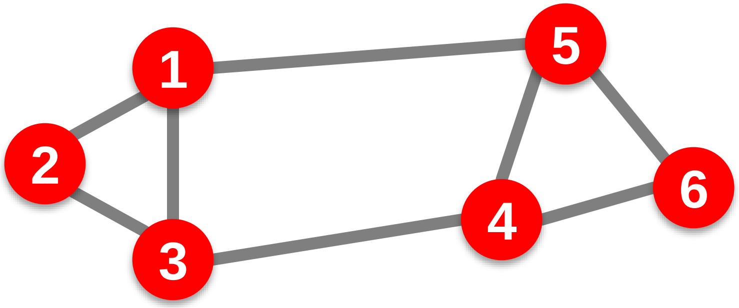
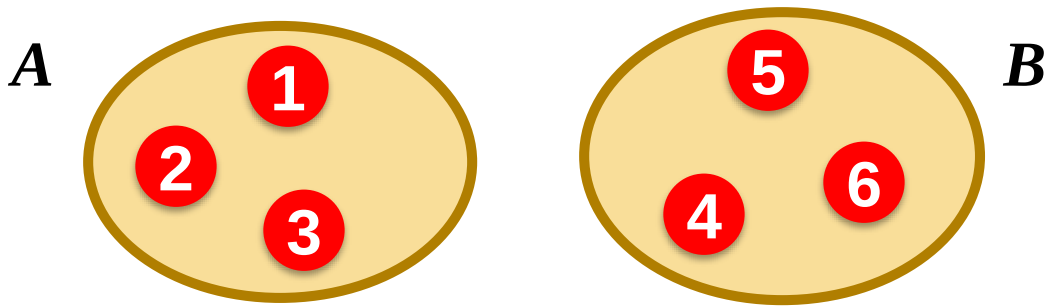
Spectral Clustering

Graph Partitioning

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



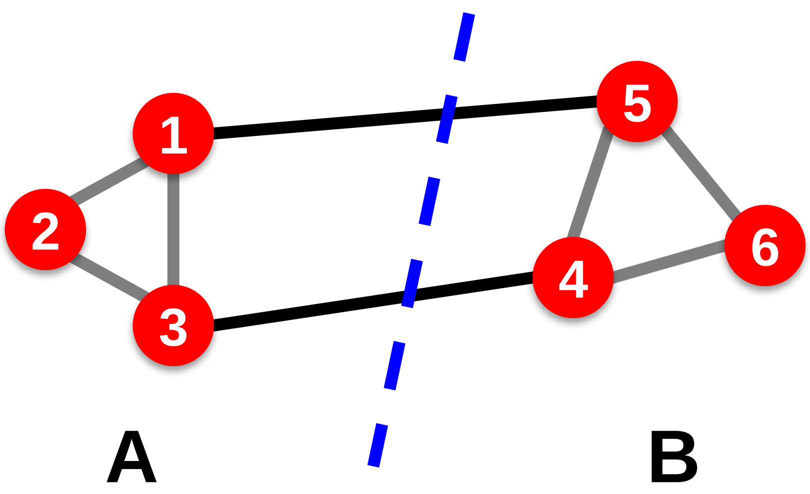
- **Bi-partitioning task:**
  - Divide vertices into two disjoint groups A, B



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

Graph Partitioning

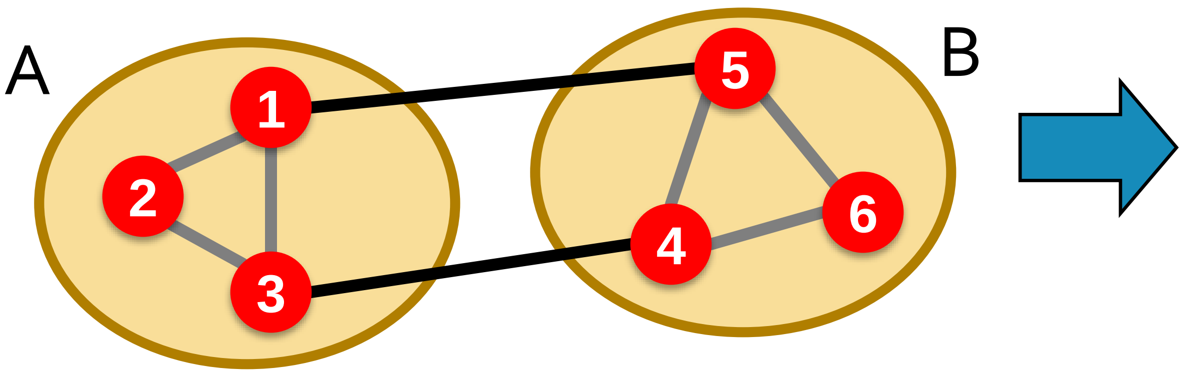
- **What makes a good partition?**
  - Maximize the number of within-group connections
  - Minimize the number of between-group connections



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:

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



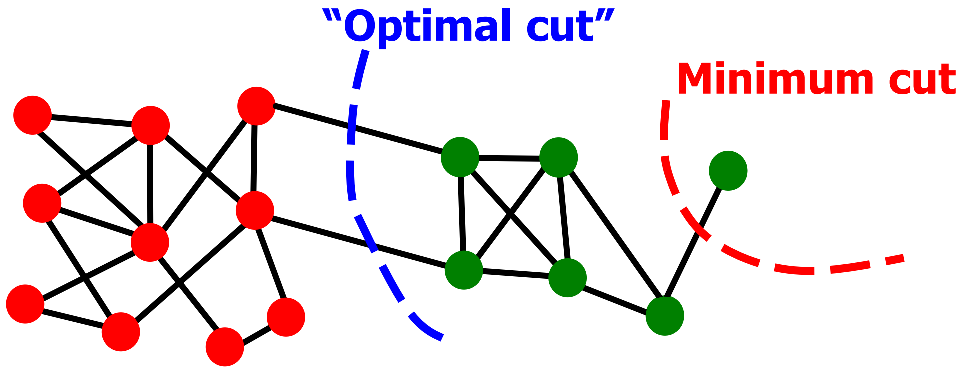
$cut(A,B) = 2$

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

Graph Cut Criteria

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

$$ncut(A,B) = \frac{cut(A,B)}{vol(A)} + \frac{cut(A,B)}{vol(B)}$$

$\boldsymbol{vol(A)}$: total weight of the edges with at least one endpoint in $\boldsymbol{A}$: $\boldsymbol{vol(A) = \sum_{i \in A} k_i}$

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

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 $(x_1, \dots, x_n)$
  - Think of it as a label/value of each node of G
- **What is the meaning of $A \cdot x$?**

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

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

- **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

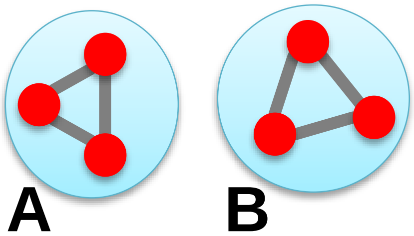
$$\begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{n1} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \lambda \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$$

$$\boldsymbol{A \cdot x = \lambda \cdot x}$$

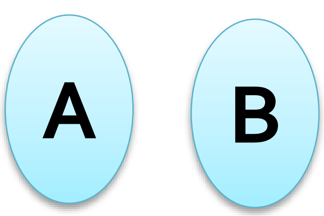
- **Spectral Graph Theory:**
  - Analyze the "spectrum" of matrix representing G
  - **Spectrum:** Eigenvectors x_i of a graph, ordered by the magnitude (strength) of their corresponding eigenvalues λ_i: $\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_n\}$

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

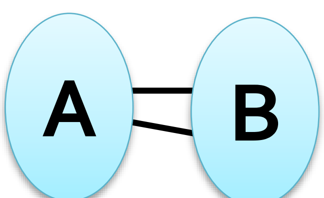
Example: Graph on 2 components



- **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
    - $x' = (\underbrace{1, \dots, 1}_{|A|}, \underbrace{0, \dots, 0}_{|B|})$ **then** $A \cdot x' = (d, \dots, d, 0, \dots, 0)$
    - $x'' = (0, \dots, 0, 1, \dots, 1)$ **then** $A \cdot x'' = (0, \dots, 0, d, \dots, d)$
    - And so in both cases the corresponding $\lambda = d$
- **A bit of intuition:**



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

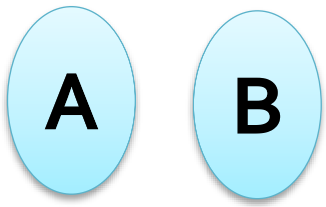


$\lambda_n - \lambda_{n-1} \approx 0$

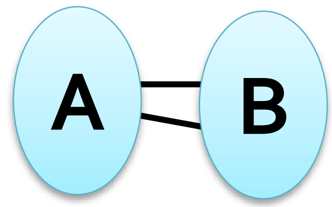
2[nd] largest eigval. λ_{n-1} now has value very close to λ_n

More Intuition

- **More intuition:**



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



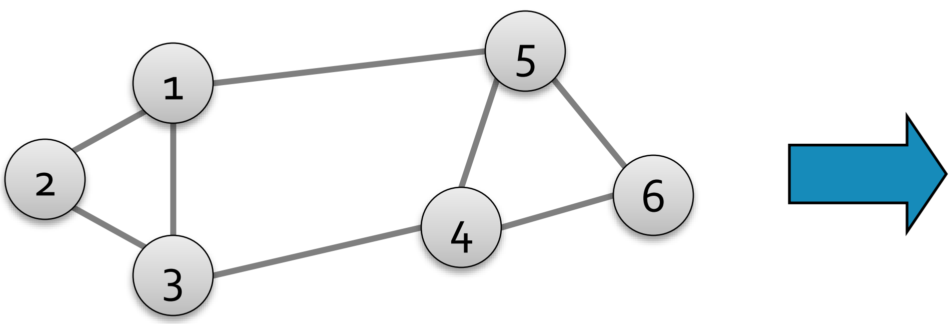
$\lambda_n - \lambda_{n-1} \approx 0$

2[nd] 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 $\boldsymbol{x_n} = (\mathbf{1}, \dots \mathbf{1})$ is an eigenvector
  - Since eigenvectors are orthogonal then the components of $\boldsymbol{x_{n-1}}$ sum to **0**.
    - Why? Because $\boldsymbol{x_n} \cdot \boldsymbol{x_{n-1}} = \sum_i \boldsymbol{x_n}[\boldsymbol{i}] \cdot \boldsymbol{x_{n-1}}[\boldsymbol{i}]$
  - So we can look at the eigenvector of the 2[nd] 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 $\boldsymbol{i}$ and $\boldsymbol{j}$

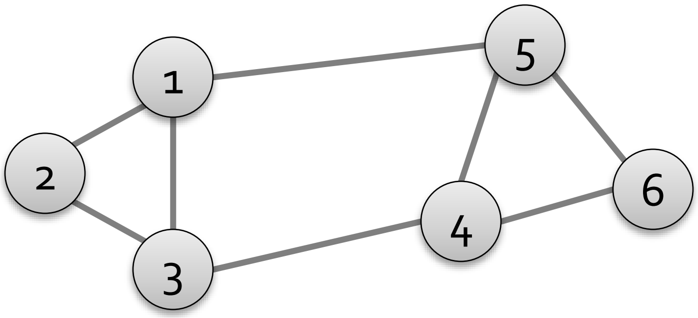


	1	2	3	4	5	6
1	0	1	1	0	1	0
2	1	0	1	0	0	0
3	1	1	0	1	0	0
4	0	0	1	0	1	1
5	1	0	0	1	0	1
6	0	0	0	1	1	0

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

Matrix Representations

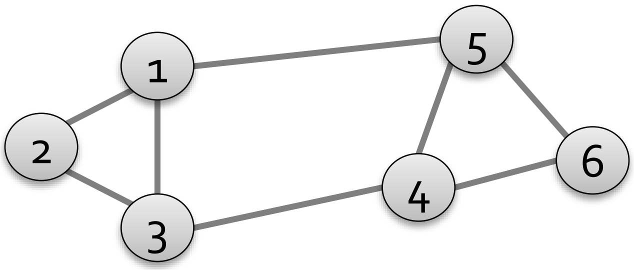
- **Degree matrix (D):**
  - $n \times n$ diagonal matrix
  - $D=[d_{ii}]$, d_{ii} = degree of node i



	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

$$\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}$$

- **What is trivial eigenpair?**
  - $\boldsymbol{x} = (1, \dots, 1)$ then $\boldsymbol{L} \cdot \boldsymbol{x} = \mathbf{0}$ and so $\boldsymbol{\lambda} = \boldsymbol{\lambda_1} = \mathbf{0}$
- **Important properties:**
  - **Eigenvalues** are non-negative real numbers
  - **Eigenvectors** are real and orthogonal

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:**
  - **(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:**

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

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

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

Details!

- Write x in axes of eigenvecotrs $w_1, w_2, \dots, w_n$ 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 $\boldsymbol{x^T M x}$?
  - $x^T M x = (\sum_i \alpha_i w_i)(\sum_i \alpha_i \lambda_i w_i) = \sum_{ij} \alpha_i \lambda_j \alpha_j \overbrace{w_i w_j}^{= 0 \text{ if } i \neq j \text{; } 1 \text{ otherwise}}$

    $= \sum_i \alpha_i \lambda_i w_i w_i = \sum_i \boldsymbol{\lambda_i \alpha_i^2}$
  - To minimize this over all unit vectors x orthogonal to:

    w = min over choices of $(\alpha_1, \dots \alpha_n)$ so that:

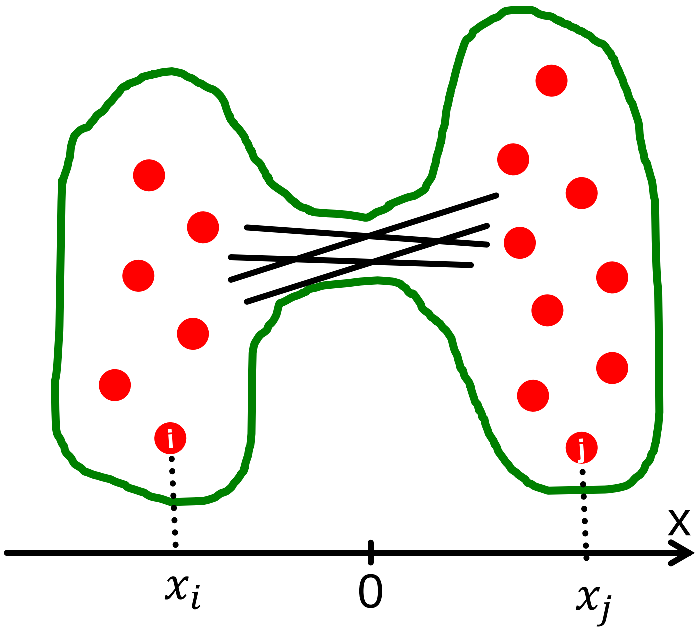
    $\sum \alpha_i^2 = 1$ (unit length) $\sum \alpha_i = 0$ (orthogonal to w_1)
  - To minimize this, set $\boldsymbol{\alpha_2 = 1}$ and so $\sum_i \boldsymbol{\lambda_i \alpha_i^2 = \lambda_2}$

λ_2 as optimization problem

- **What else do we know about x?**
  - $\boldsymbol{x}$ is unit vector: $\sum_i x_i^2 = 1$
  - $\boldsymbol{x}$ is orthogonal to **1**st eigenvector $(\mathbf{1}, \dots, \mathbf{1})$ 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]

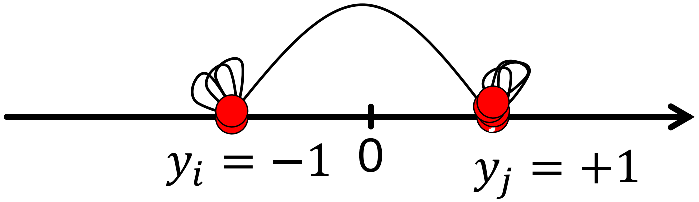
- **Back to finding the optimal cut**
- **Express partition (A,B) as a vector**

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

- We can minimize the cut of the partition by finding a non-trivial vector x that **minimizes**:

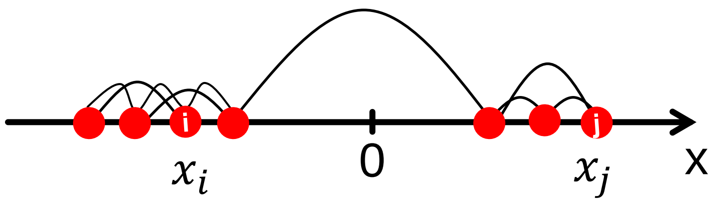
$$\arg\min_{y \in [-1,+1]^n} f(y) = \sum_{(i,j)\in E} (y_i - y_j)^2$$

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



Rayleigh Theorem

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



- $\lambda_2 = \min_{y} f(y)$: The minimum value of $f(y)$ is given by the 2nd smallest eigenvalue λ_2 of the Laplacian matrix L
- $\mathbf{x} = \arg\min_{y} f(y)$: The optimal solution for y is given by the corresponding eigenvector x, referred as the **Fiedler vector**

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 $2\boldsymbol{\alpha} \geq \boldsymbol{\lambda_2}$
  - This is the approximation guarantee of the spectral clustering. It says the cut spectral finds is at most **2** away from the optimal one of score $\boldsymbol{\alpha}$.
- **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: $\lambda_2 \leq \underbrace{\frac{\sum (x_i - x_j)^2}{\sum_i x_i^2}}_{\lambda_2 \text{ is only smaller}} \leq 2\alpha$ (while also $\sum_i x_i = 0$)

Approx. Guarantee of Spectral

Details!

- **Proof (continued):**
  - **1) Let's set:** $\boldsymbol{x_i} = \begin{cases} -\frac{\mathbf{1}}{\boldsymbol{a}} & \boldsymbol{if\ i \in A} \\ +\frac{\mathbf{1}}{\boldsymbol{b}} & \boldsymbol{if\ i \in B} \end{cases}$
    - Let's quickly verify that $\sum_i x_i = 0$: $a\left(-\frac{1}{a}\right) + b\left(\frac{1}{b}\right) = \mathbf{0}$
  - **2) Then:** $\frac{\sum(x_i - x_j)^2}{\sum_i x_i^2} = \frac{\sum_{i\in A, j\in B}\left(\frac{1}{b}+\frac{1}{a}\right)^2}{a\left(-\frac{1}{a}\right)^2 + b\left(\frac{1}{b}\right)^2} = \frac{e\cdot\left(\frac{1}{a}+\frac{1}{b}\right)^2}{\frac{1}{a}+\frac{1}{b}} =$

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

e … number of edges between A and B

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

Approx. Guarantee of Spectral

Details!

- **Putting it all together:**

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

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

So far…

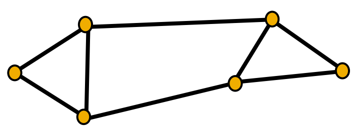
- **How to define a "good" partition of a graph?**
  - Minimize a given graph cut criterion
- **How to efficiently identify such a partition?**
  - Approximate using information provided by the eigenvalues and eigenvectors of a graph
- **Spectral Clustering**

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
  - Map vertices to corresponding components of λ_2

$\lambda =$

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

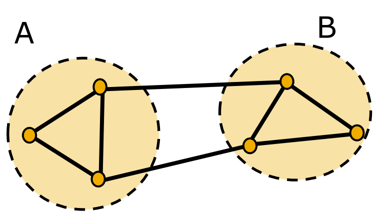
Split at 0:

Cluster A: Positive points

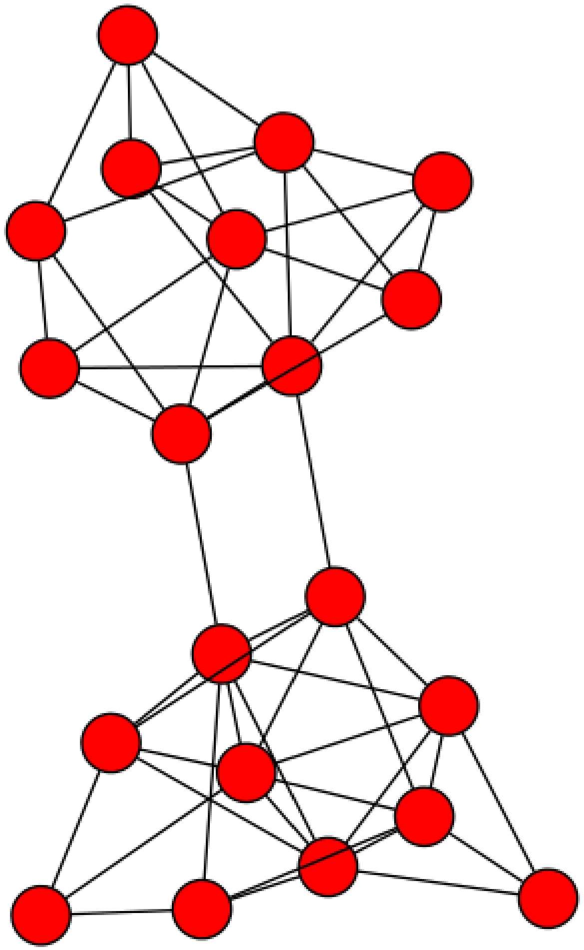
Cluster B: Negative points

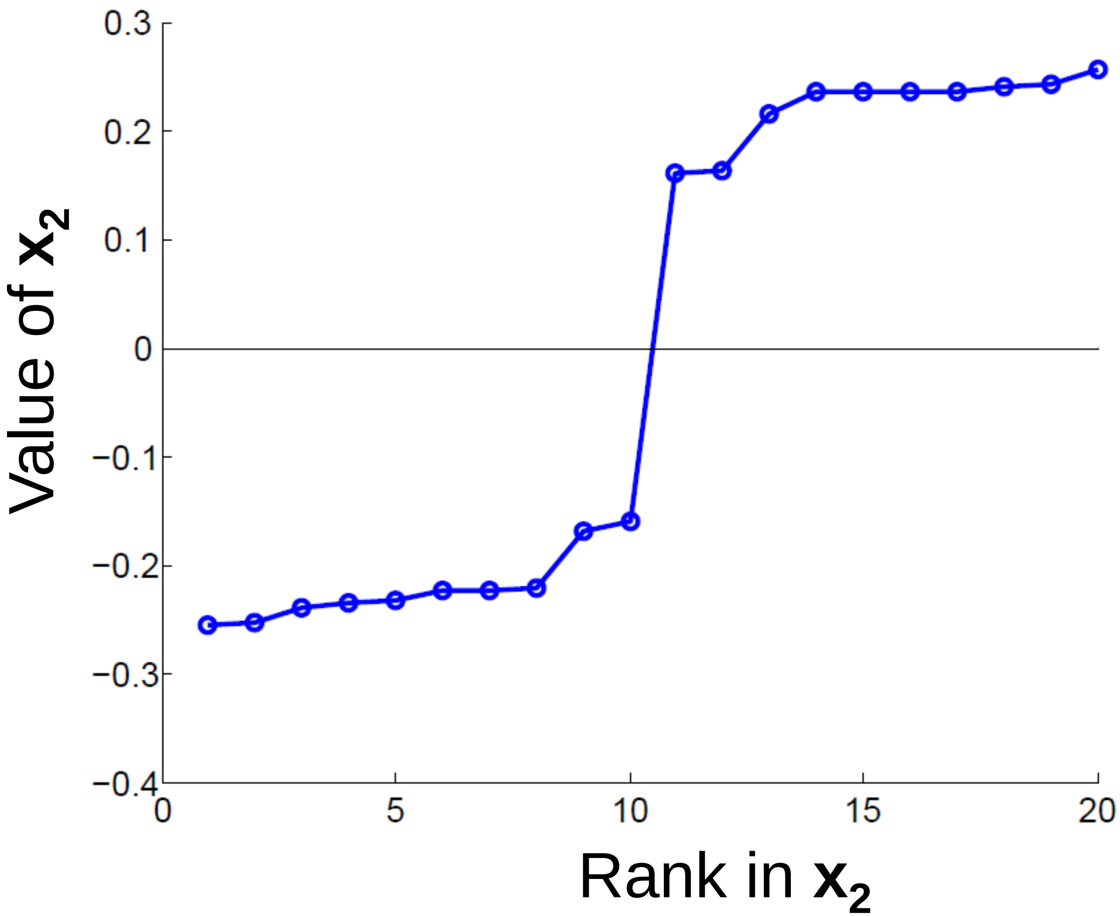
1	0.3
2	0.6
3	0.3

4	-0.3
5	-0.3
6	-0.6

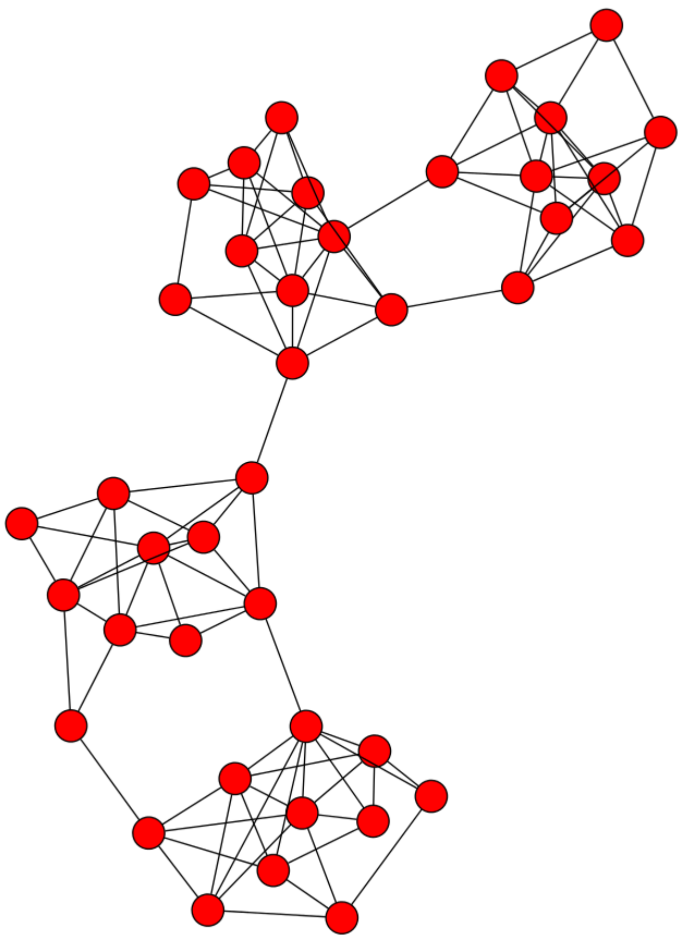


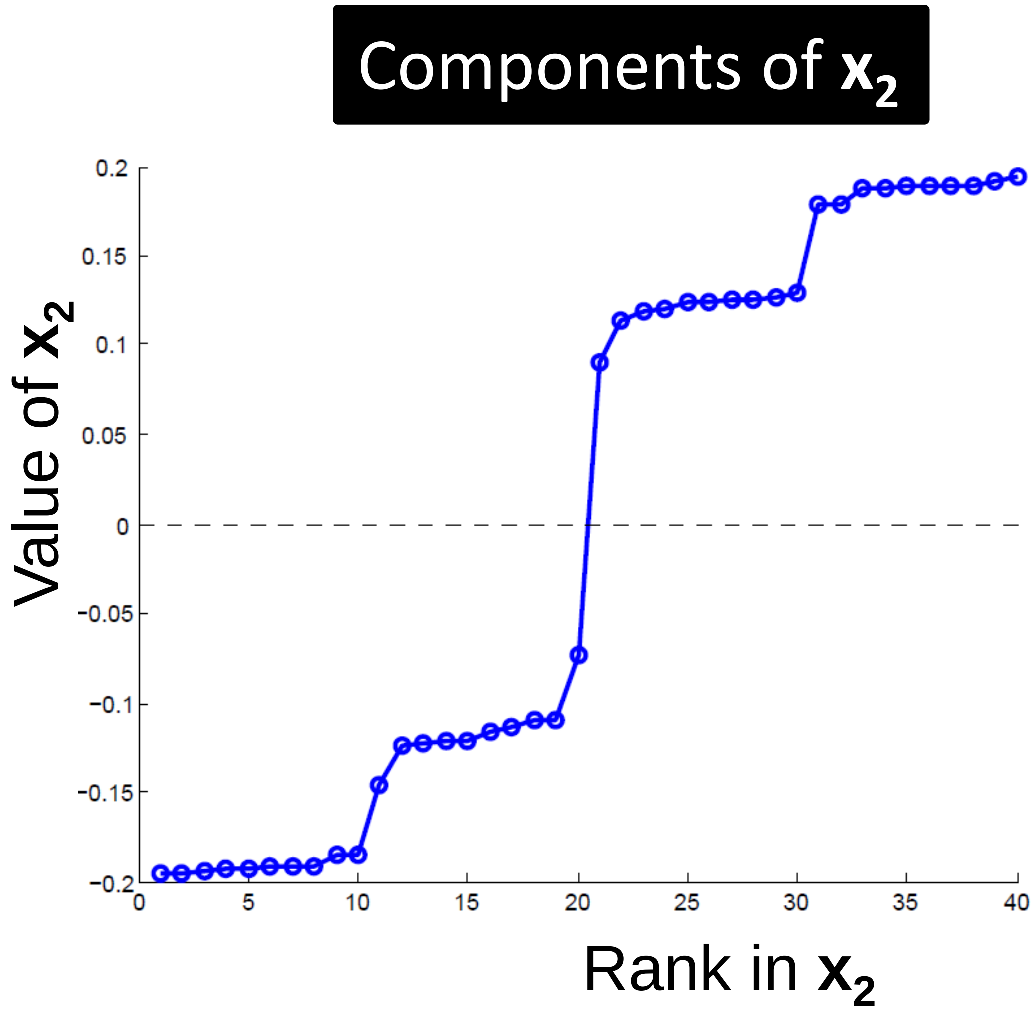
Example: Spectral Partitioning





Example: Spectral Partitioning





Example: Spectral partitioning