MA2000: Combinatorial Optimization

Madhab Barman and Nachiketa Mishra

Indian Institute of Information Technology,
Design & Manufacturing, Kancheepuram

Cut

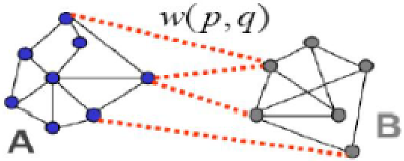
- **Cut:** A weighted graph $G = \langle V, E \rangle$ can be partitioned into two disjoint sets:

$$A \cup B = V \text{ and } A \cap B = \Phi$$

by simply removing the edges connecting the two parts.
- A weighted graph is the one in which weight is associated with each edge.
- The degree of dissimilarity between these two pieces can be computed as the total weight of the edges that have been removed. In graph theory, it is called the cut:

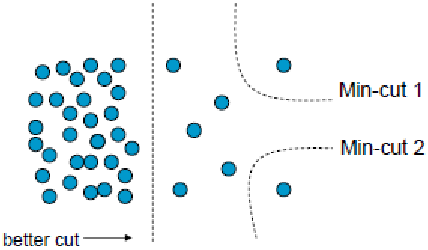
$$cut(A, B) = \sum_{p \in A} \sum_{q \in B} w(p, q),$$

where $w(p, q)$ is the weight of the edge that connects p and q



Min cut and Drawbacks

- By minimizing this cut value, one can optimally bi-partition the graph and achieve good segmentation: $\min cut(A, B)$.
- The minimum cut occasionally supports cutting isolated nodes in the graph due to the small values achieved by partitioning such nodes.



- Need to account for cluster similarity.

Normalized cut

- Computes the cut cost as a fraction of the total edge connections to all nodes in the graph.
- Fix bias of min cut by normalizing for size of segments:

$$Ncut(A,B) = \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(A,B)}{assoc(B,V)} \tag{1}$$

  where $assoc(A,V)$ defines the total weights of connection from nodes A to all nodes in the graph G.

$$vol(A) = assoc(A,V) = \sum_{p \in A} \sum_{q \in V} w(p,q)$$

- **Advantage:** Being unbiased measure: the isolated nodes, $Ncut$ value will no longer be small, as the cut value will almost always be a high percentage of the total connection from the isolated node to all other nodes.

Normalized cut

- Normalized association:

$$Nassoc(A,B) = \frac{assoc(A,A)}{assoc(A,V)} + \frac{assoc(B,B)}{assoc(B,V)}$$

  Defines how tightly on average within the cluster are connected to each others.
- Compute:

$$\begin{aligned} Ncut(A,B) &= \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(A,B)}{assoc(B,V)} \\ &= \frac{assoc(A,V) - assoc(A,A)}{assoc(A,V)} + \frac{assoc(B,V) - assoc(B,B)}{assoc(B,V)} \\ &= 2 - Nassoc(A,B) \end{aligned}$$

- Problem of minimizing $Ncut(A,B)$ is same as maximize the $Nassoc(A,B)$.
- Which make sense, as minimizing disassociation between the groups and maximize the association within the group identical.

Computation of minimum cut

- Convert *Ncut* equation (1) into metrices using following method:

$$Ncut(A,B) = \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(A,B)}{assoc(B,V)}$$
$$= \frac{\sum_{x_i>0,x_j<0} -w_{ij}x_ix_j}{\sum_{x_i>0} d_i} + \frac{\sum_{x_i<0,x_j>0} -w_{ij}x_ix_j}{\sum_{x_i<0} d_i}$$

  where x is an N dimensional indicator vector such that $x_i = 1$ if i is in A, and $x_i = -1$ if i is in B. Degree of node i: $d_i = \sum_j w_{ij}$

- Degree matrix: Let D be and $N \times N$ diagonal matrix, with $d_i = \sum_j w_{ij}$
- Affinity matrix or Weight matrix or Adjacent matrix: Let W be and $N \times N$ symmetric matrix with $W(i,j) = w_{i,j}$
- Then *Ncut* can be simplified by

$$\min_x Ncut(x) = \min_x \frac{x^T L x}{x^T D x} \qquad \text{subject to } x^T D x = 1$$

  where the Laplacian matrix $L = D - W$

Computation of minimum cut

- This Optimization problem can be solved by solving generalized eigenvalue equation:

$$Lx = \lambda x$$

- Note: the first eigenvector is x_1, with the eigenvalue $\lambda_1 = 0$.(we discard it)
- We pick the second smallest eigenvector λ_2 which is the solution of our problem.
- **NCuts Matlab code available at** https://www.cis.upenn.edu/~jshi/software/
  - Data Clustering with Normalized Cuts
  - Image Segmentation with Normalized Cuts

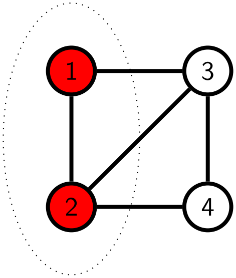
Weighted grpah

- For every vertex $v_i \in V$, the degree $d(v_i)$ of v_i is the sum of the weights of the edges adjacent to v_i

$$d(v_i) = \sum_{j=1}^{n} w_{ij}$$

- **Degree matrix:** $D = diag(d_1, d_2, \ldots, d_n)$, where $d_i = d(v_i)$
- Given subset of vertices $S \subset V$, we define the volume by

$$vol(A) = \sum_{v_i \in A} d(v_i) = \sum_{v_i \in A} \left(\sum_{j=1}^{n} w_{ij} \right)$$

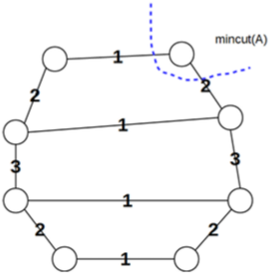


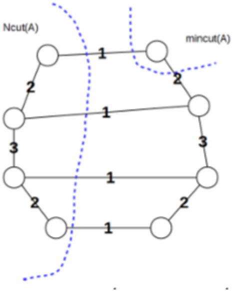
- If $vol(A) = 0$, all the vertices in A are isolated.
- If $V = \{v_1, v_2\}$, then

$$\begin{aligned} vol(A) &= d(v_1) + d(v_2) \\ &= (w_{12} + w_{13}) + (w_{21} + w_{23} + w_{24}) \end{aligned}$$

- **Remarks:**
  - **cut(A)** measures how many edges escape from A;
  - **assoc(A, A)** measures how many edges stay within A;
  - $cut(A, B) + assoc(A, A) = vol(A)$

How to calculate Mincut and Ncut

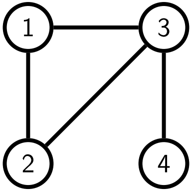




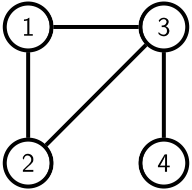
- Here $\mathrm{mincut}(A, B) = 1 + 2 = 3$
- Ncut:

$$Ncut(A, B) = \frac{cut(A, B)}{assoc(A, V)} + \frac{cut(A, B)}{assoc(B, V)}$$
$$= \frac{4}{3+6+6+3} + \frac{4}{3+6+6+3} = \frac{4}{9}$$

Graph Laplacian matrix



Graph Laplacian matrix



$$D = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, W = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}, L = D - W = \begin{bmatrix} 2 & -1 & -1 & 0 \\ -1 & 2 & -1 & 0 \\ -1 & -1 & 3 & -1 \\ 0 & 0 & -1 & 1 \end{bmatrix}$$

- Here $V = \{1, 2, 3, 4\}$ and E = set of edges

Properties of the Laplacian

Lemma

Graph Laplacian is always semi positive definite.

Proof. Need to show $x^T Lx \geq 0$ for any $x \in \mathbb{R}^n$

$$
\begin{aligned}
x^T Lx &= x^T Dx - x^T Wx \\
&= \sum_{i=1} d_i x_i^2 - \sum_{i,j} w_{ij} x_i x_j \\
&= \sum_{i=1} \sum_{j=1} w_{ij} x_i^2 - \sum_{i,j} w_{ij} x_i x_j \\
&= \sum_{i,j} \frac{w_{ij}}{2} (x_i^2 + x_j^2) - \sum_{i,j} w_{ij} x_i x_j \\
&= \frac{1}{2} \sum_{ij} w_{ij} (x_i - x_j)^2
\end{aligned}
$$

- For every vector $x \in \mathbb{R}^n$, and $w_{ij} = w_{ji} \geq 0$,

$$x^T Lx = \frac{1}{2} \sum_{ij} w_{ij} (x_i - x_j)^2 \geq 0.$$

Lemma

The smallest eigenvalue is 0 with eigenvector equal to a constant vector.

Proof.

$$L\mathbf{1} = D\mathbf{1} - W\mathbf{1} = d - d = 0,$$

where $d = [d_1, d_2, \ldots, d_n]^T$.

Or,

$$x^T L x = \frac{1}{2} \sum_{ij} w_{ij} (x_i - x_j)^2.$$

For eigenvalue $\lambda = 0$, $Lx = \lambda x = 0x = 0$, which gives

$$0 = x^T L x = \frac{1}{2} \sum_{ij} w_{ij} (x_i - x_j)^2 = 0.$$

Since $w_{ij} > 0$; they are connected, $x_i = x_j$ for all i, j. So eigenvector x is a constant vector. For undirected graph, the graph is connected by a path.

- This is why only the second smallest eigenvector is needed when grouping the data into two partitions.

Fiedler Method

Lemma

If G is a simple connected graph with n vertices and L is the Laplacian matrix for G then L has n- real eigenvalues satisfying

$$0 = \lambda_1 < \lambda_2 \leq \lambda_3 \leq \cdots \leq \lambda_n.$$

The Fiedler Value gives a measurement as to how well connected the graph is.

Definition (Fiedler Value)

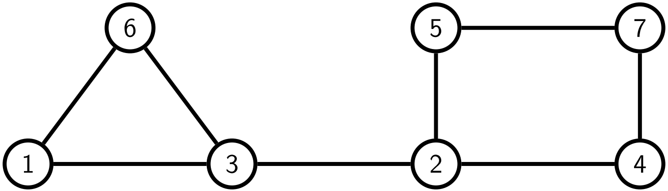
The Fiedler Value or the algebraic connectivity of a graph is the second smallest eigenvalue of its Laplacian matrix L.

Definition (Fiedler Vector)

A Fiedler Vector of a graph is an eigenvector correspond- ing to the Fiedler Value.

Notice: the eigenspace corresponding to the Fiedler Value may be multidimensional.

Example



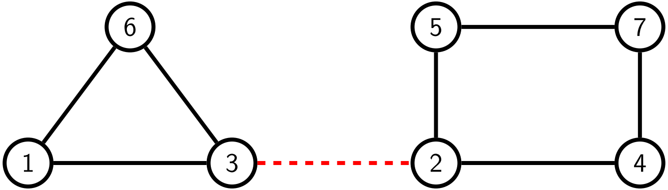
$$D = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 \end{bmatrix}, \; W = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}, \; L = \begin{bmatrix} 2 & 0 & -1 & 0 & 0 & -1 & 0 \\ 0 & 3 & -1 & -1 & -1 & 0 & 0 \\ -1 & -1 & 3 & 0 & 0 & -1 & 0 \\ 0 & -1 & 0 & 2 & 0 & 0 & -1 \\ 0 & -1 & 0 & 0 & 2 & 0 & -1 \\ -1 & 0 & -1 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & -1 & -1 & 0 & 2 \end{bmatrix}$$

Eigenvalues: 0, 0.3588, 2.0000, 2.2763, 3.0000, 3.5892, 4.7757

Fiedler value: $\lambda_2 = 0.3588$

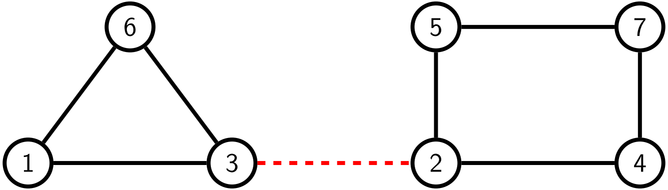
Fiedler vector: $v_2 = [0.48, \; -0.15, \; 0.31, \; -0.35, \; -0.35, \; 0.48, \; 0.42]^T$

- Fiedler Method: we can achieve a "reasonable" partition into two subgraphs by separating the vertices according to the sign of the values in a Fiedler Vector v_2, where each entry corresponds to a vertex.
- This means we group together the vertices i with $v_i = +\text{sign}$, and we group together the vertices i with $v_i = -\text{sign}$.
- In the case that $v_i = 0$, for some i, we simply have to make a choice.
- By "reasonable" we mean that an attempt is made to remove as few edges as possible while keeping the resulting subgraphs of approximately equal size.



Fiedler vector: $v_2 = [0.48,\ -0.15,\ 0.31,\ -0.35,\ -0.35,\ 0.48, -0.42]^T$

- Fiedler Method: we can achieve a "reasonable" partition into two subgraphs by separating the vertices according to the sign of the values in a Fiedler Vector v_2, where each entry corresponds to a vertex.
- This means we group together the vertices i with v_i = +sign, and we group together the vertices i with v_i = −sign.
- In the case that $v_i = 0$, for some i, we simply have to make a choice.
- By "reasonable" we mean that an attempt is made to remove as few edges as possible while keeping the resulting subgraphs of approximately equal size.



Fiedler vector: $v_2 = [0.48,\ -0.15,\ 0.31,\ -0.35,\ -0.35,\ 0.48, -0.42]^T$

- +sign $A = \{1, 3, 6\}$ and -sign $B = \{2, 4, 5, 7\}$
- This means we separate the vertices accordingly.

What are We Wishing For?

- Ideally for a partition $P = (A, B)$ of a graph G we would like to minimize $cut(P) = cut(A, B)$ while keeping $|A| \approx |B|$.
- To formalize this: consider $x \in \mathbb{R}^n$ with $x_i = \pm 1$.
- Having such a vector we can then create a partition by taking the vertices i with $x_i = +1$ as one subset and the vertices i with $x_i = -1$. More formally,

$$P = \big(\{i \,:\, x_i = +1\}, \{i \,:\, x_i = -1\}\big)$$

- Keeping the sizes of the subsets equal amounts to having $\sum_{i=1}^{n} x_i = 0$, and keeping them close amounts to having $\sum_{i=1}^{n} x_i \approx 0$.

Cut partition

Lemma

For any partition $P = (A, B)$ of a graph G with edge set E we have, then

$$cut(P) = \frac{1}{4} \sum_{(i,j) \in E} (x_i - x_j)^2.$$

Proof. Consider that

$$\begin{aligned}
\sum_{(i,j) \in E} (x_i - x_j)^2 &= \sum_{\substack{(i,j) \in E \\ x_i = -x_j}} (x_i - x_j)^2 + \sum_{\substack{(i,j) \in E \\ x_i = x_j}} (x_i - x_j)^2 \\
&= \sum_{\substack{(i,j) \in E \\ x_i = -x_j}} (\pm 2)^2 + \sum_{\substack{(i,j) \in E \\ x_i = x_j}} (0)^2 \\
&= 4cut(P).
\end{aligned}$$

Note: The $\frac{1}{4}$ doesn't matter for minimizing so the goal can be rephrased as trying to minimize $\sum_{(i,j) \in E} (x_i - x_j)^2$ with the conditions that $\sum_{i=1}^{n} x_i \approx 0$ and $x_i = \pm 1$.

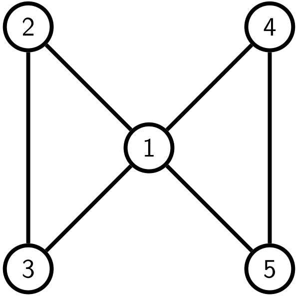
More and Trickier Examples

- There is a 0 in the Fiedler Vector.

More and Trickier Examples

- There is a 0 in the Fiedler Vector.
- Repeated values in the Fiedler Vector might yield choices.
- We might choose a k-partition with $k > 2$.
- The eigenspace corresponding to the Fiedler Value has dimension greater than 1.

0 in the Fiedler Vector



The Laplacian matrix

$$L = \begin{bmatrix} 4 & -1 & -1 & -1 & -1 \\ -1 & 2 & -1 & 0 & 0 \\ -1 & -1 & 2 & 0 & 0 \\ -1 & 0 & 0 & 2 & -1 \\ -1 & 0 & 0 & -1 & 2 \end{bmatrix}$$

0 in the Fiedler Vector contd.

- Eigenvalues: $0,\ 1,\ 3,\ 3,\ 5$
- Fiedler value: 1
- A Fiedler Vector: $[0, -0.5, -0.5, 0.5, 0.5]^T$
- It's clear both from the graph and from the vector that the 1 vertex is difficult to categorize.
- Even though the Fiedler Method doesn't explicitly tell us what to do with that vertex the way that the values are spread out makes our options fairly clear.

0 in the Fiedler Vector contd.

- Eigenvalues: 0, 1, 3, 3, 5
- Fiedler value: 1
- A Fiedler Vector: $[0, -0.5, -0.5, 0.5, 0.5]^T$
- It's clear both from the graph and from the vector that the 1 vertex is difficult to categorize.
- Even though the Fiedler Method doesn't explicitly tell us what to do with that vertex the way that the values are spread out makes our options fairly clear.
- We can either partition as $A = \{2, 3, 1\}, B = \{4, 5\}$, or $A = \{2, 3\}, B = \{4, 5, 1\}$

Theorem

The Fiedler vector $x = v_2$ solves the binary spectral clustering problem:

$$\textit{Minimize } x^T L x \textit{ over } x \in \mathbb{R}^n, \textit{ subjected to } \mathbf{1}^T x = 0 \textit{ and } ||x||^2 = 1.$$

Proof. Let x be a minimizer.

- We can write

$$x = a_1 v_1 + a_2 v_2 + \cdots + a_n v_n$$

where $v_1 = \frac{\mathbf{1}}{\sqrt{n}}$, $v_i^T v_j = 0$ and $||v_i|| = 1$.

- We have

$$\begin{aligned} 0 = \mathbf{1}^T x = \mathbf{1}^T (a_1 v_1 + a_2 v_2 + \cdots + a_n v_n) &= a_1 \mathbf{1}^T v_1 + a_2 \mathbf{1}^T v_2 + \cdots + a_n \mathbf{1}^T v_n \\ &= \frac{\sqrt{n}}{\sqrt{n}} (a_1 \mathbf{1}^T v_1 + a_2 \mathbf{1}^T v_2 + \cdots + a_n \mathbf{1}^T v_n) = \sqrt{n}(a_1 v_1^T v_1 + a_2 v_1^T v_2 + \cdots + a_n v_1^T v_n) \end{aligned}$$

$$0 = \sqrt{n} a_1 ||v_1||^2 \implies a_1 = 0$$

- Again we have

$$1 = ||x||^2 = \sum_{i=1}^{n} a_i^2 = \sum_{i=2}^{n} a_i^2$$

- Therefore,

$$\begin{aligned} x^T L x &= x^T L \sum_{i=2}^{n} a_i v_i = x^T \sum_{i=2}^{n} a_i L v_i = x^T \sum_{i=2}^{n} a_i \lambda_i v_i \\ &= \sum_{i=2}^{n} a_i \lambda_i x^T v_i = \sum_{i=2}^{n} \lambda_i a_i^2 \\ &\geq \lambda_2 \sum_{i=2}^{n} a_i^2 = \lambda_2 \end{aligned}$$

- Setting $a_2 = 1, a_3 = a_4 = \cdots = 0$, $x = v_2$. Thus $x^T L x = \lambda_2$.

k-way spectral clustering

- How do we partition a graph into k clusters ?

  Recursive bi-partitioning

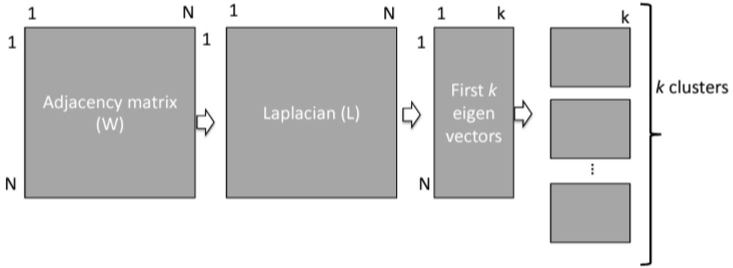
  - Recursively apply bi-partitioning algorithm in a hierarchical divisive manner.
  - Disadvantages: Inefficient, unstable

  Cluster multiple eigenvectors (Notes)

  - Build a reduced space from multiple eigenvectors.
  - k eigenvectors in a natural way to cluster a graph into k clusters.
  - Commonly used in recent papers
  - A preferable approach

k-way spectral clustering

- Given graph G
- Find graph Laplacian $L = D - W$
- Obtain the k eigen vectors associated with k smallest eigen values of L
- Represent each node as the k-dimensional vector
- Cluster nodes based on k-means clustering (Notes)



K-mean vs Spectral clustering

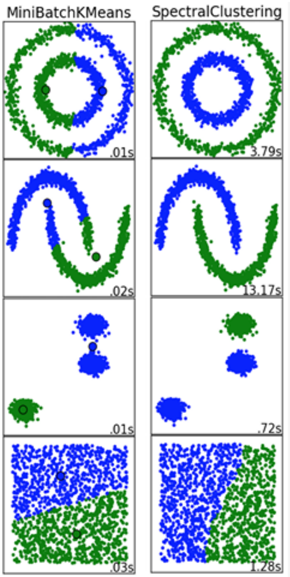
K-Means

- FAST
- Will fail sometimes
- Not very useful on anisotropic data

Spectral clustering (More detailed)

- Excellent quality under many different data forms
- Much slower than KMeans

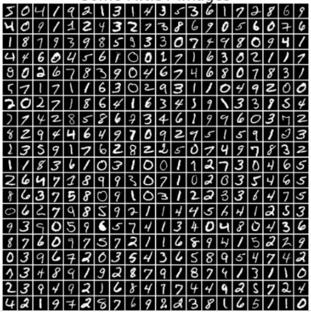
(Python Code: Colab notebook)



Applications

- Spectral Clustering in Machine Learning (Python: Click here)
- Spectral clustering on MNIST (Python: Click here)

Some MNIST Images



- Spectral clustering image segmentation (Python, R : Click here)
- **NCuts Matlab code available at** https://www.cis.upenn.edu/~jshi/software/
  - Data Clustering
  - Image Segmentation