^{k} cut(A_i, \bar{A}_i)/vol(A_i)$$

- This tries to balance cluster sizes… but makes the problem NP-hard...

Intuition: Random Walks POV

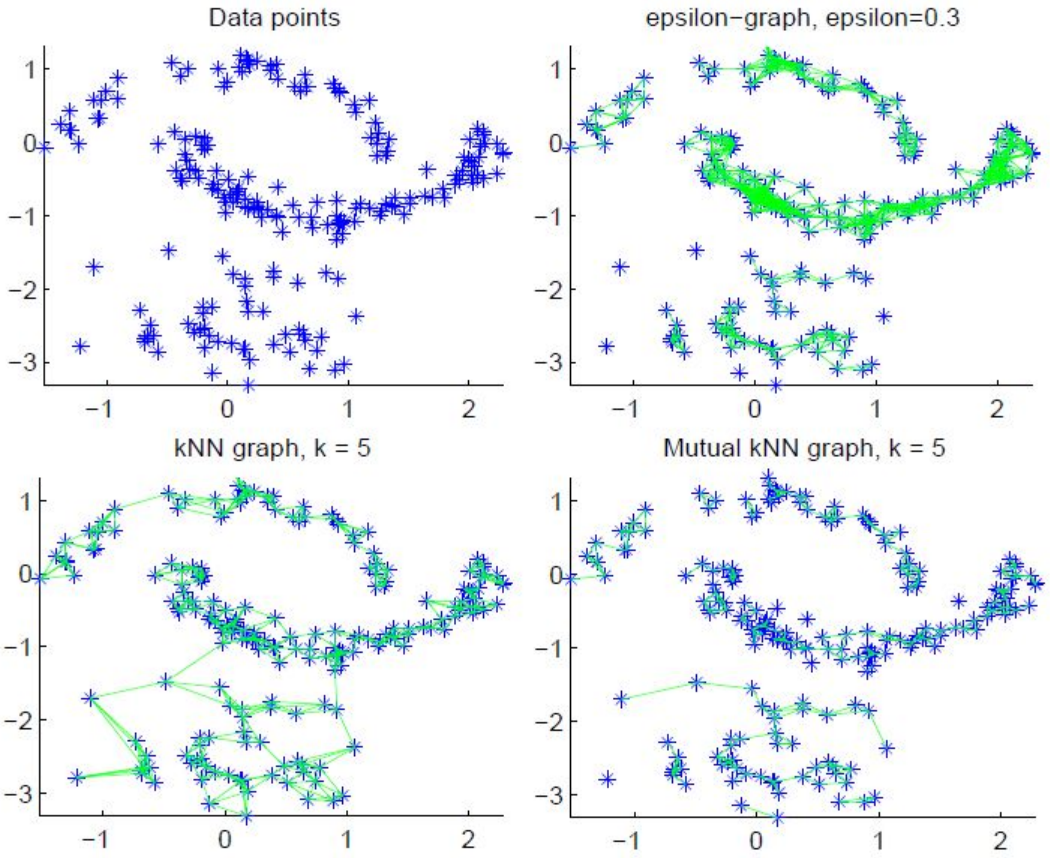
- Stochastic process, randomly jumping from vertex to vertex
- Goal: Try to find partitioning s.t. the random walk stays long within the same cluster - seldom jumping between clusters
  - Transition probability of jumping from v_i to v_j is:
    $$\mathbf{p_{ij} := w_{ij}/d_i}$$
  - Thus the transition matrix $P = (p_{ij})_{i,j=1,\ldots,n}$ is:
    $$\boldsymbol{P = D^{-1}W}$$
  - 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
- Can connect random walks and Laplacian via the ‘commute distance’; the expected time it takes a random walk to travel between two vertices
  - Note: this decreases if there are multiple different short paths between two vertices
  - Points connected by a (set of) short path(s) in the graph lie in the same local neighbourhood

Intuition: Perturbation theory POV

- Recall: Monday's presentation
- Extend this to how the eigenvalues/eigenvectors of A would change if perturbed
- Caveats:
  - Ensure that the order of the eigenvalues is meaningful. In the case of Laplacians this is always true, as we know that any connected component possesses exactly one eigenvector which has eigenvalue 0. (may not be true of S or W)
  - The entries of the eigenvectors on the components should be "safely bounded away" from 0: a perturbed eigenvector is unlikely to have any zero-valued. (Only for L_{sym})

Practical Details:

- The similarity function:
  - Must be intrinsically “meaningful” in scoring different points distances
- Constructing *S* graph:
- ε-neighbourhood: difficult to choose appropriate/useful ε (specifically on data with different ‘scaled’ clusters)
- KNN: *usually* works on “different scales”, but can break graph into several disconnected components if there are high-density regions reasonably far apart
- Fully-connected: applicable if *S* is not sparse, but choosing σ is similar to choosing ε.
- In general, work with KNN first



Practical Details (Cont.):

- Parameters of S: *k* or ε
  - But little theory is known…
  - Rule of thumb, with kNN, the connectivity parameter should dictate a connected graph - or at least have significantly fewer connected components than clusters.
  - For ε-neighbourhood, aim for a resulting graph that is “safely connected”: ε = min(longest edge in MST of the fully connected graph) - but fails on outliers.
  - For fully-connected, aim instead to have s(xi,xj) scaled to result similarly to a corresponding kNN or ε-neighbourhood graph.
- Number of clusters...
- The k-means step:
  - Could use any algorithm - but Euclidean distance seems “meaningful”
- Which Graph Laplacian?
  - Check degree of distribution of *S*: if very regular, all 3 are fine, otherwise suggestion is normalized and L_{rw}

Conclusion:

- Popular in ML
- Extended to many areas…
- “[overall] spectral clustering is successful because it does not make strong assumptions on the form of the clusters” (as opposed to k-means, where clusters form convex sets)
- Can be efficiently implemented for large data sets (if *S* is sparse)

- Choosing a good *S* is not trivial
- Spectral clustering can be unstable under different parameter choice for the neighbourhood graphs

- Extra links: Dr. Ali Ghodsi (UW) lectures on SC1, SC2