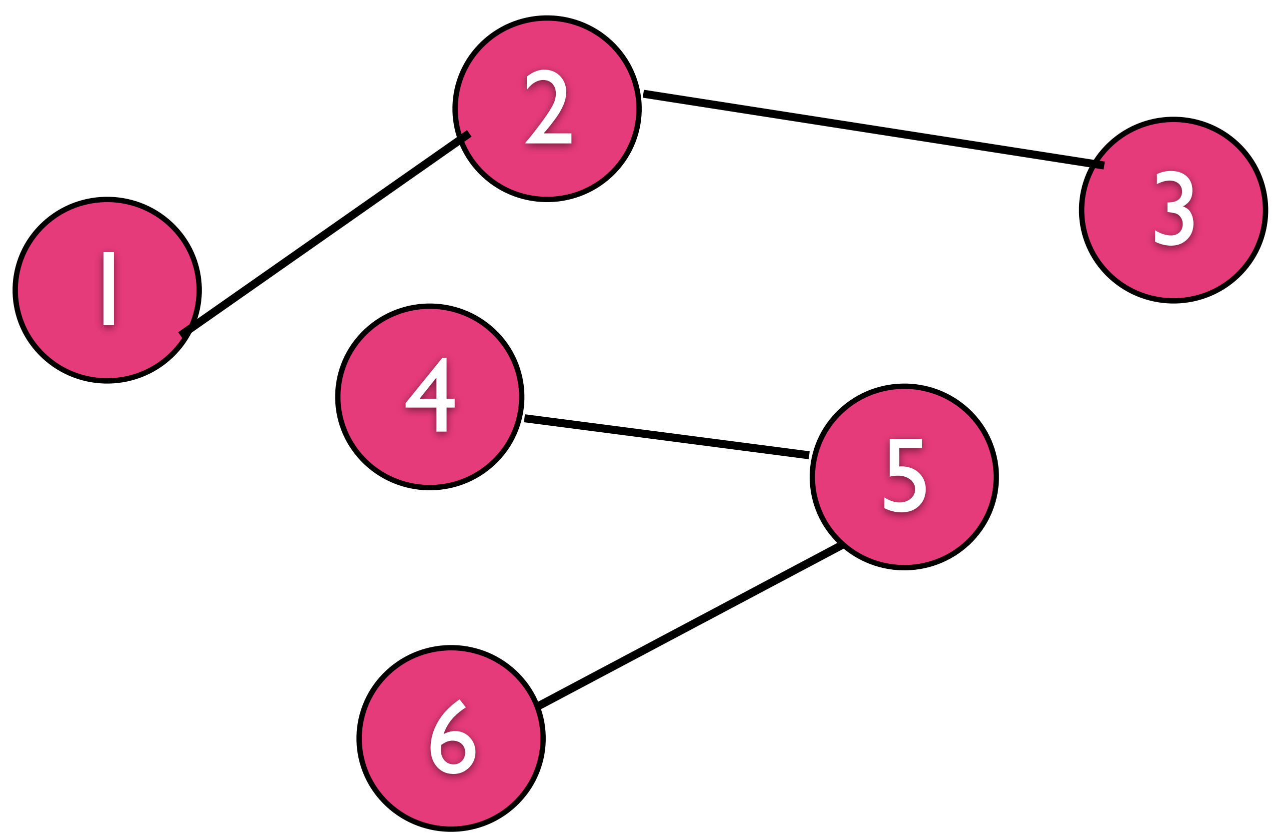
Graph Mining

Danushka Bollegala

UNIVERSITY OF LIVERPOOL

Graphs

- A Graph G can be defined as a set of vertices (nodes) V connected by a set of edges (links) E
- A graph G(V,E) is fully defined by specifying the two sets V and E

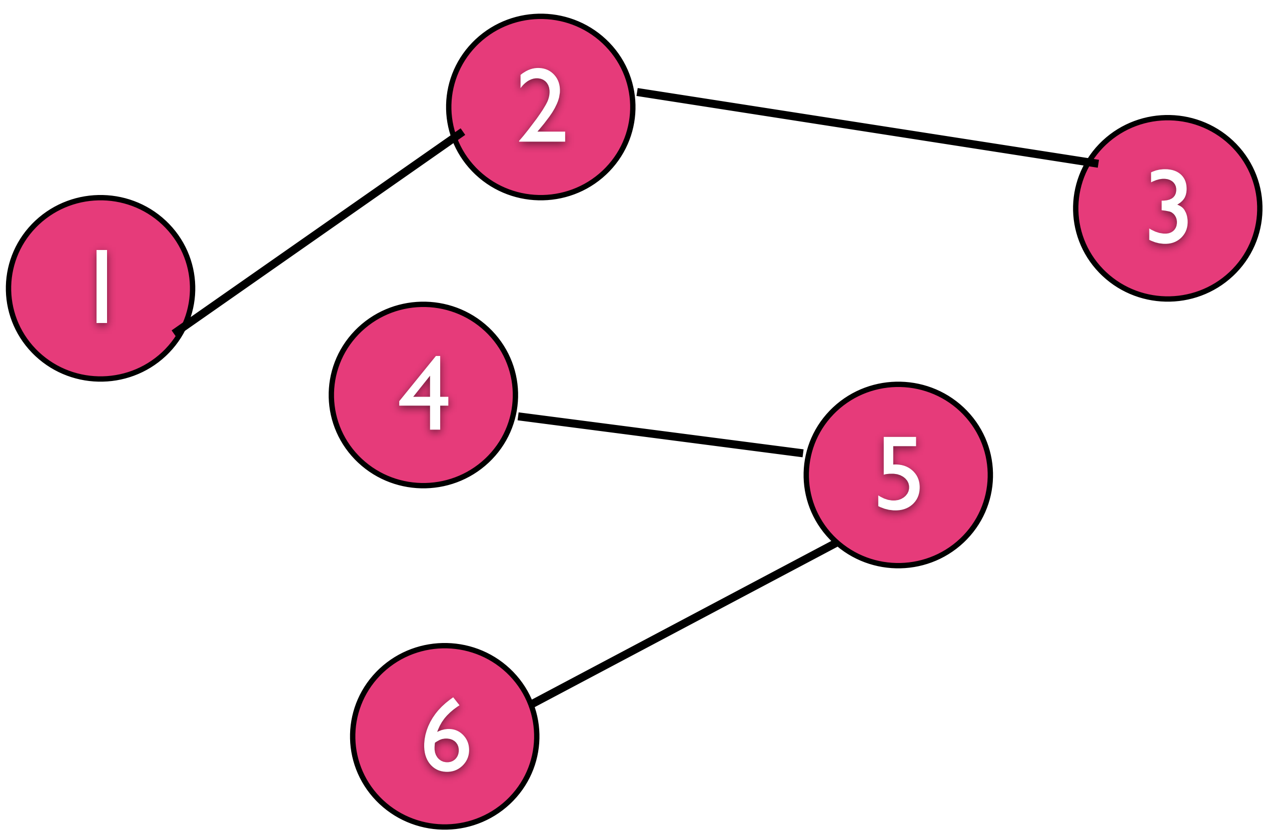


Types of Graphs

- Undirected Graph
  - There are no directional edges in the graph
- Directed Graph
  - There are directional edges in the graph
- Labeled/Coloured Graph
  - Vertex-Labeled Graph
    - Vertices are labeled (coloured)
  - Edge-Labeled Graph
    - Edges are labeled (coloured)
- Weighted Graph
  - Edges have weights associated with them
- Unweighted Graph
  - Edges have no weights associated with them. All edges have an equal weight.

Adjacency Matrix

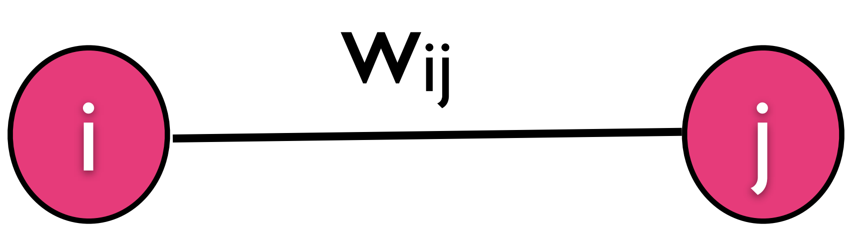
- If two vertices v_i and v_j are connected by an edge in an graph G, then the element a_{ij} in the incidence matrix will be set to 1, otherwise it will be set to 0.



$$\begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

Weight Matrix

- The weight matrix W of a weighted graph G denotes the weight of the edge between vertices v_i and v_j by the element w_{ij}
- Notes
  - A negative weight does not indicate a reverse link always (however, some abuse of notation is possible, if defined in advance)



For undirected graphs, $w_{ij} = w_{ji}$
(W is becomes a symmetric matrix)

State Transitions

- At a given time t=T, the probability of being at each vertex can be represented by a |V| dimensional vector **x**, where |V| is the total number of vertices in the graph.
- Question
  - What is the probability of being at each vertex at t =(T+1)
- Answer
  - B**x**
    - **B** is the state transition matrix
    - The probability of being at vertex V_j at t=T+1, when we are at vertex V_i at t = T is given by B_{ij}
- What about t=(T+2) then
  - $B(B\mathbf{x}) = B^2\mathbf{x}$
- What about t = (T+n) then
  - $B^n\mathbf{x}$

Random Walk in a Graph

- Assume that you are walking in a graph
- You start with some vertex and randomly move to a vertex that is connected to the current vertex
- All connected vertices have an equal probability of getting selected for the next move
- After you have moved infinite amount of time in this graph according to the previously described mechanism, what is the probability of you ending up in some vertex v_i in the graph?

Random Walk

- If the state transition has reached a stable state, then we have the situation
  - $Ax = \lambda x$
- This means that x is the eigenvector of A corresponding to the eigenvalue λ, which is a scalar.
- Instead of moving around the graph for infinite time we can simply perform eigenvalue decomposition of A to find the final state (if it exists!)
- Moreover, final state (if exists) does not depend on the initial state!

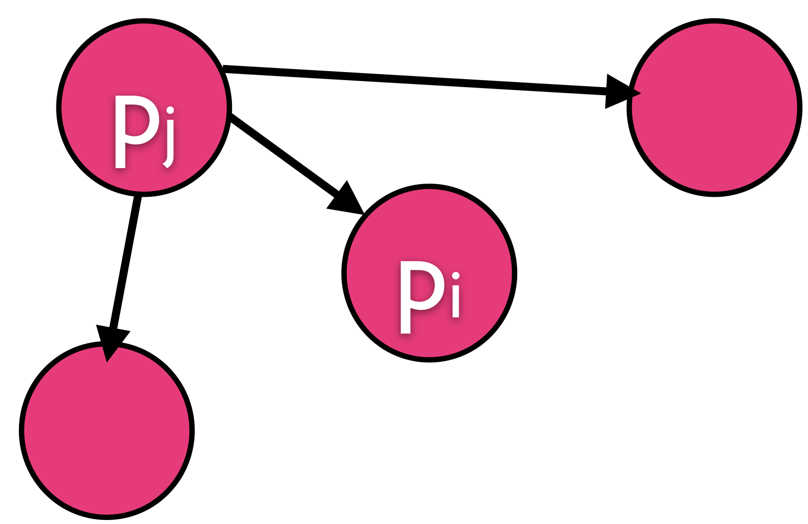
What can we learn from a Random Walk?

- Connectivity of the graph
  - If there are *islands* in the graph (ie. subgraphs that are not connected), then no matter how much we perform this random walk, we will not be able to reach those islands.
- Importance of the vertices
  - If there is a close connection between two vertices v_i and v_j, then the probability of ending up in v_j, when we start from v_i will be higher
  - But, it does not matter from where we start
    - which means that the probability of ending up at a particular vertex is an indicator of how *important* that vertex (measured by its connectivity to other vertices in the graph) in the graph
  - Highly connected people are more important/influential?

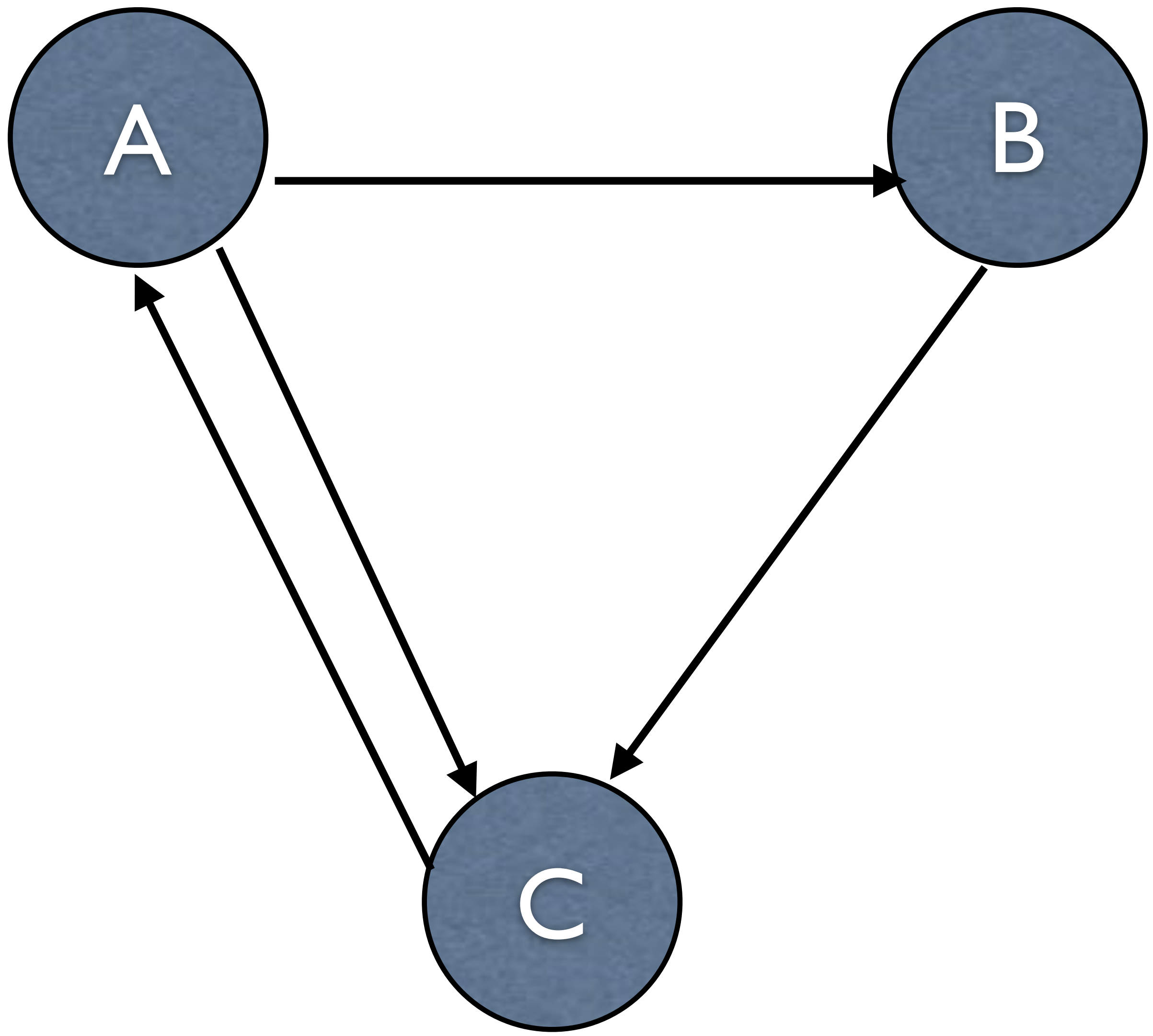
PageRank Algorithm

- One of many algorithms that are based on the idea of random walks in a graph
- Proposed by Larry Page
- Original objective
  - Compute the rank of web pages
  - vertices = web pages
  - edges = hyperlinks
- Can be applied to any graph, not limiting to web graph, to induce a ranking for the vertices.
- $PR(p_i)$: page rank of page p_i
- $L(p_j)$: number of outbound links on p_j

$$PR(p_i) = \sum_{p_j \in M(p_i)} \frac{PR(p_j)}{L(p_j)}$$



Quiz: Compute the PageRanks for the following graph.



$$p_A = p_C$$

$$p_B = \frac{p_A}{2}$$

$$p_C = p_B + \frac{p_A}{2}$$

Issues with simple PageRank

- If the random walker gets trapped/struck inside a particular node, then the simple PageRank algorithm we discussed previously will fail.
- This is called "a leak" of PageRank
- To overcome this problem we use *teleportation*
  - At each node p we will select a node from the set of nodes connected via in-bound links to p, M(p), with a probability *d-1*.
  - Or, we randomly jump (teleport) to any of the remaining (N-1) nodes with probability *d*.
- This gives rise to the *damped* version of PageRank discussed in the next slide.

Damping Factor

- It is possible that a random surfer (walker) might not surf (walk) over the graph eternally (until infinite number of iterations) but will stop after a while (tired/damping).
- The following version of the PageRank algorithm takes this into consideration
  - d is the damping factor and is set to 0.85 in most practical cases
  - N is the total number of vertices (pages)

$$PR(p_i) = \frac{1-d}{N} + d \sum_{p_j \in \mathcal{M}(p_i)} \frac{PR(p_j)}{L(p_j)}$$

Clustering on Graphs

- Given a set of N items $\{\mathbf{x}_n\}$ to cluster, we can represent those items as vertices in an undirected graph, where the weight associated with an edge e_{ij} corresponds to the similarity between the items $\mathbf{x}_i$ and $\mathbf{x}_j$ represented by the vertices v_i and v_j.
- By removing *k* edges of this graph (such that some objective function defined over the graph is optimised), we can create *k* clusters (of vertices) from this graph.
- Graph cut = clustering!
  - Normalised Graph Cut gives Spectral Clustering

Spectral Graph Theory

- Spectrum = the set of eigenvalues
- By looking at the *spectrum* we can know about the graph itself!
- A way of normalising data (canonical form) and then perform clustering (e.g. via k-means) on this normalised/reduced space.
- Input: A similarity matrix
- Output: A set of (non-overlapping/hard) clusters.

Terminology

- Undirected Graph G(V, E)
- V: set of vertices (nodes in the network)
- E: set of edges (links in the network)
- Weight w_{ij} is the weight of the edge connecting vertex I and j (represented by the affinity matrix.)
- Degree: sum of weights on outgoing edges of a vertex. $$d_i = \sum_{j=1}^{n} w_{ij}.$$
- Measuring the size of a subset A of V

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

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

How to create W?

- How to create the affinity matrix W from the similarity matrix S?
- ε-neighbourhood graph
  - Connect all vertices that have similarity greater than ε
- k-nearest neighbour graph
  - Connect the k-nearest neighbours of each vertex.
- Mutual k-nearest neighbour graphs for asymmetric S.
- Fully connected graph
- Use the Gaussian similarity function (kernel)

$$\exp(-\|x_i - x_j\|^2/(2\sigma^2))$$

Unnormalised Graph Laplacian

- L = D – W
- D: degree matrix. A diagonal matrix diag(d1,...,dn)
- Properties
- For every vector $\mathbf{f} \in R^n$ $$\mathbf{f}^T \mathbf{L} \mathbf{f} = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij} (f_i - f_j)^2$$
- L is symmetric and positive semi-definite
- The smallest eigenvalue of L is zero and the corresponding eigenvector is 1 = (1,...,1)T
- L has n non-negative, real-valued eigenvalues

Normalised Graph Laplacians

- Two versions exist
- $L_{sym} = D^{-1/2} L D^{-1/2} = I - D^{-1/2} W D^{-1/2}$
- $L_{rw} = D^{-1} L = I - D^{-1} W$

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

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

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 eigenproblem* $Lu = \lambda D u$.
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* $D^{1/2}\mathbb{1}$.
5. L_{sym} *and* L_{rw} *are positive semi-definite and have* n *non-negative real-valued eigenvalues* $0 = \lambda_1 \leq \ldots \leq \lambda_n$.

Positive Semi-definite Matrix

- A symmetric, real matrix $M \in \mathbb{R}^{n \times n}$ is defined to be positive semi-definite (PSD), if the scalar product $\mathbf{z}^T M \mathbf{z}$ is positive for every non-zero column vector $\mathbf{z} \in \mathbb{R}^n$
- Useful fact to remember
  - All eigenvalues of a PSD matrix are non-negative.

Spectral Clustering (L)

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 $u_1, \ldots, u_k$ of L.
- Let $U \in \mathbb{R}^{n \times k}$ be the matrix containing the vectors $u_1, \ldots, u_k$ as columns.
- For $i = 1, \ldots, n$, let $y_i \in \mathbb{R}^k$ be the vector corresponding to the i-th row of U.
- Cluster the points $(y_i)_{i=1,\ldots,n}$ in $\mathbb{R}^k$ with the k-means algorithm into clusters $C_1, \ldots, C_k$.

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

Spectral Clustering (L_{rw})

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 $u_1, \ldots, u_k$ of the generalized eigenproblem $Lu = \lambda Du$.
- Let $U \in \mathbb{R}^{n \times k}$ be the matrix containing the vectors $u_1, \ldots, u_k$ as columns.
- For $i = 1, \ldots, n$, let $y_i \in \mathbb{R}^k$ be the vector corresponding to the i-th row of U.
- Cluster the points $(y_i)_{i=1,\ldots,n}$ in $\mathbb{R}^k$ with the k-means algorithm into clusters $C_1, \ldots, C_k$.

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

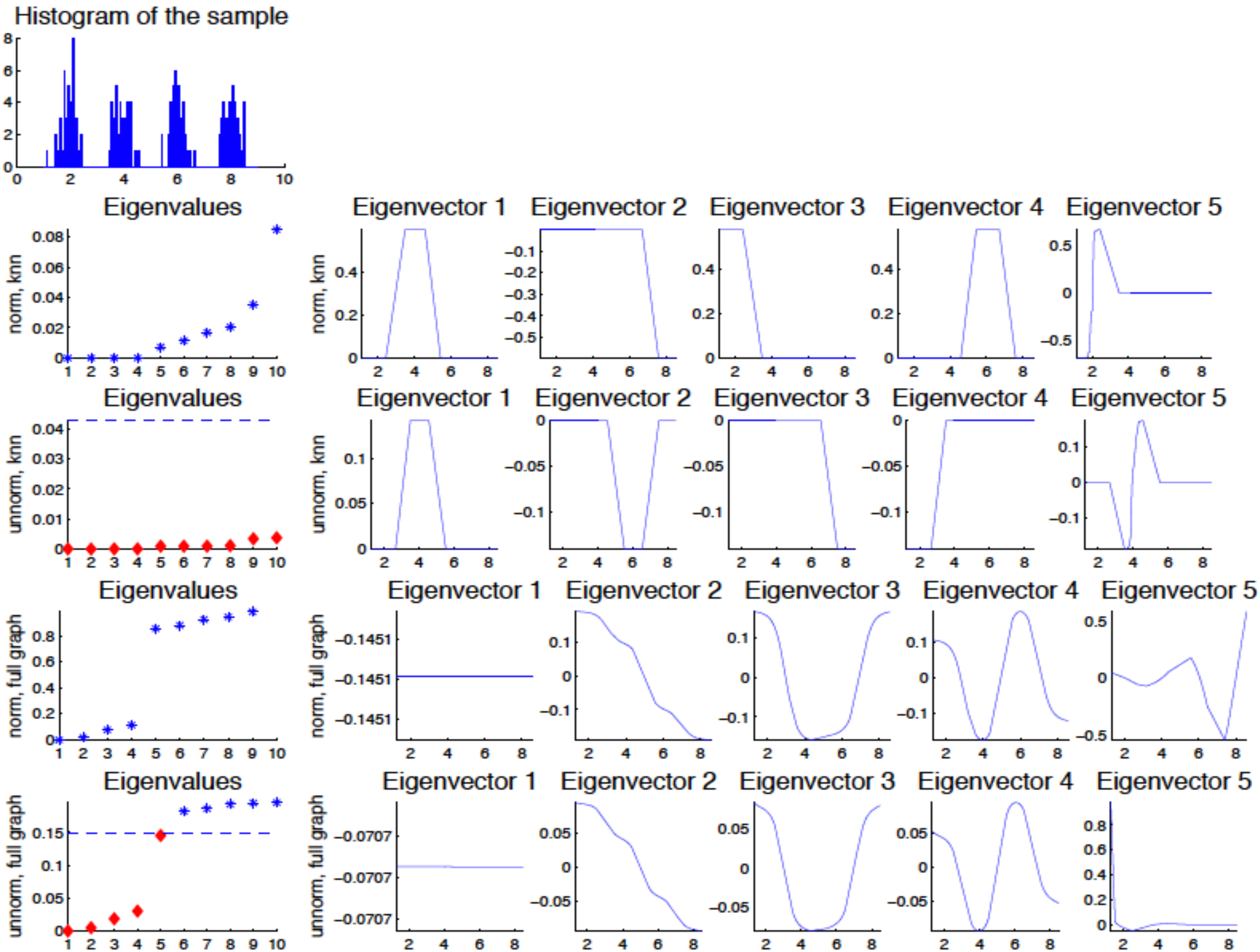
Spectral Clustering (L_{sym})

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_{sym}.
- Compute the first k eigenvectors $u_1, \ldots, u_k$ of L_{sym}.
- Let $U \in \mathbb{R}^{n \times k}$ be the matrix containing the vectors $u_1, \ldots, u_k$ 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 $i = 1, \ldots, n$, let $y_i \in \mathbb{R}^k$ be the vector corresponding to the i-th row of T.
- Cluster the points $(y_i)_{i=1,\ldots,n}$ with the k-means algorithm into clusters $C_1, \ldots, C_k$.

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


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

Graph cut Point of View

- The partition (A1,...,Ak) induces a cut on the graph
- Two types of graph cuts exist
- W(A,B) is the sum of edge weights of edges that start from a vertex in cluster A and end at a vertex in cluster B.

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

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

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

- Spectral clustering solves a relaxed version of the mincut problem (therefore it is an approximation)

Example: RatioCut, k=2

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

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

$$\sum_{i=1}^{n} f_i = \sum_{i \in A} \sqrt{\frac{|\overline{A}|}{|A|}} - \sum_{i \in \overline{A}} \sqrt{\frac{|A|}{|\overline{A}|}} = |A| \sqrt{\frac{|\overline{A}|}{|A|}} - |\overline{A}| \sqrt{\frac{|A|}{|\overline{A}|}} = 0.$$

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

$$\boxed{\min_{f \in \mathbb{R}^n} f'Lf \text{ subject to } f \perp \mathbb{1}, \ \|f\| = \sqrt{n}.}$$

$$\begin{aligned} f'Lf &= \frac{1}{2} \sum_{i,j=1}^{n} w_{ij} (f_i - f_j)^2 \\ &= \frac{1}{2} \sum_{i \in A, j \in \overline{A}} w_{ij} \left(\sqrt{\frac{|\overline{A}|}{|A|}} + \sqrt{\frac{|A|}{|\overline{A}|}} \right)^2 + \frac{1}{2} \sum_{i \in \overline{A}, j \in A} w_{ij} \left(-\sqrt{\frac{|\overline{A}|}{|A|}} - \sqrt{\frac{|A|}{|\overline{A}|}} \right)^2 \\ &= \text{cut}(A, \overline{A}) \left(\frac{|\overline{A}|}{|A|} + \frac{|A|}{|\overline{A}|} + 2 \right) \\ &= \text{cut}(A, \overline{A}) \left(\frac{|A| + |\overline{A}|}{|A|} + \frac{|A| + |\overline{A}|}{|\overline{A}|} \right) \\ &= |V| \cdot \text{RatioCut}(A, \overline{A}). \end{aligned}$$

By Rayleigh-Ritz theorem the solution to this problem is given by the second smallest eigenvalue of L. (Recall that 0 is always an eigenvalue of L, which is PSD)

Recommendations

- L_{rw} based spectral clustering (Shi & Malik,2000) is better (especially when the degree distribution is uneven).
- Use k-nearest neighbour graphs
- How to set the number of clusters:
- k=log(n)
- Use the eigengap heuristic
- If using Gaussian kernel how to set sigma
  - Mean distance of a point to its log(n)+1 nearest neighbours.

References

- A Tutorial on Spectral Clustering, Ulrike von Luxburg, Statistics and Computing, Vol. 17, no 4, pp. 395-416, 2007.
- On spectral clustering: Analysis and algorithm, Andrew Y. Ng, In Proc. of Neural Information Processing (NIPS), 2001.
- Normalized cuts and Image Segmentation, Jianbo Shi and Jitendra Malik, IEEE Transactions on Pattern Analysis and Machine Learning, Vol. 22, no 8, pp. 888-905, 2000.