Community Detection: Modularity Optimization & Spectral Clustering
Modularity of partitioning $S$ of graph $G$:

- $Q \propto \sum_{s \in S} \left[ (\text{# edges within group } s) - (\text{expected # edges within group } s) \right]$

$$Q(G, S) = \frac{1}{2m} \sum_{s \in S} \left[ \sum_{i,j \in s} A_{ij} - \sum_{i,j \in s} \frac{k_i k_j}{2m} \right]$$

$$= \frac{1}{2m} \sum_{s \in S} \sum_{i \in s} \sum_{j \in s} \left( A_{ij} - \frac{k_i k_j}{2m} \right)$$

Normalizing const.: $-1 < Q < 1$

Modularity $Q$ tells us whether $S$ represents any significant community structure

- So, let’s find $S$ that maximizes modularity itself!
Method 2: Modularity Optimization

- Let’s split the graph into 2 communities
- Want to directly optimize modularity:
  \[
  \max_S Q(G, S) = \frac{1}{2m} \sum_{s \in S} \sum_{i \in s} \sum_{j \in s} (A_{ij} - \frac{k_i k_j}{2m})
  \]

- Community membership vector \( s \):
  - \( s_i = 1 \) if node \( i \) is in community 1
  - \( s_i = -1 \) if node \( i \) is in community -1

\[
Q(G, s) = \frac{1}{2m} \sum_{i \in N} \sum_{j \in N} \left( A_{ij} - \frac{k_i k_j}{2m} \right) \frac{s_i s_j + 1}{2}
= \frac{1}{4m} \sum_{i, j \in N} \left( A_{ij} - \frac{k_i k_j}{2m} \right) s_i s_j
\]
Define:

- Modularity matrix: $B_{ij} = A_{ij} - \frac{k_i k_j}{2m}$
- Membership: $s = \{-1, +1\}$

Then: $Q(G, s) = \frac{1}{4m} \sum_{i \in N} \sum_{j \in N} \left( A_{ij} - \frac{k_i k_j}{2m} \right) s_i s_j$

$= \frac{1}{4m} \sum_{i,j \in N} B_{ij} s_i s_j$

$= \frac{1}{4m} \sum_i s_i \sum_j B_{ij} s_j = \frac{1}{4m} s^T B s$

$= B_{ii} \cdot s$

Task: Find $s \in \{-1, +1\}^n$ that maximizes $Q(G, s)$

Note: each row/col of $B$ sums to 0: $\sum_j A_{ij} = k_i$, $\sum_j \frac{k_i k_j}{2m} = k_i$ but $\sum_j \frac{k_j}{2m} = k_i$
Symmetric matrix $A$
- is Positive Semidefinite: $A = U \cdot U^T$
- Then solutions $\lambda, x$ to equation $A \cdot x = \lambda \cdot x$ :
  - Eigenvectors $x_i$ ordered by the magnitude of their corresponding eigenvalues $\lambda_i$ ($\lambda_1 \leq \lambda_2 \ldots \leq \lambda_n$)
  - $x_i$ are orthonormal (orthogonal and unit length)
  - $x_i$ form a coordinate system (basis)
  - If $A$ is Positive Semidefinite: $\lambda_i \geq 0$ (and they always exist)
- Eigendecomposition theorem: Can rewrite matrix $A$ in terms of its eigenvectors and eigenvalues:
  $$A = \sum_i x_i \cdot \lambda_i \cdot x_i^T$$
Modularity Optimization

- **Rewrite:** \( Q(G, s) = \frac{1}{4m} s^T B s \) in terms of its eigenvectors \( x \) and eigenvalues \( \lambda \):

\[
\begin{align*}
  &= s^T \left[ \sum_{i=1}^{n} x_i \lambda_i x_i^T \right] s \\
  &= \sum_{i=1}^{n} s^T x_i \lambda_i x_i^T s \\
  &= \sum_{i=1}^{n} \left( s^T x_i \right)^2 \lambda_i
\end{align*}
\]

- So, if there would be no other constraints on \( s \) then to maximize \( Q \), we make \( s = x_n \)

  - **Why?** Because \( \lambda_n \geq \lambda_{n-1} \geq \cdots \)
    - Remember \( s \) has fixed length (\( ||s|| = 1 \))!
    - Assigns all weight in the sum to \( \lambda_n \) (largest eigenvalue)
    - All other \( s^T x_i \) terms are zero because of orthonormality
Let’s consider only the first term in the summation (because $\lambda_n$ is the largest):

$$\max\limits_s Q(G, s) = \sum_{i=1}^n (s^T x_i)^2 \lambda_i \approx (s^T x_n)^2 \lambda_n$$

Let’s maximize: $\sum_{j=1}^n s_j \cdot x_{n,j}$ where $s_j \in \{-1, +1\}$

To do this, we set:

- $s_j = \begin{cases} +1 & \text{if } x_{n,j} \geq 0 \\ -1 & \text{if } x_{n,j} < 0 \end{cases}$ (j−th coordinate of $x_n \geq 0$)

- Continue the bisection hierarchically
Summary: Modularity Optimization

- Fast Modularity Optimization Algorithm:
  - Find leading eigenvector $x_n$ of modularity matrix $B$
  - Divide the nodes by the signs of the elements of $x_n$
  - Repeat hierarchically until:
    - If a proposed split does not cause modularity to increase, declare community indivisible and do not split it
    - If all communities are indivisible, stop

- How to find $x_n$? Power method!
  - Start with random $v^{(0)}$, repeat:
  - When converged ($v^{(t)} \approx v^{(t+1)}$), set $x_n = v^{(t)}$

$$v^{(t+1)} = \frac{Bv^{(t)}}{|Bv^{(t)}|}$$
Girvan-Newman (previous lecture):
- Based on the “strength of weak ties”
- Remove edge of highest betweenness

Modularity:
- Overall quality of the partitioning of a graph
- Use to determine the number of communities

Fast Modularity Optimization:
- Transform the modularity optimization into an eigenvalue problem
Spectral Clustering for Graph Partitioning
Three basic stages:

1) **Pre-processing**
   - Construct a matrix representation of the graph

2) **Decomposition**
   - Compute eigenvalues and eigenvectors of the matrix
   - Map each point to a lower-dimensional representation based on one or more eigenvectors

3) **Grouping**
   - Assign points to two or more clusters, based on the new representation

But first, let’s define the problem
Graph Partitioning

- **Undirected graph** $G(V, E)$:

- **Bi-partitioning task:**
  - Divide vertices into two disjoint groups $A, B$

- **Questions:**
  - How can we define a “good” partition of $G$?
  - How can we efficiently identify such a partition?
What makes a good partition?

- Maximize the number of within-group connections
- Minimize the number of between-group connections
Graph Cuts

- Express partitioning objectives as a function of the “edge cut” of the partition

- **Cut:** Set of edges with only one vertex in a group:

\[
cut(A, B) = \sum_{i \in A, j \in B} w_{ij}
\]

A

[Diagram showing two sets A and B with edges between them.]

cut(A, B) = 2
**Graph Cut Criterion**

- **Criterion:** Minimum-cut
  - Minimize weight of connections between groups
    \[
    \arg \min_{A,B} \text{cut}(A,B)
    \]
- **Degenerate case:**
- **Problem:**
  - Only considers external cluster connections
  - Does not consider internal cluster connectivity
Graph Cut Criteria

- **Criterion:** Conductance [Shi-Malik, ’97]
  - Connectivity between groups relative to the density of each group
    \[
    \phi(A, B) = \frac{\text{cut}(A, B)}{\min(\text{vol}(A), \text{vol}(B))}
    \]
    \[
    \text{vol}(A): \text{total weight of the edges with at least one endpoint in } A: \text{vol}(A) = \sum_{i \in A} k_i
    \]

  - Why use this criterion?
    - Produces more balanced partitions

- How do we efficiently find a good partition?
  - **Problem:** Computing optimal cut is NP-hard
A: adjacency matrix of undirected G

- $A_{ij} = 1$ if $(i, j)$ is an edge, else 0
- $x$ is a vector in $\mathbb{R}^n$ with components $(x_1, \ldots, x_n)$
- Think of it as a label/value of each node of $G$

What is the meaning of $A \cdot x$?

$$
\begin{bmatrix}
    a_{11} & \cdots & a_{1n} \\
    \vdots & \ddots & \vdots \\
    a_{n1} & \cdots & a_{nn}
\end{bmatrix}
\begin{bmatrix}
    x_1 \\
    \vdots \\
    x_n
\end{bmatrix} =
\begin{bmatrix}
    y_1 \\
    \vdots \\
    y_n
\end{bmatrix}
$$

$$
y_i = \sum_{j=1}^{n} A_{ij} x_j = \sum_{(i,j) \in E} x_j
$$

- Entry $y_i$ is a sum of labels $x_j$ of neighbors of $i$
What is the meaning of $Ax$?

- $j^{th}$ coordinate of $A \cdot x$:
  - Sum of the $x$-values of neighbors of $j$
  - Make this a new value at node $j$

- Spectral Graph Theory:
  - Analyze the “spectrum” of matrix representing $G$
  - **Spectrum**: Eigenvectors $x_i$ of a graph, ordered by the magnitude (strength) of their corresponding eigenvalues $\lambda_i$: $\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_n\}$

  $$\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$$

  Note: We sort $\lambda_i$ in ascending (not descending) order!

$$A \cdot x = \lambda \cdot x$$
Example: d-regular graph

- Suppose all nodes in $G$ have degree $d$ and $G$ is connected
- What are some eigenvalues/vectors of $G$?

$$Ax = \lambda x$$

What is $\lambda$? What $x$?

- Let’s try: $x = (1, 1, \ldots, 1)$
- Then: $Ax = (d, d, \ldots, d) = \lambda x$. So: $\lambda = d$
- We found an eigenpair of $G$: $x = (1, 1, \ldots, 1)$, $\lambda = d$

Remember the meaning of $y = Ax$:

Note, this is just one eigenpair. An $n$ by $n$ matrix can have up to $n$ eigenpairs.
$d$ is the largest eigenvalue of $A$

- $G$ is $d$-regular connected, $A$ is its adjacency matrix
- **Claim:**
  - $d$ is the largest eigenvalue of $A$,
  - $d$ has multiplicity of 1 (there is only 1 eigenvector associated with eigenvalue $d$)
- **Proof:** *Why no eigenvalue $d' > d$?*
  - To obtain $d$ we needed $x_{1i} = x_{1j}$ for every $i, j$
  - This means $x_1 = c \cdot (1,1, \ldots, 1)$ for some const. $c$
  - **Define:** $S =$ nodes $i$ with maximum possible value of $x_{1i}$
  - Then consider some vector $y$ which is not a multiple of vector $(1, \ldots, 1)$. So not all nodes $i$ (with labels $y_i$) are in $S$
  - Consider some node $j \in S$ and a neighbor $i \notin S$ then node $j$ gets a value strictly less than $d$
  - So $y$ is not eigenvector! And so $d$ is the largest eigenvalue!
Example: Graph on 2 components

- **What if** $G$ is not connected?
  - $G$ has 2 components, each $d$-regular
- **What are some eigenvectors?**
  - $x = \text{Put all 1s on } A \text{ and 0s on } B \text{ or vice versa}$
    - $x' = (1, \ldots, 1, 0, \ldots, 0)$ then $A \cdot x' = (d, \ldots, d, 0, \ldots, 0)$
    - $x'' = (0, \ldots, 0, 1, \ldots, 1)$ then $A \cdot x'' = (0, \ldots, 0, d, \ldots, d)$
    - And so in both cases the corresponding $\lambda = d$

- **A bit of intuition:**
  - $\lambda_n = \lambda_{n-1}$
  - $\lambda_n - \lambda_{n-1} \approx 0$

$2^{\text{nd}}$ largest eigval. $\lambda_{n-1}$ now has value very close to $\lambda_n$
More intuition:

- If the graph is connected (right example) then we already know that $x_n = (1, \ldots, 1)$ is an eigenvector.
- Since eigenvectors are orthogonal then the components of $x_{n-1}$ sum to 0.
  - Why? Because $x_n \cdot x_{n-1} = \sum_i x_n[i] \cdot x_{n-1}[i]$.
- So we can look at the eigenvector of the 2nd largest eigenvalue and declare nodes with positive label in $A$ and negative label in $B$.
- But there is still lots to sort out.
### Adjacency matrix \( A \):

- \( n \times n \) matrix
- \( A=[a_{ij}], a_{ij}=1 \) if edge between node \( i \) and \( j \)

### Important properties:

- Symmetric matrix
- Eigenvectors are real and orthogonal
Degree matrix (D):

- $n \times n$ diagonal matrix
- $D = [d_{ii}]$, $d_{ii} =$ degree of node $i$
Matrix Representations

- Laplacian matrix \((L)\):
  - **\(n \times n\)** symmetric matrix
  - \[
    L = D - A
  \]

- What is trivial eigenpair?
  - \(x = (1, \ldots, 1)\) then \(L \cdot x = 0\) and so \(\lambda = \lambda_1 = 0\)

- Important properties:
  - **Eigenvalues** are non-negative real numbers
  - **Eigenvectors** are real and orthogonal

11/12/2014
3 Facts about the Laplacian $L$

(a) All eigenvalues are $\geq 0$
(b) $x^T Lx = \sum_{ij} L_{ij} x_i x_j \geq 0$ for every $x$
(c) $L = N^T \cdot N$

- That is, $L$ is positive semi-definite

**Proof: (the 3 facts are saying the same thing)**

- (c)$\Rightarrow$(b): $x^T Lx = x^T N^T N x = (x N)^T (N x) \geq 0$
  - As it is just the square of length of length of $N x$

- (b)$\Rightarrow$(a): Let $\lambda$ be an eigenvalue of $L$. Then by (b)
  $x^T Lx \geq 0$ so $x^T Lx = x^T \lambda x = \lambda x^T x \Rightarrow \lambda \geq 0$

- (a)$\Rightarrow$(c): is also easy! Do it yourself.
### $\lambda_2$ as optimization problem

- **Fact:** For symmetric matrix $M$:

  $$\lambda_2 = \min_x \frac{x^T M x}{x^T x}$$

- **What is the meaning of $\min x^T L x$ on $G$?**

  - $x^T L x = \sum_{i,j=1}^n L_{ij} x_i x_j = \sum_{i,j=1}^n (D_{ij} - A_{ij}) x_i x_j$
  
  - $= \sum_i D_{ii} x_i^2 - \sum_{(i,j) \in E} 2x_i x_j$

  - $= \sum_{(i,j) \in E} (x_i^2 + x_j^2 - 2x_i x_j) = \sum_{(i,j) \in E} (x_i - x_j)^2$

  Node $i$ has degree $d_i$. So, value $x_i^2$ needs to be summed up $d_i$ times. But each edge $(i,j)$ has two endpoints so we need $x_i^2 + x_j^2$.

---

11/12/2014

Proof:

\[
\lambda_2 = \min_x \frac{x^T M x}{x^T x}
\]

- Write \(x\) in axes of eigenvects \(w_1, w_2, \ldots, w_n\) of \(M\).
  So, \(x = \sum_i^n \alpha_i w_i\)
- Then we get: \(Mx = \sum_i \alpha_i Mw_i = \sum_i \alpha_i \lambda_i w_i\)
- So, what is \(x^T M x\)?
  \[
x^T M x = (\sum_i \alpha_i w_i)(\sum_i \alpha_i \lambda_i w_i) = \sum_{ij} \alpha_i \lambda_j \alpha_j w_i w_j
  = \sum_i \alpha_i \lambda_i w_i w_i = \sum_i \lambda_i \alpha_i^2
  \]
  To minimize this over all unit vectors \(x\) orthogonal to: \(w = \min\) over choices of \((\alpha_1, \ldots, \alpha_n)\) so that:
  \[\sum \alpha_i^2 = 1\text{ (unit length)}\]
  \[\sum \alpha_i = 0\text{ (orthogonal to }w_1)\]
  To minimize this, set \(\alpha_2 = 1\) and so \(\sum \lambda_i \alpha_i^2 = \lambda_2\)
### λ₂ as optimization problem

**What else do we know about x?**

- \(x\) is unit vector: \(\sum_i x_i^2 = 1\)
- \(x\) is orthogonal to 1\(^{st}\) eigenvector \((1, \ldots, 1)\) thus: \(\sum_i x_i \cdot 1 = \sum_i x_i = 0\)

**Remember:**

\[
\lambda_2 = \min \sum_{(i,j) \in E} (x_i - x_j)^2 \quad \text{subject to} \quad \sum_i x_i = 0
\]

We want to assign values \(x_i\) to nodes \(i\) such that few edges cross 0.
(we want \(x_i\) and \(x_j\) to subtract each other)

Balance to minimize
Find Optimal Cut [Fiedler’73]

- Back to finding the optimal cut
- Express partition \((A,B)\) as a vector
  \[
  y_i = \begin{cases} 
  +1 & \text{if } i \in A \\
  -1 & \text{if } i \in B
  \end{cases}
  \]
- We can minimize the cut of the partition by finding a non-trivial vector \(x\) that minimizes:
  \[
  \arg \min_{y \in [-1,+1]^n} f(y) = \sum_{(i,j) \in E} (y_i - y_j)^2
  \]

Can’t solve exactly. Let’s relax \(y\) and allow it to take any real value.
Rayleigh Theorem

\[
\min_{y \in \mathbb{R}^n} f(y) = \sum_{(i,j) \in E} (y_i - y_j)^2 = y^T Ly
\]

- \( \lambda_2 = \min_y f(y) \): The minimum value of \( f(y) \) is given by the 2\(^{nd} \) smallest eigenvalue \( \lambda_2 \) of the Laplacian matrix \( L \)
- \( x = \arg \min_y f(y) \): The optimal solution for \( y \) is given by the corresponding eigenvector \( x \), referred as the Fiedler vector
Suppose there is a partition of $G$ into $A$ and $B$ where $|A| \leq |B|$, s.t. $\alpha = \frac{(\# \text{ edges from } A \text{ to } B)}{|A|}$ then $\lambda_2 \leq 2\alpha$

This is the approximation guarantee of the spectral clustering: Spectral finds a cut that has at most twice the conductance as the optimal one of conductance $\alpha$.

Proof:

- Let: $a=|A|$, $b=|B|$ and $e=\# \text{ edges from } A \text{ to } B$
- Enough to choose some $x_i$ based on $A$ and $B$ such that:
  \[
  \lambda_2 \leq \frac{\sum (x_i-x_j)^2}{\sum_i x_i^2} \leq 2\alpha \quad \text{(while also } \sum_i x_i = 0)\]

Note: $|A|<|B|$
Proof (continued):

1) Let’s set: 
\[ x_i = \begin{cases} 
-\frac{1}{a} & \text{if } i \in A \\
\frac{1}{b} & \text{if } i \in B 
\end{cases} \]

Let’s quickly verify that \( \sum x_i = 0 \):
\[ a \left( -\frac{1}{a} \right) + b \left( \frac{1}{b} \right) = 0 \]

2) Then:
\[ \frac{\sum (x_i - x_j)^2}{\sum x_i^2} = \frac{\sum_{i \in A, j \in B} (\frac{1}{b} + \frac{1}{a})^2}{a \left( -\frac{1}{a} \right)^2 + b \left( \frac{1}{b} \right)^2} = e \cdot \frac{(\frac{1}{a} + \frac{1}{b})^2}{\frac{1}{a} + \frac{1}{b}} = \]
\[ e \left( \frac{1}{a} + \frac{1}{b} \right) \leq e \left( \frac{1}{a} + \frac{1}{a} \right) = e \frac{2}{a} \leq 2 \alpha \]

Which proves that the cost achieved by spectral is better than twice the OPT cost.

\[ e \ldots \text{number of edges between } A \text{ and } B \]
Putting it all together: The Cheeger inequality

\[ \frac{\alpha^2}{2k_{\text{max}}} \leq \lambda_2 \leq 2\alpha \]

where \( k_{\text{max}} \) is the maximum node degree in the graph

- Note we only provide the 1\text{st} part: \( \lambda_2 \leq 2\alpha \)
- We did not prove \( \frac{\alpha^2}{2k_{\text{max}}} \leq \lambda_2 \)
- Overall this always certifies that \( \lambda_2 \) always gives a useful bound
So far...

- **How to define a “good” partition of a graph?**
  - Minimize a given graph cut criterion

- **How to efficiently identify such a partition?**
  - Approximate using information provided by the eigenvalues and eigenvectors of a graph

- **Spectral Clustering**
Three basic stages:

1) **Pre-processing**
   - Construct a matrix representation of the graph

2) **Decomposition**
   - Compute eigenvalues and eigenvectors of the matrix
   - Map each point to a lower-dimensional representation based on one or more eigenvectors

3) **Grouping**
   - Assign points to two or more clusters, based on the new representation
1) **Pre-processing:**
- Build Laplacian matrix $L$ of the graph

2) **Decomposition:**
- Find eigenvalues $\lambda$ and eigenvectors $x$ of the matrix $L$
- Map vertices to corresponding components of $\lambda_2$

How do we now find the clusters?
3) **Grouping:**
- Sort components of reduced 1-dimensional vector
- Identify clusters by splitting the sorted vector in two

**How to choose a splitting point?**
- Naïve approaches:
  - Split at 0 or median value
- More expensive approaches:
  - Attempt to minimize normalized cut in 1-dimension (sweep over ordering of nodes induced by the eigenvector)

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.3</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.6</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>-0.3</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>-0.3</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>-0.6</td>
<td></td>
</tr>
</tbody>
</table>

**Split at 0:**
- **Cluster A:** Positive points
- **Cluster B:** Negative points
Example: Spectral Partitioning
Example: Spectral Partitioning

Components of $x_2$

Value of $x_2$

Rank in $x_2$
Example: Spectral partitioning

Components of $x_1$

Components of $x_3$
How do we partition a graph into $k$ clusters?

Two basic approaches:

- **Recursive bi-partitioning** [Hagen et al., ’92]
  - Recursively apply bi-partitioning algorithm in a hierarchical divisive manner
  - Disadvantages: Inefficient, unstable

- **Cluster multiple eigenvectors** [Shi-Malik, ’00]
  - Build a reduced space from multiple eigenvectors
  - Commonly used in recent papers
  - A preferable approach...
Why use multiple eigenvectors?

- **Approximates the optimal cut** [Shi-Malik, ’00]
  - Can be used to approximate optimal $k$-way normalized cut
- **Emphasizes cohesive clusters**
  - Increases the unevenness in the distribution of the data
  - Associations between similar points are amplified, associations between dissimilar points are attenuated
  - The data begins to “approximate a clustering”
- **Well-separated space**
  - Transforms data to a new “embedded space”, consisting of $k$ orthogonal basis vectors
  - Multiple eigenvectors prevent instability due to information loss
Many other partitioning methods

- **METIS:**
  - Heuristic but works really well in practice
  - [http://glaros.dtc.umn.edu/gkhome/views/metis](http://glaros.dtc.umn.edu/gkhome/views/metis)

- **Graclus:**
  - Based on kernel k-means

- **Louvain:**
  - Based on Modularity optimization

- **Clique percorlation method:**
  - For finding overlapping clusters
  - [http://angel.elte.hu/cfinder/](http://angel.elte.hu/cfinder/)