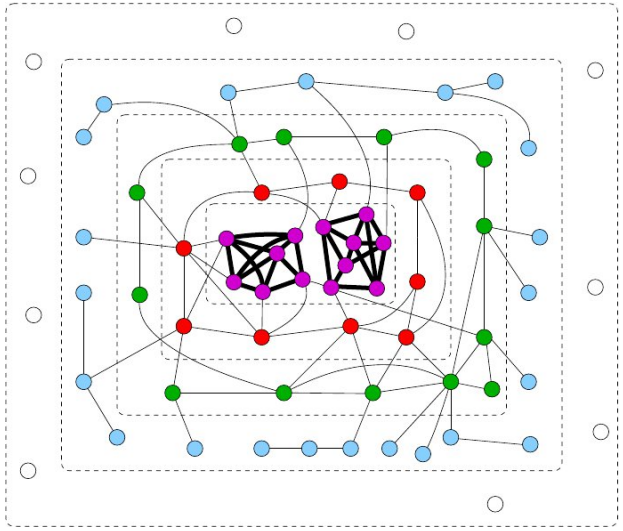
Graph Mining – II



Michalis Vazirgiannis

LIX @ Ecole Polytechnique

Outline

1. Introduction & Motivation
2. Graph Generators
3. **Supervised Learning for graphs**
    1. Graph Kernels – graph embeddings
    2. Graph classification
4. Unsupervised learning
    1. Community detection

Elements og Learning from Graph data

- **Graph models/ graph generators** graph generators (erdos reyni, preferential attachment, kronecker graphs)
- **Node base metrics**: - Ranking algorithms (Pagerank), Ranking evaluation measures (Kendal Tau, NDCG),
- **Graph exploration/preprocessing**: degree distributions, visualization
- **Supervised learning for graphs**: link prediction, graph kernels, graph classification
- **Unsupervised learning:** clustering, community mining, degeneracy.
- **Learning theory in graphs:** model ensembling/selection…

Graph Mining – pattern discovery perspective

Graph Pattern Mining

- Frequent graph patterns
- Pattern summarization
- Optimal graph patterns
- Graph patterns with constraints
- Approximate graph patterns

Graph Classification

- Pattern-based approach
- Decision tree
- Decision stumps

Graph Compression

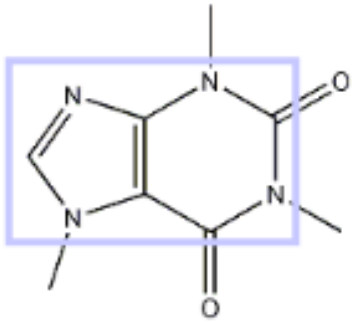
- Other important topics (graph model, laws, graph dynamics, social network analysis, visualization, summarizati
- on, graph clustering, link analysis, …)

Applications of Graph Patterns

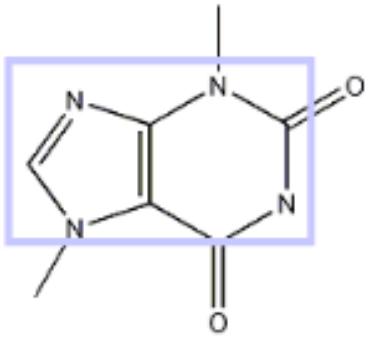
- Mining biochemical structures
- Finding biological conserved sub-networks
- Finding functional modules
- Program control flow analysis
- Intrusion network analysis
- Mining communication networks
- Anomaly detection
- Building blocks for graph classification, clustering, compression,
- comparison, correlation analysis, and indexing

Frequent sub-graph Mining

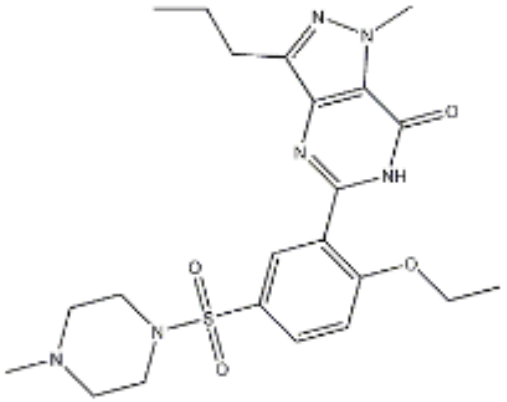
CHEMICAL COMPOUNDS



(a) caffeine



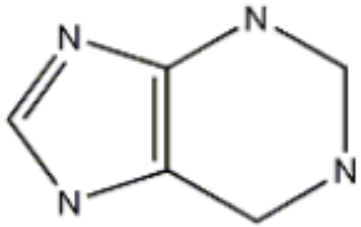
(b) diurobromine



(c) viagra

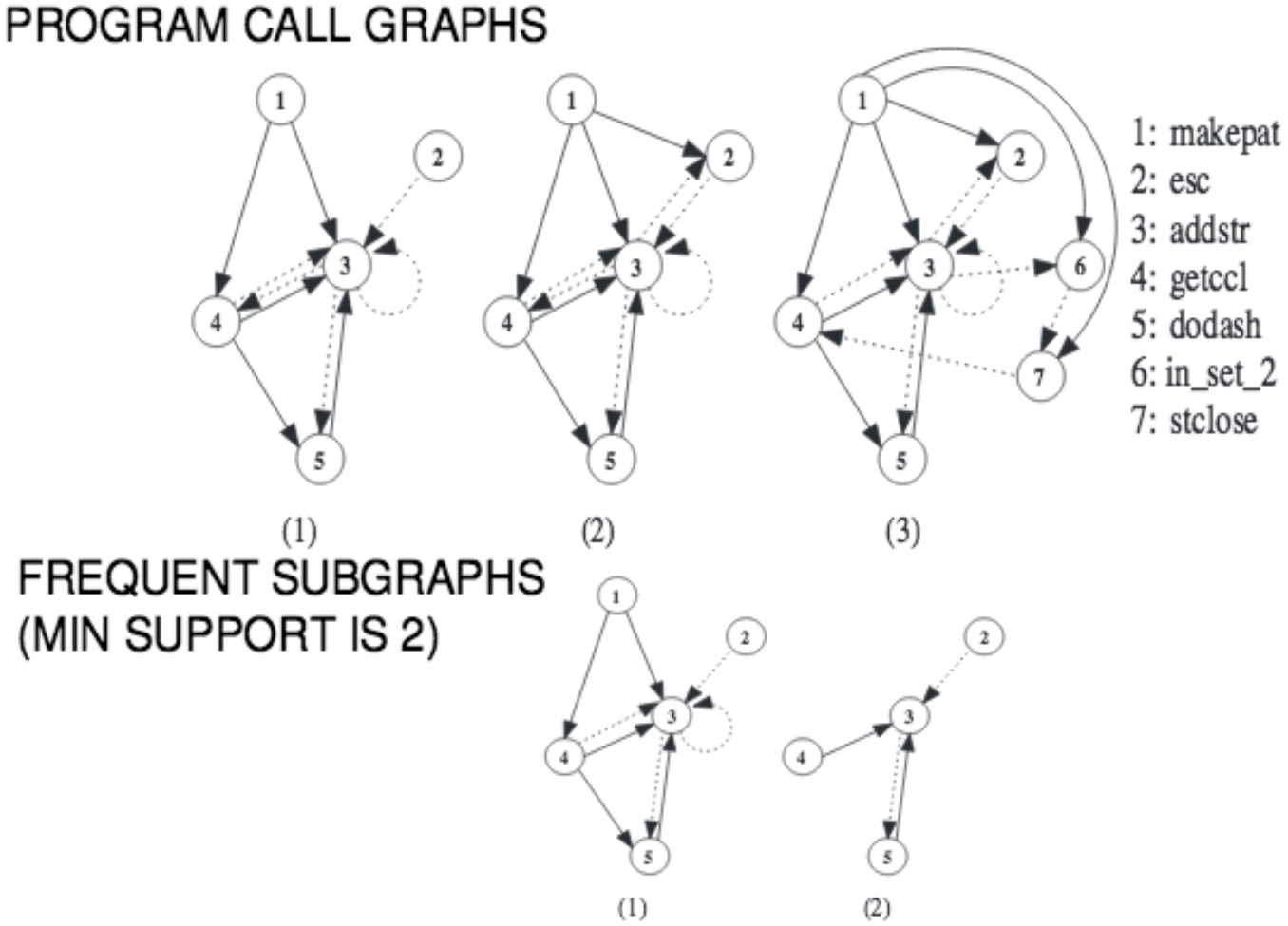
...

FREQUENT SUBGRAPH



https://www.cs.ucsb.edu/~xyan/tutorial/KDD08_graph_partI

Frequent sub-graph Mining



https://www.cs.ucsb.edu/~xyan/tutorial/KDD08_graph_partI

Frequent sub-graph mining

Apriori-based approaches

- –AGM/AcGM: Inokuchi, et al. (PKDD'00)
- –FSG: Kuramochi and Karypis (ICDM'01)
- –PATH#: Vanetik and Gudes (ICDM'02, ICDM'04)
- –FFSM: Huan, et al. (ICDM'03) and SPIN: Huan et al. (KDD'04)
- –FTOSM: Horvath et al. (KDD'06)

Pattern growth approach

- –Subdue: Holder et al. (KDD'94)
- –MoFa: Borgelt and Berthold (ICDM'02)
- *–gSpan: Yan and Han (ICDM'02)*
- –Gaston: Nijssen and Kok (KDD'04)
- –CMTreeMiner: Chi et al. (TKDE'05)
- –LEAP: Yan et al. (SIGMOD'08)

Apriori approach

- If a subgraph is frequent then all its sub-subgraphs should be frequent
- Cost of operation

$$T_{total} \propto \sum_{k} D_k * t_k^{isom}$$

- k: # elements to be checked (frequent, infrequent)
- D: data
- t_k^{isom} : time for checking isomorphism

Properties of Graph Mining Algorithms

Search Order

- breadth vs. depth
- complete vs. incomplete

Generation of Candidate Patterns

- apriori vs. pattern growth

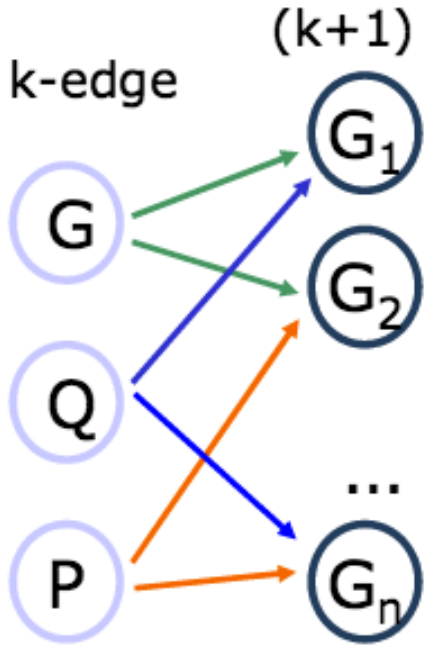
Discovery Order of Patterns

- DFS order,
- path, tree, graph

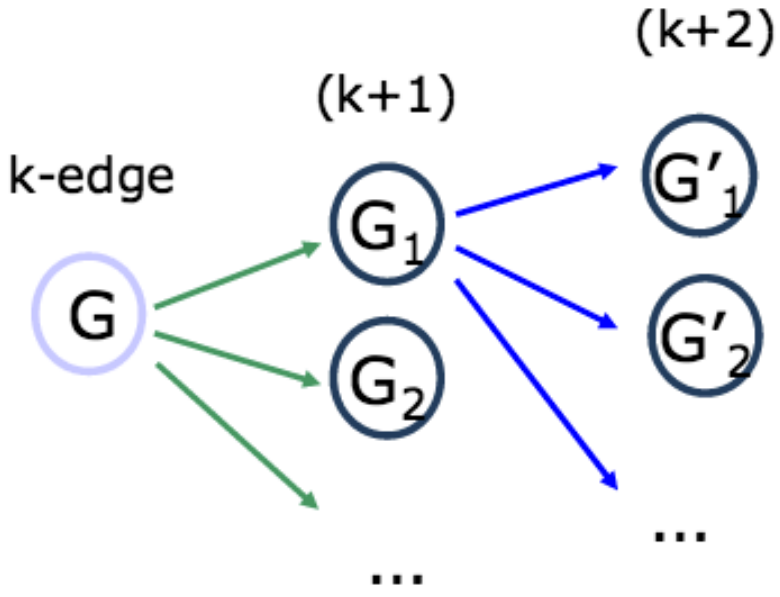
Elimination of Duplicate Subgraphs

- passive vs. active

Generation of Candidate Patterns



join



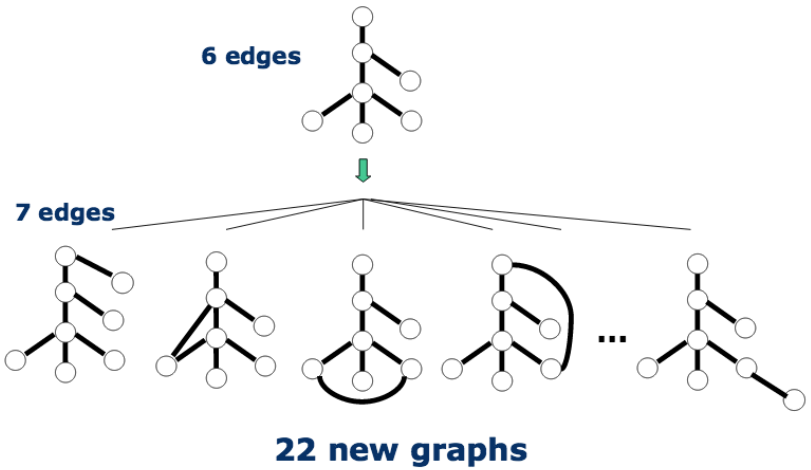
grow

Apriori-Based Approach VS. Pattern-Growth Approach

Graph pattern explosion problem…

- If a graph is frequent, all of its subgraphs are frequent – the Apriori property
- An n-edge frequent graph may have 2^n subgraphs!
- In the AIDS antiviral screen dataset with 400+ compounds, at the support level 5%, there are > 1M frequent graph patterns

- Many enumeration algorithms available (AGM, FSG, gSpan, Path-Join, MoFa, FFSM, SPIN, Gaston…)
- Two significant problems : Exponential pattern set, Threshold setting



Outline

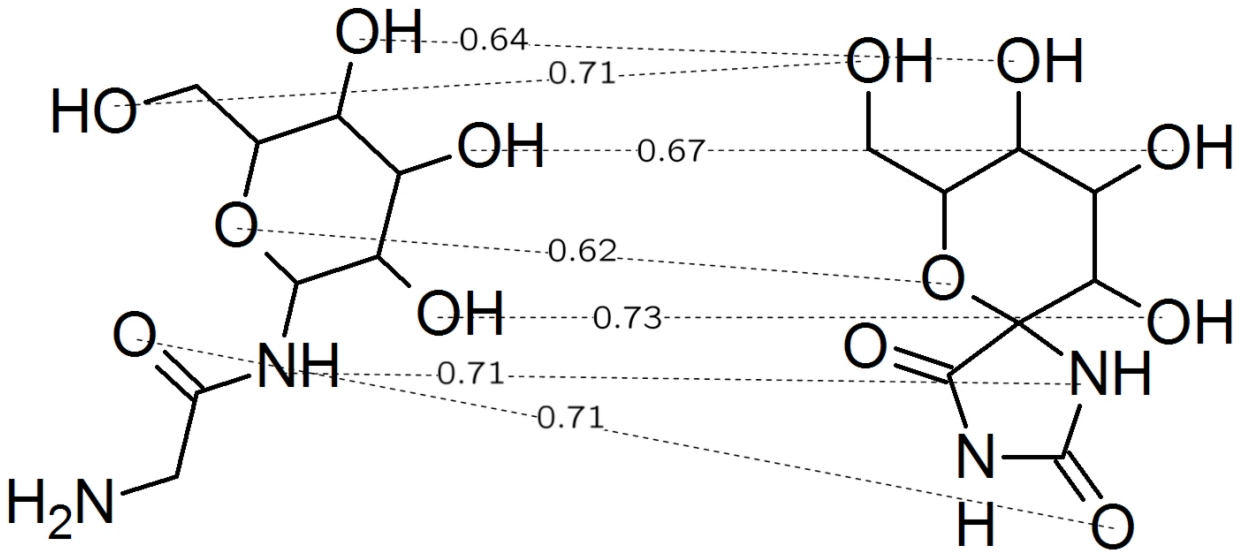
1. Introduction & Motivation
2. Graph Generators
3. **Supervised Learning for graphs**
    1. **Graph Kernels – graph embeddings**
    2. Graph classification
4. Unsupervised learning
    1. Community detection

Outline

1. Introduction & Motivation
2. Graph Generators
3. **Supervised Learning for graphs**
    1. Graph Kernels – graph embeddings
    2. Graph classification
4. Unsupervised learning
    1. Community detection

Graph Kernels

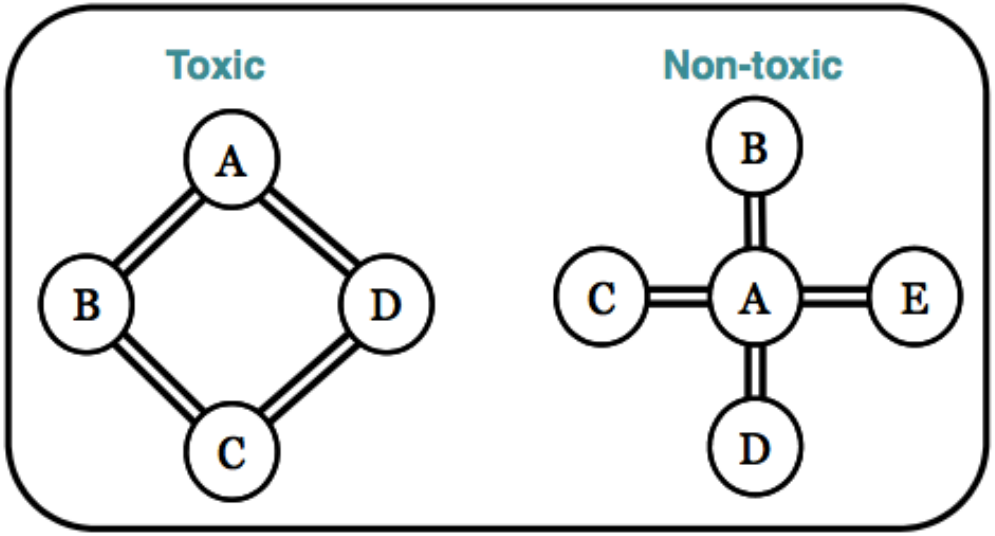
- Frequent Subgraph Mining seeks to find patterns in a dataset of graphs = pattern mining
- Graph Kernels aim at computing similarity scores between graphs in a dataset = graph similarity



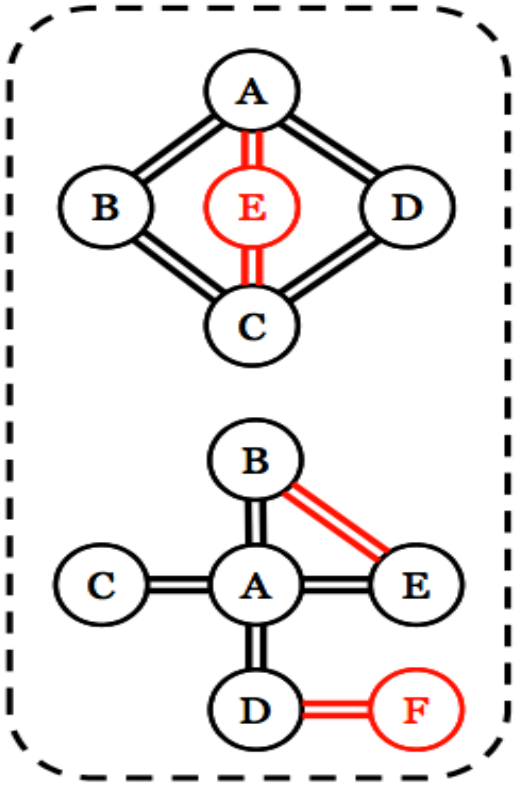
http://www.ra.cs.uni-tuebingen.de/forschung/terminiert/molsim/Mapping-GPB-Topological.jpg

Graph classification

Chemical compounds



Known



unknown=>

Predict toxic molecules assuming the known toxic ones

http://www.csc.ncsu.edu/faculty/samatova/practical-graph-mining-with-R/slides/pdf/Classification.pdf

Graph classification applications

Graphs classification applications

- Chemical compounds
- Molecular structures
- Textual data – text categorization, opinion mining, sentiment analysis, metaphor detection …
- Social networks – community similarity

Graph Classification

Structure-based Approach

- Local structures in a graph, e.g., neighbors surrounding a vertex, paths with fixed length

Pattern-based Approach

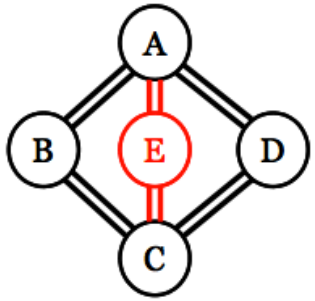
- Subgraph patterns from domain knowledge or from graph mining
- Decision Tree (Fan et al. KDD'08)
- Boosting (Kudo et al. NIPS'04)
- LAR-LASSO (Tsuda, ICML'07)

Kernel-based Approach

- Random walk (Gärtner '02, Kashima et al. '02, ICML'03, Mahé et al. ICML'04)
- Optimal local assignment (Fröhlich et al. ICML'05)
- ….

Classification based on graph structures

- Graph classification – i.e. recognize graph topologies
  - i.e. molecular graphs:



- Vertex classification – label prediction
  - i.e. predict in a social graph is the node is male/female.

Graph Isomorphism

- $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are isomorphic if
  - there exists a bijection(isomorphism) $f : V_1 \rightarrow V_2$ such that $(v_i , v_j) \in E_1$ if and only if $(f (v_i), f (v_j)) \in E_2$
  - the problem is NP-complete

Subgraph isomorphism:

- Aims to find a subset of edges and vertices of G1 that is isomorphic to a subset of G2
- could be a potential solution to graph similarity
- finding the largest isomorphic subgraphs is NP-hard
- Need a polynomial-time similarity measure for graphs

Graph Edit Distances

Principle

- Count operations that are necessary to transform G1 into G2
- Assign costs to different types of operations (edge/node insertion/ deletion, modification of labels)

Advantages

- Captures partial similarities between graphs
- Allows for noise in the nodes, edges and their labels
- Flexible way of assigning costs to different operations

Disadvantages

- Contains sub-graph isomorphism check as one intermediate step
- Choosing cost function for different operations is difficult

Topological Descriptors

Principle

- Map each graph to a feature vector
- Use distances and metrics on vectors for learning on graphs

Advantages

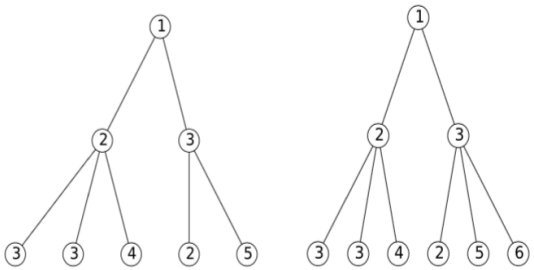
- Reuses known and efficient tools for feature vectors

Disadvantages

- Efficiency comes at a price: feature vector transformation leads to loss of topological information (or includes subgraph isomorphism as one step)
- Can use embedding to metric spaces as well (i.e. NNs)

Graph Embeddings

- Graphs non-vectorial form makes the task of using them as input to machine learning algorithms very complex,
- Challenge: representing graphs or *graph nodes* as vectors
- graph representations in vector spaces to enable input to ML algorithms.
- Embeddings based on the instances of a substructure
  - Sub trees, connected sub graphs of size k, shortest paths



- Evaluate the graph embedding methods for graph classification
  - Compare their performance to state-of-the-art graph kernels.
  - Graph Embeddings outperform kernels
  - time complexity remains very attractive.

Nikoletzos, Vazirgiannis 2016, submitted

Graph kernels

Kernels on *pairs of graphs*

- Not pairs of nodes… although this is interesting as well
- A graph kernel makes the whole family of kernel methods applicable to graphs

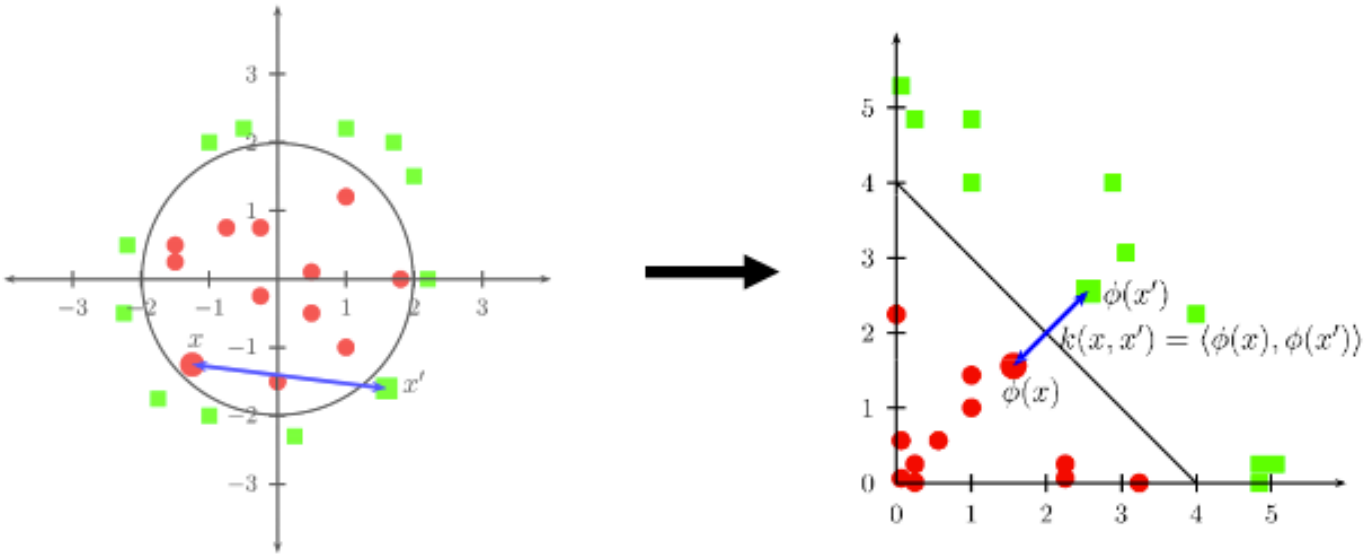
Compare substructures of graphs that are computable in polynomial time.

Define a similarity measure between graphs computable in polynomial time

- Desired Properties of Graph Kernels
  - Symmetric, Semi-positive definite
  - Fast to compute
  - Expressive (captures topological similarity between graphs)
- Categories of Graph Kernels – based on:
  - walks and paths
  - limited-sized subgraphs
  - Sub tree patterns
  - others (based on graph edit distance etc.)

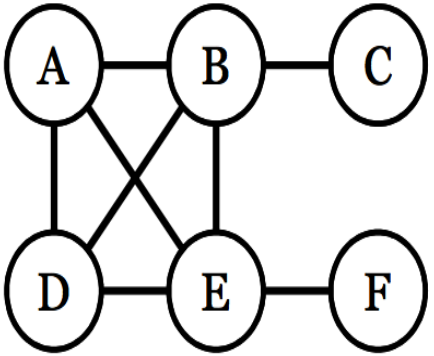
Kernel Notion

- Map two objects x and x' via mapping ϕ into feature space H
- Measure their similarity in H as
- Kernel Trick: Compute inner product in K as kernel in input space $k(x,x')=<\phi(x),>$

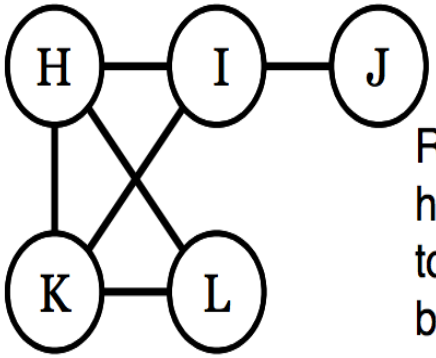


Random walk Graph kernels

Random walk vertices heavily distributed towards A,B,D,E

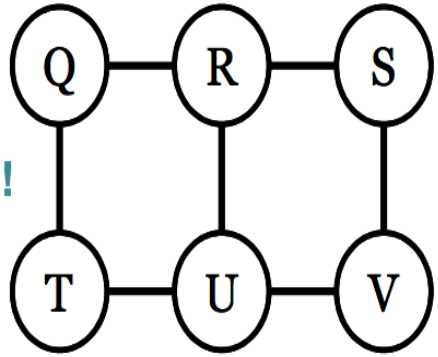


Similar!



Random walk vertices heavily distributed towards H,I,K with slight bias towards L

Not Similar!



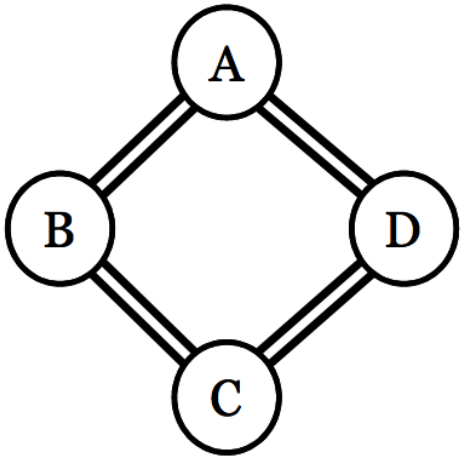
Random walk vertices evenly distributed

http://www.csc.ncsu.edu/faculty/samatova/practical-graph-mining-with-R/slides/pdf/Classification.pdf

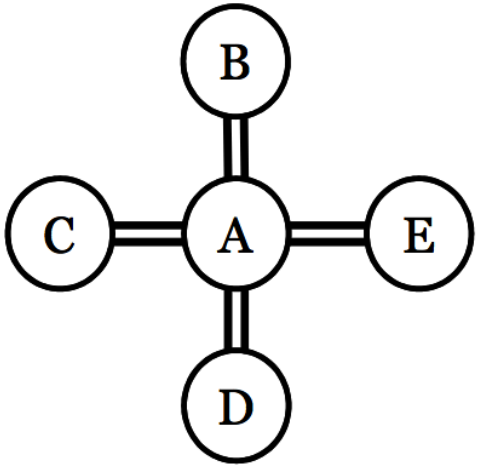
Product graph

- Assume two graphs $G_1(V_1, E_1)$, $G_2(V_2, E_2)$
- Direct Product notation: $G_x = G_1 x\ G_2$
- Direct product vertices: $V(G_x) = \{(a,b) \in (V_1 x V_2)\}$
- Direct product edges: $E(G_x) = \{(a,b),(c,d) \mid (a,c) \in E_1\ \&\ (b,d) \in E_2\}$

Graph Product - example



Type-A



Type-B

Type-A	A	B	C	D
A	0	1	1	0
B	1	0	0	1
C	1	0	0	1
D	0	1	1	0

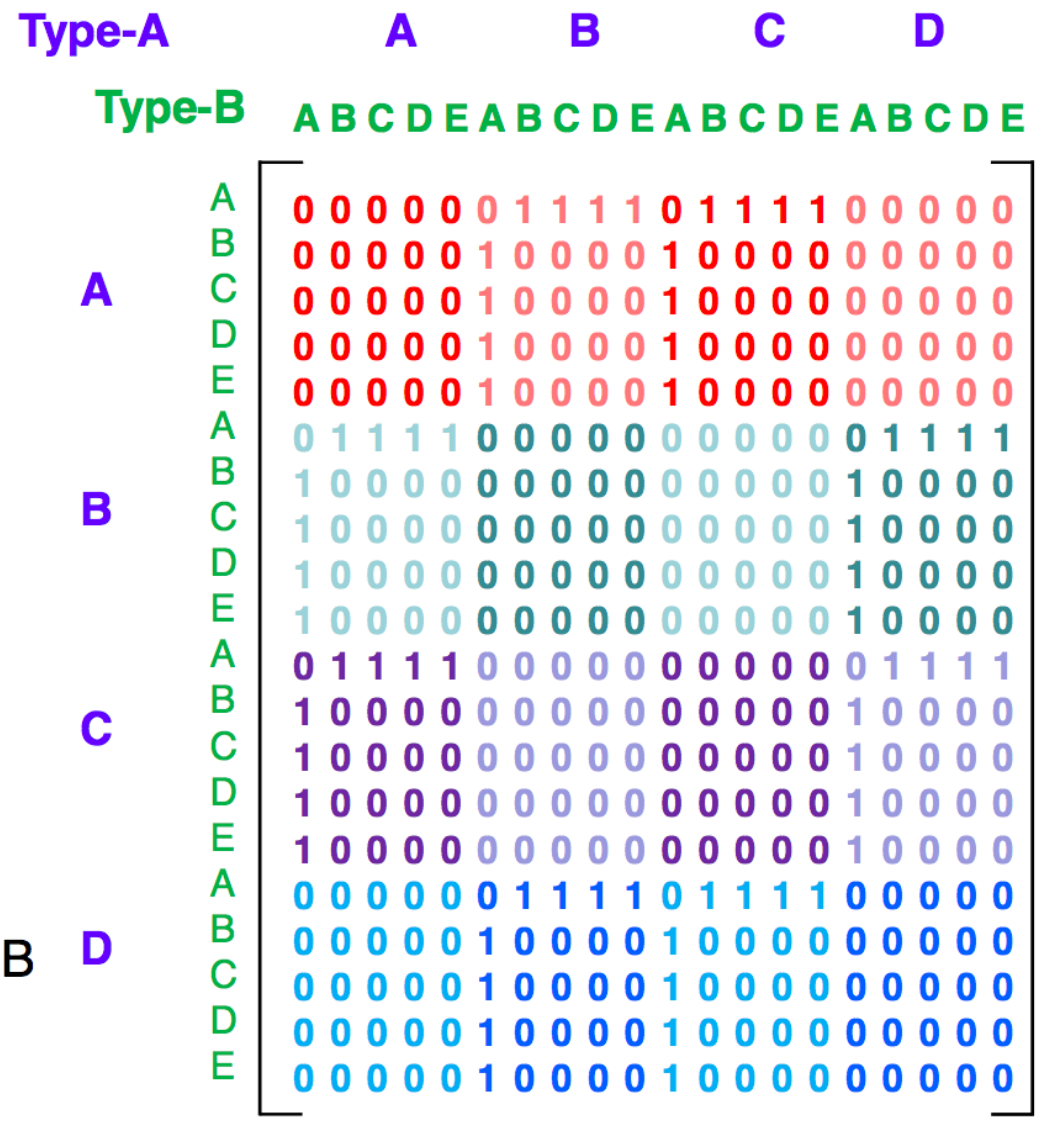
Type-B	A	B	C	D	E
A	0	1	1	1	1
B	1	0	0	0	0
C	1	0	0	0	0
D	1	0	0	0	0
E	1	0	0	0	0

Graph Product - example

Type-A	A	B	C	D
A	0	1	1	0
B	1	0	0	1
C	1	0	0	1
D	0	1	1	0

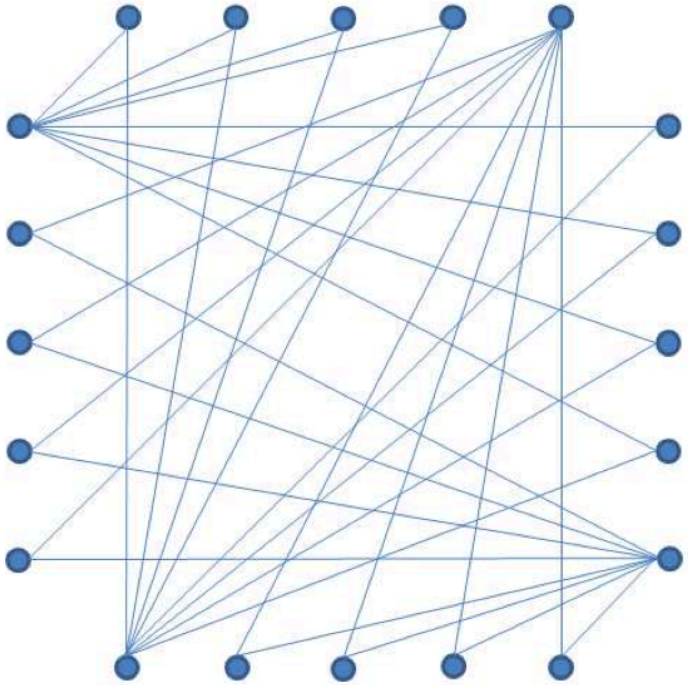
Type-B	A	B	C	D	E
A	0	1	1	1	1
B	1	0	0	0	0
C	1	0	0	0	0
D	1	0	0	0	0
E	1	0	0	0	0

Intuition: multiply each entry of Type-A by ***entire matrix*** of Type-B



Direct Product Kernel

- Compute direct product graph
- Decay constant $\gamma < 1/\min(d_i, d_o)$
  - d_i/d_o: max in/out degree in G_x
- Compute the weighted geometric series of walks (array A).
- Sum over all vertex pairs.



$$k(G_1, G_2) = \sum_{ij} \left(I - \frac{A_{ij}}{\gamma} \right)^{-1}$$

Random Walk Kernels - Preliminaries

- graph $G = (V,E),\ V = \{v_1, v_2, \cdots, v_n\}$ ordered set of n vertices, $E \subset V \times V$ set of edges
- The adjacency matrix A of G is defined as $[A_{ij}] = \{1 \text{ if } (v_i, v_j) \in E, 0 \text{ otherwise}\}$
- A walk w on the graph is a sequence of nodes $w = (v_1, v_2, \cdots, v_n)\ , (v_i, v_{i+1},) \in E$

Random Walk Kernel

- Graphs with matching walks are similar
- Performing a random walk on the direct product graph $G_\times$ is equivalent to simultaneous random walk on the graphs G_1 and G_2

  - The A^k_X entry represents the probability of simultaneous length-k random walks on G and G'.
- Number of walks of length *k* from v_i to v_j can be computed by A^k_{ij}
- Discount longer walks by the factor of λ

How to compute common walks between two graphs?

- The product graph $G\times = (V_\times, E_\times)$ of G1 and G2:

$$V_\times = \{(v,w) \in V_1 \times V_2 : l(v) = l(w)\},\ l(v):\ \text{label of vertex } v$$

$$E_\times = \{((v_1,w_1),(v_2,w_2)) \in V_\times^2 : (v1,v2) \in E1 \text{ and } (w_1,w_2) \in E_2 \text{ and } l(v_1,w_1) = l(v_2,w_2)\}$$

- Common walks can be thought of as walks within the product graph

Random Walk Kernel

- Random walk kernel between two graphs G1 and G2 is defined as

$$k_x\left(G_1, G_2\right) = \sum_{i,j=1}^{|V_x|} [\sum_{n=0}^{\infty} \lambda^n A_X^n]_{ij} = e^T (I - \lambda \mathrm{A}_x)^{-1} e$$

- where $e = (1,1,\cdots,1)^T$ start/termination prob distribution
- A_x, un-normalized adjacency matrix of the direct product graph.
- A_X^n represents similarity between simultaneous length *n* walks on G and G', measured via the kernel function κ

- Requires matrix inversion of $n^2 \times n^2$ matrix
- Time complexity $O(n^6)$
- Can be improved to $O(n^3)$ (Vishwanathan et al., 2010)
  - Employ Sylvester equation $M = SMT + M_0$ where $S,T,M_0 \in Rn \times n$ are given and we need to solve for $M \in Rn \times n$

Random walk kernels alternatives

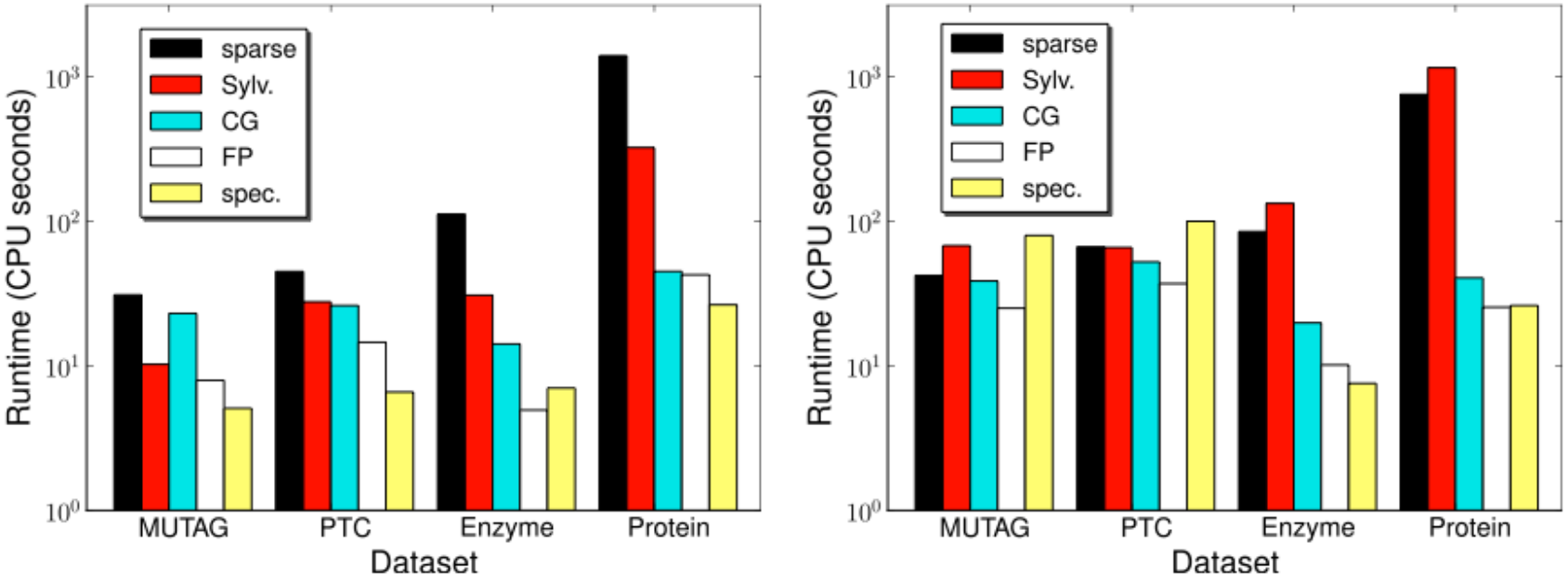


Figure 5: Time (in seconds on a log-scale) to compute 100×100 kernel matrix for unlabeled (left) *resp*. labeled (right) graphs from several data sets, comparing the conventional sparse method to our fast Sylvester equation, conjugate gradient (CG), fixed-point iteration (FP), and spectral approaches.

Kernel Matrix

- Assum

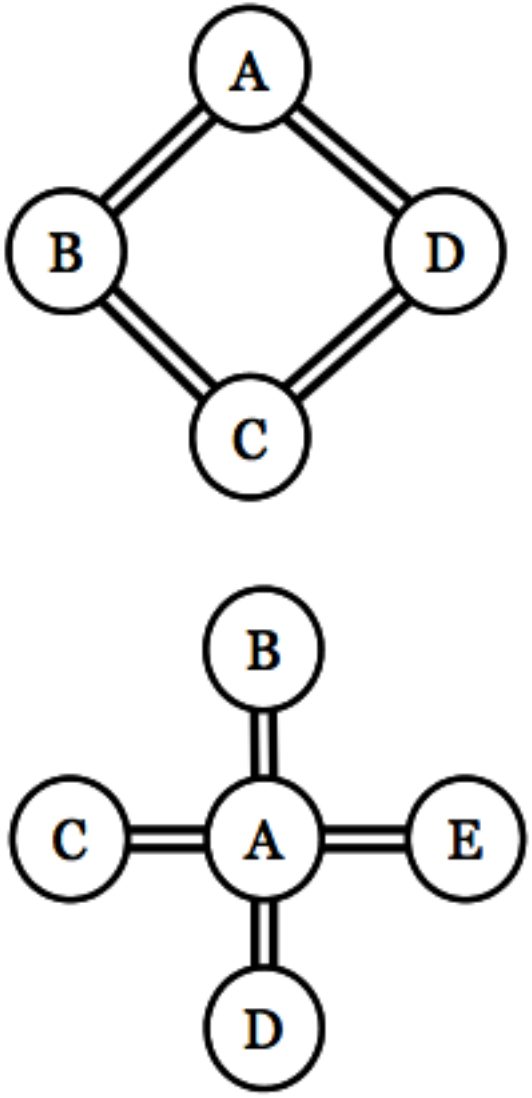
$$\begin{bmatrix} K(G_1, G_1), K(G_1, G_2), \ldots, K(G_1, G_n) \\ K(G_2, G_1), K(G_2, G_2), \ldots, K(G_2, G_n) \\ \ldots \\ K(G_n, G_1), K(G_n, G_2), \ldots, K(G_n, G_n) \end{bmatrix}$$

- This matrix is used as input to the classification learning method (i.e. SVM) function to learn the model.

Predicting toxic compounds

Assume the PTC dataset of molecules tested for positive or negative toxicity

```
# Learning pipeline (SVM ) in R
data("PTCData") # graph data
data("PTCLabels") # toxicity
information
# select 5 molecules to build model
on
sTrain= sample(1:length(PTCData),5)
PTCDataSmall<-PTCData[sTrain]
PTCLabelsSmall<-PTCLabels[sTrain]
# generate kernel matrix
K =
generateKernelMatrix(PTCDataSmall,
PTCDataSmall)
# create SVM model
model =ksvm(K, PTCLabelsSmall,
kernel='matrix')
```



Outline

1. Introduction & Motivation
2. Graph Generators
3. Supervised Learning for graphs
    1. Graph Kernels – graph embeddings
    2. Graph classification
4. **Unsupervised learning**
    1. Community detection

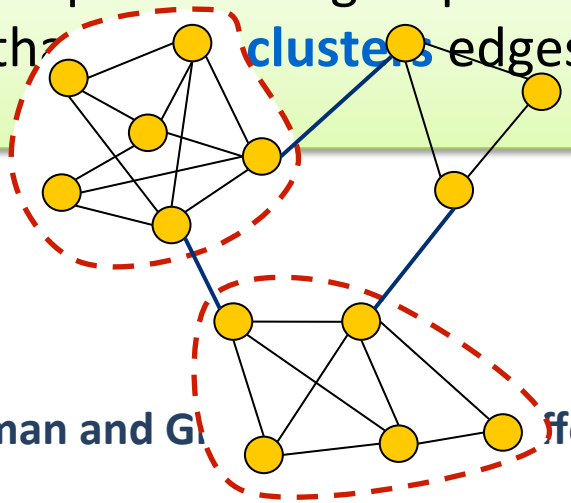
Outline

1. Introduction & Motivation
2. **Community evaluation measures**
3. Graph clustering
4. Graph classification

Basics

- The notion of **community structure** captures the tendency of nodes to be organized into modules (communities, clusters, groups)
  - Members within a community are **more similar** among each other
- Typically, the communities in graphs (networks) correspond to **densely connected** entities (nodes)

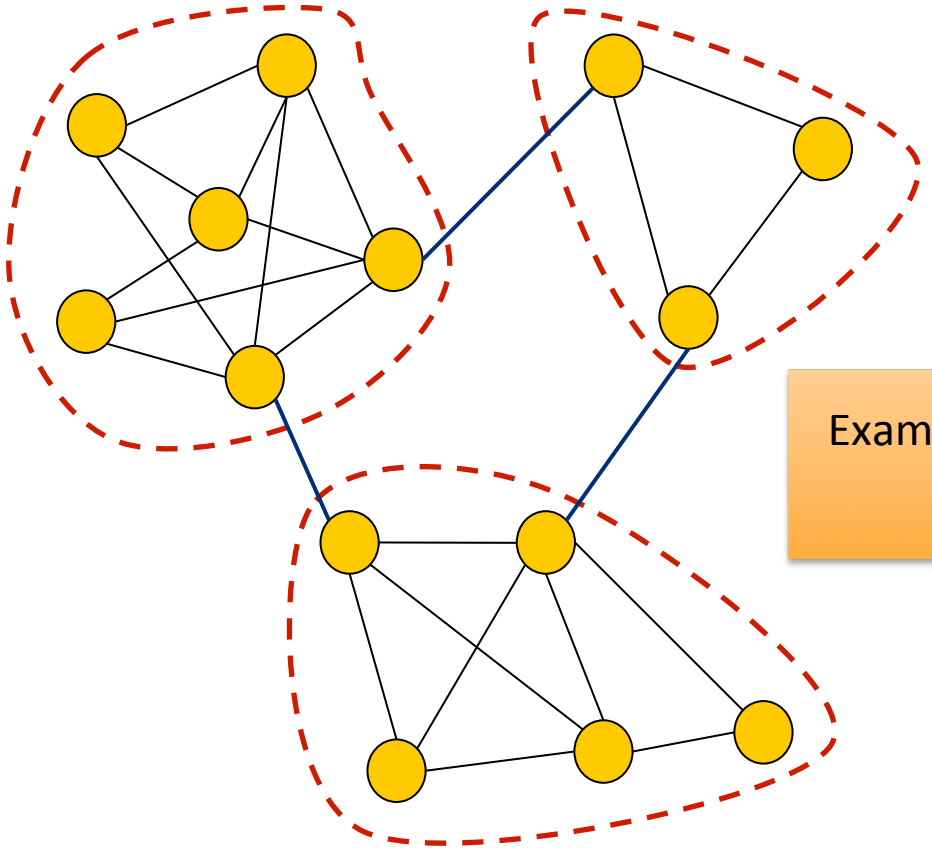
A community corresponds to a group of nodes with more **intra-cluster** edges than **cluster** edges



Example graph with three communities

[Newman '03], [Newman and G ffer '07], [Fortunato '10],
[Danon et al. '05], [Coscia et al. 11]

Schematic representation of communities



Example graph with three communities

Community detection in graphs

- How can we extract the inherent communities of graphs?
- Typically, a two-step approach
  1. Specify a **quality measure** (evaluation measure, objective function) that quantifies the desired properties of communities
  2. Apply **algorithmic techniques** to assign the nodes of graph into communities, optimizing the objective function
- Several measures for quantifying the quality of communities have been proposed
- They mostly consider that communities are set of nodes with many edges between them and few connections with nodes of different communities
  - Many possible ways to formalize it

Community evaluation measures

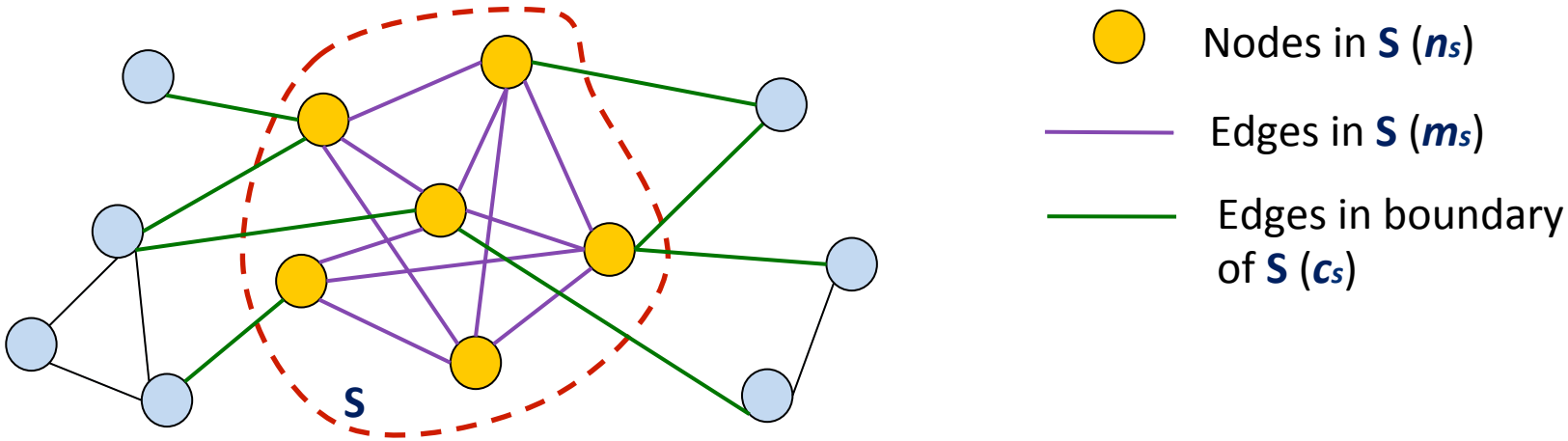
- Focus on
  - Intra-cluster edge density (# of edges within community),
  - Inter-cluster edge density (# of edges across communities)
  - Both two criteria

- We group the community evaluation measures according to
  - Evaluation based on **internal** connectivity
  - Evaluation based on **external** connectivity
  - Evaluation based on **internal and external** connectivity
  - Evaluation based on **network model**

[Leskovec et al. '10], [Yang and Leskovec '12], [Fortunato '10]

Notation

- **G = (V, E)** is an undirected graph, **|V| = n**, **|E| = m**
- **S** is the set of nodes in the cluster
- n_s **= |S|** is the number of nodes in **S**
- m_s is the number of edges in **S**, $m_s = \left|\{(u,v): u \in S, v \in S\}\right|$
- c_s is the number of edges on the boundary of **S**, $c_s = \left|\{(u,v): u \in S, v \notin S\}\right|$
- d_u is the degree of node u
- f **(S)** represent the clustering quality of set S



Evaluation based on external connectivity

- **Expansion** [Radicchi et al. '04]

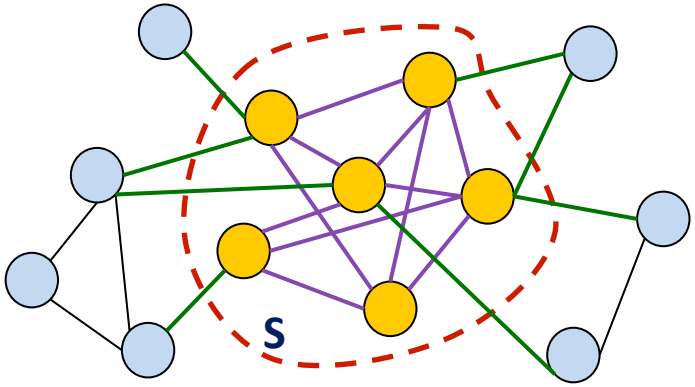
$$f(S) = \frac{c_s}{n_s}$$

Measures the number of edges per node that point outside **S**

- **Cut ratio** [Fortunato '10]

$$f(S) = \frac{c_s}{n_s(n - n_s)}$$

Fraction of existing edges – out of all possible edges – that leaving **S**

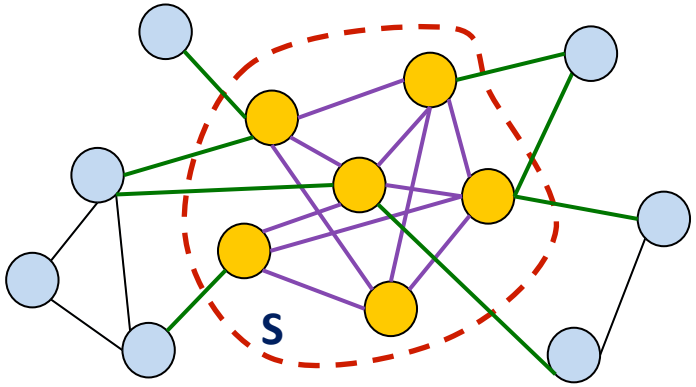


Evaluation based on internal connectivity (1)

- **Internal density** [Radicchi et al. '04]

$$f(S) = \frac{m_s}{n_s(n_s - 1)/2}$$

Captures the internal edge density of community **S**



- **Edges inside** [Radicchi et al. '04]

$$f(S) = m_s$$

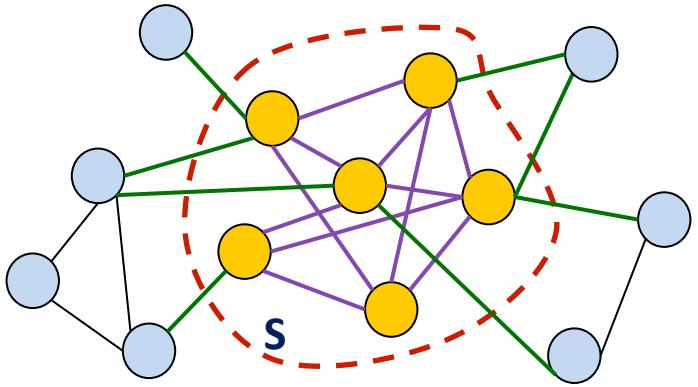
Number of edges between the nodes of **S**

Evaluation based on internal and external connectivity (2)

- **Conductance** [Chung '97]

$$f(S) = \frac{c_s}{2m_s + c_s}$$

Measures the fraction of total edge volume that points outside **S**



- **Normalized cut** [Shi and Malic '00]

$$f(S) = \frac{c_s}{2m_s + c_s} + \frac{c_s}{2(m - m_s) + c_s}$$

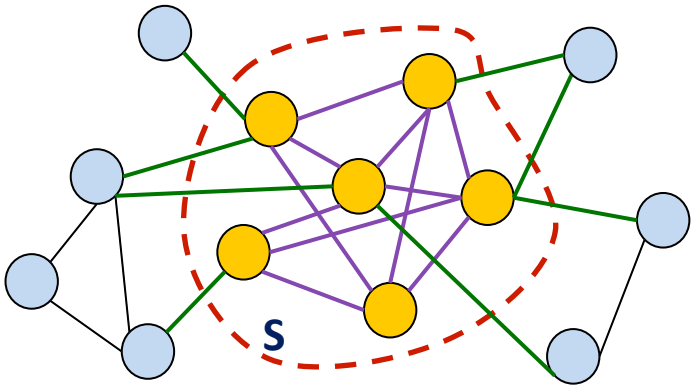
Measures the fraction of total edge volume that points outside **S** normalized by the size of **S**

Evaluation based on internal connectivity (3)

- **Triangle participation ratio (TPR)** [Yang and Leskovec '12]

$$f(S) = \frac{\left|\left\{u : u \in S, \left\{(v,w) : v,w \in S, (u,v) \in E, (u,w) \in E, (v,w) \in E\right\} \neq \varnothing\right\}\right|}{n_s}$$

Fraction of nodes in **S** that belong to a triangle



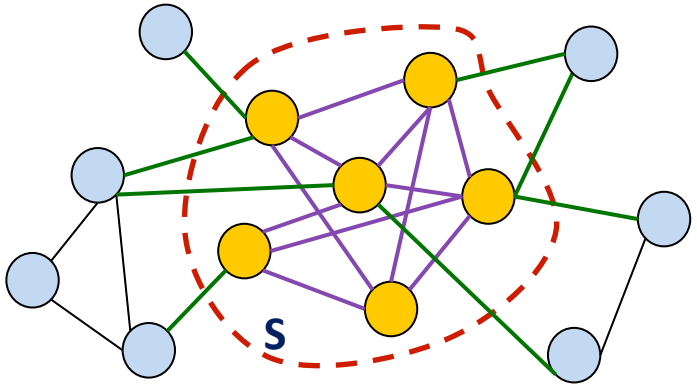
Evaluation based on network model

- **Modularity** [Newman and Girvan '04], [Newman '06]

$$f(S) = \frac{1}{4}\left(m_s - E\left(m_s\right)\right)$$

Measures the difference between the number of edges in **S** and the expected number of edges **E(m_s)** in case of a configuration model

- Typically, a random graph model with the same degree sequence



Outline

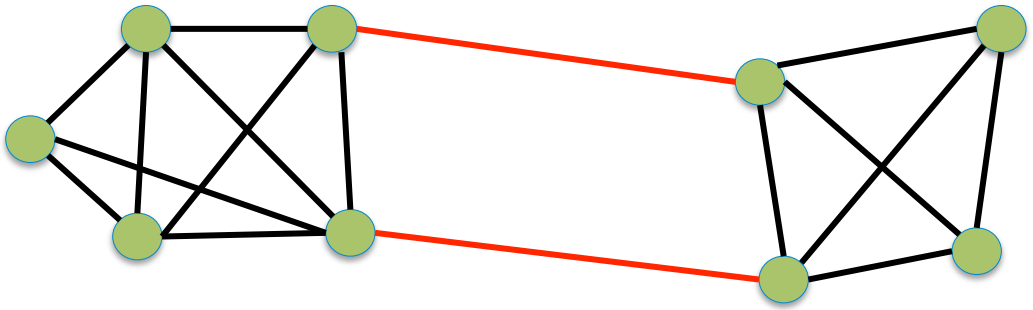
1. Introduction & Motivation
2. Community evaluation measures
3. **Graph clustering**
4. Graph classification

Notations

- Given Graph G=(V,E) undirected:
  - Vertex Set V={v_1,…..v_n}, Edge e_{ij} between v_i and v_j
    - we assume weight $w_{ij}>0$ for e_{ij}
  - |V| : number of vertices
  - d_i degree of v_i : $d_i = \sum_{v_j \in V} w_{ij}$
  - $v(V) = \sum_{v_i \in V} d_i$
  - for $A \subset V$ $\bar{A} = V - A$
  - Given $A, B \subset V$ & $A \cap B = \emptyset, w(A,B) = \sum_{v_i \in A, vj \in B} w_{ij}$
  - D : Diagonal matrix where $D(i,i) = d_i$
  - W : Adjacency matrix $W(i,j) = w_{ij}$

Graph-Cut

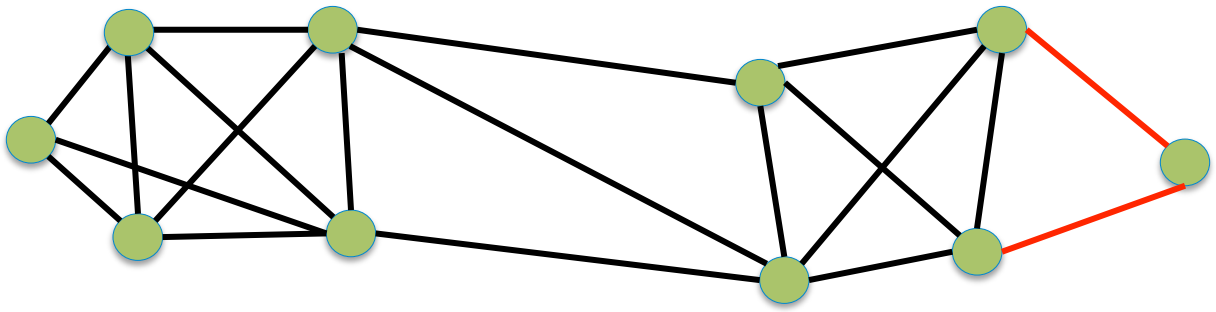
- For k clusters:
  - $cut(A_1, .., Ak) = 1/2 \sum_{i=1}^{k} w(Ai, \overline{A_i})$
    - undirected graph:1/2 we count twice each edge



- Min-cut:Minimize the edges' weight a cluster shares with the rest of the graph

Min-Cut

- Easy for k=2 : Mincut(A_1,A_2)
  - Stoer and Wagner: “A Simple Min-Cut Algorithm”
- In practice one vertex is separated from the rest
  - The algorithm is drawn to outliers



Normalized Graph Cuts

- We can normalize by the size of the cluster (size of sub-graph) :
  - number of Vertices (Hagen and Kahng, 1992):

    $$Ratiocut(A_1, \dots Ak) = \sum_{i=1}^{k} \frac{cut(Ai, \overline{A_i})}{|Ai|}$$

  - sum of weights (Shi and Malik, 2000) :

    $$Ncut(A_1, \dots Ak) = \sum_{i=1}^{k} \frac{cut(Ai, \overline{A_i})}{v(A_i)}$$

- Optimizing these functions is NP-hard
- Spectral Clustering provides solution to a relaxed version of the above

From Graph Cuts to Spectral Clustering

- For simplicity assume k=2:
  - Define $f: V \to \mathbb{R}$ for Graph G :

$$f_i = \begin{cases} 1 & v_i \in A \\ -1 & v_i \in \overline{A} \end{cases}$$

- Optimizing the original cut is equivalent to an optimization of:

$$\sum_{i,j=1}^{n} w_{ij}\left(f_i - f_j\right)^2$$

$$= \sum_{v_i \in A, vj \in \overline{A}} w_{ij}(1+1)^2 + \sum_{v_i \in \overline{A}, vj \in A} w_{ij}(-1-1)^2$$

$$= \mathbf{8} * \boldsymbol{cut}\left(\boldsymbol{A}, \overline{\boldsymbol{A}}\right)$$

Graph Laplacian

- How is the previous useful in Spectral clustering?

$$\sum_{i,j=1}^{n} w_{ij}(f_i - fj)^2$$

$$= \sum_{i,j=1}^{n} w_{ij} {f_i}^2 - 2\sum_{i,j=1}^{n} w_{ij} f_i f_j + \sum_{i,j=1}^{n} w_{ij} {f_j}^2$$

$$= \sum_{i,j=1}^{n} d_i {f_i}^2 - 2\sum_{i,j=1}^{n} w_{ij} f_i f_j + \sum_{i,j=1}^{n} d_j {f_j}^2$$

$$= 2\left(\sum_{i,j=1}^{n} d_{ii} {f_i}^2 - \sum_{i,j=1}^{n} w_{ij} f_i f_j\right)$$

$$= 2\left(\boldsymbol{f^T D f - f^T W f}\right) = 2\boldsymbol{f^T (D - W) f} = 2\boldsymbol{f^T L f}$$

- **f:**a single vector with the cluster assignments of the vertices
- **L=D-W :** the Laplacian of a graph

Properties of L

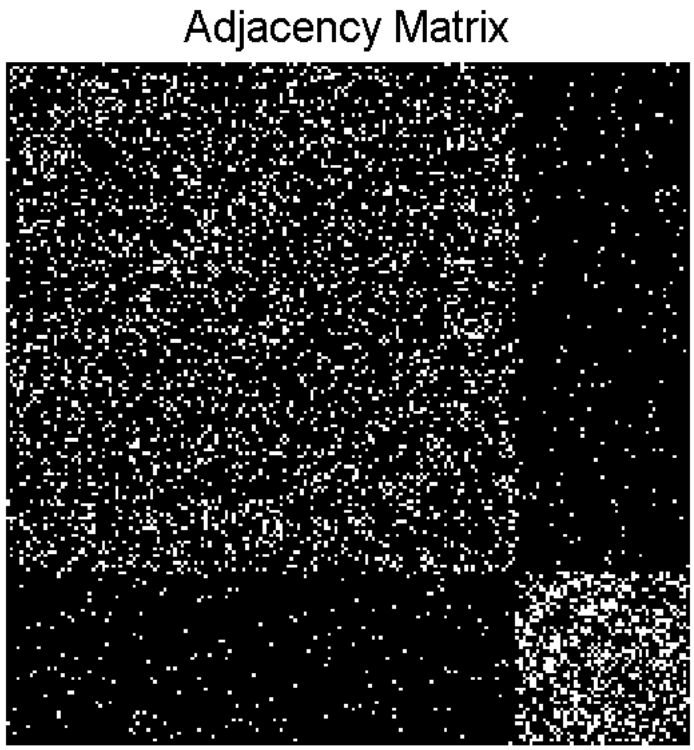
- L is
  - Symmetric
  - Positive
  - Semi-definite
- The smallest eigenvalue of L is 0
  - The corresponding eigenvector is $\mathbb{1}$
- L has n non-negative, real valued eigenvalues
  - $0 = \lambda 1 \leq \lambda 2 \leq \ldots \leq \lambda n$

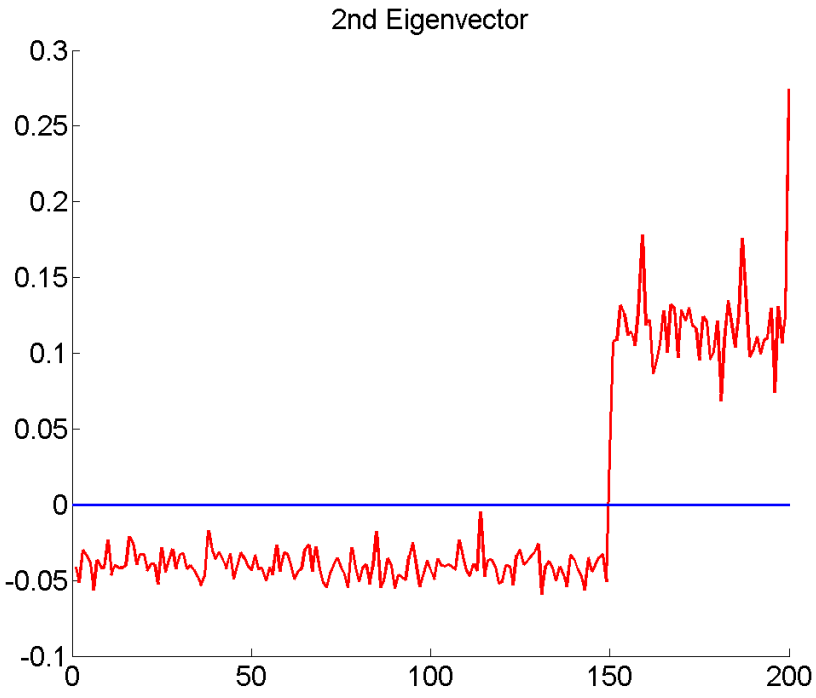
Two Way Cut from the Laplacian

- We could solve $min_f f^T L f$ where $f \in \{-1,1\}^n$
- NP-Hard for discrete cluster assignments
  - Relax the constraint $to\ f \in R^n$:

    $min_f f^T L f$ subject to $f^T f = n$
- The solution to this problem is given by:
  - (**Rayleigh-Ritz Theorem**) the eigenvector corresponding to smallest eigenvalue: 0 and the corresponding eigenvector (full of 1s) offers no information
- We use the second eigenvector as an approximation
  - $f_i > 0$ the vertex belongs to one cluster , $f_i < 0$ to the other

Example


Adjacency Matrix


2nd Eigenvector
0.3
0.25
0.2
0.15
0.1
0.05
0
-0.05
-0.1
0
50
100
150
200

Ratio Cut

- $Ratiocut(A_1, \dots Ak) = \sum_{i=1}^{k} \frac{cut(Ai, \overline{A_i})}{|Ai|}$
  - Define $f: V \to \mathbb{R}$ for Graph G :

$$f_i = \begin{cases} \sqrt{\frac{|\overline{A}|}{|A|}} & vi \in A \\ -\sqrt{\frac{|A|}{|\overline{A}|}} & v_i \in \overline{A} \end{cases}$$

  - $\sum_{i,j=1}^{n} w_{ij}(f_i - fj)^2 = 2cut(\mathrm{A}, \overline{A}) \left(\sqrt{\frac{|\overline{A}|}{|A|}} + \sqrt{\frac{|A|}{|\overline{A}|}} + 2 \right)$

    $= 2|V|Ratiocut(A, \overline{A})$

Ratio Cut

- We have $min_f f^T L f$ subject to

$$f^T 1 = 0, \; fTf = n$$

$$f^T 1 = \sum_i^n f_i = \sum_{v_i \in A} \sqrt{\frac{|\bar{A}|}{|A|}} + \sum_{v_i \in \bar{A}} - \sqrt{\frac{|A|}{|\bar{A}|}} = |A| \sqrt{\frac{|\bar{A}|}{|A|}} - |\bar{A}| \sqrt{\frac{|A|}{|\bar{A}|}} = 0$$

$$f^T f = \sum_i^n f_i^2 = |\bar{A}| + |A| = n$$

- The second smallest eigenvalue of $Lf = \lambda f$ approximates the solution

Normalized Cut

- $Ncut(A_1, \dots Ak) = \sum_{i=1}^{k} \frac{cut(Ai, \overline{A_i})}{v(A_i)}$
- Define $f: V \to \mathbb{R}$ for Graph G :

$$f_i = \begin{cases} \sqrt{\frac{v(\overline{A})}{v(A)}} & vi \in A \\ -\sqrt{\frac{v(A)}{v(\overline{A})}} & vi \in \overline{A} \end{cases}$$

- $\sum_{i,j=1}^{n} w_{ij}(f_i - fj)^2 = 2cut(\mathrm{A}, \overline{A}) \left(\sqrt{\frac{v(\overline{A})}{v(A)}} + \sqrt{\frac{v(A)}{v(\overline{A})}} + 2 \right)$

$$= 2v(\mathrm{V})\mathrm{Ncut}(\mathrm{A}, \overline{A})$$

Normalized Cut

- Similarly: $\min_f f^T L f$ subject to

$$f^T D 1 = 0, f^T D f = v(V)$$

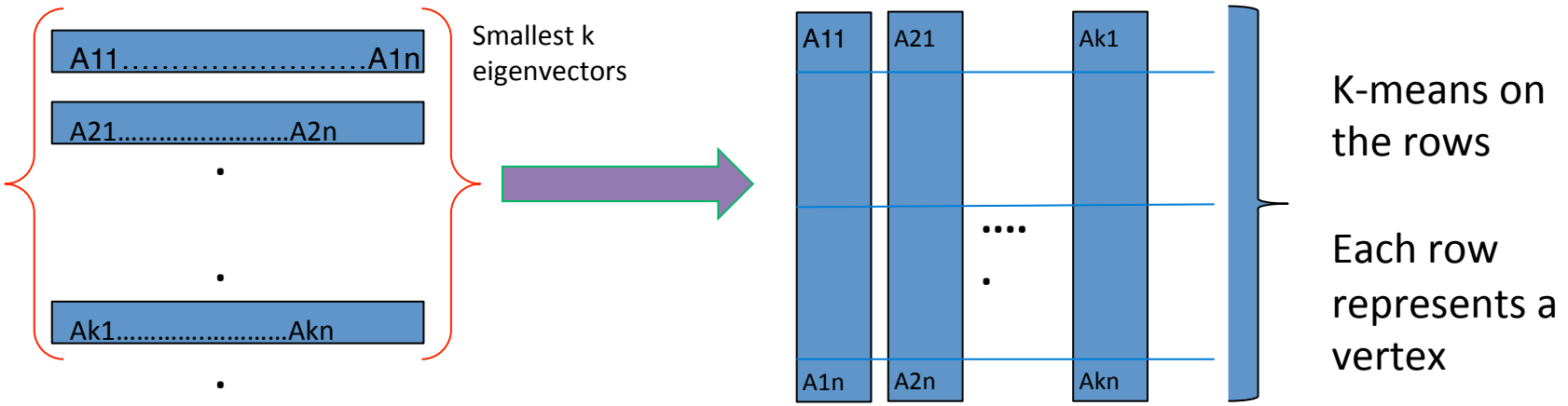
- Assume $h = D^{1/2} f$
  - $min_h h^T D^{-1/2} \mathrm{L} D^{-1/2} \mathrm{h}$ subject to

$$h^T D^{1/2} 1 = 0, \qquad h^T h = v(V)$$

  - The answer is in the eigenvector of the second smallest eigenvalue of $L_{sym} = D^{-1/2} \mathrm{L} D^{-1/2}$
    Shi and Malik (2000)
- L_{sym} is the normalized Laplacian
  - has n non-negative, real valued eigenvalues
  - $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$

Multi-Way Graph Partition

- The cluster assignment is given by the smallest k eigenvectors of L
- The real values need to be converted to cluster assignments
  - We use k-means to cluster the rows
  - We can substitute *L* with L_{sym}

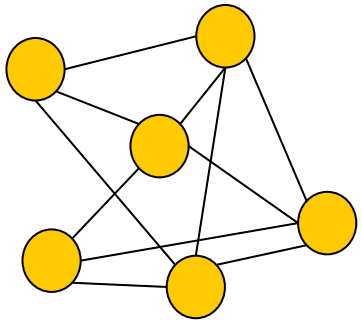


Graph Clustering Algorithms

- Spectral Clustering
- **Modularity Based Methods**

Main idea

- **Modularity** function **[Newman and Girvan '04], [Newman '06]**
- Initially introduced as a measure for assessing the strength of communities
  - **Q = (fraction of edges within communities) –**
    (expected number of edges within communities)
- What is the **expected** number of edges?
- Consider a configuration model
  - **Random graph** model with the same degree distribution
  - Let P_{ij} = probability of an edge between nodes **i** and **j** with degrees k_i and k_j respectively
  - Then $P_{ij} = k_i k_j / 2m$, where $m = |E| = \frac{1}{2} \sum_i k_i$



Formal definition of modularity

- **Modularity Q**

$$Q = \frac{1}{2m} \sum_{ij} \left(A_{ij} - \frac{k_i k_j}{2m} \right) \delta(C_i, C_j)$$

where

- **A** is the adjacency matrix
- k_i, k_j the degrees of nodes **i** and **j** respectively
- **m** is the number of edges
- C_i is the community of node **i**
- **δ(.)** is the Kronecker function: 1 if both nodes **i** and **j** belong on the same community ($C_i = C_j$), 0 otherwise

[Newman and Girvan '04], [Newman '06]

Properties of modularity

$$Q = \frac{1}{2m}\sum_{ij}\left(A_{ij} - \frac{k_i k_j}{2m}\right)\delta(c_i, c_j)$$

- **Larger** modularity **Q** indicates **better** communities (more than random intra-cluster density)
  - The community structure would be better if the number of internal edges exceed the expected number
- Modularity value is always **smaller than 1**
- It can also take **negative values**
  - E.g., if each node is a community itself
  - No partitions with positive modularity → No community structure
  - Partitions with large negative modularity → Existence of subgraphs with small internal number of edges and large number of inter-community edges

[Newman and Girvan '04], [Newman '06], [Fortunato '10]

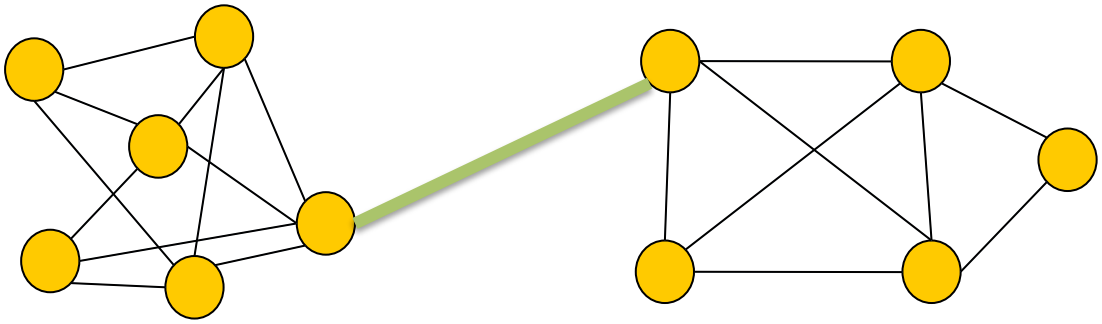
Applications of modularity

- Modularity can be applied:
  - As **quality function** in clustering algorithms
  - As **evaluation measure** for comparison of different partitions or algorithms
  - As a community detection tool itself
    - ❑ **Modularity optimization**
  - As criterion for reducing the size of a graph
    - ❑ Size reduction preserving modularity **[Arenas et al. '07]**

[Newman and Girvan '04], [Newman '06], [Fortunato '10]

Modularity-based community detection

- Modularity was first applied as a **stopping criterion** in the Newman-Girvan algorithm
- Newman-Girvan algorithm **[Newman and Girvan '04]**
  - A **divisive** algorithm (detect and remove edges that connect vertices of different communities)
  - **Idea:** try to identify the edges of the graph that are most between other vertices → responsible for connecting many node pairs
  - Select and remove edges based to the value of **betweenness centrality**
  - **Betweenness centrality:** number of **shortest paths** between every pair of nodes, that pass through an edge



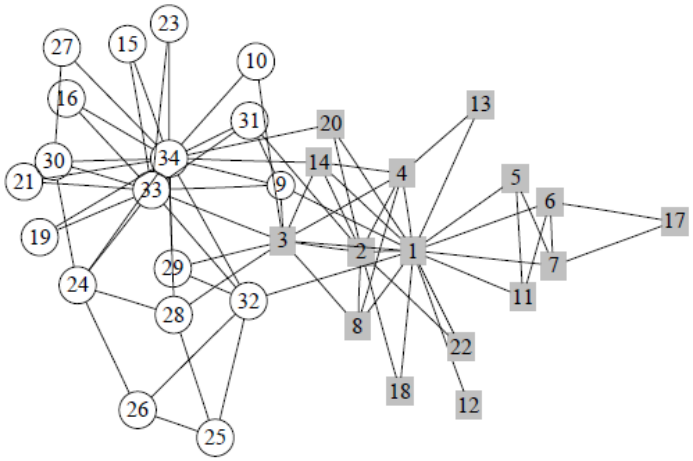
Edge betweenness is higher for edges that connect different communities

Newman-Girvan algorithm (1)

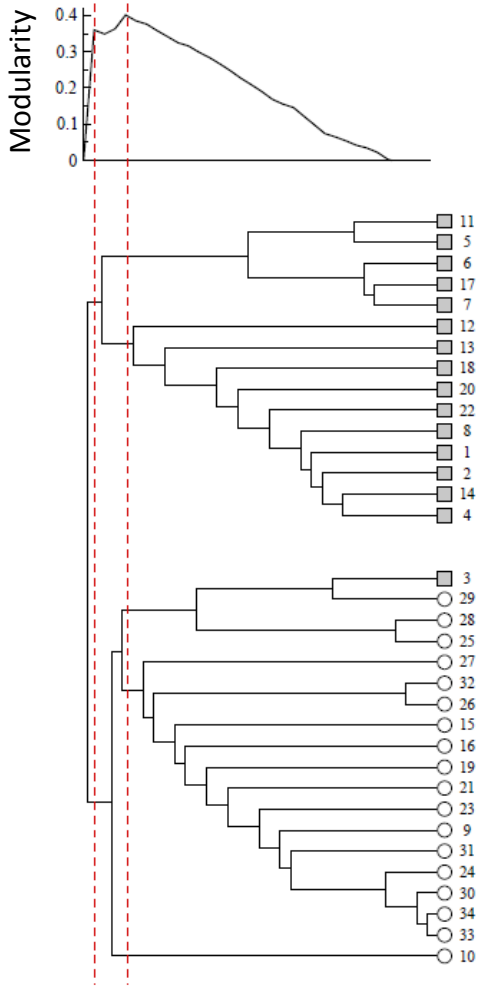
- **Basic steps:**
  1. Compute betweenness centrality for all edges in the graph
  2. Find and remove the edge with the highest score
  3. Recalculate betweenness centrality score for the remaining edges
  4. Go to step 2
- How do we know if the produced communities are **good ones** and stop the algorithm?
  - The output of the algorithm is in the form of a **dendrogram**
  - Use **modularity** as a criterion to cut the dendrogram and terminate the algorithm (Q ~= 0.3-0.7 indicates good partitions)
- Complexity: **$O(m^2n)$** (or **$O(n^3)$** on a sparse graph)

[Newman and Girvan '04], [Girvan and Newman '02]

Newman-Girvan algorithm (2)



Zachary's karate club



Community structure

[Newman and Girvan '04]

Modularity optimization

- High values of modularity indicate good quality of partitions
- **Goal:** find the partition that corresponds to the maximum value of modularity
- **Modularity maximization** problem
  - Computational difficult problem **[Brandes et al. '06]**
  - Appoximation techniques and heuristics
- Four main categories of techniques
  1. Greedy techniques
  2. **Spectral optimization**
  3. Simulated annealing
  4. Extremal optimization

[Fortunato '10]

Spectral optimization (1)

- **Idea:** Spectral techniques for modularity optimization
- **Goal:** Assign the nodes into two communities, **X** and **Y**
- Let $s_i, \forall i \in V$ an indicator variable where $s_i = +1$ if **i** is assigned to **X** and $s_i = -1$ if **i** is assigned to **Y**

$$Q = \frac{1}{2m}\sum_{ij}\left(A_{ij} - \frac{k_i k_j}{2m}\right)\delta(C_i, C_j)$$

$$= \frac{1}{4m}\sum_{ij}\left(A_{ij} - \frac{k_i k_j}{2m}\right)(s_i s_j + 1)$$

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

- **B** is the **modularity matrix**

$$B_{ij} = A_{ij} - \frac{k_i k_j}{2m}$$

[Newman '06], [Newman '06b]

Spectral optimization (2)

- Modularity matrix **B** $$B_{ij} = A_{ij} - \frac{k_i k_j}{2m}$$

- Vector **s** can be written as a linear combination of the eigenvectors $\mathbf{u_i}$ of the modularity matrix **B** $a_i = u_i^T s$

$$\text{where} \quad s = \sum_i a_i \, u_i$$

- Modularity can now expressed as

$$Q = \frac{1}{4m} \sum_i a_i u_i^T \; B \; \sum_j a_j u_j^T = \frac{1}{4m} \sum_{i=1}^{n} \left(u_i^T s \right)^2 \beta_i$$

Where $\boldsymbol{\beta_i}$ is the eigenvalue of **B** corresponding to eigenvector $\mathbf{u_i}$

[Newman '06], [Newman '06b]

Spectral optimization (3)

- Spectral modularity optimization algorithm
  1. Consider the eigenvector **u_1** of **B** corresponding to the largest eigenvalue
  2. Assign the nodes of the graph in one of the two communities **X** (**$s_i = +1$**) and **Y** (**$s_i = -1$**) based on the **signs** of the corresponding components of the eigenvector

$$s_i = \begin{cases} 1 & if\ u_1(i) \geq 0 \\ -1 & if\ u_1(i) < 0 \end{cases}$$

  – More than two partitions?

  1. **Iteratively**, divide the produced partitions into two parts
  2. If at any step the split does not contribute to the modularity, leave the corresponding subgraph as is
  3. End when the entire graph has been splinted into no further divisible subgraphs

  – Complexity: **$O(n^2 \log n)$** for sparse graphs

[Newman '06], [Newman '06b]

References – Graph clustering

- UIrike von Luxburg, A Tutorial on Spectral Clustering, Statistics and Computing, 2007
- Davis, C., W. M. Kahan (March 1970). The rotation of eigenvectors by a perturbation. III. SIAM J. Numerical Analysis 7
- Shi, Jianbo, and Jitendra Malik. "Normalized cuts and image segmentation, "*Pattern Analysis and Machine Intelligence, IEEE Transactions on* (2000).
- Mechthild Stoer and Frank Wagner. 1997. A simple min-cut algorithm. *J. ACM*
- Ng, Jordan & Weiss, K-means algorithm on the embedded eigen-space, NIPS 2001
- Hagen, L. Kahng, , "New spectral methods for ratio cut partitioning and clustering," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on* , 1992

Extensions of modularity

- Modularity has been extended in several directions
  - Weighted graphs **[Newman '04]**
  - Bipartite graphs **[Guimera et al '07]**
  - Directed graphs (next in this tutorial) **[Arenas et al. '07], [Leicht and Newman '08]**
  - Overlapping community detection (next in this tutorial) **[Nicosia et al. '09]**
  - Modifications in the configuration model – local definition of modularity **[Muff et al. '05]**

References (modularity)

- M.E.J. Newman and M. Girvan. Finding and evaluating community structure in networks. Physical Review E 69(02), 2004.
- M.E.J. Newman. Modularity and community structure in networks. PNAS, 103(23), 2006.
- S.E. Schaeffer. Graph clustering. Computer Science Review 1(1), 2007.
- S. Fortunato. Community detection in graphs. Physics Reports 486 (3-5), 2010.
- M. Coscia, F. Giannotti, and D. Pedreschi. A classification for community discovery methods in complex networks. Statistical Analysis and Data Mining 4 (5), 2011.
- A. Arenas, J. Duch, A. Fernandez, and S. Gomez. Size reduction of complex networks preserving modularity. New J. Phys., 9(176), 2007.
- M. Girvan and M.E.J. Newman. Community structure in social and biological networks. PNAS 99(12), 2002.
- U. Brandes, D. Delling, M. Gaertler, R. Gorke, M. Hoefer, Z. Nikoloski, and D. Wagner. On Modularity Clustering. IEEE TKDE 20(2), 2008.
- M.E.J. Newman. Fast algorithm for detecting community structure in networks. Phys. Rev. E 69, 2004.
- A. Clauset, M.E.J. Newman, and C. Moore. Finding community structure in very large networks. Phys. Rev. E 70, 2004.

References (modularity)

- M.E.J. Newman. Finding community structure in networks using the eigenvectors of matrices. Phys. Rev. E 74, 2006.
- R. Guimera, M. Sales-Pardo, L.A.N. Amaral. Modularity from Fluctuations in Random Graphs and Complex Networks. Phys. Rev. E 70, 2004.
- J. Duch and A. Arenas. Community detection in complex networks using Extremal Optimization. Phys. Rev. E 72, 2005.
- A. Arenas, J. Duch, A. Fernandez, and S. Gomez. Size reduction of complex networks preserving modularity. New Journal of Physics 9(6), 2007.
- E.A. Leicht and M.E.J. Newman. Community structure in directed networks. Phys. Rev. Lett. 100, 2008.
- V. Nicosia, G. Mangioni, V. Carchiolo, and M. Malgeri. Extending the definition of modularity to directed graphs with overlapping communities. J. Stat. Mech. 03, 2009.
- S. Muff, F. Rao, A. Caflisch. Local modularity measure for network clusterizations. Phys. Rev. E, 72, 2005.
- S. Fortunato and M. Barthelemy. Resolution limit in community detection. PNAS 104(1), 2007.

References graph kernels

- P. Mahe, N. Ueda, T. Akutsu, J.-L. Perret, and J.-P. Vert, Extensions of Marginalized Graph Kernels, in Proceedings of the Twenty-first International Conference on Machine Learning, New York, NY, USA, 2004, p. 70-.
- N. Shervashidze, Scalable graph kernels, Dissertation, Universitait Tuebingen, 2012.
- J. Ramon and T. Gartner, Expressivity versus efficiency of graph kernels, in Proceedings of the First International Workshop on Mining Graphs, Trees and Sequences, 2003, pp. 65-74.
- S. V. N. Vishwanathan, N. N. Schraudolph, R. Kondor, and K. M. Borgwardt, Graph Kernels, Journal of Machine Learning Research, vol. 11, p. 1201-1242, Apr. 2010.
- https://www.cs.ucsb.edu/~xyan/tutorial/KDD08_graph_partII.pdf
- https://www.cs.ucsb.edu/~xyan/tutorial/KDD08_graph_partI.pdf

References (community evaluation measures)

- M.E.J. Newman. The structure and function of complex networks. SIAM REVIEW 45, 2003.
- M.E.J. Newman and M. Girvan. Finding and evaluating community structure in networks. Physical Review E 69(02), 2004.
- S.E. Schaeffer. Graph clustering. Computer Science Review 1(1), 2007.
- S. Fortunato. Community detection in graphs. Physics Reports 486 (3-5), 2010.
- L. Danon, J. Duch, A. Arenas, and A. Diaz-guilera. Comparing community structure identification. Journal of Statistical Mechanics: Theory and Experiment 9008 , 2005.
- M. Coscia, F. Giannotti, and D. Pedreschi. A classification for community discovery methods in complex networks. Statistical Analysis and Data Mining 4 (5), 2011.
- J. Leskovec, K.J. Lang, and M.W. Mahoney. Empirical comparison of algorithms for network community detection. In: WWW, 2010.
- F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, and D. Parisi. Defining and identifying communities in networks. PNAS, 101(9), 2004.
- J. Yang and J. Leskovec. Defining and Evaluating Network Communities based on Ground-Truth. In: ICDM, 2012.
- Fan Chung. Spectral Graph Theory. CBMS Lecture Notes 92, AMS Publications, 1997.