Network community detection:

-- Trawling
-- Spectral graph partitioning
-- Overlapping communities

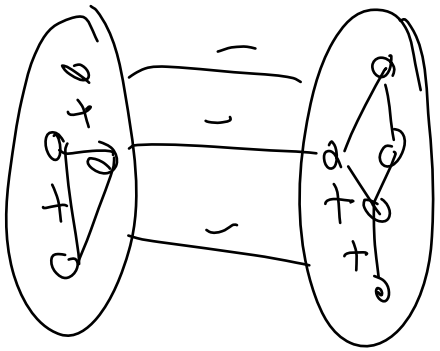
CS224W: Social and Information Network Analysis

Jure Leskovec, Stanford University

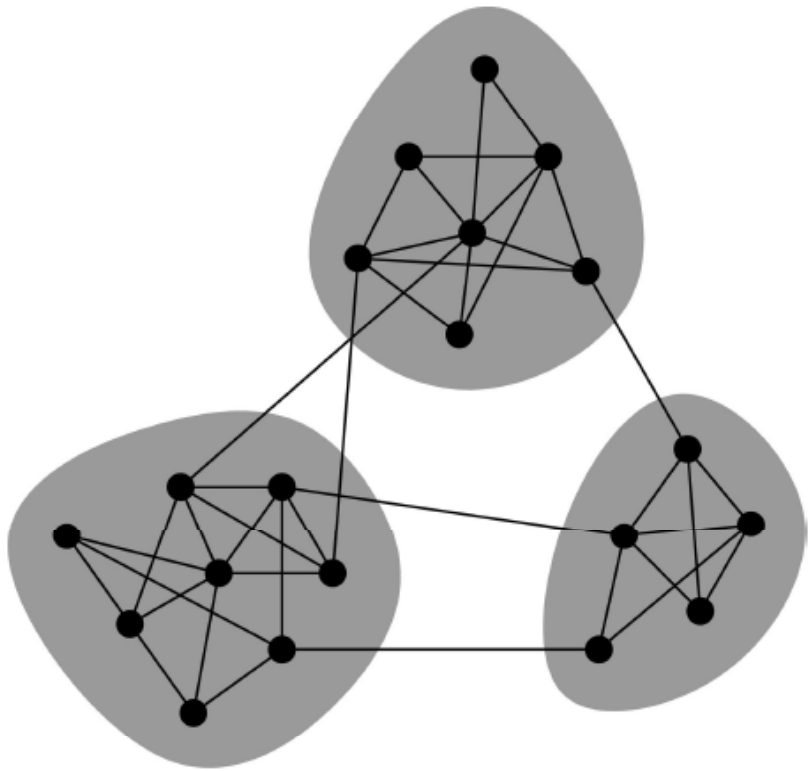
http://cs224w.stanford.edu

Announcement: HW3-Competition

- Task:
  - Find coalitions in signed networks
- Incentives:
  - **European chocolates!**
  - Fame
  - Up to 10% extra credit
- Due:
  - Friday midnight
  - No late days!



Community structure

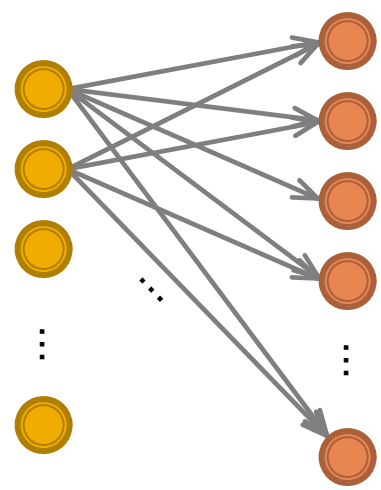


- Today: 3 methods
  (3) Trawling: Community signatures that can be efficiently extracted
  (4) Spectral graph partitioning: Laplacian matrix, 2nd eigenvector
  (5) Overlapping communities: Clique percolation method

Method3: Trawling

[Kumar et al. '99]

- Searching for small communities in Web graph
- (1) What is the signature of a community/discussion in a Web graph



A dense 2-layer graph

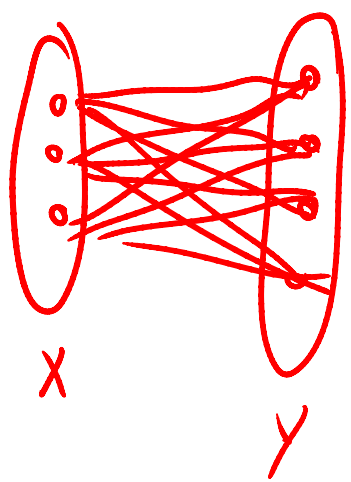
Use this to define topics: What the same people on the left talk about on the right

Intuition: many people all talking about the same things

[Kumar et al. '99]

Searching for small communities

- (2) A more well-defined problem:
  Enumerate complete bipartite subgraphs $K_{s,t}$
  - Where $K_{s,t}$ = s nodes where each links to the same t other nodes



$|X| = s$

$|Y| = t$

$K_{s,t}$

[Kumar et al. '99]

The Plan: (1), (2) and (3)

- Two points:
  - (1) The signature of a community/discussion
  - (2) Complete bipartite subgraph $K_{s,t}$
    - $K_{s,t}$ = graph on s nodes, each links to the same t other nodes
- Plan:
  - (A) From (2) get back to (1):
    - Via: Any dense enough graph contains a smaller $K_{s,t}$ as a subgraph
  - (B) How do we solve (2) in a giant graph?
    - What similar problems have been solved on big non-graph data?
    - (3) Frequent itemset enumeration [Agrawal-Srikant '99]

[Agrawa-Srikant '99]

Frequent itemset enumeration

- Marketbasket analysis:
  - What items are bought together in a store?
- Setting:
  - Universe U of n items items we sell
  - m subsets of U: $S_1, S_2, \ldots, S_m \subseteq U$ MARKET BASKETS
    (S_i is a set of items one person bought)
  - Frequency threshold f
- Goal:
  - Find all subsets T s.t. $T \subseteq S_i$ of $\geq f$ sets S_i
    (items in T were bought together $\geq f$ times)

Frequent itemsets: Example

- Example:
  - Universe of items:
    - $U=\{1,2,3,4,5\}$
  - Itemsets:
    - $S_1=\{1,3,5\}$, $S_2=\{2,3,4\}$, $S_3=\{2,4,5\}$, $S_4=\{3,4,5\}$, $S_5=\{1,3,4,5\}$, $S_6=\{2,3,4,5\}$
  - Minimum support:
    - $f=3$
- Algorithm: Build up the lists
  - Insight: for a frequent set of size k, all its subsets are also frequent

[Agrawa-Srikant '99]

Example: Apriori algorithm

- $U=\{1,2,3,4,5\}$, $f=3$
- $S_1=\{1,3,5\}$, $S_2=\{2,3,4\}$, $S_3=\{2,4,5\}$, $S_4=\{3,4,5\}$, $S_5=\{1,3,4,5\}$, $S_6=\{2,3,4,5\}$

set size							
1	~~1~~	2	3		4		5
2		~~2,3~~	2,4	~~2,5~~	3,4	3,5	4,5
3	~~2,3,5~~	~~2,3,4~~			3,4,5		

[Agrawa-Srikant '99]

Frequent itemset: Apriori algorithm

- For $i = 1,\ldots,k$
  - Find all frequent sets of size i by composing sets of size i-1 that differ in 1 element

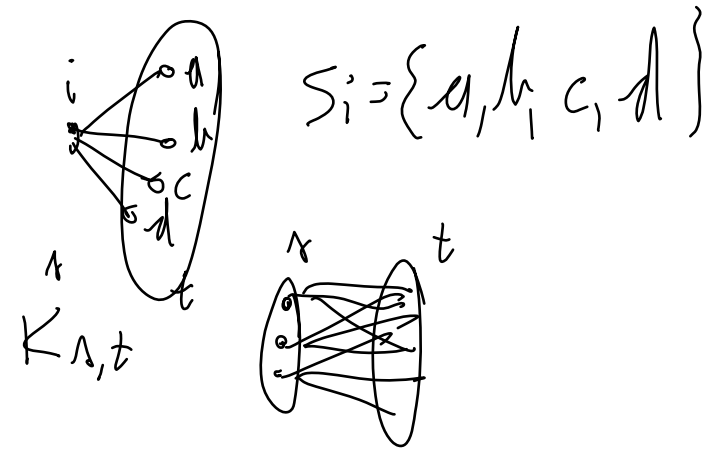
- Open question:
  - Efficiently find only maximal frequent sets

[Kumar et al. '99]

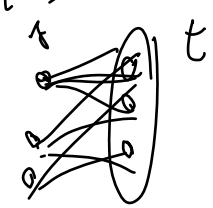
From Itemsets to bipart graphs $K_{s,t}$

- <u>Claim:</u> (3) (itemsets) solves (2) (bipartite graphs)

- How?
  - View each node i as a set S_i of nodes i points to
  - $K_{s,t}$ = a set y of size t that occurs in s sets S_i
  - Looking for $K_{s,t}$ → set of frequency threshold to s and look at layer t – all frequent sets of size t



$K_{s,t}$:
- set s
- itemsets of t elements



[Kumar et al. '99]

From $K_{s,t}$ to Communities

- (2) $\Rightarrow$ (1): Informally, every dense enough graph G contains a bipartite $K_{s,t}$ subgraph where s and t depend on size (# of nodes) and density (avg. degree) of G [Kovan-Sos-Turan '53]

- Theorem: Let $G=(X,Y,E)$, $|X|=|Y|=n$ with avg. degree: $\overline{d} = s^{1/t} n^{1-1/t} + t$

  then G contains $K_{s,t}$ as a subgraph

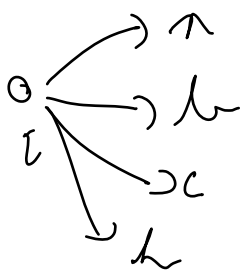
  [Will not prove it here. See online slides]

$K_{s,t}$ and Communities

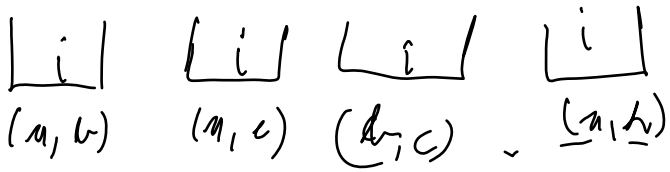
- For the proof we will need the following fact
  - Recall: $\binom{a}{b} = \frac{a(a-1)\ldots(a-b+1)}{b!}$
    - Let $f(x) = x(x-1)(x-2)\ldots(x-k)$
      Once $x \geq k$, $f(x)$ curves upward (convex)
  - Suppose a setting:
    - $g(y)$ is convex
    - Want to minimize $\sum_{i=1}^{n} g(x_i)$
    - where $\sum_{i=1}^{n} x_i = x$
  - To minimize $\sum_{i=1}^{n} g(x_i)$ make each $x_i = x/n$

Nodes and buckets

- Node i, degree d_i and neighbor set S_i



- Put node i in buckets for all size t subsets of its neighbors



Potential right-hand sides of $K_{s,t}$ (*i.e.*, all size t subsets of S_i)

Nodes and buckets

- Note: As soon as s nodes appear in a bucket we have a $K_{s,t}$
- How many buckets node i contributes?
  - d_i … degree of node i

$$\binom{d_i}{t}$$

- What is the total size of all buckets?

$$\sum_{i=1}^{n} \binom{d_i}{t} \geq \sum_{i=1}^{n} \binom{\bar{d}}{t} = \underline{n \binom{\bar{d}}{t}}$$

by convexity

Nodes and buckets

$$\binom{a}{b} = \frac{a(a-1)\ldots(a-b+1)}{b!}$$

- So, the total height of all buckets is…

$$n\binom{\bar{d}}{t} \geq n\frac{(\bar{d}-t)^t}{t!} = n\frac{s\,n^{t-1}}{t!} = \frac{n^t s}{t!}$$

Plug in:

$$\bar{d} = s^{1/t} n^{1-1/t} + t$$

And we are done!

- We have: Total height of all buckets $\geq \frac{n^t s}{t!}$
- How many buckets are there? $\binom{n}{t} \leq \frac{n^t}{t!}$
- What is the average height of buckets?

$$\geq \frac{n^t s}{t!} \frac{t!}{n^t} = s$$

So, avg. bucket height $\geq s$

- So by pigeonhole principle, there must be at least one bucket with more than s nodes in it.

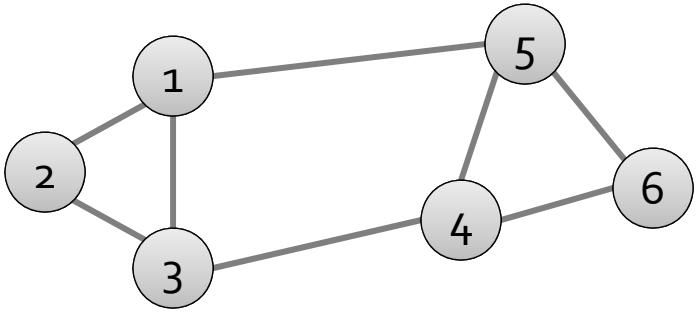
[Kumar et al. '99]

Method3: Trawling — Summary

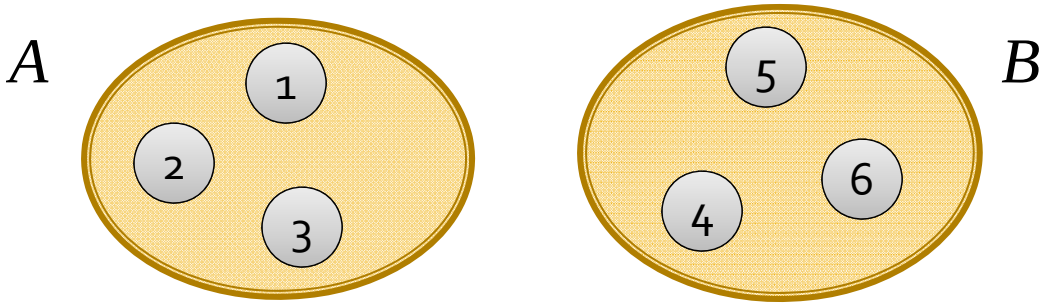
- Theoretical result:
  - Complete bipartite subgraphs $K_{s,t}$ are embedded in larger dense enough graphs (i.e., the communities)
    - i.e., biparite subgraphs as signatures of communities
- Algorithmic result:
  - Frequent itemset extraction and dynamic programming
  - **SCALABLE!!!**

Method4: Graph partitioning

- Undirected graph G(V,E):



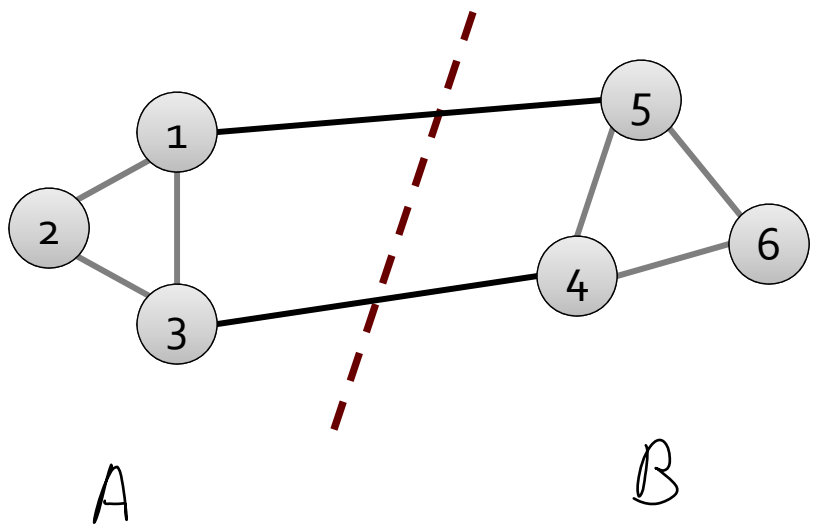
- Bi-partitioning task:
  - Divide vertices into two disjoint groups (A,B)



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

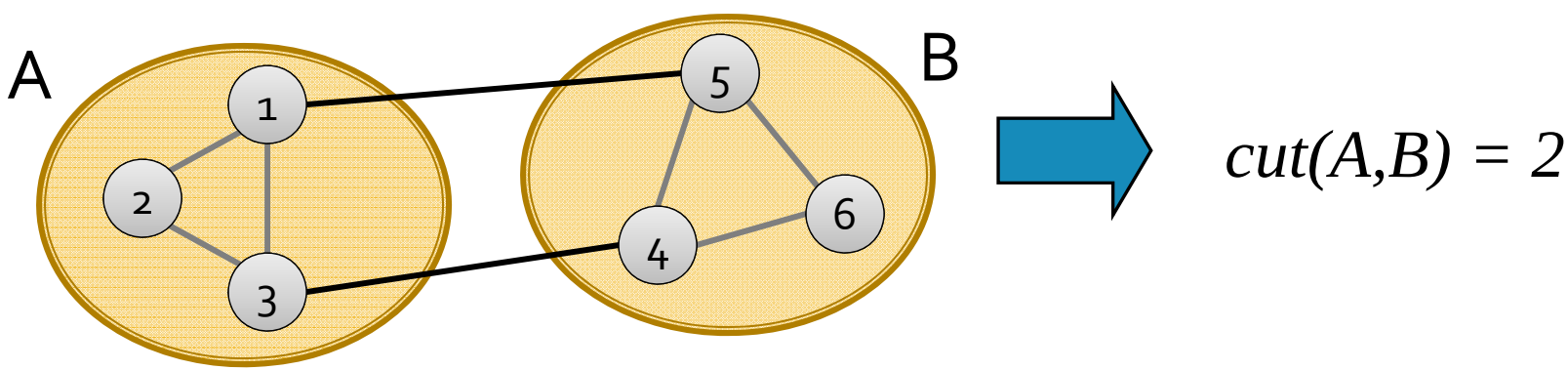
Graph partitioning

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

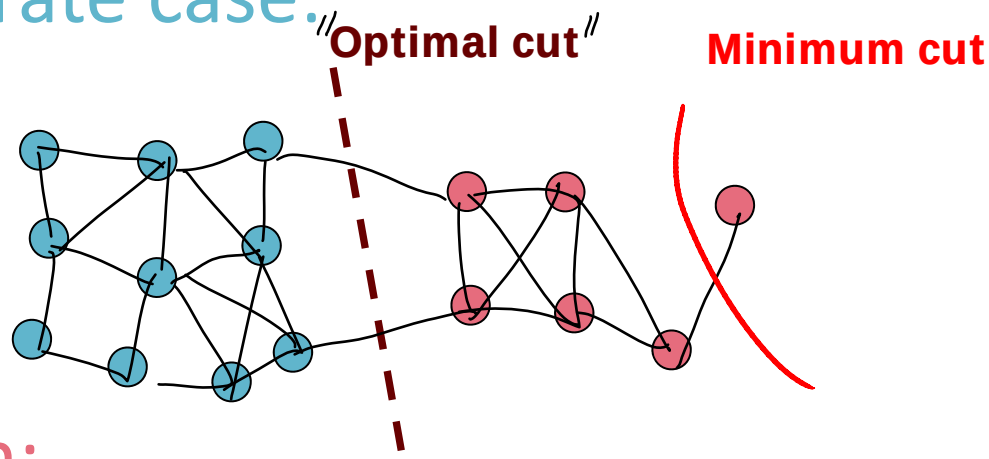


Graph Cut Criterion

- Criterion: Minimum-cut
  - Minimise weight of connections between groups

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

- Degenerate case:



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

[Shi-Malik]

Graph Cut Criteria

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

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

Vol(A): The total weight of the edges originating from group 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 $(x_1,\ldots, x_n)$
  - just a label/value of each node of G
- What is the meaning of $A \cdot x$?

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

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

- Entry y_j is a sum of labels x_i of neighbors of j

What is the meaning of *Ax*?

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

- j^{th} coordinate of Ax:
  - 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 of a graph, ordered by the magnitude (strength) of their corresponding eigenvalues: $\Lambda = \{\lambda_1, \lambda_2, \dots, \lambda_n\}$

$$\lambda_1 \le \lambda_2 \le \dots \le \lambda_n$$

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 λ? What x?

$x = (1, 1, \ldots, 1)$

$A \cdot x = (d, d, d, \ldots, d)$

$\Rightarrow \lambda = d$

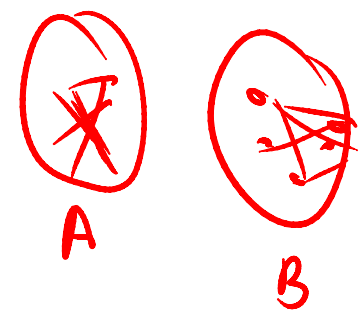
$Ax = \lambda x$

d

$(1, \ldots, 1)$ $(1, 1, \ldots, 1)$

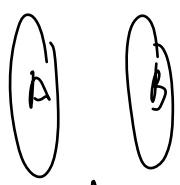
Example: Graph on 2 components

- What if G is not connected?
  - Say G has 2 components, each d-regular
- What are some eigenvectors?
  - x= Put all 1s on A and 0s on B or vice versa

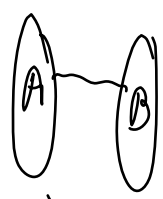


$x' = (\underbrace{0, 0, 0 \ldots 0}_{A}, \underbrace{1, 1, 1, 1}_{B})$ $\quad Ax = (0, 0, 0, d, d \ldots)$

$x'' = (1, \ldots 1, 0 \ldots 0)$ $\quad Ax = (d \ldots d, 0 \ldots 0)$



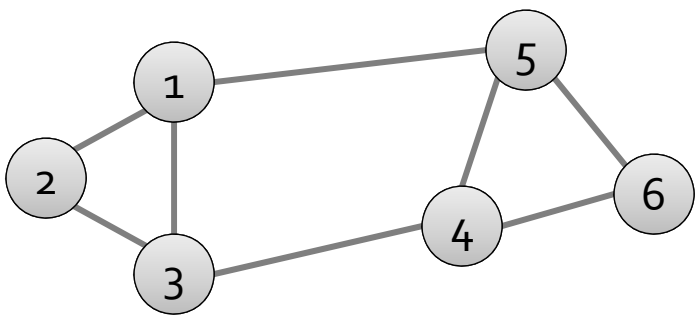
2 dim



$\lambda_1 - \lambda_2 \approx 0$

Matrix Representations

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

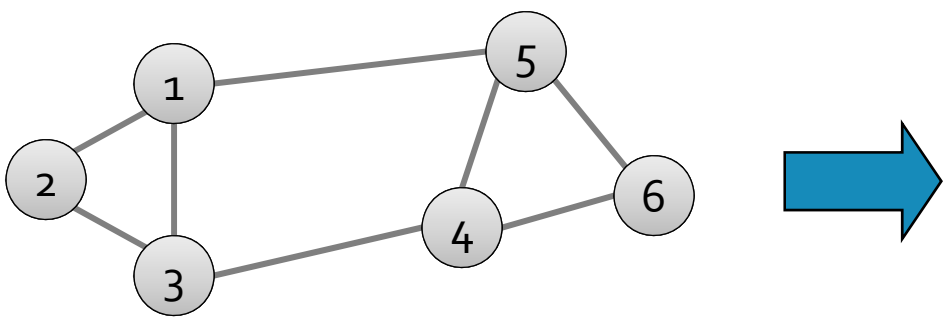


	1	2	3	4	5	6
1	0	1	1	0	1	0
2	1	0	1	0	0	0
3	1	1	0	1	0	0
4	0	0	1	0	1	1
5	1	0	0	1	0	1
6	0	0	0	1	1	0

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

Matrix Representations (continued)

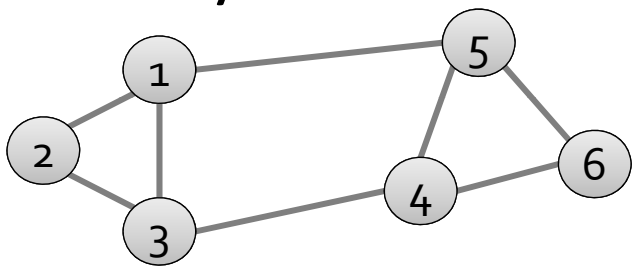
- Degree matrix (D):
  - $n \times n$ diagonal matrix
  - $D=[d_{ii}]$, d_{ii} = degree of node i



	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 eigenvector/ eigenvalue?

$x = (1, 1, 1, 1 \ldots 1)$

$Lx = 0 \cdot x$ $\quad \lambda = 0$

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

λ_2 as optimization problem

- For symmetric matrix M:

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

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

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

$$x^T L x = \sum_{i,j=1}^{m} L_{ij} x_i x_j = \sum_{i,j=1}^{m} (D_{ij} - A_{ij}) x_i x_j$$

λ_2 as optimization problem

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

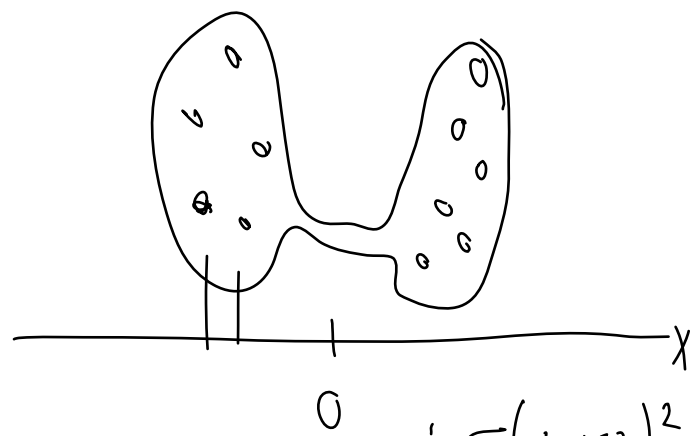
$$\sum x_i \cdot 1 = \sum_i x_i = 0$$

- Then:

$$\lambda_2 = \min_{\substack{\text{All labelings of} \\ \text{nodes so that} \\ \text{sum}(x_i)=0}} \frac{\sum (x_i - x_j)^2}{\underbrace{\sum x_i^2}_{=1}}$$

$\sum x_i^2 = 1$

$\sum x_i = 0$



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

$\sum x_i = 0, \quad \sum x_i^2 = 1$

Find Optimal Cut [Fiedler'73]

- Express partition (A,B) as a vector

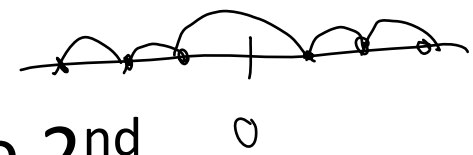
$$x_i = \begin{cases} +1 & \text{if } i \in A \\ -1 & \text{if } i \in B \end{cases}$$

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

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

Rayleigh Theorem

$$\min_x f(x) = \sum_{(i,j)\in E} (x_i - x_j)^2 = x^T L x$$



- The minimum value is given by the 2nd smallest eigenvalue λ_2 of the Laplacian matrix L
- The optimal solution for x is given by the corresponding eigenvector λ_2, referred as the Fiedler vector

So far...

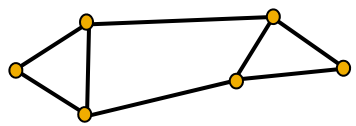
- How to define a "good" partition of a graph?
  - Minimise a given graph cut criterion
- How to efficiently identify such a partition?
  - Approximate using information provided by the eigenvalues and eigenvectors of a graph
- Spectral Clustering

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

- 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

- 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

x

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 (continued)

- 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, mean or median value
  - More expensive approaches:
    - Attempt to minimise normalized cut criterion in 1-dimension

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

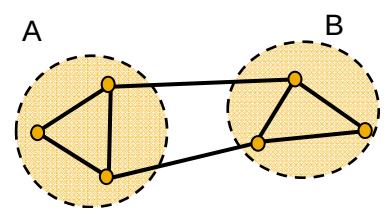
Split at 0:

Cluster A: Positive points

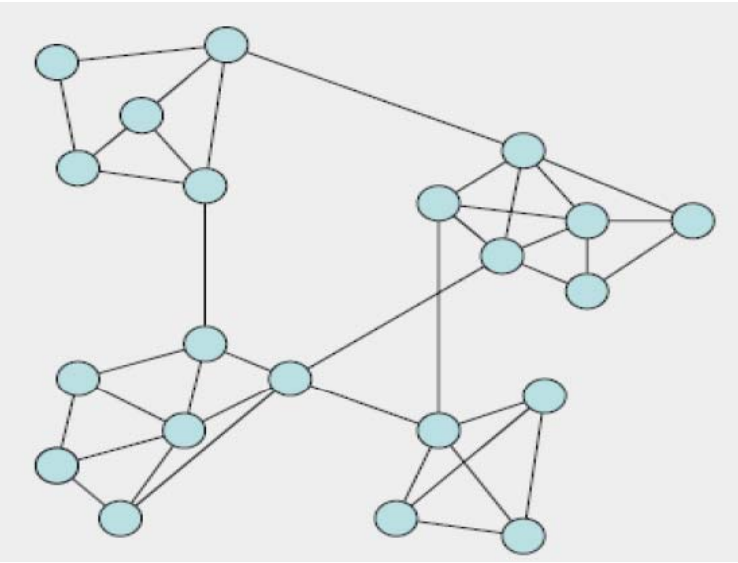
Cluster B: Negative points

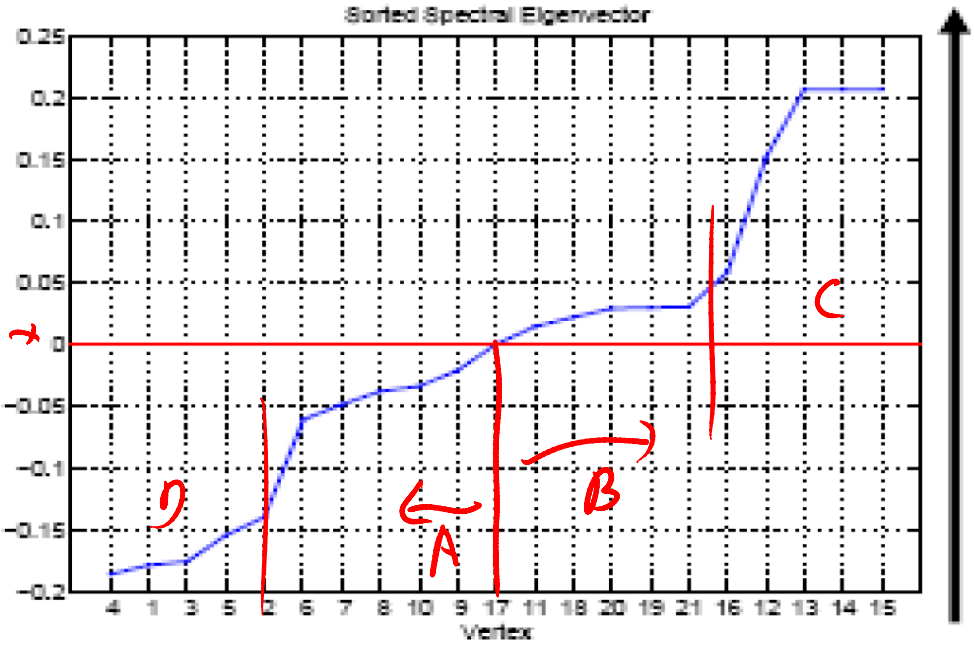
1	0.3
2	0.6
3	0.3

4	-0.3
5	-0.3
6	-0.6



Example: Spectral partitioning





K-Way Spectral Clustering

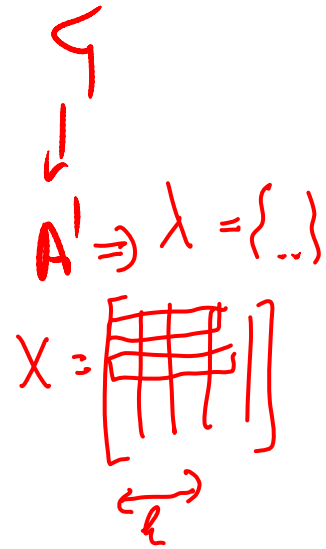
- How do we partition a graph into k clusters?
- Two basic approaches:
  - Recursive bi-partitioning [Hagen et al., '92]
    - Recursively apply bi-partitioning algorithm in a hierarchical divisive manner
    - Disadvantages: Inefficient, unstable
  - Cluster multiple eigenvectors [Shi-Malik, '00]
    - Build a reduced space from multiple eigenvectors
    - Commonly used in recent papers
    - A preferable approach…

k-Eigenvector Clustering

- k-eigenvector Algorithm [Ng et al.,'01]
  - Pre-processing:
    - Construct the scaled adjacency matrix A':

$$A' = D^{-1/2} A D^{-1/2}$$

  - Decomposition:
    - Find the eigenvalues and eigenvectors of A'
    - Build embedded space from the eigenvectors corresponding to the k largest eigenvalues
  - Grouping:
    - Apply k-means to reduced $n \times k$ space to get k clusters

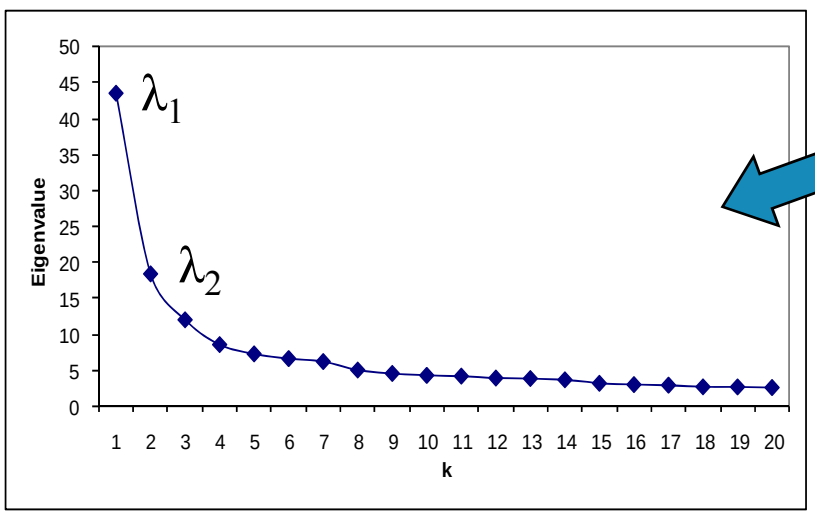


Why use multiple eigenvectors?

- Approximates the optimal cut [Shi-Malik, '00]
  - Can be used to approximate the 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
- NB: Multiple eigenvectors prevent instability due to information loss

How to select k?

- Eigengap:
  - The difference between two consecutive eigenvalues
- Most stable clustering is generally given by the value k that maximises eigengap: $\Delta_k = |\lambda_k - \lambda_{k-1}|$
- 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
- Cluto:
  - http://glaros.dtc.umn.edu/gkhome/views/cluto/
- Clique percorlation method:
  - For finding overlapping clusters
  - http://angel.elte.hu/cfinder/

Overlapping communities

- Non-overlapping vs. overlapping communities