Community Detection: Modularity Optimization & Spectral Clustering

CS224W: Social and Information Network Analysis
Jure Leskovec, Stanford University
http://cs224w.stanford.edu

Modularity

- **Modularity of partitioning S of graph G:**
  - Q $\propto \sum_{s \in S}$ [(# edges within group s) – (expected # edges within group s)]
  - $$Q(G,S) = \frac{1}{2m}\sum_{s\in S}\left[\sum_{i,j\in s} A_{ij} - \sum_{i,j\in s}\frac{k_i k_j}{2m}\right]$$
    $$= \underbrace{\frac{1}{2m}}\sum_{s\in S}\sum_{i\in s}\sum_{j\in s}\left(A_{ij} - \frac{k_i k_j}{2m}\right)$$

Normalizing const.: -1<Q<1

A_{ij} = 1 if i→j, 0 else

- **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:**
  - $\max_S Q(G,S) = \frac{1}{2m}\sum_{s\in S}\sum_{i\in s}\sum_{j\in s}\left(A_{ij} - \frac{k_i k_j}{2m}\right)$
- **Community membership vector s:**
  - s_i **= 1** if node ***i*** is in community **1**
    -1 if node ***i*** is in community **-1**

$\frac{s_i s_j + 1}{2}$ = 1.. if $s_i = s_j$, 0.. else

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

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

Modularity Matrix

Note: each row/col of **B** sums to **0:** $\sum_j A_{ij} = k_i$, $\sum_j \frac{k_i k_j}{2m} = k_i \sum_j \frac{k_j}{2m} = k_i$

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

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

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

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

Quick Review of Linear Algebra

$$\begin{bmatrix} a_{11} & \dots & a_{1n} \\ \vdots & & \vdots \\ a_{n1} & \dots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \lambda \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$$

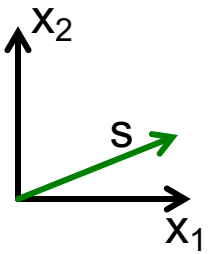
- **Symmetric** matrix **A**
  - is Positive Semidefinite: $\boldsymbol{A = U \cdot U^T}$
- Then solutions $\boldsymbol{\lambda, x}$ to equation $\boldsymbol{A \cdot x = \lambda \cdot x}$:
  - **Eigenvectors** $\boldsymbol{x_i}$ ordered by the magnitude of their corresponding **eigenvalues** λ_i ($\lambda_1 \leq \lambda_2 \dots \leq \lambda_n$)
  - $\boldsymbol{x_i}$ are **orthonormal** (orthogonal and unit length)
  - $\boldsymbol{x_i}$ form a coordinate system (basis)
  - If $\boldsymbol{A}$ is Positive Semidefinite: $\lambda_i \geq 0$ (and they always exist)
- **Eigendecomposition theorem:** Can rewrite matrix $\boldsymbol{A}$ in terms of its eigenvectors and eigenvalues: $\boldsymbol{A = \sum_i x_i \cdot \lambda_i \cdot x_i^T}$

Modularity Optimization

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

$$= s^{\mathrm{T}} \left[\sum_{i=1}^{n} x_i \lambda_i x_i^T \right] s = \sum_{i=1}^{n} s^T x_i \lambda_i x_i^T s = \sum_{i=1}^{n} (s^T \mathrm{x}_i)^2 \lambda_i$$

- **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 λ_n (largest eigenvalue)
      - All other $s^T x_i$ terms are **zero** because of orthonormality



Finding the vector *s*

- **Let's consider only the first term in the summation (because λ_n is the largest):**

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

- **Let's maximize:** $\sum_{j=1}^{n} s_j \cdot x_{n,j}$ where $s_j \in \{-1,+1\}$
- To do this, we set:
  - $s_j = \begin{cases} +1 & if\ x_{n,j} \geq 0 \ \ (\text{j—th coordinate of } x_n \geq 0) \\ -1 & if\ x_{n,j} < 0 \ \ (\text{j—th coordinate of } x_n < 0) \end{cases}$
- **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 $x_n = v^{(t)}$

$$v^{(t+1)} = \frac{Bv^{(t)}}{\left\| Bv^{(t)} \right\|}$$

Summary: Modularity

- **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

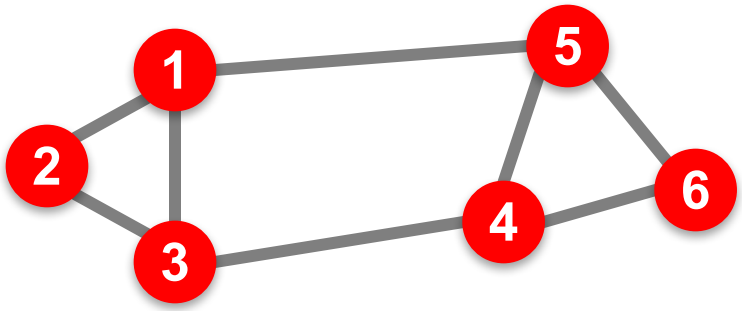
Spectral Clustering for Graph Partitioning

Spectral Clustering Algorithms

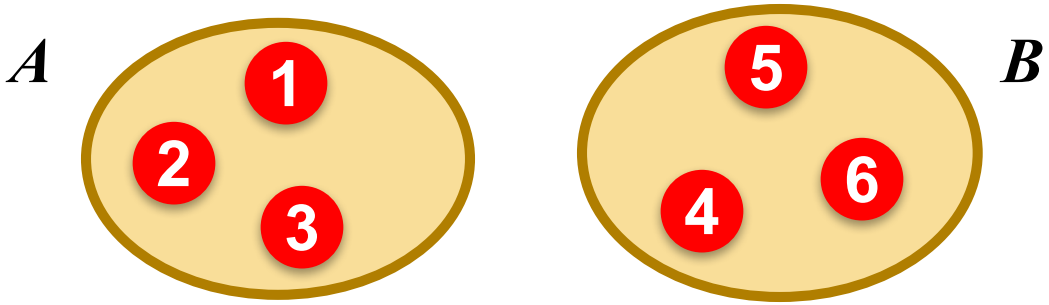
- **Three basic stages:**
  - **1) Pre-processing**
    - Construct a matrix representation of the graph
  - **2) Decomposition**
    - Compute eigenvalues and eigenvectors of the matrix
    - Map each point to a lower-dimensional representation based on one or more eigenvectors
  - **3) Grouping**
    - Assign points to two or more clusters, based on the new representation
- But first, let's define the problem

Graph Partitioning

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



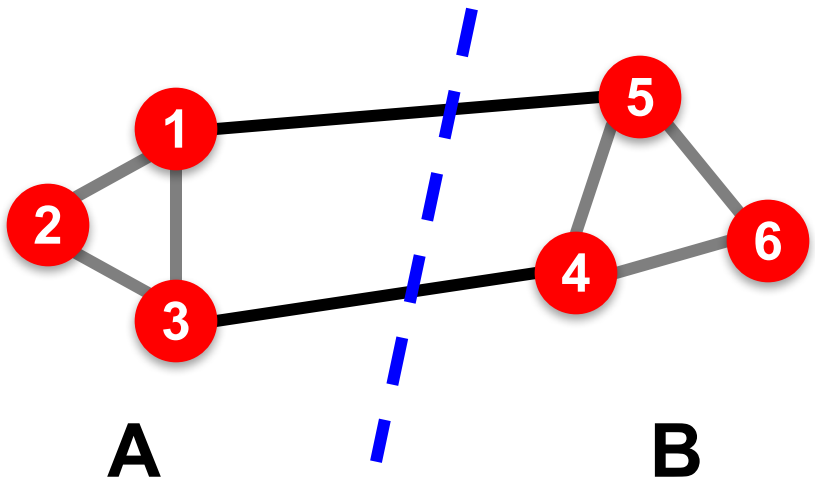
- **Bi-partitioning task:**
  - Divide vertices into two disjoint groups A, B



- **Questions:**
  - How can we define a "good" partition of G?
  - How can we efficiently identify such a partition?

Graph Partitioning

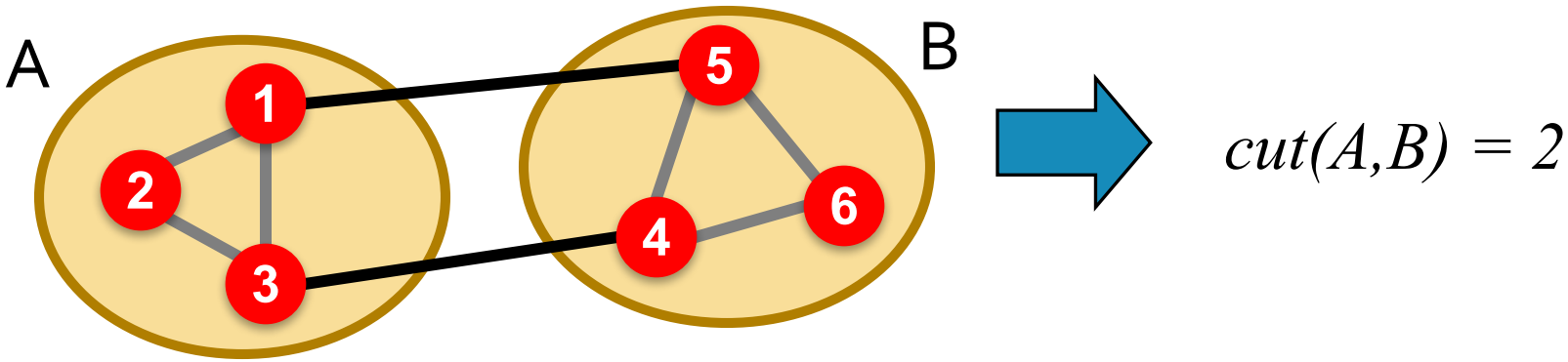
- **What makes a good partition?**
  - Maximize the number of within-group connections
  - Minimize the number of between-group connections



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:

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

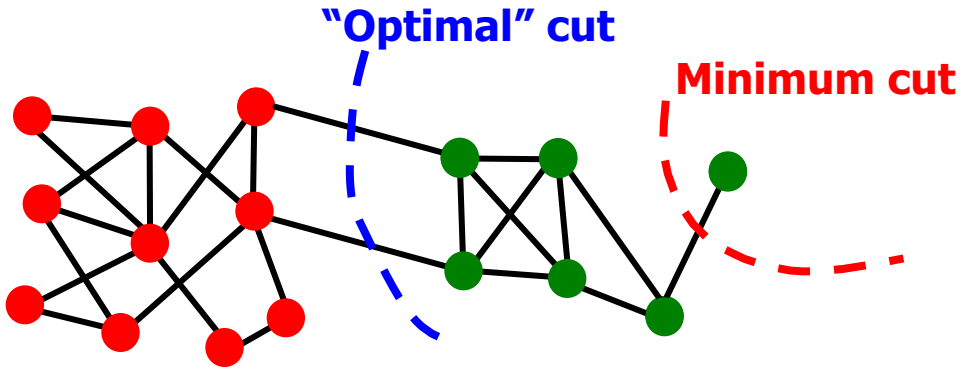


Graph Cut Criterion

- **Criterion: Minimum-cut**
  - Minimize weight of connections between groups

$$\arg\min_{A,B} cut(A,B)$$

- **Degenerate case:**



- **Problem:**
  - Only considers external cluster connections
  - Does not consider internal cluster connectivity

[Shi-Malik]

Graph Cut Criteria

- **Criterion: Conductance** [Shi-Malik, '97]
  - Connectivity between groups relative to the density of each group

$$\phi(A,B) = \frac{cut(A,B)}{\min(vol(A), vol(B))}$$

$\boldsymbol{vol(A)}$: total weighted degree of the nodes in $\boldsymbol{A}$: $\boldsymbol{vol(A)} = \sum_{i \in A} \boldsymbol{k_i}$ (number of edge end points in A)

- **Why use this criterion?**
  - Produces more balanced partitions
- **How do we efficiently find a good partition?**
  - **Problem:** Computing optimal cut is NP-hard

Spectral Graph Partitioning

- A: adjacency matrix of undirected **G**
  - A_{ij} **=1** if (i, j) is an edge, else **0**
- x is a vector in $\Re^n$ with components $(x_1, \ldots, x_n)$
  - Think of it as a label/value of each node of G
- **What is the meaning of $A \cdot x$?**

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

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

- **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

$$\begin{bmatrix} a_{11} & \dots & a_{1n} \\ \vdots & & \vdots \\ a_{n1} & \dots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \lambda \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$$

$$A \cdot x = \lambda \cdot x$$

- **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 λ_i: $\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_n\}$

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

Note: We sort λ_i in ascending (not descending) order!

Example: d-regular graph

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

  $\boldsymbol{A \cdot x = \lambda \cdot x}$ What is λ? What $\boldsymbol{x}$?

  - **Let's try: $\boldsymbol{x = (1, 1, \ldots, 1)}$**
  - **Then: $\boldsymbol{A \cdot x = (d, d, \ldots, d) = \lambda \cdot x}$. So: $\boldsymbol{\lambda = d}$**
  - **We found an eigenpair of G: $\boldsymbol{x = (1, 1, \ldots, 1)}$, $\boldsymbol{\lambda = d}$**

Remember the meaning of $y = A \cdot x$:

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

Note, this is just one eigenpair. An *n* by *n* matrix can have up to *n* eigenpairs.

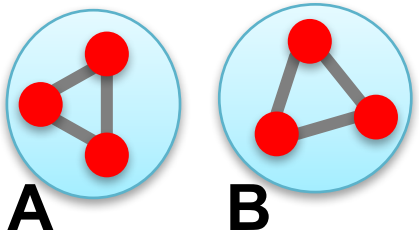
d is the largest eigenvalue of A

Details!

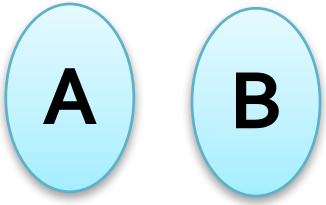
- **G** is **d**-regular connected, **A** is its adjacency matrix
- **Claim:**
  - (1) **d** has multiplicity of **1** (there is only **1** eigenvector associated with eigenvalue **d**)
  - (2) **d** is the largest eigenvalue of **A**
- **Proof:**
  - To obtain $\boldsymbol{d}$ we needed $\boldsymbol{x_{1i} = x_{1j}}$ for every i, j
  - This means $\boldsymbol{x_1} = c \cdot (1,1,\ldots,1)$ for some const. c
  - **Define:** $\boldsymbol{S}$ = nodes $\boldsymbol{i}$ with maximum possible value of $\boldsymbol{x_{1i}}$
  - Then consider some vector $\boldsymbol{y}$ which is not a multiple of vector $(\mathbf{1},\ldots,\mathbf{1})$. So not all nodes $\boldsymbol{i}$ (with labels $\boldsymbol{y_i}$) are in $\boldsymbol{S}$
  - Consider some node $\boldsymbol{j} \in \boldsymbol{S}$ and a neighbor $\boldsymbol{i} \notin \boldsymbol{S}$ then node $\boldsymbol{j}$ gets a value strictly less than $\boldsymbol{d}$
  - So y is not eigenvector! And so $\boldsymbol{d}$ is the largest eigenvalue!

Example: Graph on 2 components

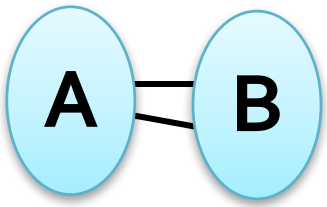
- **What if G is not connected?**
  - G has 2 components, each d-regular



- **What are some eigenvectors?**
  - x = Put all 1s on A and 0s on B or vice versa
    - $x' = (\underbrace{1, \ldots, 1}_{|A|}, \underbrace{0, \ldots, 0}_{|B|})$ 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:**



$\lambda_n = \lambda_{n-1}$

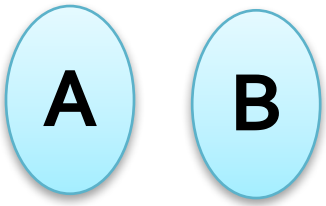


$\lambda_n - \lambda_{n-1} \approx 0$

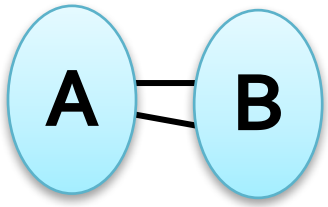
2nd largest eigval. λ_{n-1} now has value very close to λ_n

More Intuition

- **More intuition:**



$\lambda_n = \lambda_{n-1}$



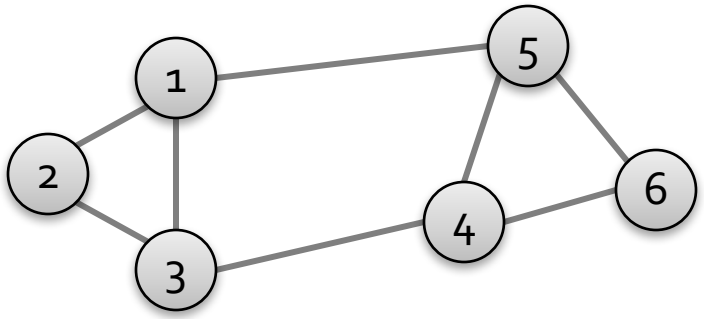
$\lambda_n - \lambda_{n-1} \approx 0$

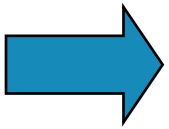
2nd largest eigval. λ_{n-1} now has value very close to λ_n

  - If the graph is connected (right example) then we already know that $\boldsymbol{x_n} = (\mathbf{1}, \ldots \mathbf{1})$ is an eigenvector
  - Since eigenvectors are orthogonal then the components of $\boldsymbol{x_{n-1}}$ sum to **0**.
    - Why? Because 0 = $\boldsymbol{x_n} \cdot \boldsymbol{x_{n-1}} = \sum_i \boldsymbol{x_n}[\boldsymbol{i}] \cdot \boldsymbol{x_{n-1}}[\boldsymbol{i}]$ = $\sum_i \boldsymbol{x_n}\,[\boldsymbol{i}]$
  - 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.**

Matrix Representations

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



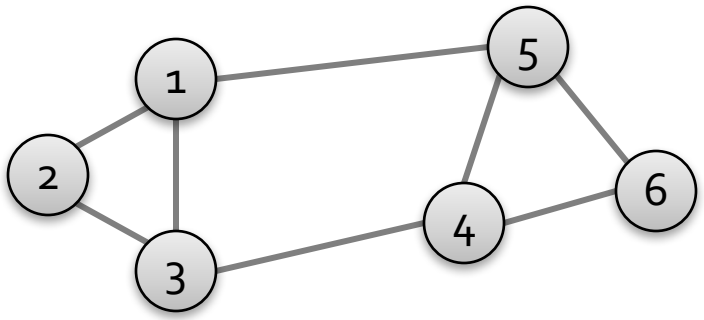


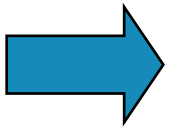
	1	2	3	4	5	6
1	0	1	1	0	1	0
2	1	0	1	0	0	0
3	1	1	0	1	0	0
4	0	0	1	0	1	1
5	1	0	0	1	0	1
6	0	0	0	1	1	0

- **Important properties:**
  - Symmetric matrix
  - Eigenvectors are real and orthogonal

Matrix Representations

- **Degree matrix (D):**
  - $n \times n$ diagonal matrix
  - $D=[d_{ii}]$, d_{ii} = degree of node i

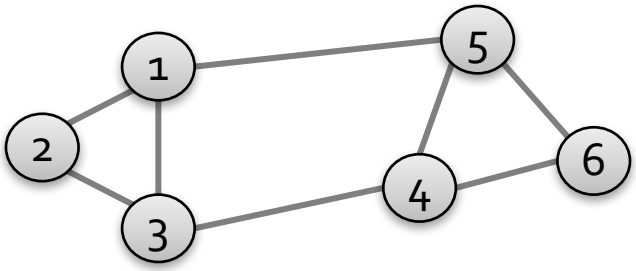




	1	2	3	4	5	6
1	3	0	0	0	0	0
2	0	2	0	0	0	0
3	0	0	3	0	0	0
4	0	0	0	3	0	0
5	0	0	0	0	3	0
6	0	0	0	0	0	2

Matrix Representations

- **Laplacian matrix (L):**
  - $n \times n$ symmetric matrix



	1	2	3	4	5	6
1	3	-1	-1	0	-1	0
2	-1	2	-1	0	0	0
3	-1	-1	3	-1	0	0
4	0	0	-1	3	-1	-1
5	-1	0	0	-1	3	-1
6	0	0	0	-1	-1	2

$$L = D - A$$

- **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

3 Facts about the Laplacian L

Details!

(a) All eigenvalues are ≥ 0

(b) $x^T L x = \sum_{ij} L_{ij} x_i x_j \geq 0$ 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)⇒(b):** $x^T L x = x^T N^T N x = (xN)^T (Nx) \geq 0$
    - As it is just the square of length of Nx
  - **(b)⇒(a):** Let $\boldsymbol{\lambda}$ be an eigenvalue of $\boldsymbol{L}$. Then by **(b)** $x^T L x \geq 0$ so $x^T L x = x^T \lambda x = \lambda x^T x \Rightarrow \boldsymbol{\lambda \geq 0}$
  - **(a)⇒(c):** is also easy! Do it yourself.

λ_2 as optimization problem

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

$$\lambda_2 = \min_{x \,:\, x^T w_1 = 0} \frac{x^T M x}{x^T x}$$

($\mathbf{w_1}$ is eigenvector corresponding to λ_1)

- **What is the meaning of min $x^T L\, x$ on G?**
  - $\mathrm{x^T L\, x} = \sum_{i,j=1}^{n} L_{ij}\, x_i x_j = \sum_{i,j=1}^{n} \left(D_{ij} - A_{ij}\right) x_i x_j$
  - $= \sum_i D_{ii} x_i^2 - \sum_{(i,j)\in E} 2 x_i x_j$
  - $= \sum_{(i,j)\in E} (x_i^2 + x_j^2 - 2x_i x_j) = \sum_{(i,j)\in E} \left(x_i - x_j\right)^2$

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 $x_i^2 + x_j^2$

Proof:

$$\lambda_2 = \min_{x\,:\,x^T w_1 = 0} \frac{x^T M x}{x^T x}$$

Details!

- Write x in basis of eigenvecs $w_1, w_2, \ldots, w_n$ of $\boldsymbol{M}$. So, $x = \sum_i^n \alpha_i w_i$
- Then we get: $Mx = \sum_i \alpha_i \underbrace{M w_i}_{\lambda_i w_i} = \sum_i \alpha_i \lambda_i w_i$
- So, what is $\boldsymbol{x^T M x}$?
  - $x^T M x = (\sum_i \alpha_i w_i)(\sum_i \alpha_i \lambda_i w_i) = \sum_{ij} \alpha_i \lambda_j \alpha_j \overbrace{w_i w_j}^{= 0 \text{ if } i \neq j,\ 1 \text{ otherwise}}$

    $= \sum_i \alpha_i \lambda_i w_i w_i = \sum_i \boldsymbol{\lambda_i \alpha_i^2}$
  - To minimize this over all unit vectors *x* orthogonal to: *w* = min over choices of $(\alpha_1, \ldots \alpha_n)$ so that: $\sum \alpha_i^2 = 1$ (unit length) $\sum \alpha_i = 0$ (orthogonal to w_1)
  - To minimize this, set $\boldsymbol{\alpha_2 = 1}$ and so $\boldsymbol{\sum_i \lambda_i \alpha_i^2 = \lambda_2}$

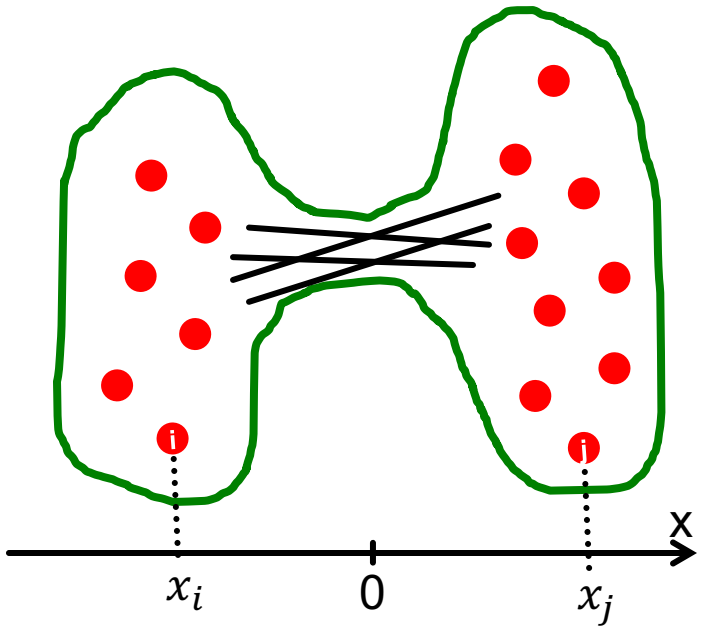
λ_2 as optimization problem

- **What else do we know about x?**
  - x is unit vector: $\sum_i x_i^2 = 1$
  - x is orthogonal to 1st eigenvector $(1, \ldots, 1)$ thus: $\sum_i x_i \cdot 1 = \sum_i x_i = 0$

- **Remember:**

$$\lambda_2 = \min_{\substack{\text{All labelings}\\ \text{of nodes } i \text{ so}\\ \text{that } \sum x_i = 0}} \frac{\sum_{(i,j)\in E} (x_i - x_j)^2}{\sum_i x_i^2}$$

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]

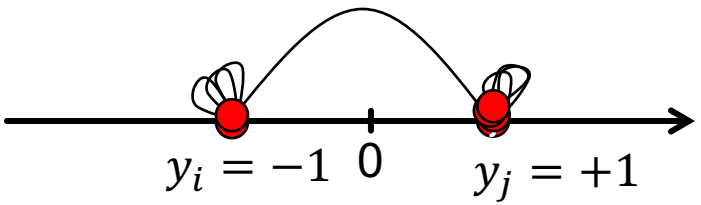
- **Back to finding the optimal cut**
- **Express partition (A,B) as a vector**

$$y_i = \begin{cases} +1 & if\ i \in A \\ -1 & if\ i \in B \end{cases}$$

- Enforce that $|\mathbf{A}| = |\mathbf{B}| \rightarrow \Sigma_j y_i = 0$ (equivalent to being orthogonal to the trivial eigenvector $(1, \ldots, 1)$)
- We can minimize the cut of the partition by finding a non-trivial vector x that **minimizes**:

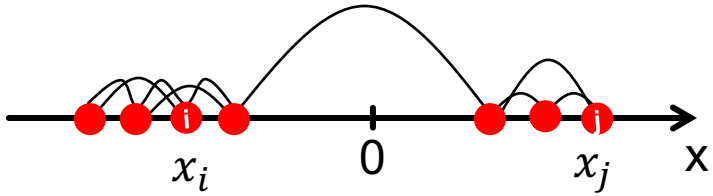
$$\arg\min_{y \in [-1,+1]^n} f(y) = \sum_{(i,j) \in E} (y_i - y_j)^2$$

Can't solve exactly. Let's relax y and allow it to take any real value.



Rayleigh Theorem

$$\min_{y \in \mathbb{R}^n \,:\, \sum_i y_i = 0} f(y) = \sum_{(i,j) \in E} (y_i - y_j)^2 = y^T L y$$



- $\boldsymbol{\lambda_2 = \min_y f(y)}$: The minimum value of $\boldsymbol{f(y)}$ is given by the 2nd smallest eigenvalue $\boldsymbol{\lambda_2}$ of the Laplacian matrix $\boldsymbol{L}$
- $\mathbf{x = \arg\min_y f(y)}$: The optimal solution for $\boldsymbol{y}$ is given by the corresponding eigenvector $\boldsymbol{x}$, referred to as the **Fiedler vector**
- can use sign of x to determine cluster assignment

Approx. Guarantee of Spectral

Details!

- Suppose there is a partition of **G** into **A** and **B** where $|A| \leq |B|$, s.t. $\boldsymbol{\alpha} = \frac{(\#\ edges\ from\ A\ to\ B)}{|A|}$ then $\boldsymbol{\lambda_2 \leq 2\alpha}$

  Note: |A|<|B|

  - 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 $\boldsymbol{\alpha}$.
- **Proof:**
  - Let: **a=|A|, b=|B|** and **e=** # edges from **A** to **B**
  - Enough to choose some $\boldsymbol{x_i}$ based on **A** and **B** such that:

$$\lambda_2 \leq \frac{\sum (x_i - x_j)^2}{\sum_i x_i^2} \leq 2\alpha \quad \text{(while also } \sum_i x_i = 0\text{)}$$

λ_2 is only smaller

Approx. Guarantee of Spectral

Details!

- **Proof (continued):**
  - **1) Let's set:** $x_i = \begin{cases} -\frac{1}{a} & \text{if } i \in A \\ +\frac{1}{b} & \text{if } i \in B \end{cases}$

    Note: |A|<|B|
    - Let's quickly verify that $\sum_i x_i = 0$: $a\left(-\frac{1}{a}\right) + b\left(\frac{1}{b}\right) = 0$
  - **2) Then:** $\frac{\sum (x_i - x_j)^2}{\sum_i x_i^2} = \frac{\sum_{i \in A, j \in B} \left(\frac{1}{b} + \frac{1}{a}\right)^2}{a\left(-\frac{1}{a}\right)^2 + b\left(\frac{1}{b}\right)^2} = \frac{e \cdot \left(\frac{1}{a} + \frac{1}{b}\right)^2}{\frac{1}{a} + \frac{1}{b}} =$

    $e\left(\frac{1}{a} + \frac{1}{b}\right) \leq e\left(\frac{1}{a} + \frac{1}{a}\right) = e\frac{2}{a} \leq 2\alpha$

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

    e … number of edges between A and B

Approx. Guarantee of Spectral

Details!

- **Putting it all together: The Cheeger inequality**

$$\frac{\alpha^2}{2k_{max}} \leq \lambda_2 \leq 2\alpha$$

  - where k_{max} is the maximum node degree in the graph
    - Note we only provide the 1st part: $\lambda_2 \leq 2\alpha$
    - We did not prove $\frac{\alpha^2}{2k_{max}} \leq \lambda_2$
  - Overall this always certifies that λ_2 always gives a useful bound

So far...

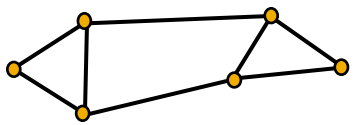
- **How to define a "good" partition of a graph?**
  - Minimize a given graph cut criterion
- **How to efficiently identify such a partition?**
  - Approximate using information provided by the eigenvalues and eigenvectors of a graph
- **Spectral Clustering**

Spectral Clustering Algorithms

- **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

Spectral Partitioning Algorithm

- **1) Pre-processing:**
  - Build Laplacian matrix L of the graph



	1	2	3	4	5	6
1	3	-1	-1	0	-1	0
2	-1	2	-1	0	0	0
3	-1	-1	3	-1	0	0
4	0	0	-1	3	-1	-1
5	-1	0	0	-1	3	-1
6	0	0	0	-1	-1	2

- **2) Decomposition:**
  - Find eigenvalues λ and eigenvectors x of the matrix L

$\lambda =$

0.0
1.0
3.0
3.0
4.0
5.0

$X =$

0.4	0.3	-0.5	-0.2	-0.4	-0.5
0.4	0.6	0.4	-0.4	0.4	0.0
0.4	0.3	0.1	0.6	-0.4	0.5
0.4	-0.3	0.1	0.6	0.4	-0.5
0.4	-0.3	-0.5	-0.2	0.4	0.5
0.4	-0.6	0.4	-0.4	-0.4	0.0

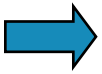
1	0.3
2	0.6
3	0.3
4	-0.3
5	-0.3
6	-0.6

How do we now find the clusters?

Spectral Partitioning

- **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)

1	0.3
2	0.6
3	0.3
4	-0.3
5	-0.3
6	-0.6



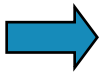
Split at 0:

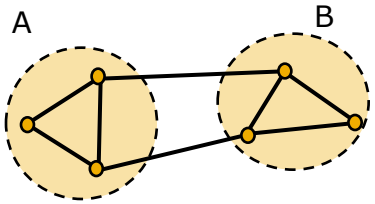
Cluster A: Positive points

Cluster B: Negative points

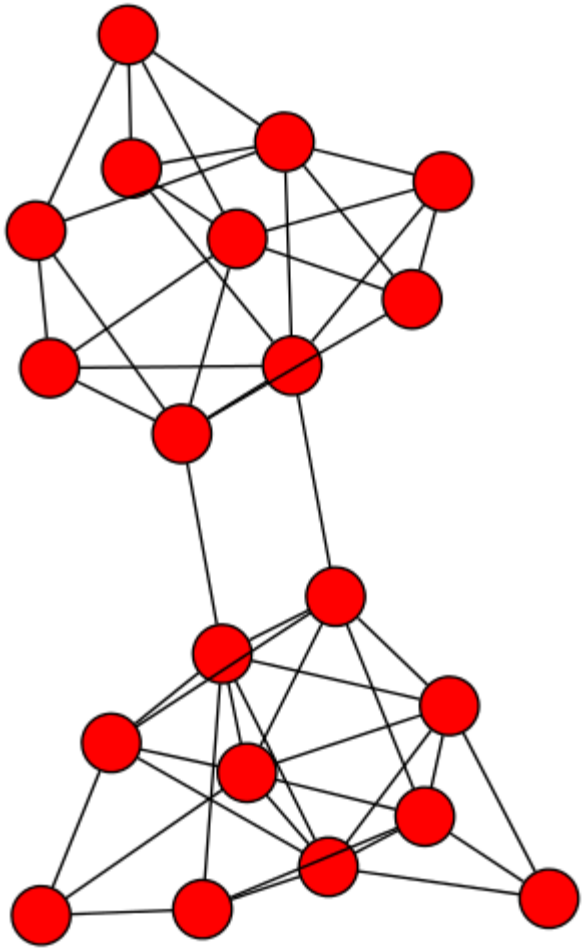
1	0.3
2	0.6
3	0.3

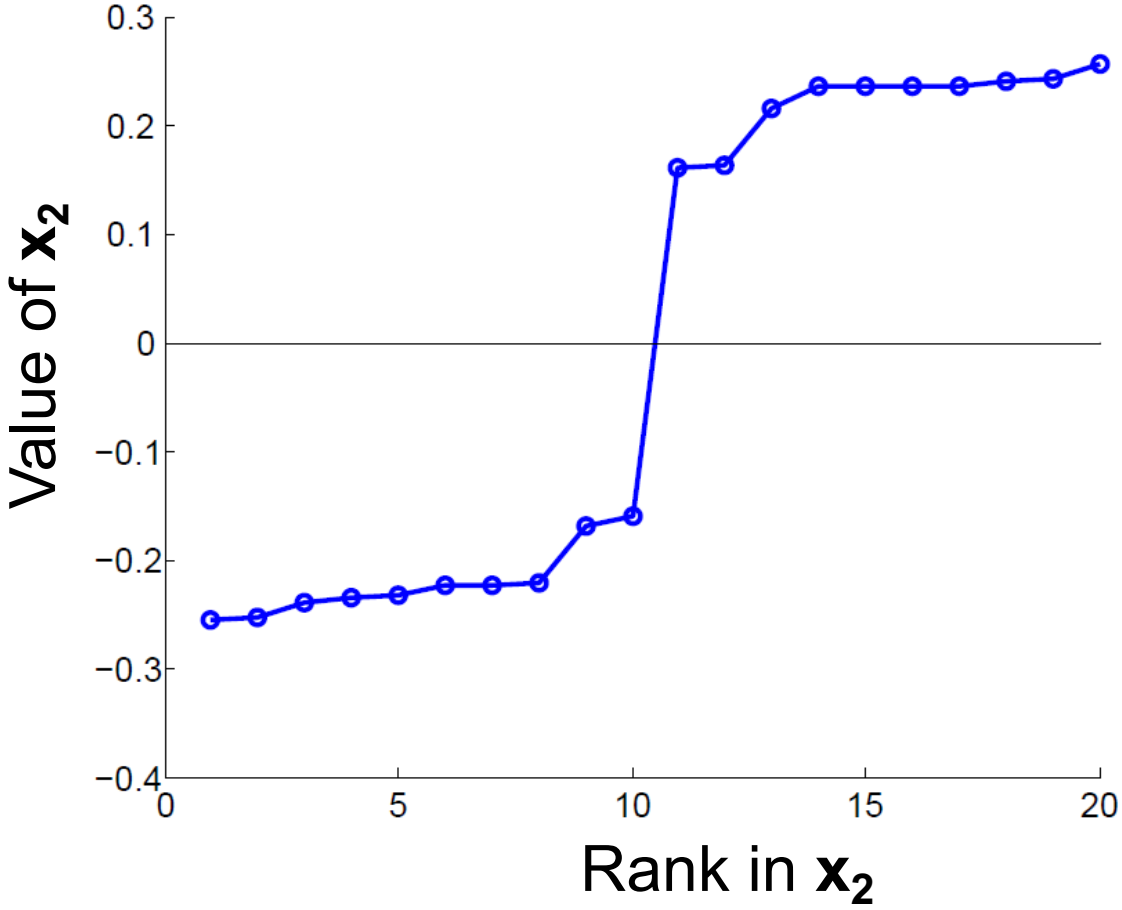
4	-0.3
5	-0.3
6	-0.6



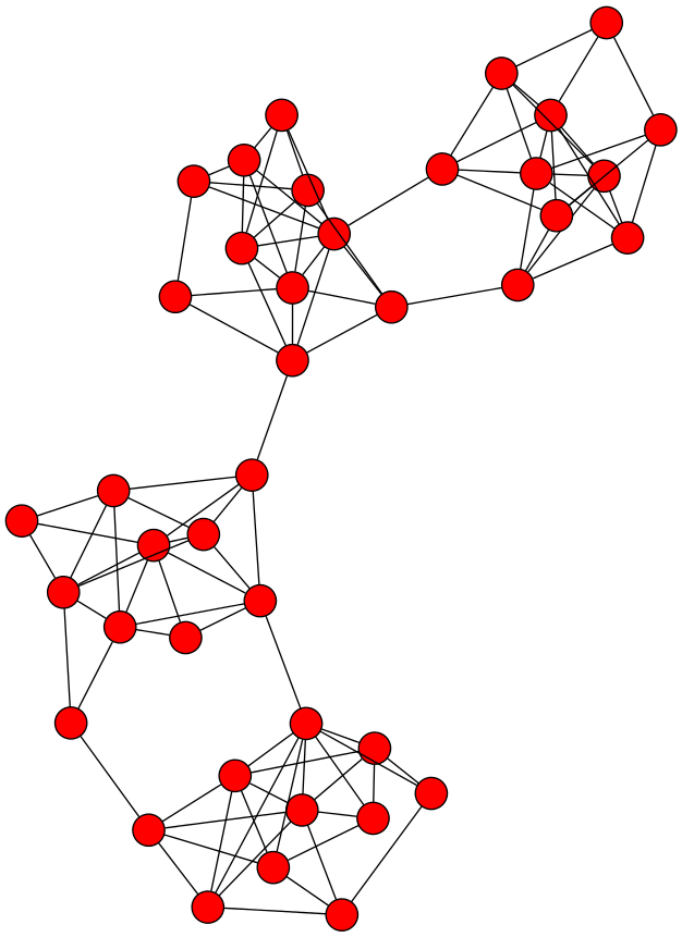


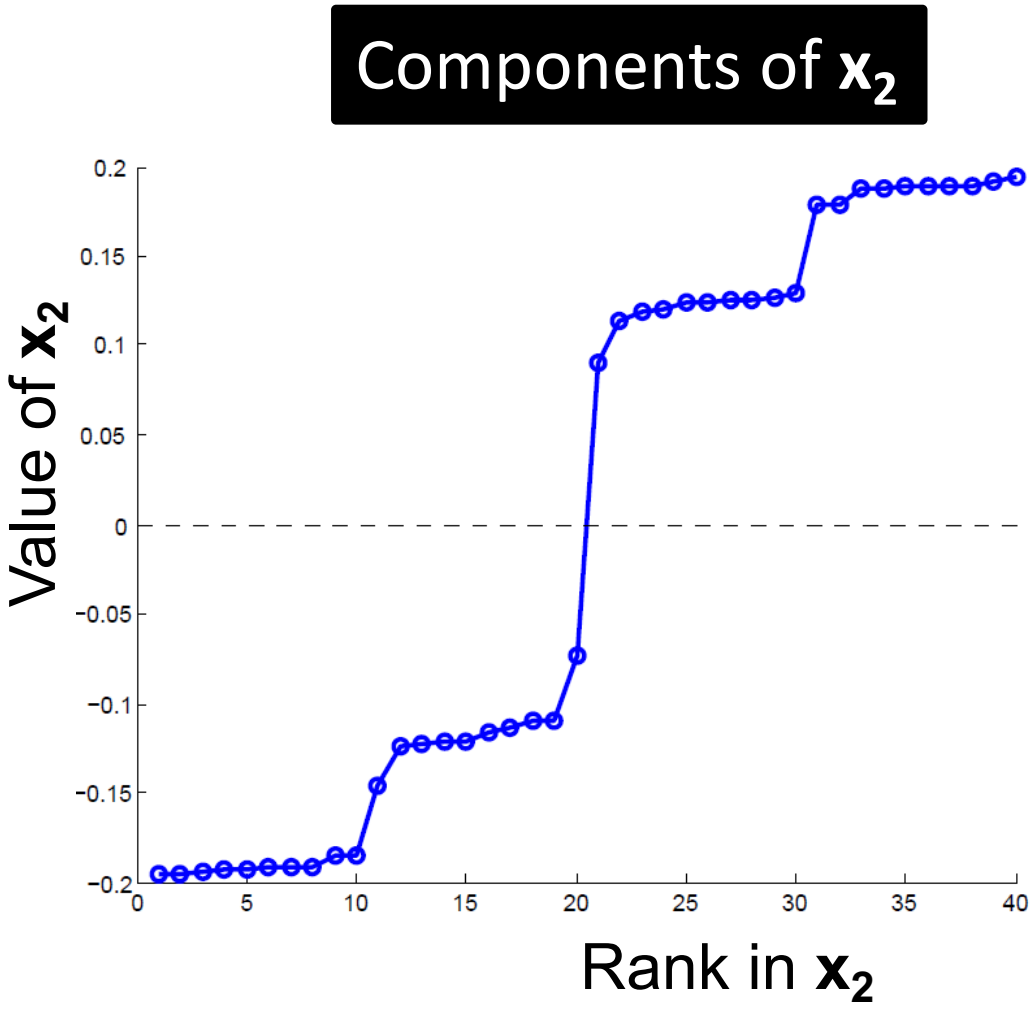
Example: Spectral Partitioning



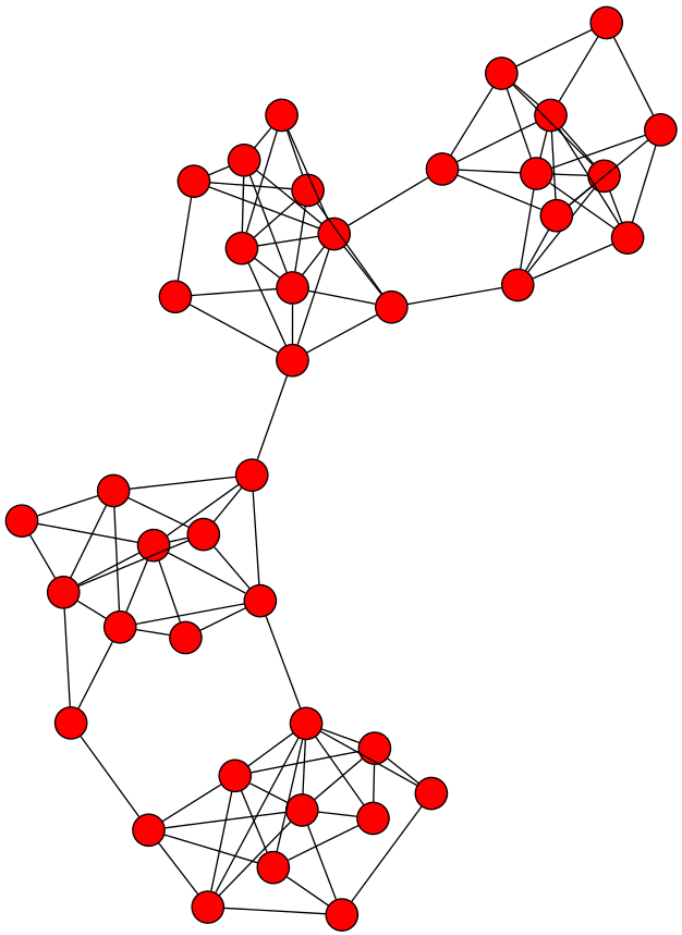


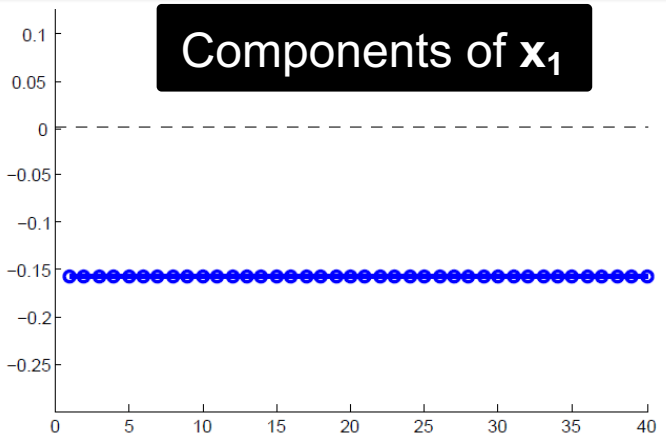
Example: Spectral Partitioning

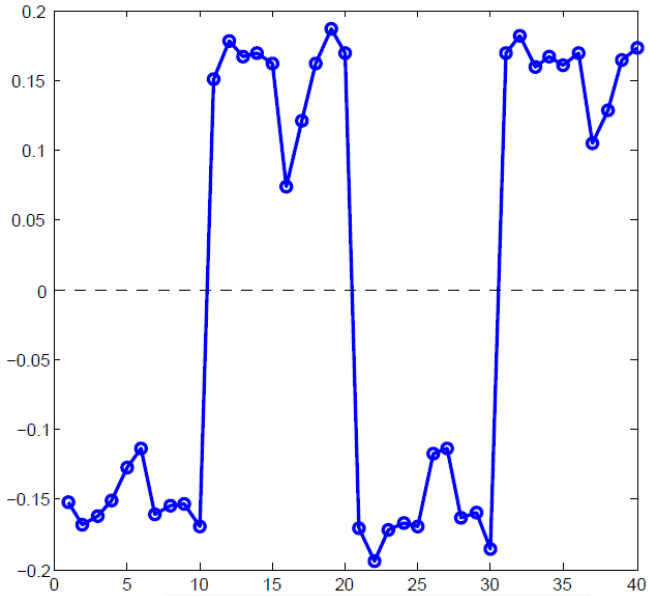




Example: Spectral partitioning







k-Way Spectral Clustering

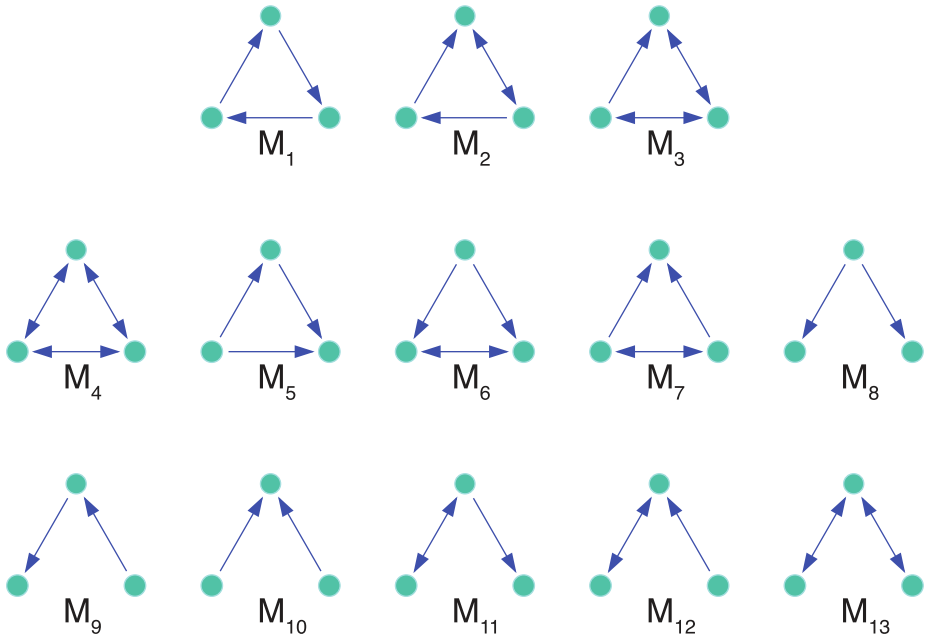
- **How do we partition a graph into k clusters?**
- **Two basic approaches:**
  - **Recursive bi-partitioning** [Hagen et al., '92]
    - Recursively apply bi-partitioning algorithm in a hierarchical divisive manner
    - Disadvantages: Inefficient, unstable
  - **Cluster multiple eigenvectors** [Shi-Malik, '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 $\boldsymbol{k}$ orthogonal basis vectors
- Multiple eigenvectors prevent instability due to information loss

Motif-based spectral clustering

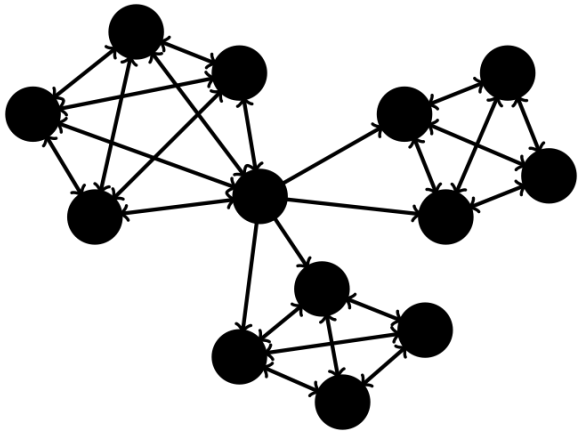
- **What if we want our clustering based on other patterns (not edges)?**



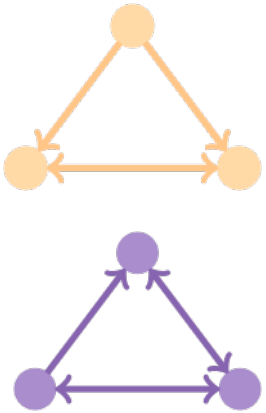
Small subgraphs (motifs, graphlets) are building blocks of networks [Milo et al., '02]

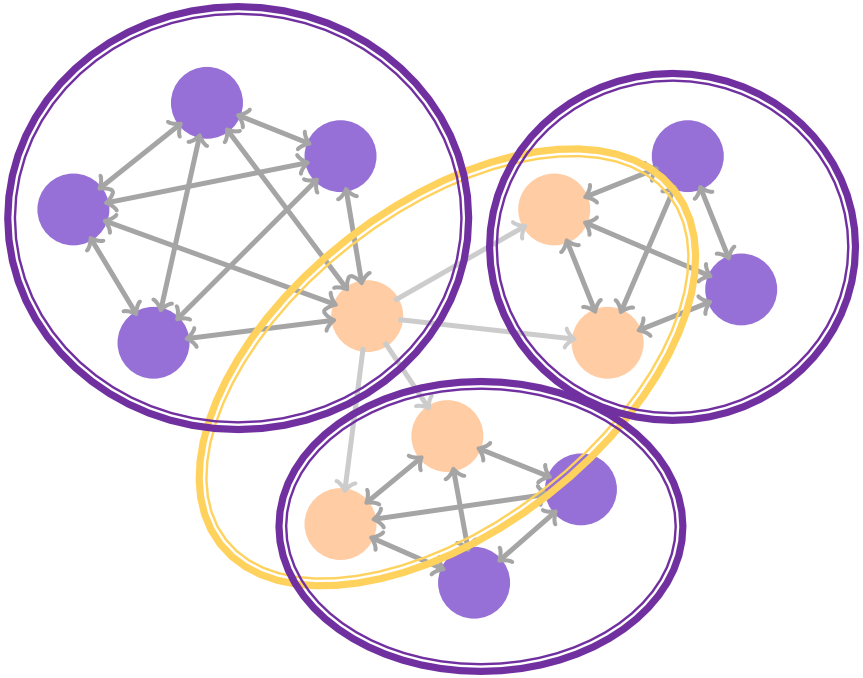
Motif-based spectral clustering

Network:



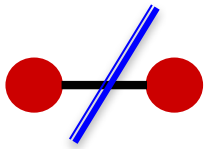
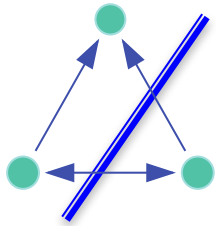
Motif:





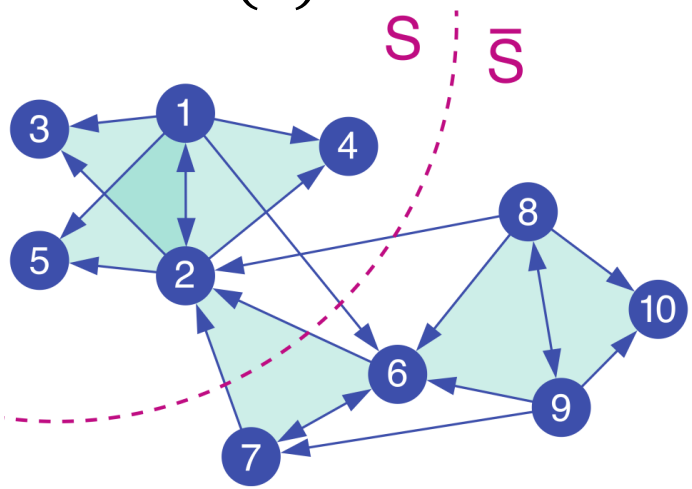
Re-define conductance for motifs

- **Generalize cuts and volumes to motifs**

$edges\ cut$  $motifs\ cut$ 

$vol(S)$ = #(edge end-points in S) → $vol_M(S)$ = #(motif end-points in S)

$$\phi(S) = \frac{\#(edges\ cut)}{vol(S)} \rightarrow \boxed{\phi_M(S) = \frac{\#(motifs\ cut)}{vol_M(S)}}$$

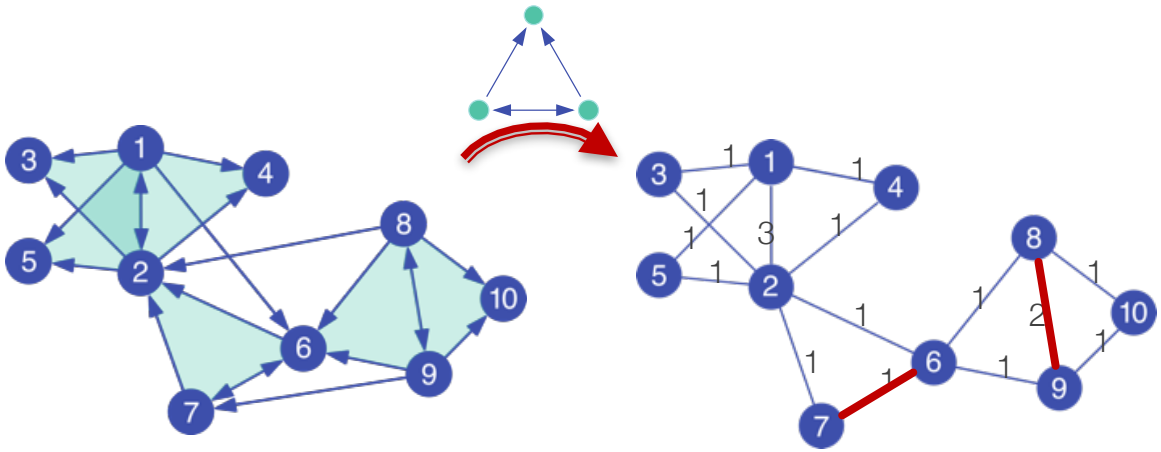


Optimize motif conductance

[Benson et al., '16]

Optimizing motif conductance

- **Three basic stages:**
  - **1) Pre-processing**
    - $W_{ij}^{(M)}$ = # times (i, j) participates in the motif



Graph G

Weighted graph $W^{(M)}$

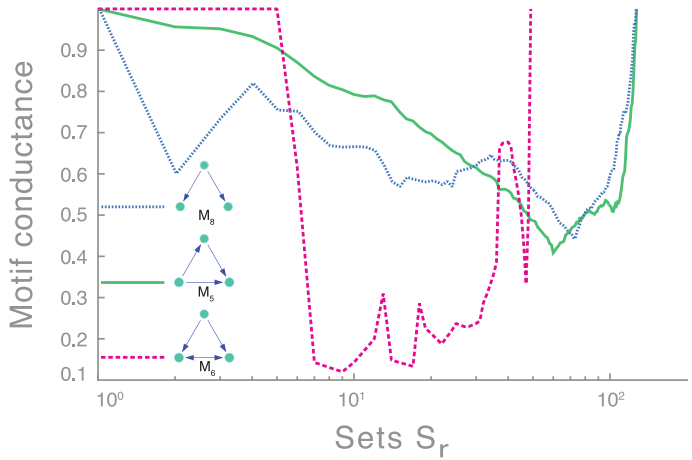
  - **2) Decomposition**
    - Same eigenvectors + embedding as standard spectral clustering (but on $W^{(M)}$)
  - **3) Grouping**
    - Same as standard spectral clustering

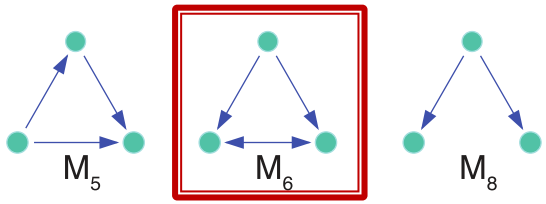
Why does this work?

- **Key ideas:**
  - $W_{ij}^{(M)}$ = # times (i, j) participates in the motif
    $D^{(M)}$ is weighted diagonal degree matrix
  - Theorem (3 node motifs):
    (1) any cut in weighted graph is proportional to motif cut
    (2) the volume of any set S in the weighted graph is proportional to the motif volume of S
  - Sketch of proof for (1): $x_i = +1$ if node i in S, -1 otherwise
  - $I(x_i = x_j = x_k) = x_i^2 + x_j^2 + x_k^2 - (x_i x_j - x_i x_k - x_j x_k)$
  - $\Sigma_{\text{motifs } i,j,k} I(x_i = x_j = x_k) = x^T(D^{(M)} - W^{(M)})x$
  - (2) is easy (prove on your own)

Motif-based clustering algorithm

- **1)** normalized motif Laplacian: $N^{(M)} = (D^{(M)})^{-1/2}[D^{(M)} - W_{ij}^{(M)}] (D^{(M)})^{-1/2}$
- **2)** (λ_2, y_2) = second eigenpair of $N^{(M)}$
- **3)** b_i = index of $(D^{(M)})^{-1/2}y_2$ with ith smallest value
- **4)** Compute motif cond. for $S_r = \{b_1, b_2, \ldots, b_r\}$, r = 1, …, n

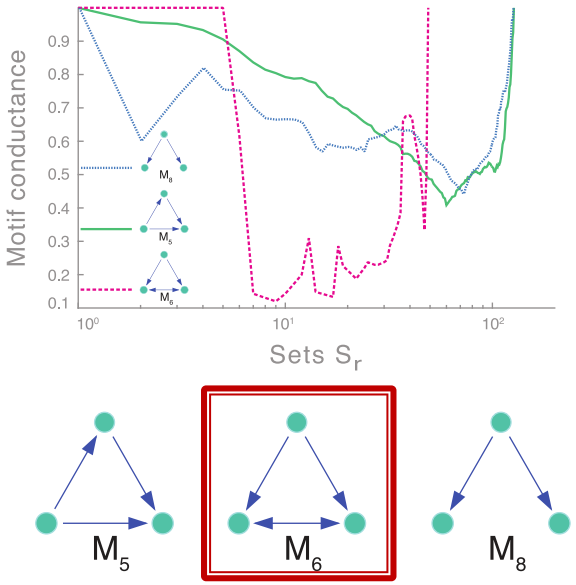




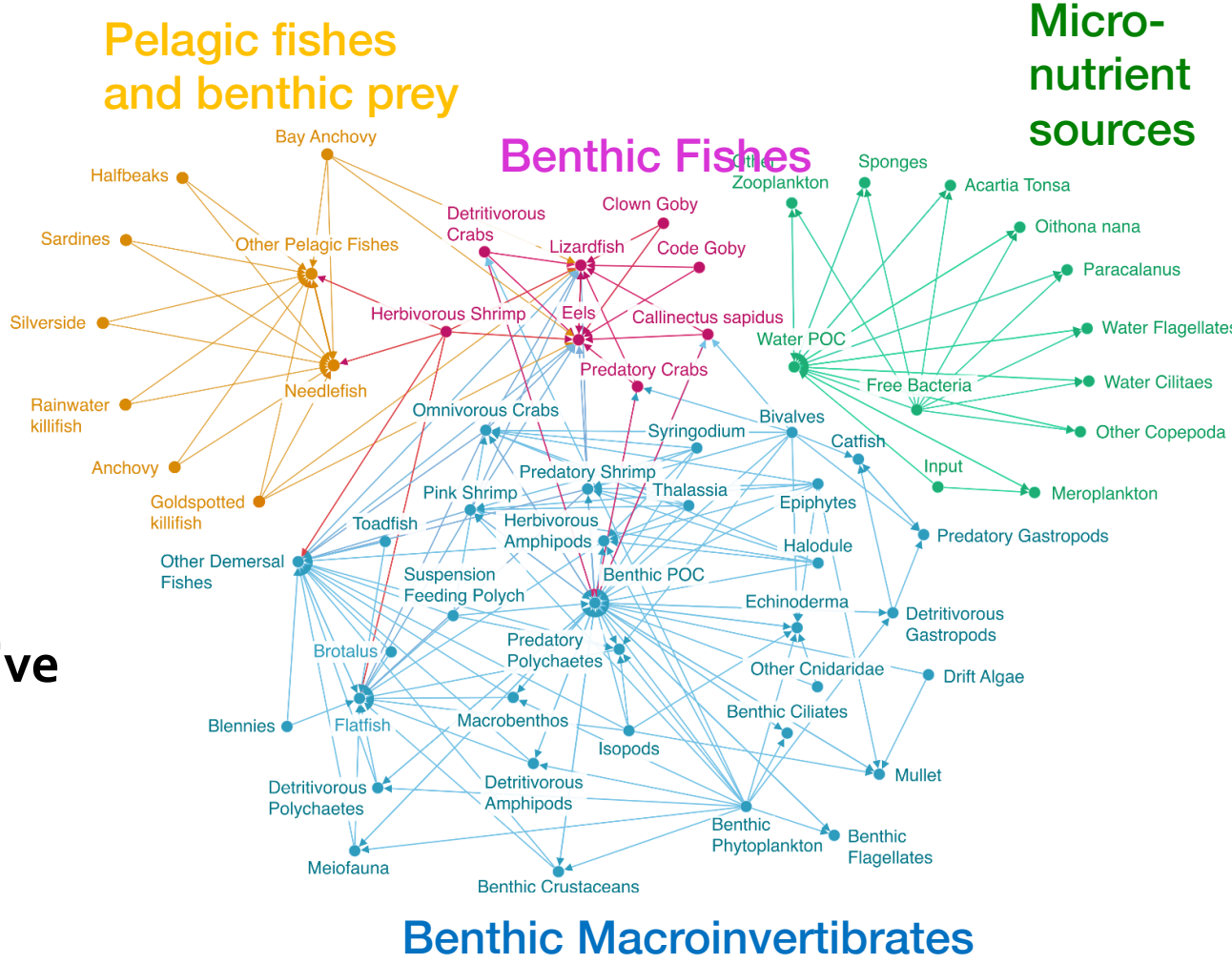
- **5)** Pick set S with smallest motif conductance (try several motifs)
- **Motif Cheeger inequality guarantee:**

$$\lambda_2/2 \leq \phi_M^{\text{opt}}, \ \phi_M(S) \leq 4\sqrt{\phi_M^{\text{opt}}}$$

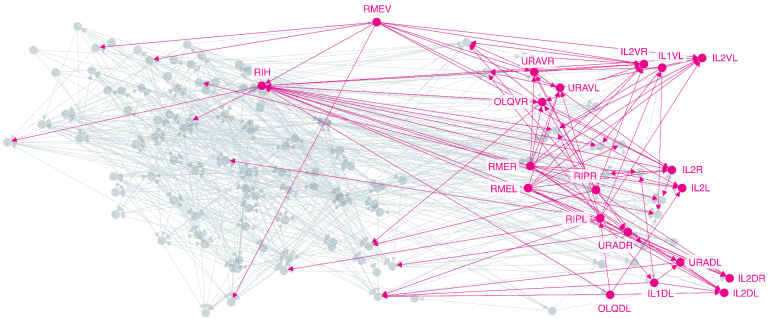
Motif-based clustering of a food web



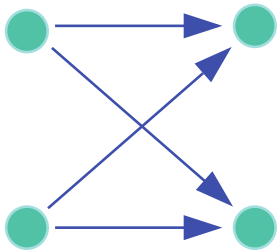
Use multiple eigenvectors or recursive bi-partitioning to get multiple clusters



Motif clustering of a neural network

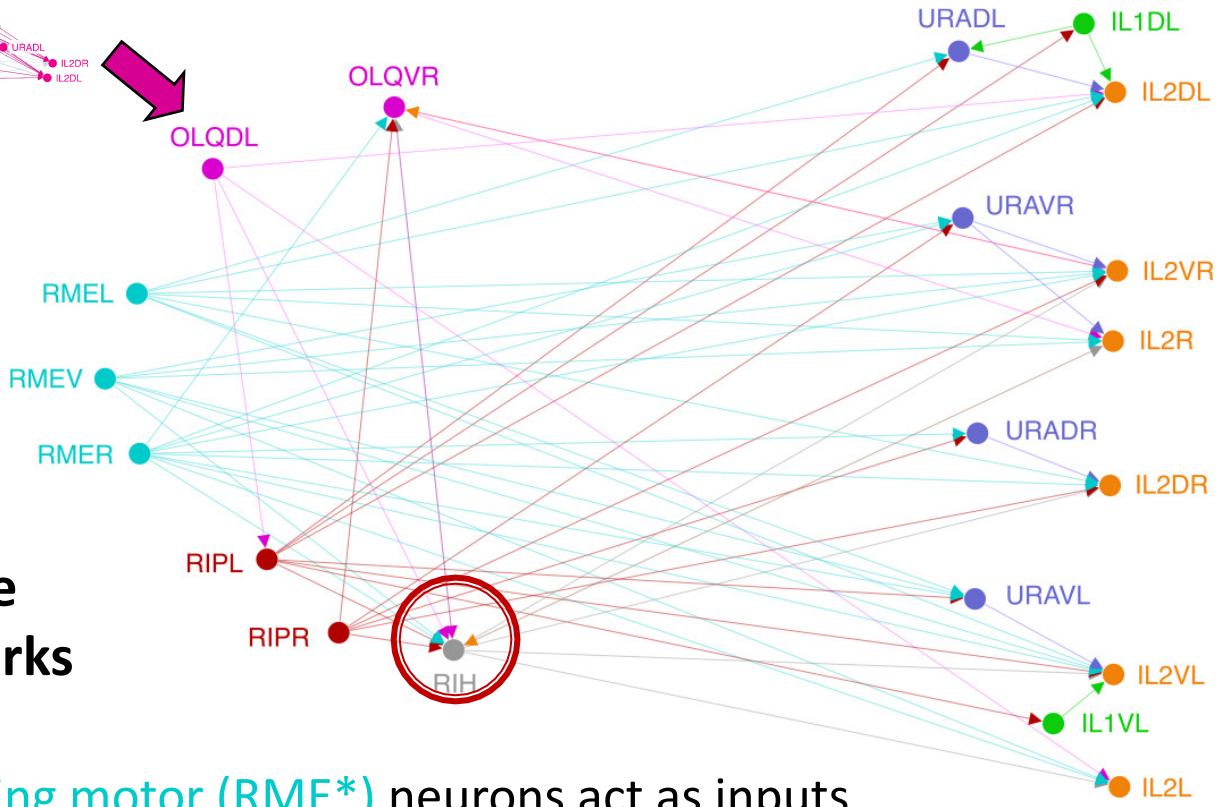


Neuron locations



"Bi-fan" motif known to be important in neural networks

[Milo et al., '02]



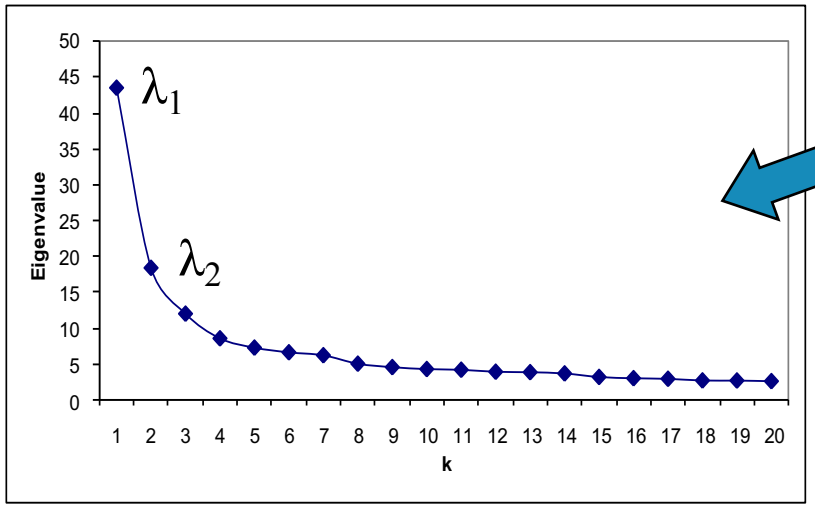
- Ring motor (RME*) neurons act as inputs
- Inner labial sensory (IL2*) neurons are the destinations
- URA neurons act as intermediaries

How to select k?

- **Eigengap:**
  - The difference between two consecutive eigenvalues
- **Most stable clustering is generally given by the value k that maximizes eigengap Δ_k:**

$$\Delta_k = |\lambda_k - \lambda_{k-1}|$$

- **Example:**



$$\max \Delta_k = |\lambda_2 - \lambda_1|$$

$\Rightarrow$ **Choose** $k = 2$

Many other partitioning methods

- **METIS:**
  - Heuristic but works really well in practice
  - http://glaros.dtc.umn.edu/gkhome/views/metis
- **Graclus:**
  - Based on kernel k-means
  - http://www.cs.utexas.edu/users/dml/Software/graclus.html
- **Louvain:**
  - Based on Modularity optimization
  - http://perso.uclouvain.be/vincent.blondel/research/louvain.html
- **Clique percorlation method:**
  - For finding overlapping clusters
  - http://angel.elte.hu/cfinder/