Foundations of Data Science

Lecture 9: Proximity

MING GAO

DaSE@ECNU
(for course related communications)
mgao@sei.ecnu.edu.cn

Nov. 4, 2016

Outline

1. Community Structures
2. Node Roles
3. Node Proximity
   - Simple Approaches
   - Graph-theoretic Approaches
   - SimRank
   - Random Walk based Approaches
4. Network Proximity
   - Known Nodes
   - Unknown Nodes

Community structures

Definition

Community structure indicates that the network divides naturally into groups of nodes with dense connections internally and sparser connections between groups.

- Global community structures: clustering-based approach, spectral clustering, modularity-based approach, etc.
- Local community structures: node-centric community, group-centric community
  - Traditional network: clique, quasi-clique, k-clique, k-core, etc.
  - Bipartite network: bi-clique, quasi-bi-clique, etc.
  - Signed network: antagonistic community, quasi-antagonistic community, etc.

Global community structures

Goal

Partition nodes of a network into disjoint sets.

- Clustering based on vertex similarity
- Latent space models
- Spectral clustering
- Modularity maximization

Spectral clustering

Let $\mathcal{L}$ be normalized Laplacian of graph G, the algorithm partitions nodes into two sets (B_1, B_2) based on the eigenvector $\mathbf{v}$ corresponding to the second-smallest eigenvalue of $\mathcal{L}$. Partitioning may be done in various ways:

- Assign all nodes whose component in $\mathbf{v}$ satisfies certain condition in B_1, and B_2 otherwise, e.g., larger than median.
- The algorithm can be used for hierarchical clustering by repeatedly partitioning the subsets in this fashion.

Modularity

Idea

Graph has community structure, if it is different from random graph (not expected to have community structure for random graph).

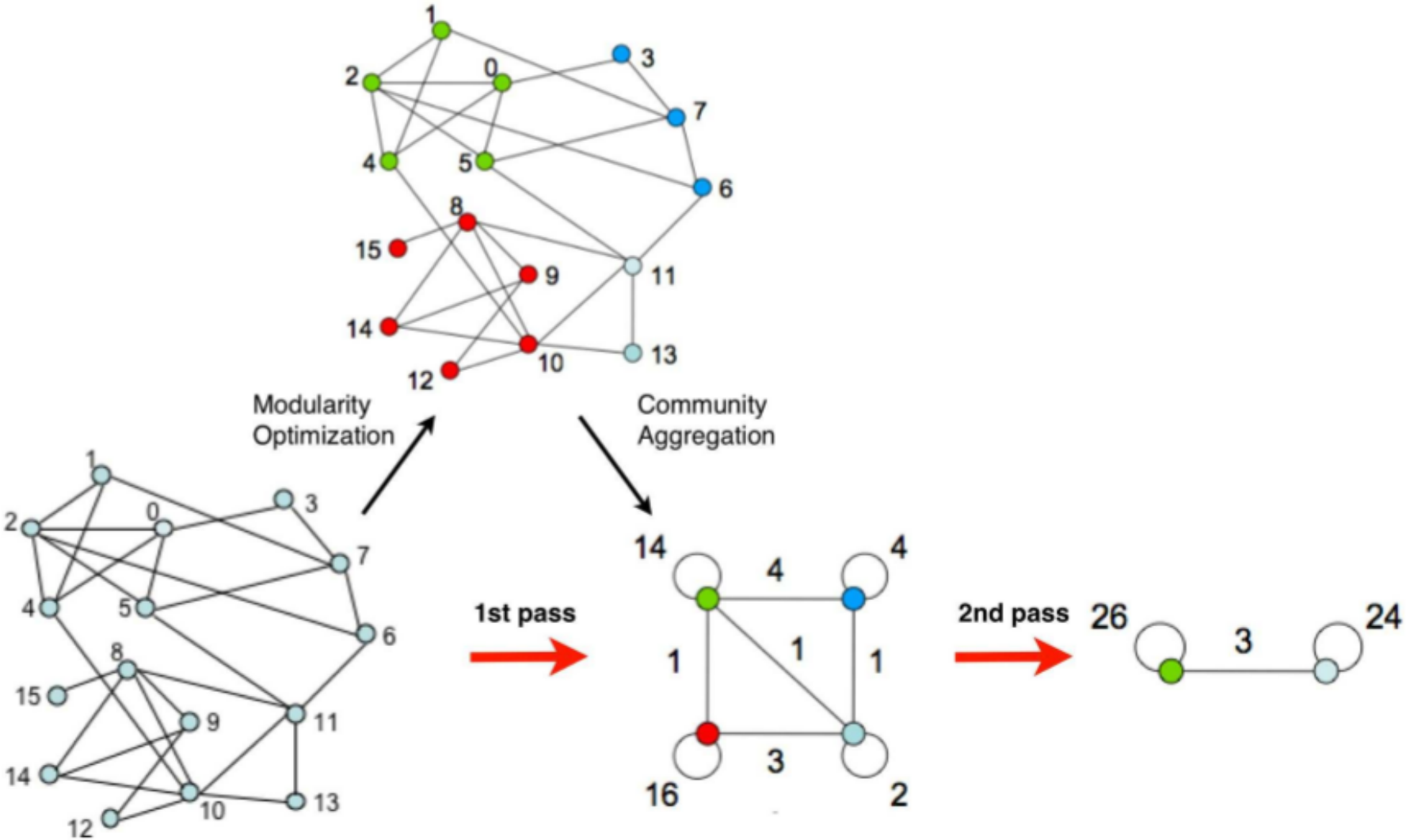
- Modularity [Newman 2006]:

$$M = \frac{1}{2m}\sum_{i,j}(A_{ij} - \frac{d(i)d(j)}{2m})\delta(C_i, C_j).$$

  where m and C_i denote # edges and the $i-$th community in the graph.

- Compares the number of edges within a community with the expected such number in a corresponding random graph.
- It can be used as a measure to evaluate the communities quality.

Louvain method

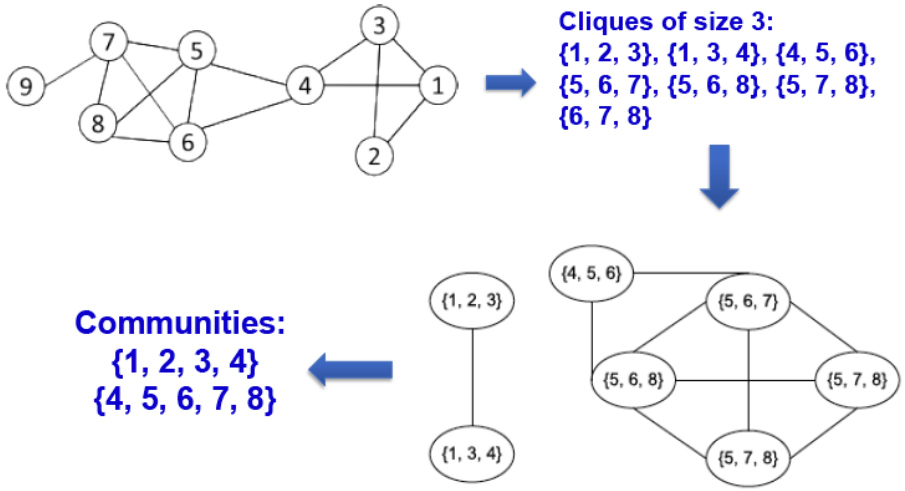


Clique

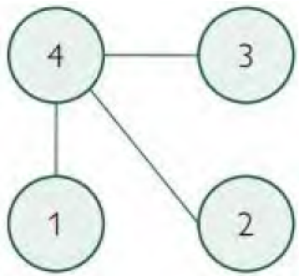
Definition

A clique is a subset of vertices of an undirected graph such that its induced subgraph is complete. A **maximal** clique is a clique that cannot be extended by including one more adjacent vertex.

- Normally use cliques as a core or a seed to find larger communities.
  - Find out all cliques of size k in a given network (NP-complete)
  - Construct a clique graph. Two cliques are adjacent if they share $k-1$ nodes
  - Each connected component in the clique graph forms a community



Extensions of clique



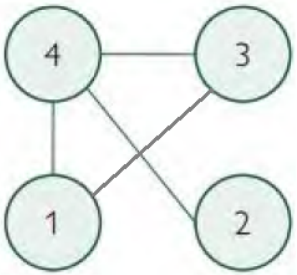
$k-$clique

Maximal subgroup, where the largest geodesic distance between any pair of nodes is not greater than k.

- It is a clique if $k = 1$.

Quasi-clique

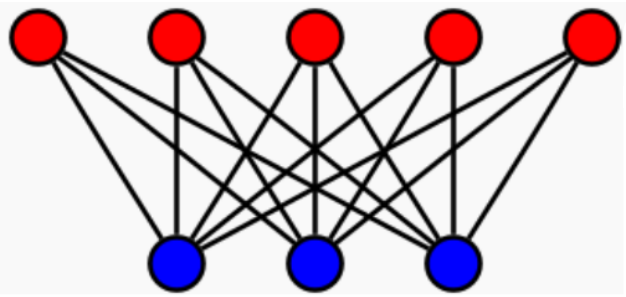
Generalize clique to dense subgraph with different definitions (degree, density).

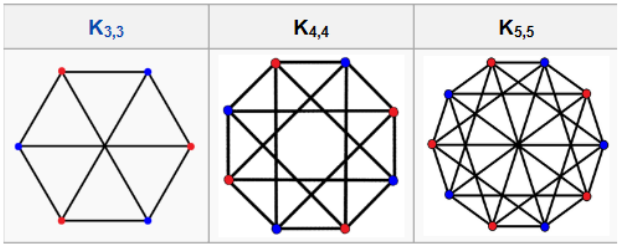


- Node degree: every node in induced subgraph is adjacent to at least $\gamma(n-1)$ other nodes in the subgraph.
- Edge density: Number of edges in subgraph is at least $\gamma n(n-1)/2$. $(\gamma = \frac{2}{3})$

where n denotes # nodes in subgraph.

Local community structure in bipartite graph

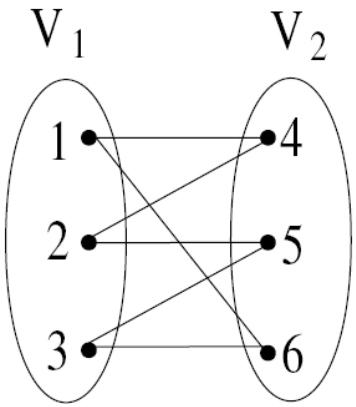




Biclique

A biclique is a special kind of bipartite graph where every vertex of the first set is connected to every vertex of the second set.

- A complete bipartite graph with partitions of size $|V_1| = m$ and $|V_2| = n$, is denoted $K_{m,n}$.



Quasi-bi-clique

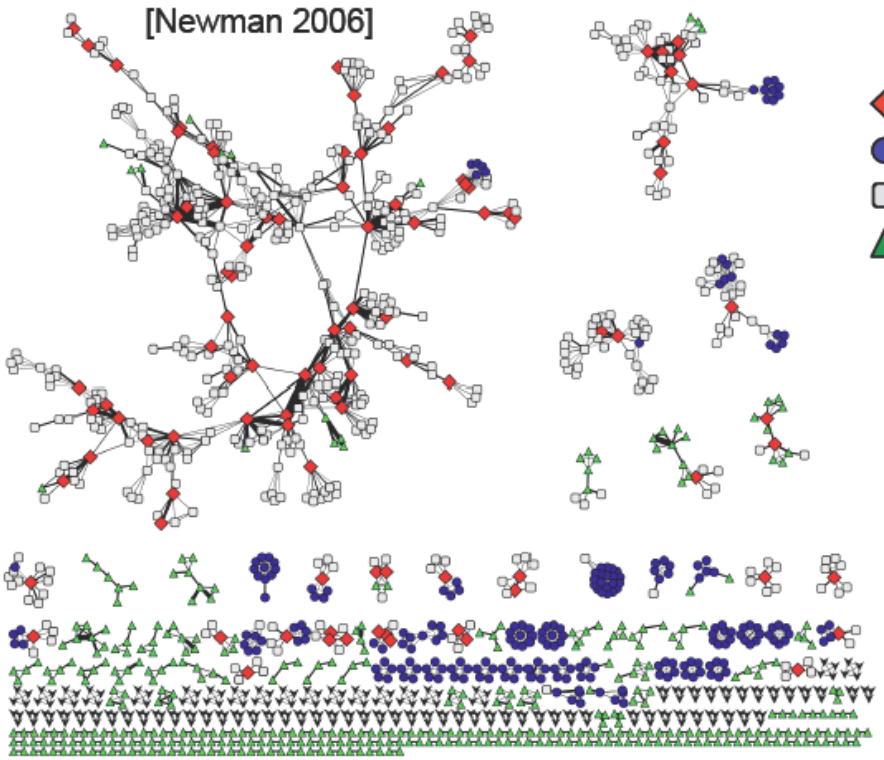
The definition of biclique is too strict. A quasi-bi-clique is a dense subgraph to relax the constraint for vertices.

- Relative version of quasi-bi-clique.
- Absolute version of quasi-bi-clique.

Node roles

What are roles?

"Functions" of nodes in the network: similar to functional roles of species in ecosystems.



- Measured by structural behaviors
- Examples
  - Centers of stars
  - Members of cliques
  - Peripheral nodes
  - ...

Motivations

Why are the roles important?

- **Role query**: identify individuals with similar behavior to a known target.
- **Role outliers**: identify individuals with unusual behavior.
- **Role dynamics**: identify unusual changes in behavior.
- **Identity resolution**: identify known individuals in a new network.
- **Role transfer**: use knowledge of one network to make predictions in another.
- **Network comparison**: determine network compatibility for knowledge transfer.

Roles and communities

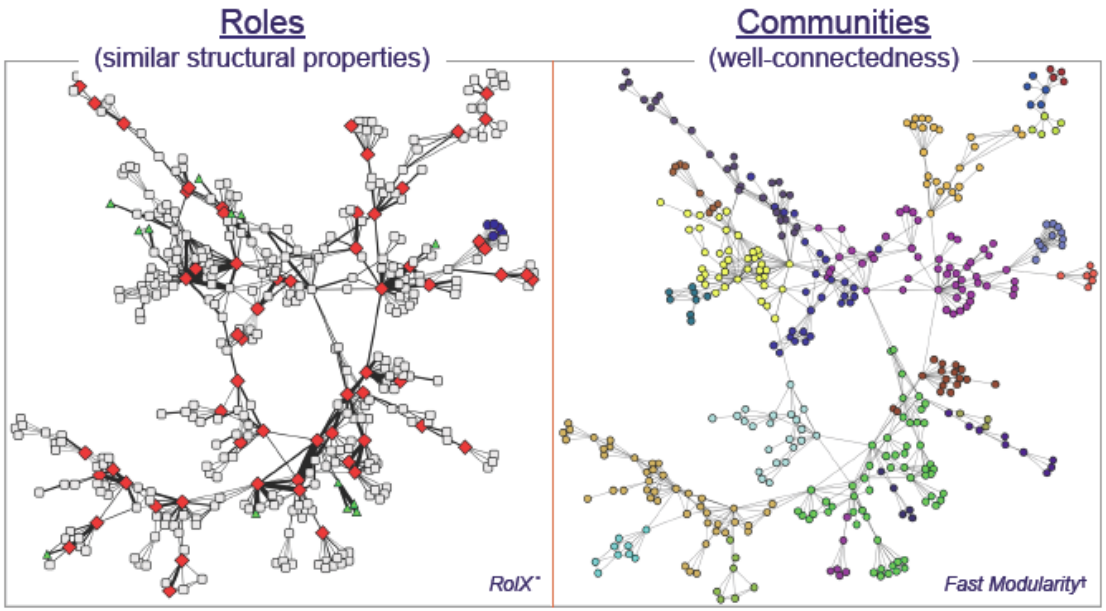
Consider the social network of a CS department

Roles

- Faculty
- Staff
- Students
- ...

Communities

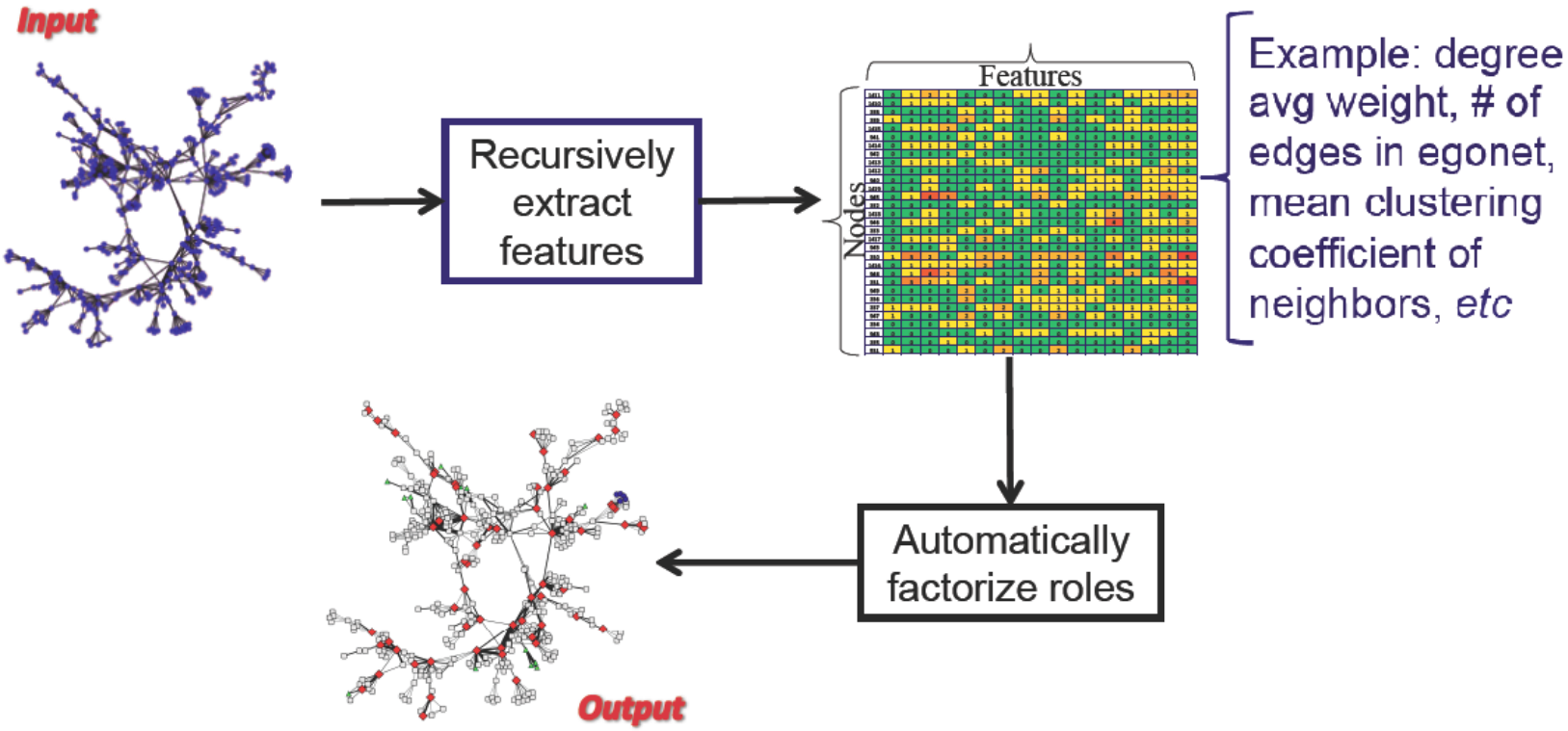
- AI lab
- Database lab
- Architecture lab
- ...



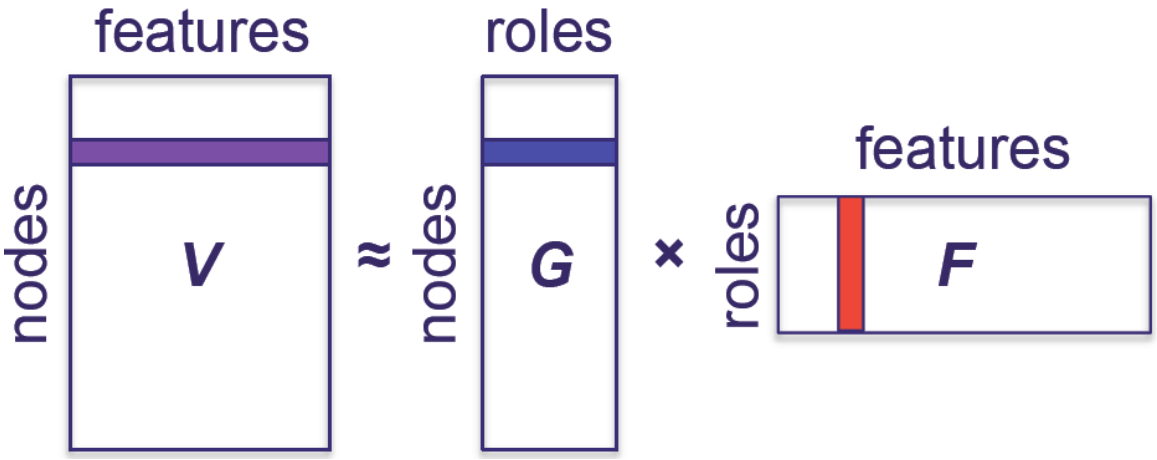
RolX: Role eXtraction [KDD 2012]

RolX

- Automatically extracts the underlying roles in a network without prior knowledge.
- Determines the number of roles automatically.
- Scales linearly on the number of edges.



Role extraction: feature grouping



ReFeX

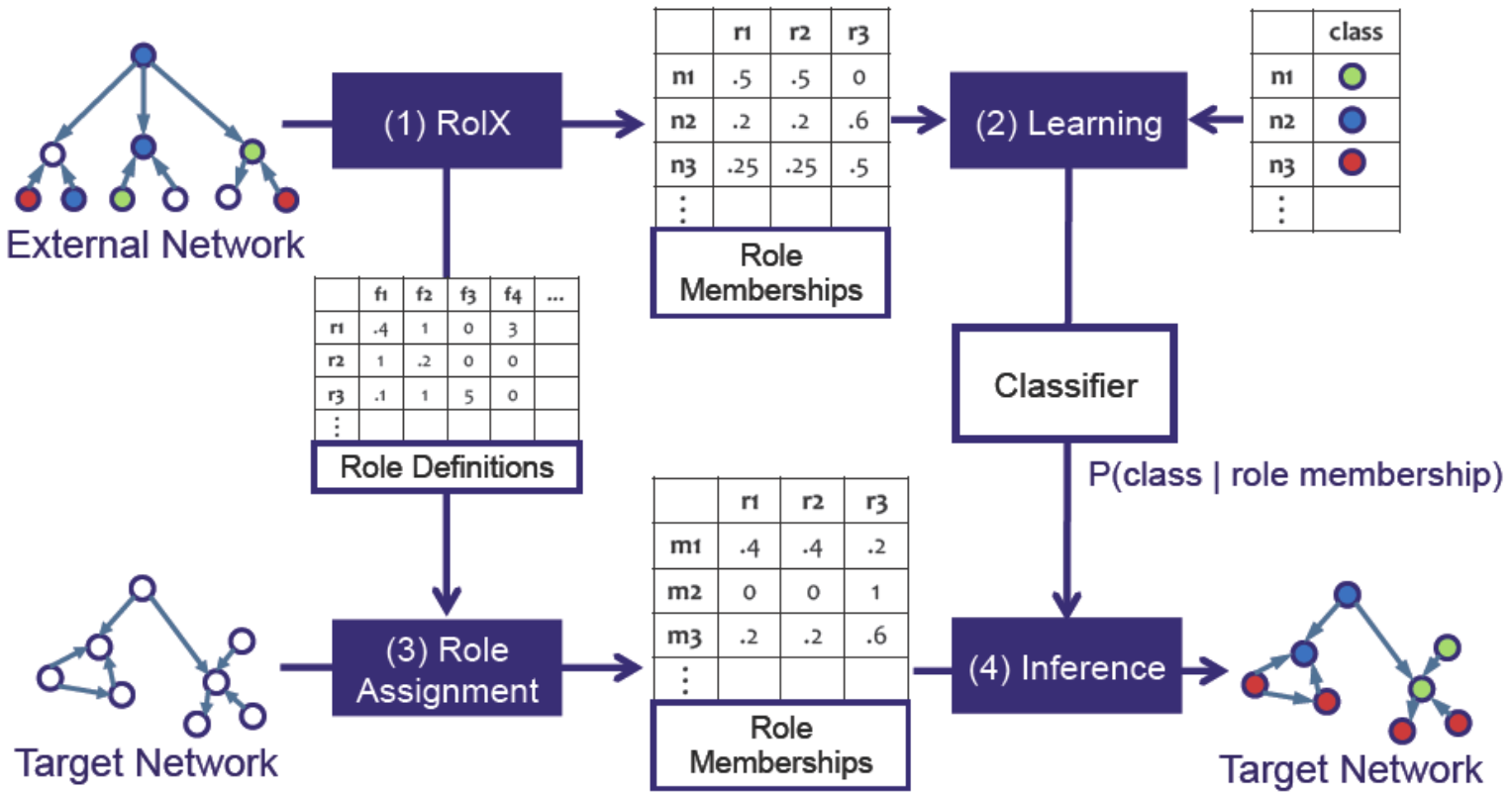
- Generate a rank r approximation of $V \approx GF$
- $$\text{Minimize } J = \frac{1}{2}\|V - GF\|^2, s.t. G \geq 0 \text{ and } F \geq 0$$
  - Computationally efficient
  - Non-negative factors simplify interpretation of roles and memberships

Experiment: role transfer

Question

How can we use labels from an external source to predict labels on a network with no labels?

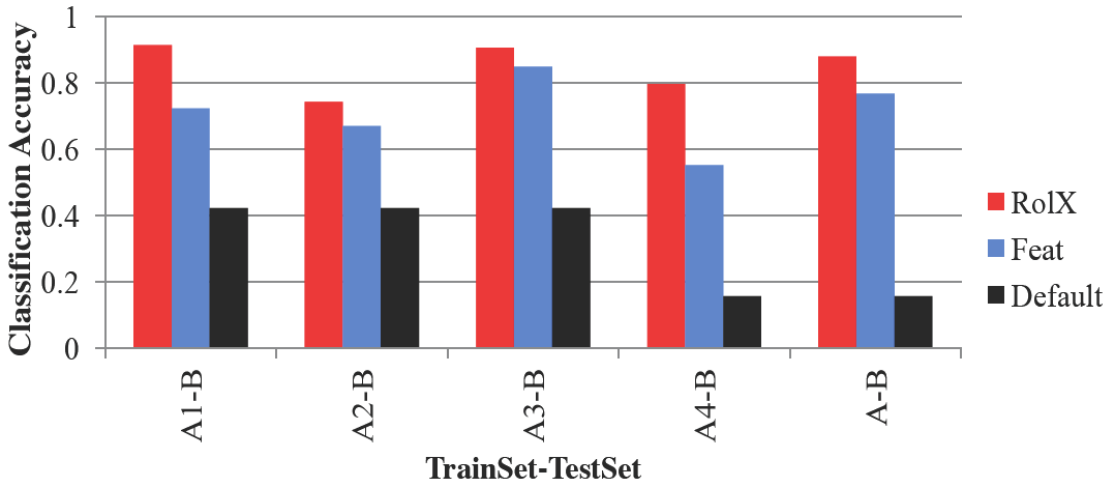
- Conjecture: nodes with similar roles are likely to have similar labels.

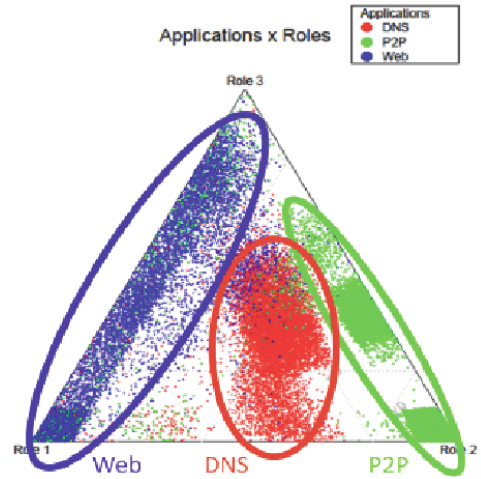


Experimental result

Observations

- Roles generalize across disjoint networks and enable prediction without re-learning.
- IP trace classes are well-separated in the RolX role space with as few as 3 roles.





Node proximity

Motivations

Node proximity (= similarity or closeness, but $\neq$ distance) measures:

- Information exchange
- Latency/speed of information exchange
- Likelihood of future link
- Propagation of a product/idea/service/ disease
- Relevance: ranking

Approaches

- Simple approaches
- Graph-theoretic approaches
- SimRank
- Random walk with restarts

Outline

Simple approaches

Similarity metrics

Given a graph $G = (V, E)$, $N(v_i)$ denotes the neighbors of node v_i.

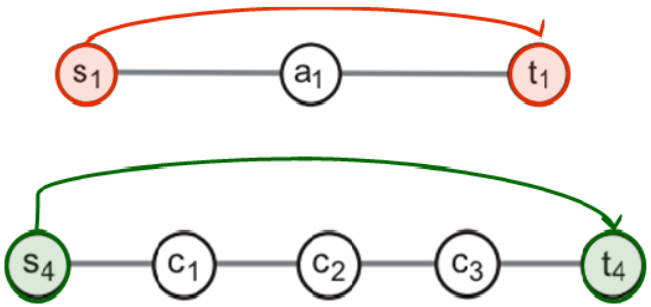
- Common neighbors: $|N(v_i) \cap N(v_j)|$.
- Jaccard coefficient: $\frac{|N(v_i) \cap N(v_j)|}{|N(v_i) \cup N(v_j)|}$.
- Adamic/Adar: $\sum_{v \in N(v_i) \cap N(v_j)} \frac{1}{\log |N(v)|}$.
- Preferential attachment: $|N(v_i)| \times |N(v_j)|$.

Drawbacks

- Jaccard coefficient treats a graph as a set of transactions which are independent.
- Thus, it loss the topological information of a graph.

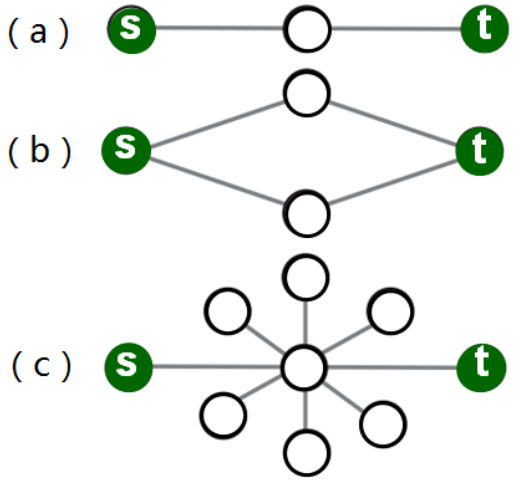
Outline

Graph-theoretic approaches



Idea

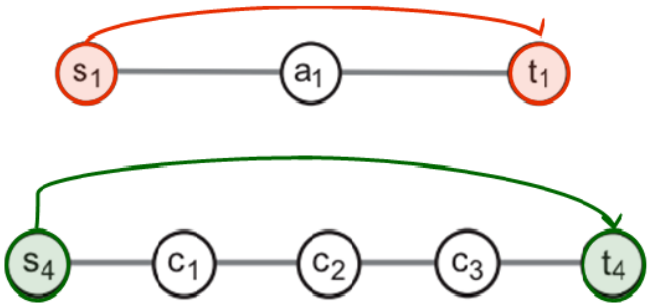
- (s_1, t_1) is more similar than (s_4, t_4).
- Simple metrics
  - Number of hops
  - Sum of weights of hops



Drawbacks

- (s, t) in (b) more similar than in (a) and (c) because of linked via more paths.
- In (c), s and t are probably unrelated since (s, t) linked via high degree node.

Graph-theoretic approaches cont.



Max-flow approach

- Heavy weighted links and number of paths matter, but path length doesnt.
- The max flow of (s_1, t_1) is the same to that of (s_4, t_4).

Katz

$$Ka = \sum_{l \geq 1} \beta^l |\text{paths}_{v_i, v_j}^{<l>}|,$$

where $\beta \in (0, 1)$ is a pre-defined parameter.

- $\text{paths}_{v_i, v_j}^{<l>}$ is the number of exact l-length paths from v_i to v_j.
- $\text{paths}_{v_i, v_j}^{<l>} = 1$ if and only if v_i and v_i are connected by a link.
- $K = \beta A + \beta^2 A^2 + \cdots = (I - \beta A)^{-1} - I$.

Outline

SimRank [G. Jeh and J. Widom KDD 2002]

Idea

Two objects are similar if they are connected to similar objects.

- Iterative computation with initial value $s^{(0)}(a,b) = \begin{cases} 1, & a = b; \\ 0, & a \neq b \end{cases}$

$$s^{(k+1)}(a,b) = \begin{cases} 1, & a = b; \\ = \frac{c}{|N(a)N(b)|} \sum_{c \in N(a)} \sum_{d \in N(b)|} s^{(k)}(c,d), & a \neq b \end{cases}$$

  where $c \in [0,1]$ is a constant.
- However, it needs $O(n^3)$ runtime. Many works improve the runtime.
  - Simrank++ [VLDB 2008]
  - $S = C(A^T S A) + I$ with Kronecker product [EDBT 2010]
  - Parallel SimRank [VLDB 2015]

Outline

Random walk

Hitting time

Hitting time H_{v_i,v_j} is the expected number of steps required for a random walk starting at v_i to reach v_j.

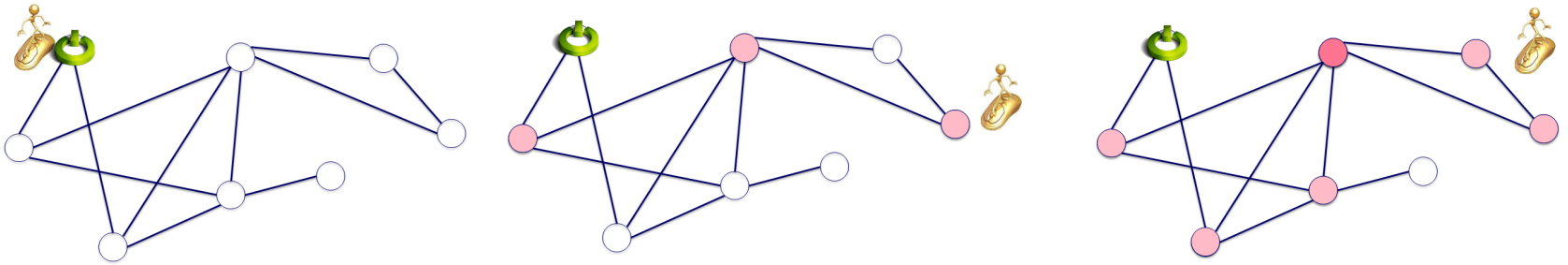
- Similar to Katz, all paths need to enumerate when hitting time is computed, but longer paths have smaller probabilities.
- $H = W + 2W^2 + \cdots = (I - W)^{-1}W(I - W)^{-1}$.

Commute time

Commute time $H_{v_i,v_j} + H_{v_j,v_i}$ is the expected number of steps required for a random walk starting at v_i to reach v_j, then return to v_i.

- For an undirected graph, $H_{v_i,v_j} = H_{v_j,v_i}$.
- Similar, we can compute commute time for every pair of nodes.

Personalized random walk with restarts



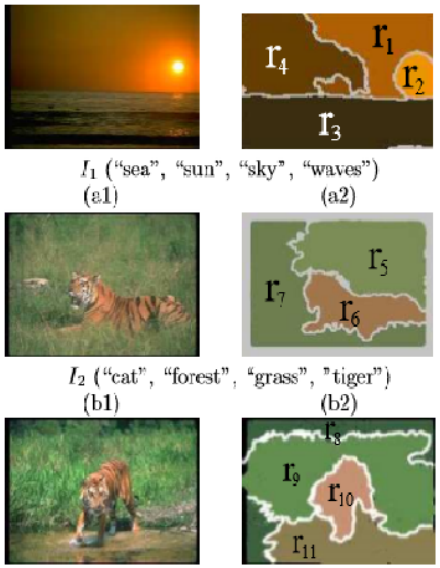
Idea

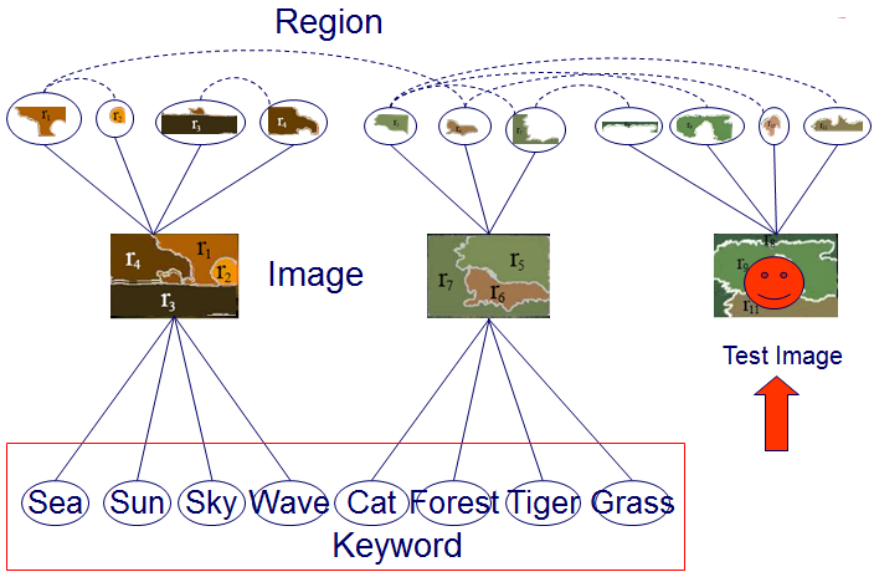
$$\mathbf{r}_i = cW\mathbf{r}_i + (1-c)\mathbf{e}_i,$$

where $\mathbf{r}_i \in R^{n\times 1}$ is ranking score w.r.t. v_i, W is the transition probability matrix, and $\mathbf{e}_i \in R^{n\times 1}$ is start score w.r.t. v_i with $r_{ii} = 1$, 0 otherwise.

- $(I - cD^{-1}A)\mathbf{r}_i = (1-c)\mathbf{e}_i$.
- $\mathbf{r}_i = (1-c)(I - cD^{-1}A)^{-1}\mathbf{e}_i$.

AutoCaptioning [KDD 2004]





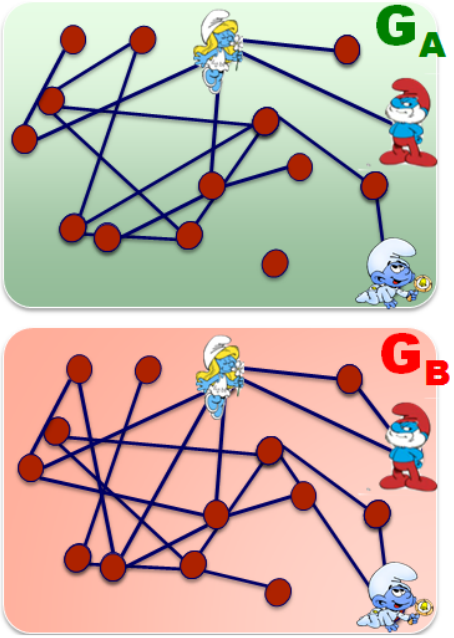
Model

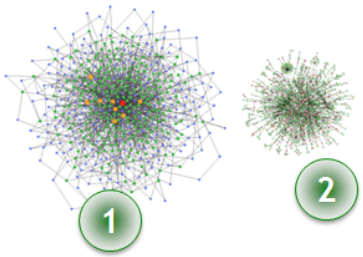
- Construct mixed media graph associated with a weighted matrix W.
- $\mathbf{r}_i = cW\mathbf{r}_i + (1-c)\mathbf{e}_i$

Even more applications

- Clustering: proximity as input.
- Classification
- Fraud detection

Network proximity





Known nodes

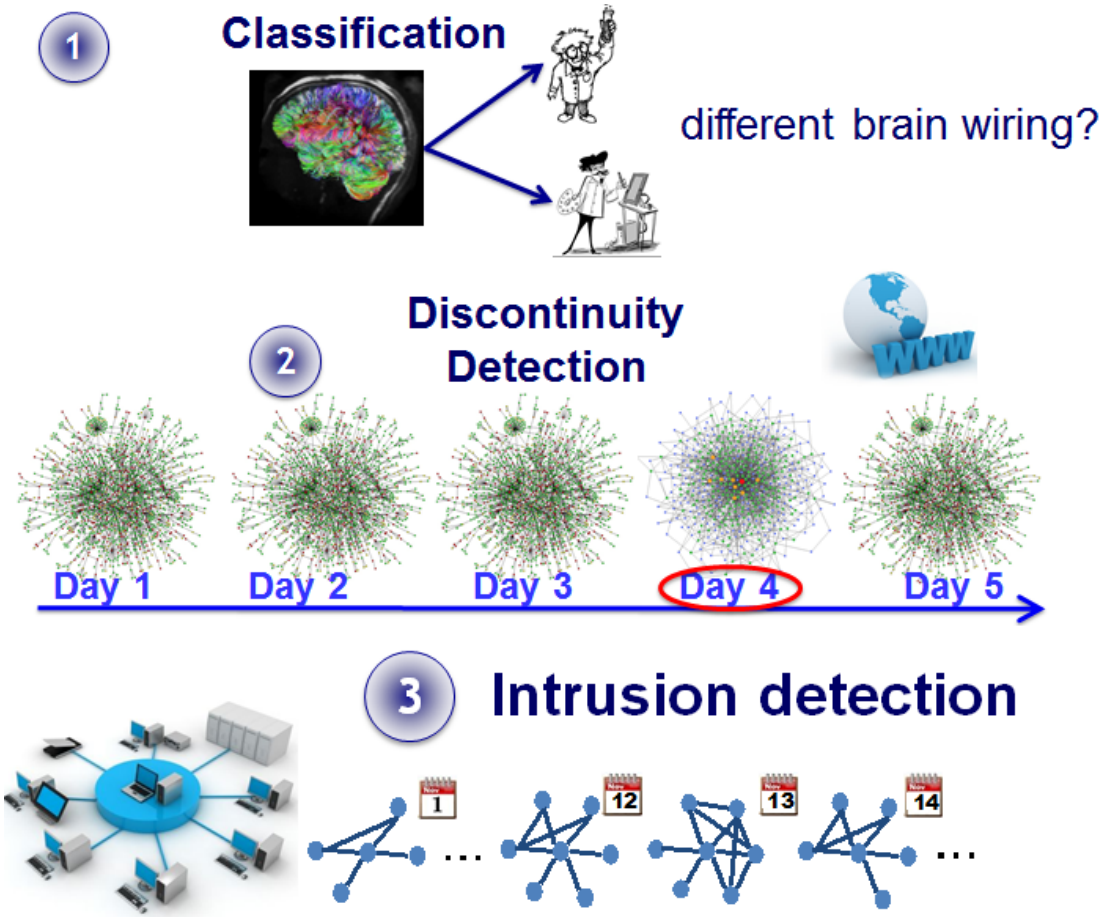
- Given:
  - 2 graphs with the same nodes and different edge sets.
  - node correspondence.
- Find: similarity score $s \in [0, 1]$.

Unknown nodes

- 2 anonymized networks with node IDs.
- Structural similarity score or node mapping.

Outline

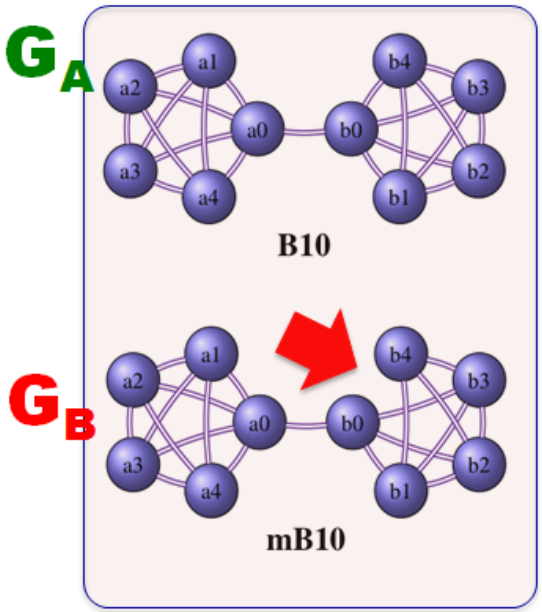
Applications

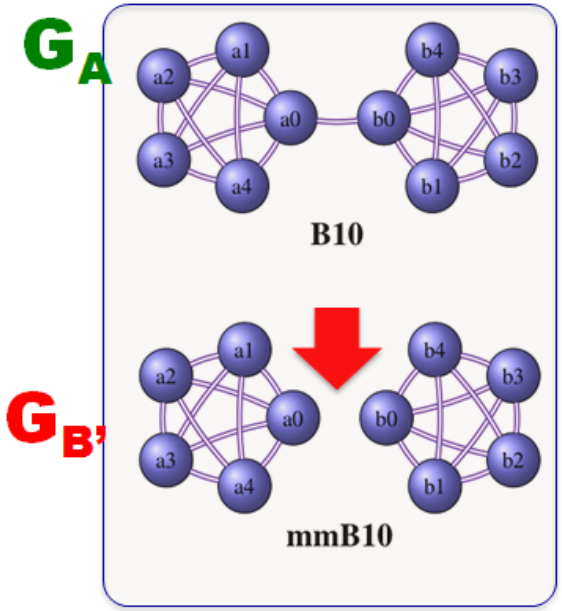


Applications

- Classification
- Discontinuity detection
- Intrusion detection
- ...

Simple approaches





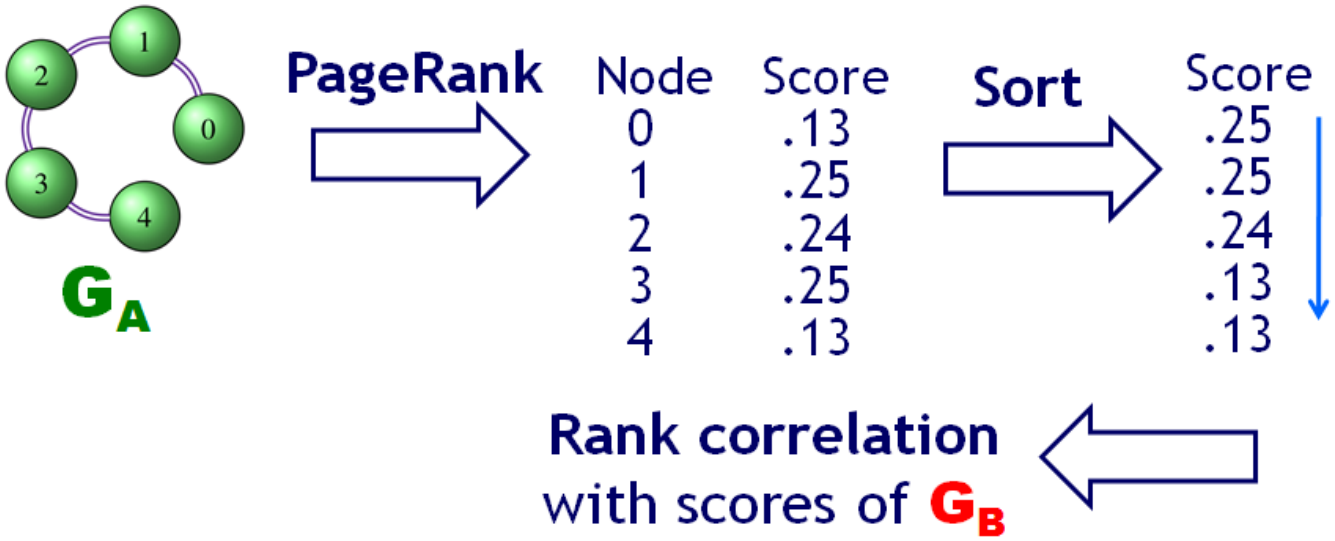
Edge overlap

- # of common edges. $E_o(G_A, G_B) = E_o(G_A, G_B')$.
- Other solutions?
  - Two graphs are similar if they share many vertices and/or edges
  - $E_o = \frac{|V_A \cap V_B| + |E_A \cap E_B|}{|V_A| + |V_B| + |E_A| + |E_B|}$.

Vertex ranking

Idea

- G_A and G_B are similar if the rankings of their vertices are similar.
- G_A and G_B are similar if their node/edge eigenvectors are close.



Edit distance

Idea

of operations to transform G_A to G_B.

- Insertion of nodes/edges.
- Deletion of nodes/edges.

Computation

$$ED(G_A, G_B) = |V_A| + |V_B| - 2|V_A \cap V_B| + |E_A| + |E_B| - 2|E_A \cap E_B|.$$

- How to assign cost per operation, e.g., weighted networks?
- For weighted graph,

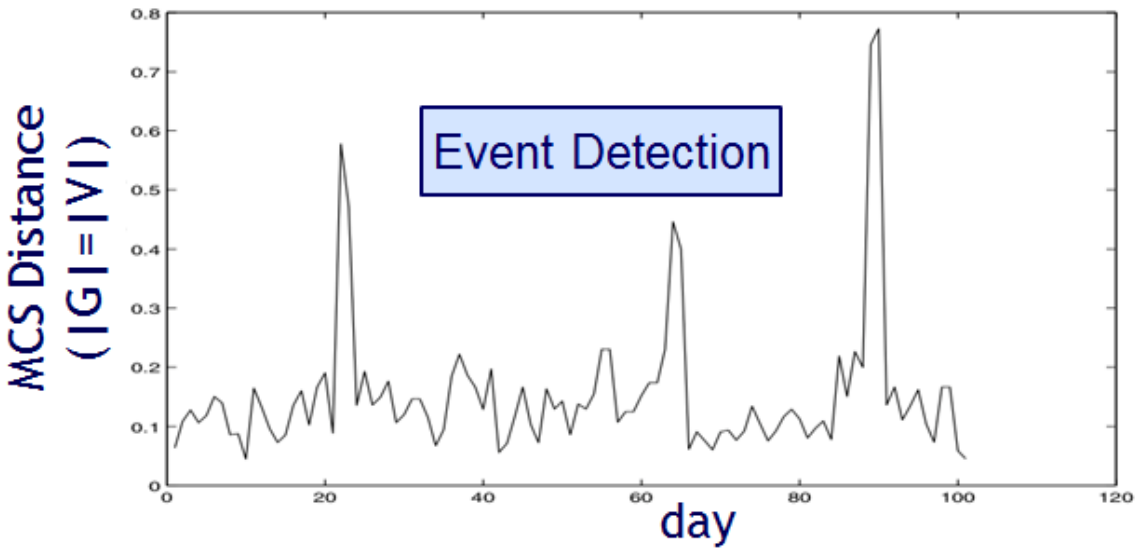
$$\begin{aligned} ED(G_A, G_B) &= c\left[|V_A| + |V_B| - 2|V_A \cap V_B|\right] \\ &+ \sum_{e \in E_A \setminus E_B} w_A(e) + \sum_{e \in E_B \setminus E_A} w_B(e) + \sum_{e \in E_A \cap E_B} |w_A(e) - w_B(e)|. \end{aligned}$$

Maximum common subgraph

Definition

$$d(G_A, G_B) = 1 - \frac{|mcs(G_A, G_B)|}{\max\{|G_A|, |G_B|\}}.$$

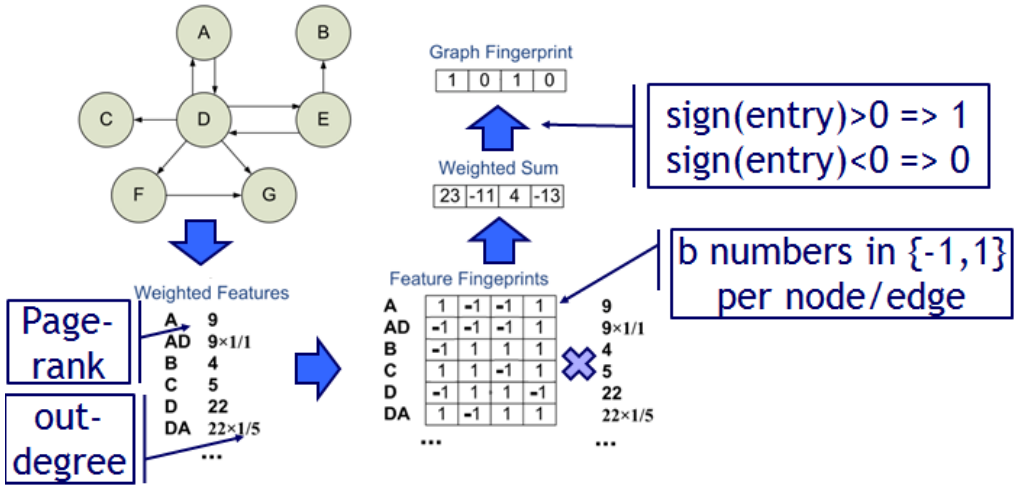
- MCS edge distance: $d(G_A, G_B) = 1 - \frac{|mcs(E_A, E_B)|}{\max\{|E_A|, |E_B|\}}$.
- MCS node distance: $d(G_A, G_B) = 1 - \frac{|mcs(V_A, V_B)|}{\max\{|V_A|, |V_B|\}}$.



Signature similarity

Approach

- Compute graph fingerprint (b bits).
- Hamming Distance between graph fingerprints.



Fingerprint of G_A: 1 0 1 0 1

Fingerprint of G_B: 0 0 1 0 1

Hamming Distance: 1

What properties should have?

Many similarity functions can be defined, what properties should a good similarity function have?

Axioms

- Identity property: $sim(G_A, G_A) = 1$.
- Symmetric property: $sim(G_A, G_B) = sim(G_B, G_A)$.
- Zero property: $sim(G_A, G_B) = 0$ if $E_A \cap E_B = \emptyset$.

Desired properties

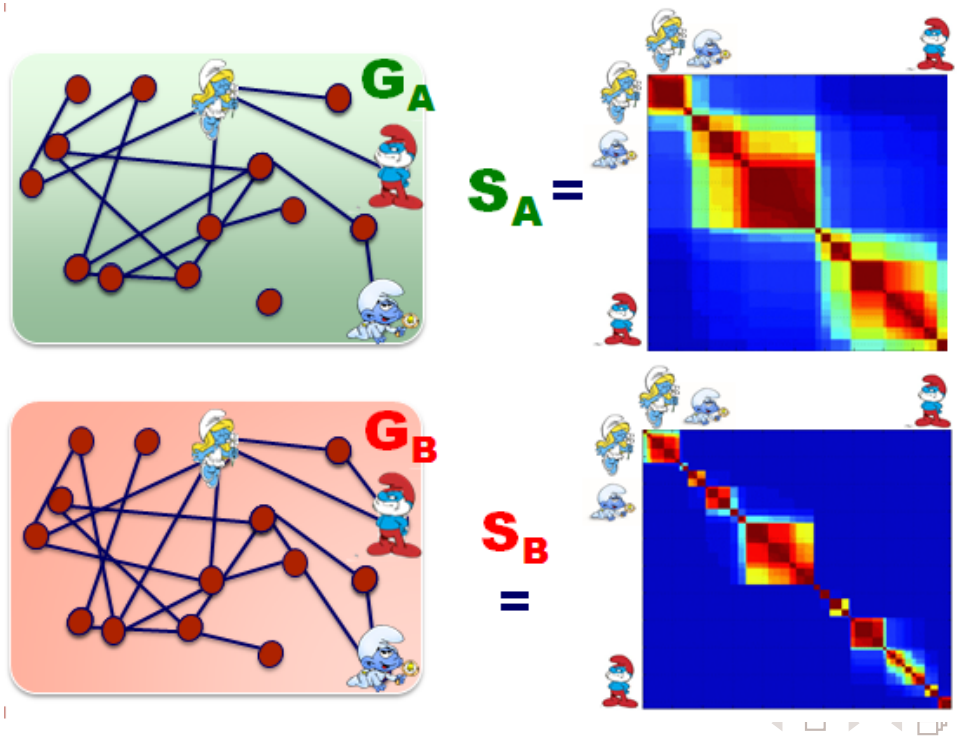
- Intuitiveness
  - Edge importance
  - Weight awareness
  - Edge-"submodularity"
  - Focus awareness
- Scalability.

Is there a method that satisfies the properties?

DeltaCon

Approach

- Find the pairwise node influence, S_A and S_B, where $S = (I + \epsilon^2 D - \epsilon A)^{-1} \approx (I - \epsilon A)^{-1} = \sum_{i \geq 0} \epsilon^i A^i$.
- Find the similarity between S_A and S_B, where $sim(S_A, S_B) = \frac{1}{\sqrt{\sum_{i,j}(\sqrt{s_{A,ij}} - \sqrt{s_{B,ij}})^2}}$.

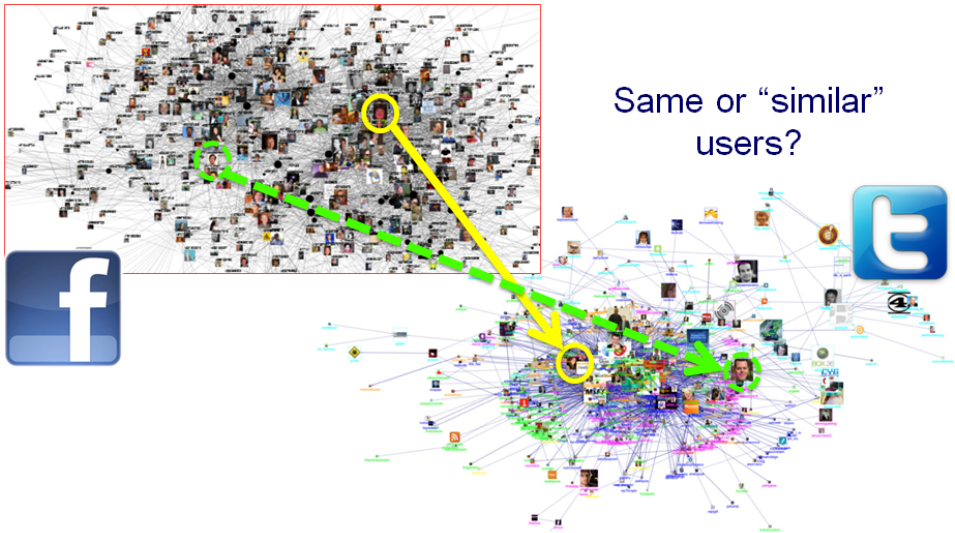


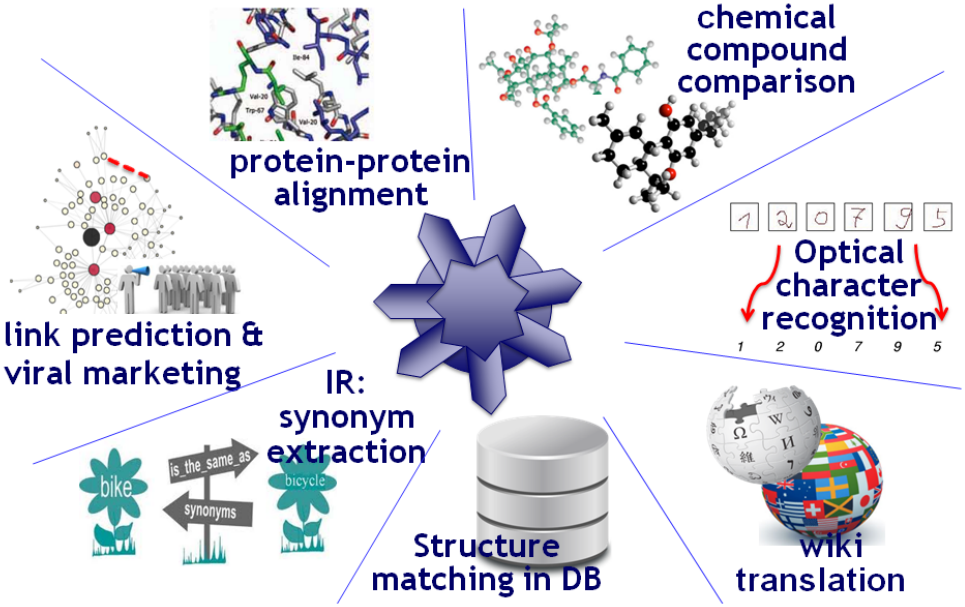
Summary

	edge	weight	returns	focus
Metric	**P1**	**P2**	**P3**	**P4**
Vertex/Edge Overlap	✗	✗	✗	?
Graph Edit Distance (XOR)	✗	✗	✗	?
Signature Similarity	✗	✓	✗	?
λ-distance (adjacency matrix)	✗	✓	✗	?
λ-distance (graph laplacian)	✗	✓	✗	?
λ-distance (normalized lapl.)	✗	✓	✗	?
DELTACON$_0$	✓	✓	✓	✓
DELTACON	✓	✓	✓	✓

Outline

Applications





Applications

- Identify users across social networks
- Protein-protein alignment
- Link prediction
- Entity resolution
- Optical character recognition: OCR
- ...

Spectral methods

$\lambda-$distance

Let $\lambda_i^{(B)}$ and $\lambda_i^{(C)}$ be the eigenvalues of graphs G_B and G_C, and $\lambda_i^{(\cdot)} \geq \lambda_{i+1}^{(\cdot)}$, the $\lambda-$distance is defined as

$$d(G_B, G_C) = \sqrt{\sum (\lambda_i^{(B)} - \lambda_i^{(C)})^2}.$$

- Adjacency matrix: A.
- (Combinatorial) Laplacian of G: $L = D - A$.
- Normailized Laplacian of G: $\mathcal{L} = D^{-1/2} L D^{-1/2}$.
- When the spectra are of different sizes, then the smaller may be padded with zero values (while maintaining the magnitude ordering), i.e., adding disjoint vertices to the smaller graph to make both graphs the same cardinality.

Isomorphism

Graph isomorphism

Find a mapping f of the vertices of G_A to the vertices of G_B such that G_A and G_B are identical; i.e. $(x, y) \in E_A$ if and only if $(f(x), f(y)) \in E_B$. Then f is an isomorphism, and G_A and G_B are called isomorphic.

- No polynomial-time algorithm is known for graph isomorphism.

Subgraph isomorphism

Subgraph isomorphism asks if there is a subset of edges and vertices of G_A that is isomorphic to a smaller graph G_B.

- Subgraph isomorphism is NP-complete.
- Runtime may grow exponentially with number of nodes.

Is there polynomial-time similarity measure for graphs?

What is Kernel? [1990s with SVM]

Kernel trick

Kernel $K(x_1, x_2)$ is a function that returns inner products between the images of data points in some space.

$$K(x_1, x_2) =< \phi(x_1), \phi(x_2) >,$$

where x_1 and x_2 are points in a low dimension space, and $\phi(x)$ maps x to a high dimension space, $< \cdot, \cdot >$ denotes the inner product of vectors.

- Choosing K is equivalent to choosing ϕ (the embedding map)to embed data in a vector space.
- K often interpreted as a similarity measure.
- If map chosen suitably, complex relations can be simplified, and easily detected by inner product.
- The inner product in high dimension can be substituted for the value of kernel function in low dimension (Kernel trick).

Existence of kernel

Mercer theorem

Function $K : \mathbb{R}^m \times \mathbb{R}^m \to R$ is a Mercer kernel if and only if all finite set $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\}$, the corresponding kernel matrix is proved to be symmetric positive semi-definite.

- Let K be the kernel matrix, where $K_{ij} = K(x_i, x_j)$.
- $K_{ij} = K(x_i, x_j) = \phi(x_1)^T \phi(x_2) = \phi(x_2)^T \phi(x_1) = K_{ji}$.
- For any $\mathbf{z} \neq \mathbf{0}$, we have

$$\begin{aligned}
\mathbf{z}^T K \mathbf{z} &= \sum_i \sum_j z_i K_{ij} z_j = \sum_i \sum_j z_i \phi(x_1)^T \phi(x_2) z_j \\
&= \sum_i \sum_j z_i \sum_k \phi_k(x_1)\phi_k(x_2) z_j = \sum_k \sum_i \sum_j z_i \phi_k(x_1)\phi_k(x_2) z_j \\
&= \sum_k \left(\sum_i z_i \phi_k(x_1) \right)^2 \geq 0.
\end{aligned}$$

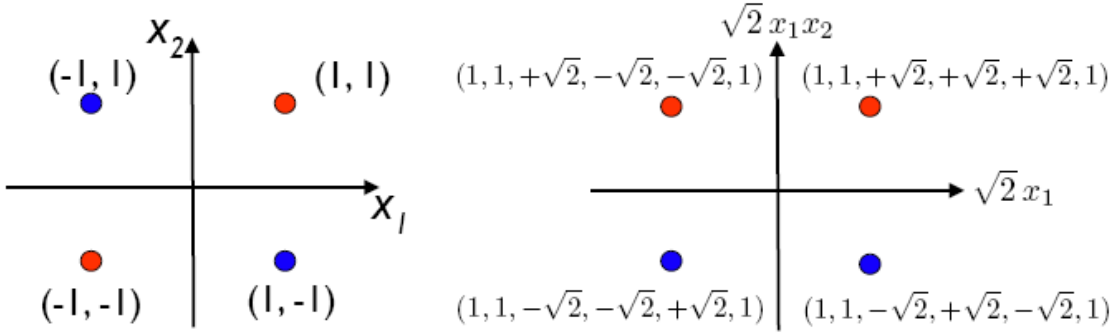
Examples of kernel

Examples

- Polynomial kernels: $\forall x, y \in \mathbb{R}^n, K(x, y) = (x^T y + c)^d, c > 0$. For $c = 2$ and $n = 2$,

$$K(\mathbf{x}, \mathbf{y}) = (x_1 y_1 + x_2 y_2 + c)^2 = \begin{pmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2} x_1 x_2 \\ \sqrt{2c} x_1 \\ \sqrt{2c} x_2 \\ c \end{pmatrix}^T \begin{pmatrix} y_1^2 \\ y_2^2 \\ \sqrt{2} y_1 y_2 \\ \sqrt{2c} y_1 \\ \sqrt{2c} y_2 \\ c \end{pmatrix}$$

- If $c = 1$



Examples and properties of kernels

Examples

- Gaussian kernels: $K(x,y) = \exp\left(-\frac{\|x-y\|^2}{2\sigma^2}\right)$, for $\sigma \neq 0$.
- Generative kernels: $K(\mathbf{x},\mathbf{y}) = \int P(\mathbf{x}|\mathbf{z})P(\mathbf{y}|\mathbf{z})P(\mathbf{z})d\mathbf{z}$, where $\mathbf{z}$ is a latent vector.
-

Properties

- The set of kernels is closed under some operations. Let K_1 and K_2 are kernels, then
  - cK_1 is a kernel, if $c > 0$.
  - $K_1 + K_2$ is a kernel.
  - Can make complex kernels from simple ones: modularity.
- Any symmetric positive semi-definite matrix can be regarded as a kernel matrix, that is as an inner product matrix in some space.

Graph kernels

Graph kernels

A graph kernel is a kernel function that computes an inner product on graphs.

- Compare substructures of graphs that are computable in polynomial time.
- Examples: walks, paths, cyclic patterns, trees.

Criteria for a good graph kernel

- Expressive
- Efficient to compute
- Positive definite
- Applicable to wide range of graphs

Random walk kernels

Principle

- Compare walks in two input graphs.
- Walks are sequences of nodes that allow repetitions of nodes.

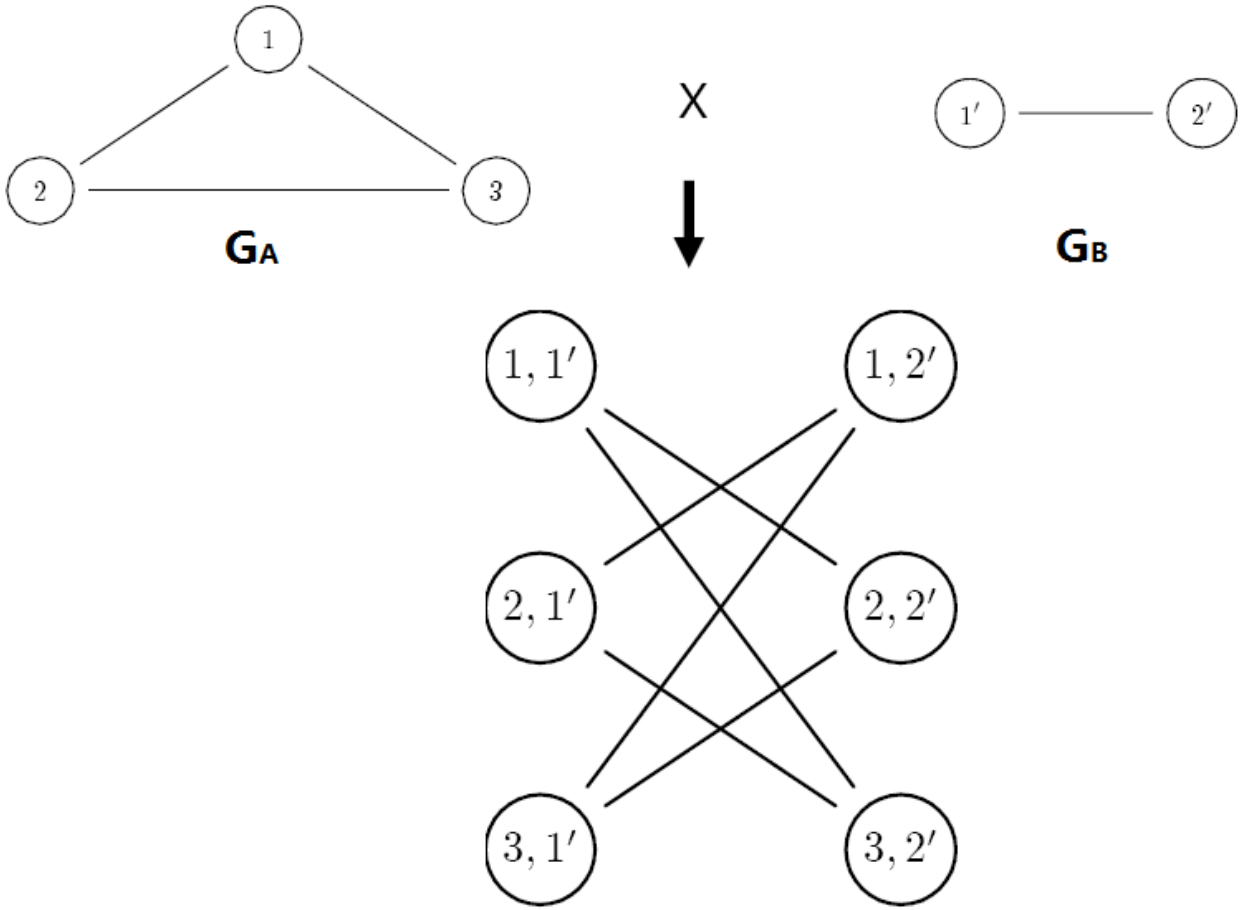
Important trick

- Walks of length k can be computed by taking the adjacency matrix A to the power of k.
- $A^k_{(i,j)} = c$ means that c walks of length k exist between v_i and v_j.
- How to find common walks in two graphs? Use the product graph of G_A and G_B.
  - Definition: $G_\times = (V_\times, E_\times)$, where $V_\times(G_A \times G_B) = \{(v_i^A, v_i^B) \in V_A \times V_B : \text{label}(v_i^A) = \text{label}(v_i^B)\}$ and $E_\times(G_A \times G_B) = \{((v_i^A, v_i^B), (v_j^A, v_j^B)) \in V_\times^2(G_A \times G_B) : (v_i^A, v_j^A) \in E_A, (v_i^B, v_j^B) \in E_B, \text{label}(v_i^A, v_j^A) = \text{label}(v_i^B, v_j^B)\}$.
  - Product graph consists of pairs of identically labeled nodes and edges from G_A and G_B.

Example of product graph

Product graph

- Each walk in $G_\times$ corresponds to one walk in G_A and G_B.
- Walks of length k can be computed by looking at $A_\times^k$.



Random walk kernels cont.

Definition

Note that common walks can now be computed from $A_{\times}^{k}$, thus

$$k_{\times}(G_A, G_B) = \sum_{i,j}^{|V_{\times}|} \Big[\sum_{n=0}^{\infty} \lambda^n A_{\times}^n \Big]_{ij} = e^T (I - \lambda A_{\times})^{-1} e.$$

- Random walk kernel counts all pairs of matching walks.
- λ is decaying factor for the sum to converge.

Important trick

- Computing product graph requires comparison of all pairs of edges in G_A and $G_B (O(n^4))$.
- Matrix multiplication or inversion for $n^2 \times n^2$ matrix runtime $(O(n^6))$.
- Complexity can be sped up to $O(n^3)$.

Fast kernel computation

Notation

- Operator *vec* flatters a matrix and vec^{-1} reconstructs it.
- The Kronecker product of A and B is written as $A \bigotimes B = (A_{ij}B)$

Product graph

- Entries in the adjacency graph are 1 iff corresponding nodes are adjacent in both G_A and G_B.
- The adjacency matrix of a product graph can be written as $A(G_A) \bigotimes A(G_B)$.

Sylvester equations

Definition

Equation of the form $M = PMQ + U$. Gory maths can help us to solve the equation.

- Rewrite: $vec(M) = vec(PMQ) + vec(U)$
- $vec(PMQ) = (Q^T \bigotimes P)vec(M)$.
- $(I - Q^T \bigotimes P)vec(M) = vec(U)$
- $vec(M) = (I - Q^T \bigotimes P)^{-1}vec(U)$.
- Multiply both sides by $vec(U)^T$:
  $vec(U)^T vec(M) = vec(U)^T (I - Q^T \bigotimes P)^{-1} vec(U)$
- Let $U = ee^T$, $Q = \lambda A(G_A)^T$ and $P = A(G_B)$, we have
  $e^T vec(M) = e^T (I - \lambda A(G_A) \bigotimes A(G_B))^{-1} e = e^T (I - \lambda A_\times)^{-1} e$.
- This is exactly the random walk graph kernel.

Take-home messages

- Centrality
  - Degree
  - Eigenvector
  - Closeness
  - Betweenness
- Weighted graph centrality
- PageRank and HITS
  - PageRank
  - HITS
  - Co-HITS
- Graph robustness

Acknowledgements

- Centrality
- Weighted graph centrality
- PageRank and HITS
- Graph robustness