• Back to graphs - define graph Laplaceans

• Properties of graph Laplaceans

• Graph partitioning –

• Introduction to clustering
“Laplace-type” matrices associated with general undirected graphs – useful in many applications

Given a graph $G = (V, E)$ define

- A matrix $W$ of weights $w_{ij}$ for each edge
- Assume $w_{ij} \geq 0$, $w_{ii} = 0$, and $w_{ij} = w_{ji} \forall (i, j)$
- The diagonal matrix $D = \text{diag}(d_i)$ with $d_i = \sum_{j \neq i} w_{ij}$

Corresponding graph Laplacean of $G$ is:

$$L = D - W$$

Gershgorin’s theorem $\rightarrow L$ is positive semidefinite.
Simplest case:

\[ w_{ij} = \begin{cases} 
1 & \text{if } (i, j) \in E & i \neq j \\
0 & \text{else} 
\end{cases} \]

\[ D = \text{diag} \left[ d_i = \sum_{j \neq i} w_{ij} \right] \]

**Example:**
Consider the graph

\[ L = \begin{pmatrix}
1 & -1 & 0 & 0 & 0 \\
-1 & 2 & 0 & 0 & -1 \\
0 & 0 & 1 & 0 & -1 \\
0 & 0 & 0 & 1 & -1 \\
0 & -1 & -1 & -1 & 3
\end{pmatrix} \]
Define the graph Laplacean for the graph associated with the simple mesh shown next. [use the simple weights of 0 or 1]. What is the difference with the discretization of the Laplace operator for case when mesh is the same as this graph?

**Proposition:**

(i) $L$ is symmetric semi-positive definite.

(ii) $L$ is singular with 1 as a null vector.

(iii) If $G$ is connected, then $\text{Null}(L) = \text{span}\{1\}$

(iv) If $G$ has $k > 1$ connected components $G_1, G_2, \cdots, G_k$, then the nullity of $L$ is $k$ and $\text{Null}(L)$ is spanned by the vectors $z^{(j)}, j = 1, \cdots, k$ defined by:

$$(z^{(j)})_i = \begin{cases} 1 & \text{if } i \in G_j \\ 0 & \text{if not.} \end{cases}$$
Proof: (i) and (ii) seen earlier and are trivial. (iii) Clearly \( u = 1 \) is a null vector for \( L \). The vector \( D^{-1/2}u \) is an eigenvector for the matrix \( D^{-1/2}LD^{-1/2} = I - D^{-1/2}WD^{-1/2} \) associated with the smallest eigenvalue. It is also an eigenvector for \( D^{-1/2}WD^{-1/2} \) associated with the largest eigenvalue. By the Perron Frobenius theorem this is a simple eigenvalue... (iv) Can be proved from the fact that \( L \) can be written as a direct sum of the Laplacian matrices for \( G_1, \cdots, G_k \). \( \blacksquare \)
A few properties of graph Laplacians

Define: oriented incidence matrix $H$: (1) First orient the edges $i \sim j$ into $i \rightarrow j$ or $j \rightarrow i$. (2) Rows of $H$ indexed by vertices of $G$. Columns indexed by edges. (3) For each $(i, j)$ in $E$, define the corresponding column in $H$ as $\sqrt{w(i, j)}(e_i - e_j)$.

Example: In previous example (P. 11-3) orient $i \rightarrow j$ so that $j > i$ [lower triangular matrix representation]. Then matrix $H$ is:

$$H = \begin{bmatrix}
1 & 0 & 0 & 0 \\
-1 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
0 & -1 & -1 & -1
\end{bmatrix}$$

Property 1: $L = HH^T$

Re-prove part (iv) of previous proposition by using this property.
A few properties of graph Laplaceans

Strong relation between $x^T L x$ and local distances between entries of $x$

Let $L$ = any matrix s.t. $L = D - W$, with $D = diag(d_i)$ and

$$w_{ij} \geq 0, \quad d_i = \sum_{j \neq i} w_{ij}$$

Property 2: for any $x \in \mathbb{R}^n$:

$$x^T L x = \frac{1}{2} \sum_{i,j} w_{ij} |x_i - x_j|^2$$
Property 3: (generalization) for any $Y \in \mathbb{R}^{d \times n}$:

$$\text{Tr} [Y L Y^\top] = \frac{1}{2} \sum_{i,j} w_{ij} \|y_i - y_j\|^2$$

Note: $y_j = j$-th column of $Y$. Usually $d < n$. Each column can represent a data sample.

Property 4: For the particular $L = I - \frac{1}{n} \mathbf{1} \mathbf{1}^\top$

$$X L X^\top = \bar{X} \bar{X}^\top = n \times \text{Covariance matrix}$$

Property 5: $L$ is singular and admits the null vector

$\mathbf{1} = \text{ones}(n,1)$
Property 6: (Graph partitioning) Consider situation when $w_{ij} \in \{0, 1\}$. If $x$ is a vector of signs ($\pm 1$) then

$$x^\top Lx = 4 \times (\text{‘number of edge cuts’})$$

edge-cut = pair $(i, j)$ with $x_i \neq x_j$

Consequence: Can be used to partition graphs

Would like to minimize $(Lx, x)$ subject to $x \in \{-1, 1\}^n$ and $e^T x = 0$ [balanced sets]
WII solve a relaxed form of this problem

What if we replace \( x \) by a vector of ones (representing one partition) and zeros (representing the other)?

Let \( x \) be any vector and \( y = x + \alpha \mathbb{1} \) and \( L \) a graph Laplacean. Compare \( (Lx, x) \) with \( (Ly, y) \).
Consider any symmetric (real) matrix $A$ with eigenvalues $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ and eigenvectors $u_1, \cdots, u_n$.

Recall that:

(Min reached for $x = u_1$)

$$\min_{x \in \mathbb{R}^n} \frac{(Ax, x)}{(x, x)} = \lambda_1$$

In addition:

(Min reached for $x = u_2$)

$$\min_{x \perp u_1} \frac{(Ax, x)}{(x, x)} = \lambda_2$$

For a graph Laplacean $u_1 = \mathbb{1} = \text{vector of all ones}$ and

...vector $u_2$ is called the Fiedler vector. It solves a relaxed form of the problem -
Define $v = u_2$ then $lab = \text{sign}(v - \text{med}(v))$
Recursive Spectral Bisection

1. Form graph Laplacean
2. Partition graph in 2 based on Fielder vector
3. Partition largest subgraph in two recursively ...
4. ... Until the desired number of partitions is reached
Three approaches to graph partitioning:

2. Geometric techniques. Coordinates are required. [Houstis & Rice et al., Miller, Vavasis, Teng et al.]
3. Graph Theory techniques – multilevel,... [use graph, but no coordinates]
   ● Currently best known technique is Metis (multi-level algorithm)
   ● Simplest idea: Recursive Graph Bisection; Nested dissection (George & Liu, 1980; Liu 1992]
   ● Advantages: simplicity – no coordinates required
Example of a graph theory approach

- Level Set Expansion Algorithm
- Given: \( p \) nodes ‘uniformly’ spread in the graph (roughly same distance from one another).
- Method: Perform a level-set traversal (BFS) from each node simultaneously.
- Best described for an example on a \( 15 \times 15 \) five – point Finite Difference grid.
- See [Goehring-Saad ’94, See Cai-Saad ’95]
- Approach also known under the name ‘bubble’ algorithm and implemented in some packages [Party, DibaP]
Clustering

Problem: we are given $n$ data items: $x_1, x_2, \cdots, x_n$. Would like to ‘cluster’ them, i.e., group them so that each group or cluster contains items that are similar in some sense.

Example: materials

Refer to each group as a ‘cluster’ or a ‘class’

‘Unsupervised learning’
What is Unsupervised learning?

“Unsupervised learning”: methods do not exploit labeled data

- Example of digits: perform a 2-D projection
- Images of same digit tend to cluster (more or less)
- Such 2-D representations are popular for visualization
- Can also try to find natural clusters in data, e.g., in materials
- Basic clustering technique: K-means
Example: Community Detection

- Communities modeled by an ‘affinity’ graph [e.g., 'user A sends frequent e-mails to user B']
- Adjacency Graph represented by a sparse matrix

Goal: Find ordering so blocks are as dense as possible →

Use ‘blocking’ techniques for sparse matrices
Advantage of this viewpoint: need not know # of clusters.

[data: www-personal.umich.edu/~mejn/netdata/]

9-19 – Glaplacians
Example of application

Data set from:

http://www-personal.umich.edu/~mejn/netdata/

- Network connecting bloggers of different political orientations [2004 US presidential election]
- ‘Communities’: liberal vs. conservative
- Graph: 1,490 vertices (blogs): first 758: liberal, rest: conservative.
- Edge: $i \rightarrow j$: a citation between blogs $i$ and $j$
- Blocking algorithm (Density thershold=0.4): subgraphs [note: density $= \frac{|E|}{|V|^2}$.]
- Smaller subgraph: conservative blogs, larger one: liberals