Spectral Clustering

Course Webpage:
http://www.cs.cornell.edu/Courses/cs4786/2016fa/
Survey

• There will be 2 surveys and the final course eval
  • If overall class participation is above 90% on all 3 I will drop all your worst assignments

• Survey one posted on CMS due by 28th sep

• Surveys are all completely anonymous and will help me make the class more fun. So be open.
Spectral Clustering

- Cluster nodes in a graph.
- Analysis of social network data.
Spectral Clustering

Input: Similarity matrix

$$A_{i,j} = \begin{cases} 1 & \text{if } (i,j) \in E \\ 0 & \text{otherwise} \end{cases}$$

A is adjacency matrix of a graph
Example

Cut as few edges as possible
Spectral Clustering

Input: Similarity matrix $A_{i,j} = A_{j,i} > 0$ indicates similarity between elements $x_i$ and $x_j$.

Example:

$$A_{i,j} = \exp(-d(x_i, x_j))$$

$A$ is the adjacency matrix of a graph.

$$L = D - A$$

$$D_{i,i} = \sum_{j=1}^{n} A_{i,j}$$
Example
Graph Clustering: Cuts

- Partition nodes so that as few edges are cut (Mincut)
- What has this got to do with the Laplacian matrix?
Consider case when we have/want 2 clusters. Let $c_j = -1$ if $x_j$ belongs to cluster 0 and $c_j = 1$ if $x_j$ belongs to cluster 1

$$\text{CUT} = \sum_{(i,j) \in E} \mathbf{1}_{c_i \neq c_j} = \frac{1}{2} c^\top L c$$
Cuts and Laplacian

\[
\text{Cut}(c) = \sum_{(i,j) \in E} 1_{c_i \neq c_j} = \sum_{i=1}^{n} \sum_{j=1}^{n} A_{i,j} 1_{c_i \neq c_j}
\]

\[
\sum_{i=1}^{n} \sum_{j=1}^{n} A_{i,j} \frac{1}{4} (c_i - c_j)^2
\]

\[
= \sum_{i=1}^{n} \sum_{j=1}^{n} A_{i,j} \frac{1}{4} (c_i^2 + c_j^2 - 2c_i c_j)
\]

\[
= \frac{1}{4} \left( \sum_{i=1}^{n} \left( \sum_{j=1}^{n} A_{i,j} \right) c_i^2 + \sum_{j=1}^{n} \left( \sum_{i=1}^{n} A_{i,j} \right) c_j^2 - 2 \sum_{i=1}^{n} \sum_{j=1}^{n} A_{i,j} c_i c_j \right)
\]

\[
= \frac{1}{4} \left( 2 \sum_{i=1}^{n} D_{i,i} c_i^2 - 2 \sum_{i=1}^{n} \sum_{j=1}^{n} A_{i,j} c_i c_j \right)
\]

\[
= \frac{1}{2} (c^\top Dc - c^\top Ac) = \frac{1}{2} c^\top Lc
\]
Hence to find the solution we need to solve for

\[
\text{Minimize } c^\top Lc \quad \text{ s.t. } \forall i \in [n], |c_i| = 1
\]

Since \( \forall i \in [n], |c_i| = 1 \), we have \( \|c\|_2 = \sqrt{n} \) and so relaxing (approximating) the optimization:

\[
\text{Minimize } c^\top Lc \quad \text{ s.t. } \|c\|_2 = \sqrt{n}
\]

Hence solution \( c \) to above is an Eigen vector, first smallest one is the all 1’s vector (for connected graph), second smallest one is our solution

To get clustering assignment we simply threshold at 0
Solution obtained by considering the second smallest up to $K^{th}$ smallest eigenvectors

If instead of $c_i = \pm 1$ make for each $k \in [K]$, $c_i^k$ to be indicator of whether point $i$ belongs to cluster $K$ or not, then

$$\text{Cut} = \sum_{k=1}^{K} (c^k)^\top Lc^k$$
Spectral Clustering Algorithm (Unnormalized)

1. Given matrix $A$ calculate diagonal matrix $D$ s.t. $D_{i,i} = \sum_{j=1}^{n} A_{i,j}$

2. Calculate the Laplacian matrix $L = D - A$

3. Find eigen vectors $v_1, \ldots, v_n$ of $L$ (ascending order of eigenvalues)

4. Pick the $K$ eigenvectors with smallest eigenvalues to get $y_1, \ldots, y_n \in \mathbb{R}^K$

5. Use K-means clustering algorithm on $y_1, \ldots, y_n$

$y_1, \ldots, y_n$ are called spectral embedding

Embeds the $n$ nodes into $K-1$ dimensional vectors
Min-cut on a graph can be efficiently computed

Why bother with the approximate algorithm

Is cut even a good measure?
Why cut is perhaps not a good measure?

Normalized Cut

\[ \text{NCUT} = \sum_{j} \text{CUT}(C_j) \]

Example \( K = 2 \)

\[ \text{CUT}(C_1, C_2) \]

Minimize \( \text{CUT}(C_1, C_2) \) s.t. \( \text{Edges}(C_1) = \text{Edges}(C_2) \)
Why cut is perhaps not a good measure?

Fixes?
Why cut is perhaps not a good measure?

Fixes? Perhaps \textbf{Ratio Cut}: \( \text{CUT}(C_1, C_2) \left( \frac{1}{|C_1|} + \frac{1}{|C_2|} \right) \)
• Normalized cut: Minimize sum of ratio of number of edges cut per cluster and number of edges within cluster

\[
\text{NCUT} = \sum_j \frac{\text{CUT}(C_j)}{\text{Edges}(C_j)}
\]

Example

\[
\text{CUT}(C_1, C_2) = \text{Edges}(C_1) + \text{Edges}(C_2)
\]

This is an NP hard problem! so relax

\[
\text{Edges}(C_i) = \text{degree}(C_i) = \sum_{t \in C_i} D_{t,t}
\]
Normalized cut: Minimize sum of ratio of number of edges cut per cluster and number of edges within cluster

\[ \text{NCUT} = \sum_j \frac{\text{CUT}(C_j)}{\text{Edges}(C_j)} \]

- Example \( K = 2 \)

\[ \text{CUT}(C_1, C_2) \left( \frac{1}{\text{Edges}(C_1)} + \frac{1}{\text{Edges}(C_2)} \right) \]

- This is an NP hard problem! … so relax
First note that \( \text{Edges} \left( C_i \right) = \sum_{k : x_k \in C_i} D_k \),

Set \( c_i = \begin{cases} \sqrt{\frac{\text{Edges}(C_2)}{\text{Edges}(C_1)}} & \text{if } i \in C_1 \\ -\sqrt{\frac{\text{Edges}(C_1)}{\text{Edges}(C_2)}} & \text{otherwise} \end{cases} \)

Verify that \( c^T L c = |E| \times \text{NCut} \) and \( c^T D c = |E| \) (and \( D c \perp 1 \))

Hence we relax Minimize \( \text{NCUT}(C) \) to

\[
\text{Minimize } \frac{c^T L c}{c^T D c} \quad \text{s.t. } D c \perp 1
\]

Solution: Find second smallest eigenvectors of \( \tilde{L} = I - D^{-1/2} A D^{-1/2} \)
Spectral Clustering Algorithm (Normalized)

1. Given matrix $A$ calculate diagonal matrix $D$ s.t. $D_{i,i} = \sum_{j=1}^{n} A_{i,j}$

2. Calculate the normalized Laplacian matrix $\tilde{L} = I - D^{-1/2} A D^{-1/2}$

3. Find eigen vectors $v_1, \ldots, v_n$ of $\tilde{L}$ (ascending order of eigenvalues)

4. Pick the $K$ eigenvectors with smallest eigenvalues to get $y_1, \ldots, y_n \in \mathbb{R}^K$

5. Use K-means clustering algorithm on $y_1, \ldots, y_n$
Normalized Cut: Alternate View

- If we perform random walk on graph, its the partition of graph into group of vertices such that the probability of transiting from one group to another is minimized

- Transition matrix: \( D^{-1}A \)

- Largest eigenvalues and eigenvectors of above matrix correspond to smallest eigenvalues and eigenvectors of \( D^{-1}L = I - D^{-1}A \)

- For \( K \)-nearest neighbor graph (K-regular), same as normalized Laplacian