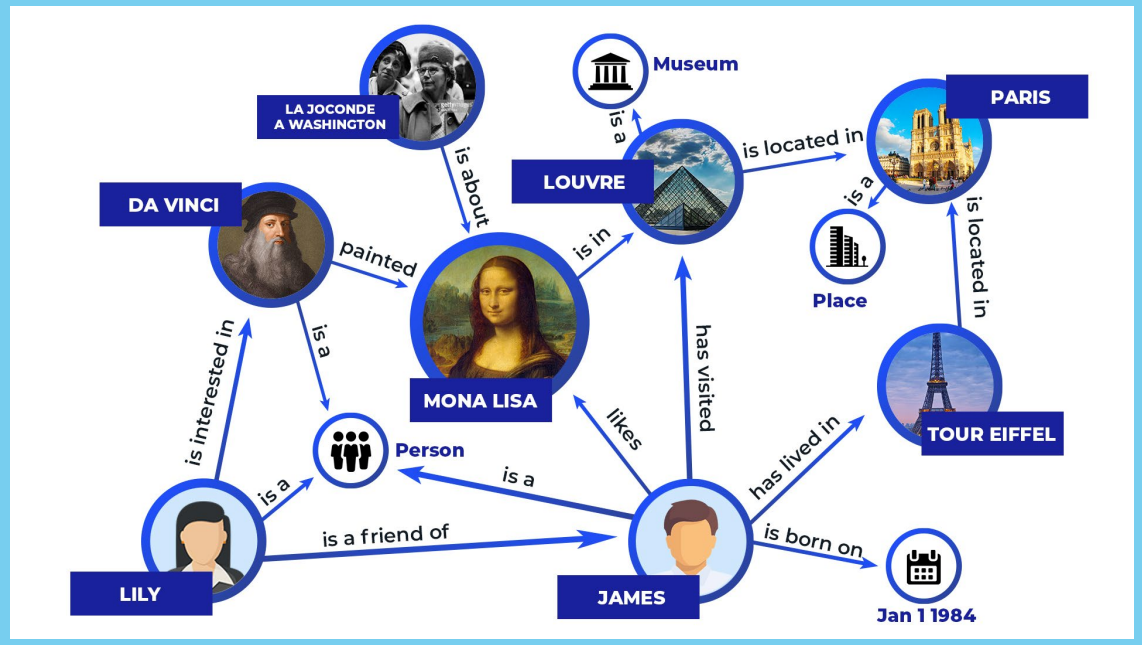


CS5344 : GRAPH MINING(I)

Anthony Tung

School of Computing

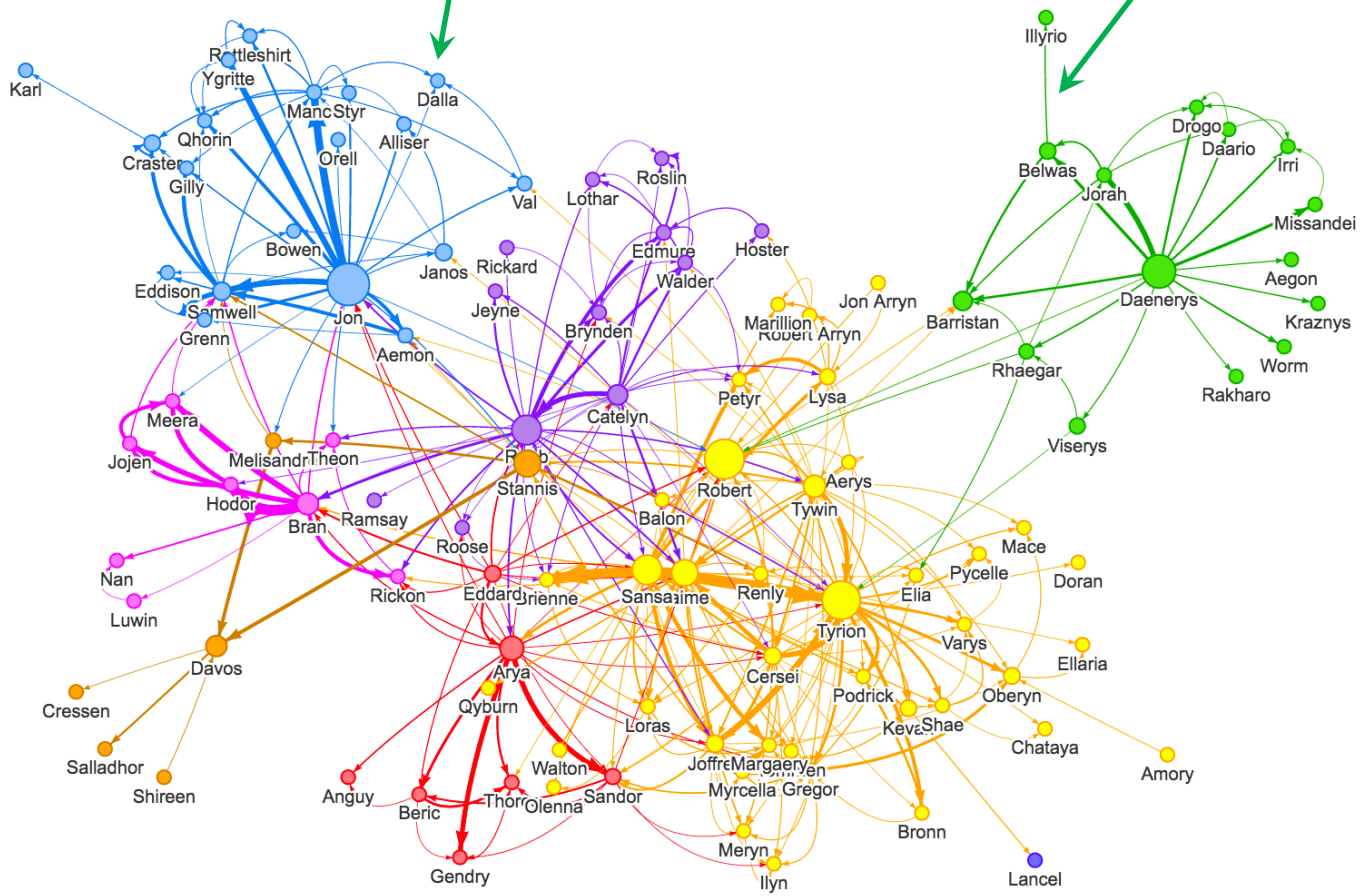
National University of Singapore

Slide Credit: Bryan Hooi, Chris von der Weth, Stanford CS246 – Mining Massive Datasets, Jure Leskovec, Anand Rajaraman, Jeff Ullman: https://mmds.org, cytoscape.org

GRAPHS: INTRODUCTION

Nodes (e.g. people)

Edges represent relationships

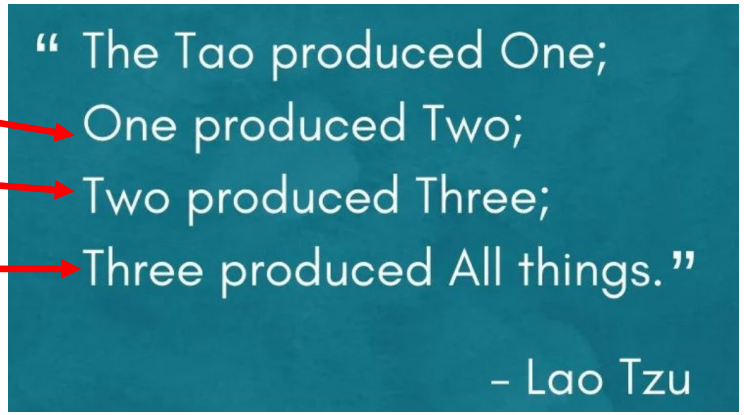


PHILOSOPHY OF GRAPHS

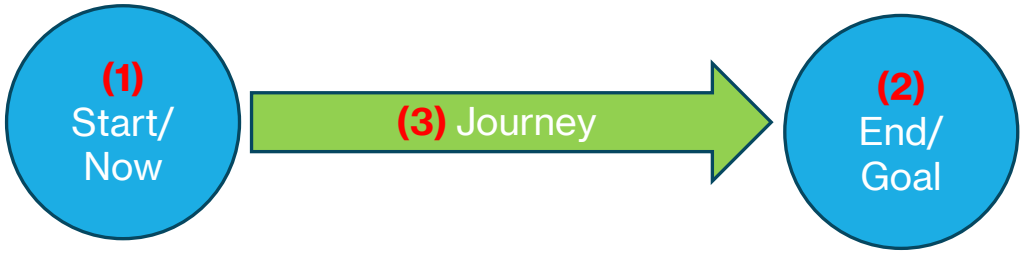
(One)source

(Two) target

(Three) change/comparison



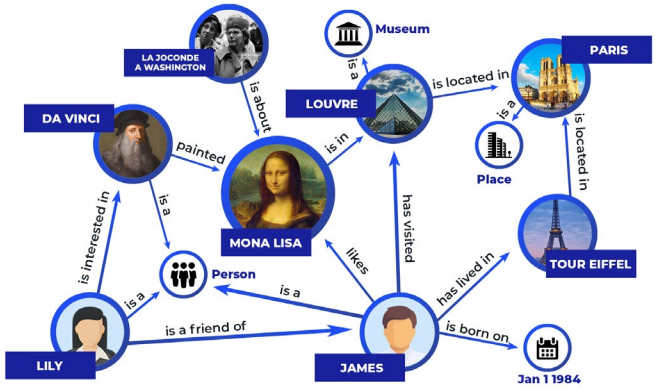
“道生一，生二，二生三，三生万物。”
《道德经》



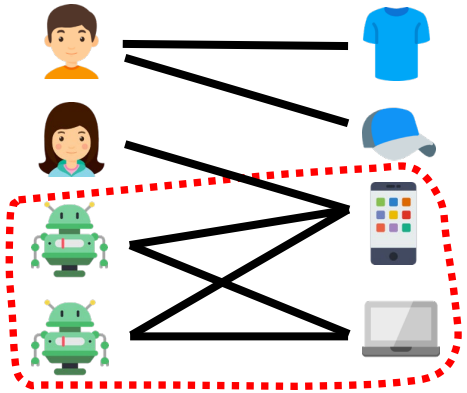
- Similarity Search/Clustering: Similarity Function
- Association rules: Connection between items
- Classification/Regression: Connection between attribute values and a target attribute
- Time series: Connection between time points
- CS5344: Connection between untrained and trained data scientist
- “Three” produce All things because it represent connections and changes

GRAPHS ARE EVERYWHERE

Graphs are ubiquitous: social, biological, chemical, web, textual, …



Knowledge graph

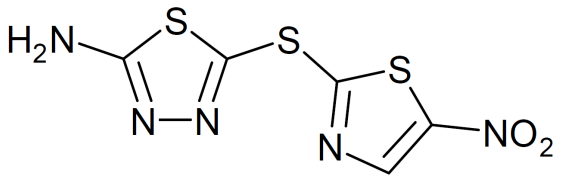


amazon

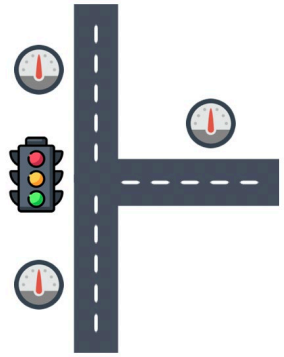
twitter

facebook

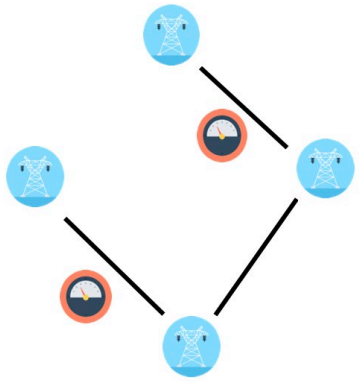
Review graph



Molecular graph



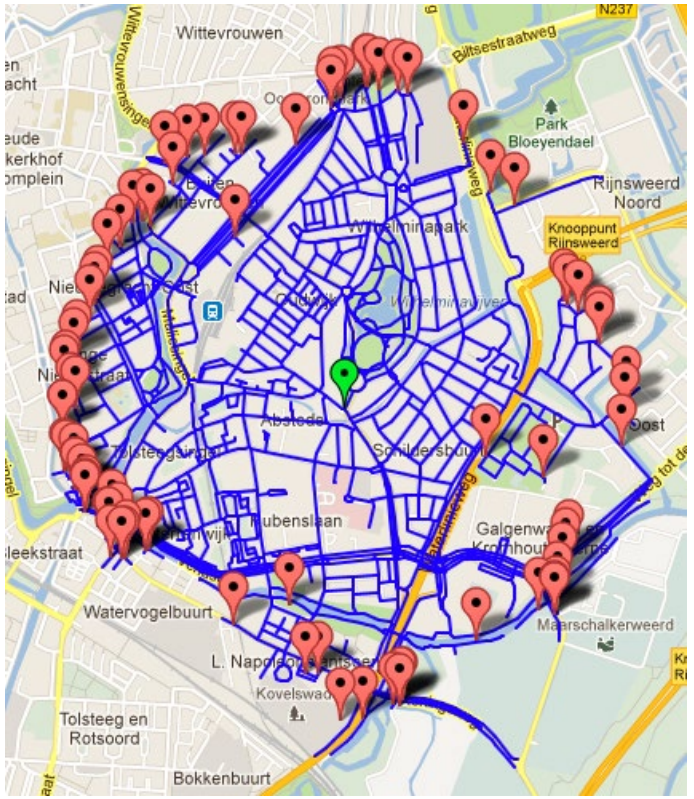
Traffic graph



Power grid graph

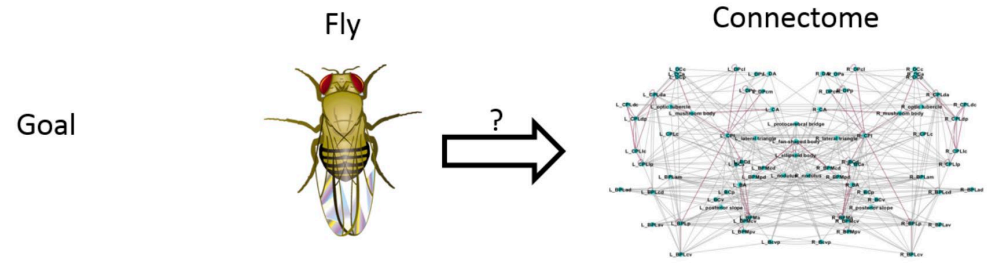
TRANSPORTATION

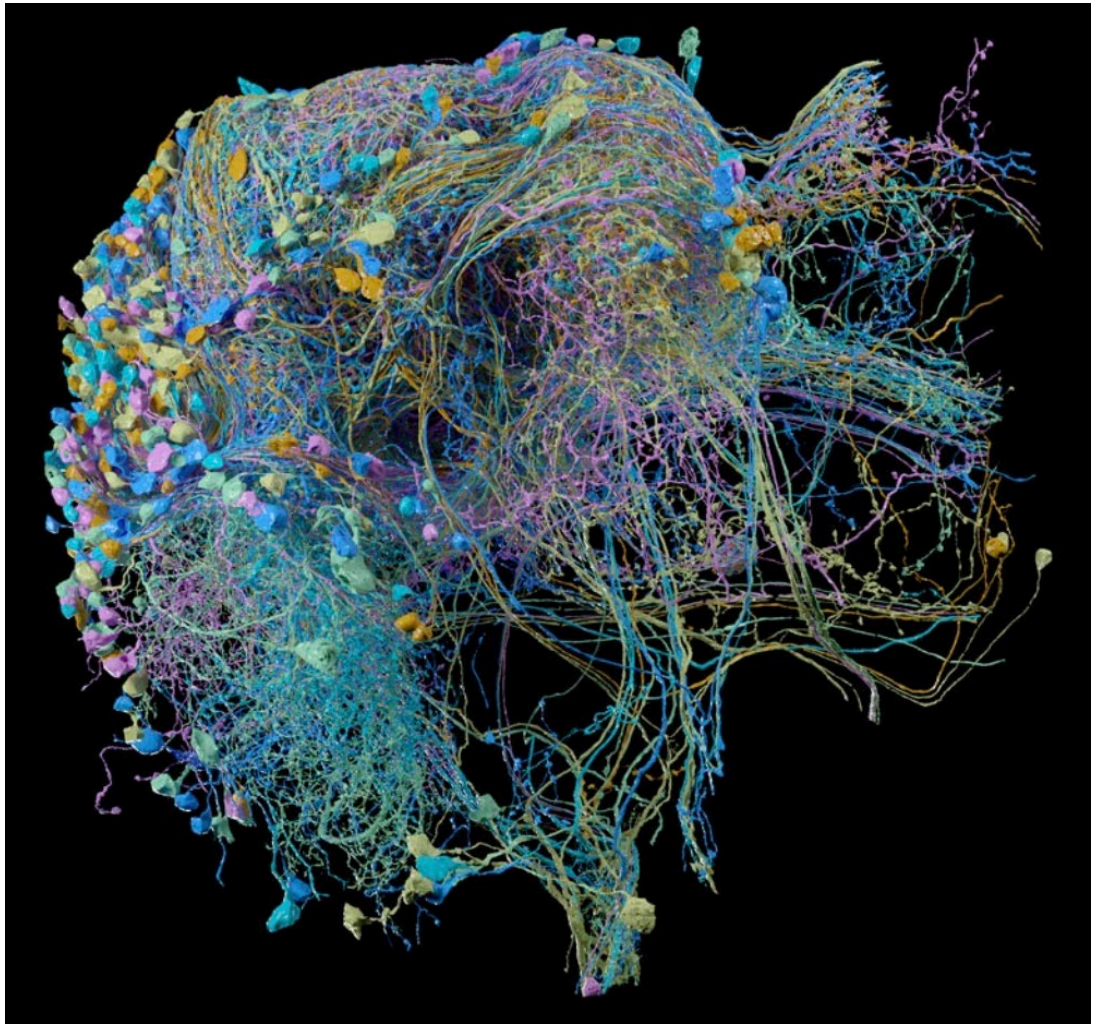
- Find shortest paths
- Plan bus routes
- Predict traffic flow / usage
- Optimize transport system design



FRUIT FLY CONNECTOME

Recently sequenced map of the fruit fly's 'hemibrain', containing 25,000 neurons and 20M neural connections





https://www.janelia.org/project-team/flyem, https://www.biorxiv.org/content/10.1101/2020.01.21.911859v1

GRAPH MINING OVERVIEW

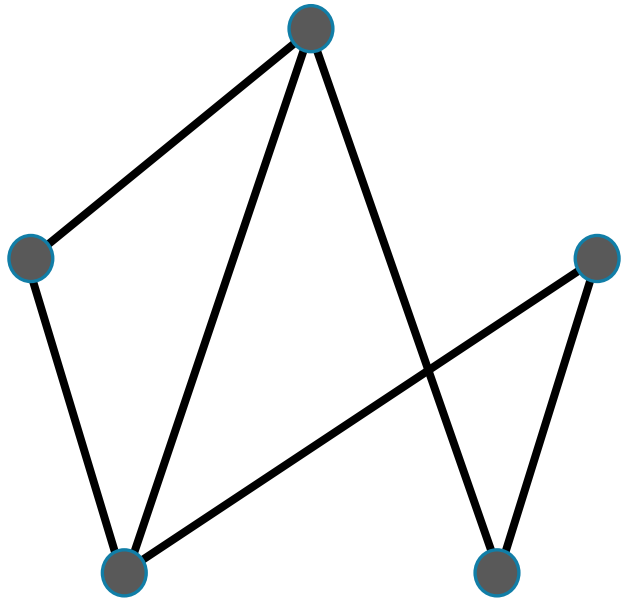
1. Introduction
2. Basic Concepts
3. Community Detection
   - Betweenness-based: Girvan Newman
   - Modularity Maximization
4. Classification Problems on Graphs

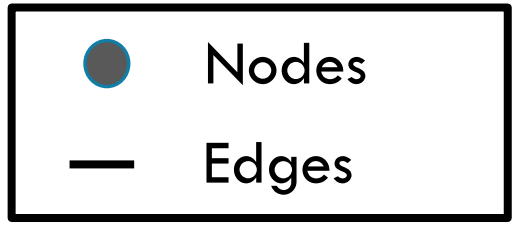
GRAPHS

A **graph** (or **network**) is a data structure used to model interactions.

Undirected graphs contain a set of nodes V, and a set of (undirected) edges E between pairs of nodes.

Example: Facebook: nodes represent people, edges represent friendships



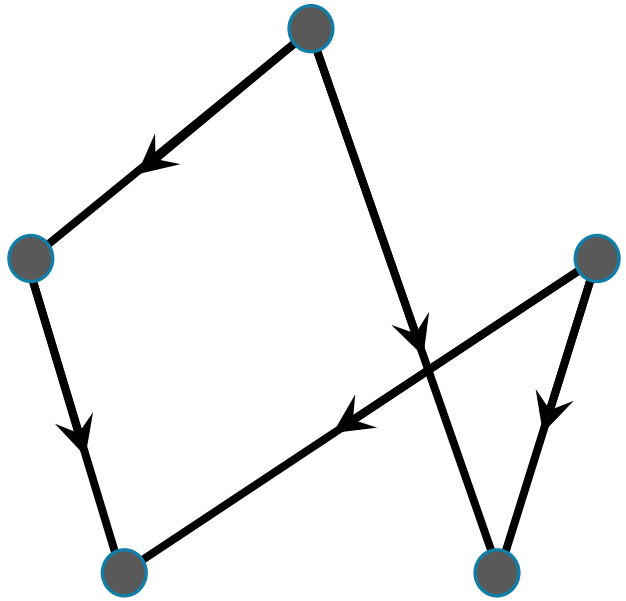


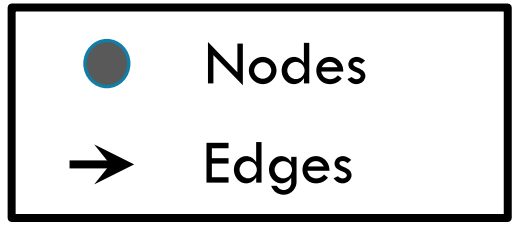
DIRECTED GRAPHS

A **graph** (or **network**) is a data structure used to model interactions.

Directed graphs contain a set of nodes V, and a set of (directed) edges between pairs of nodes.

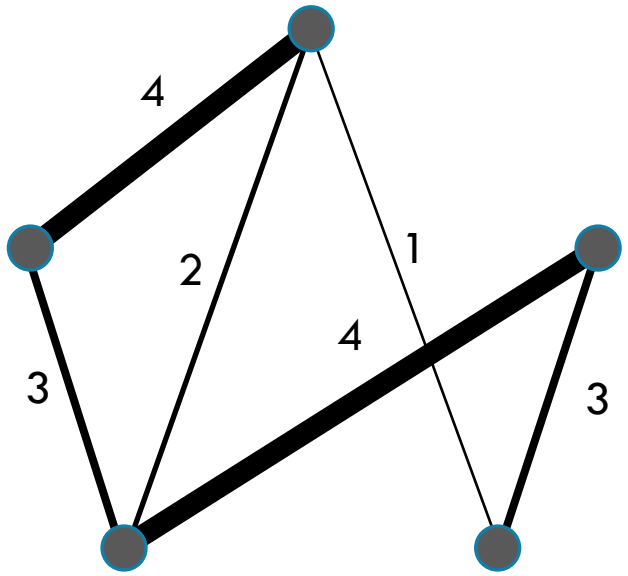
Example: Twitter: nodes represent people, edges represent follows

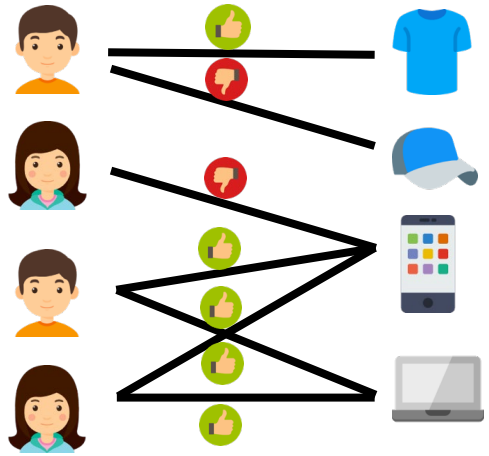




WEIGHTED GRAPH

Each edge is associated with a weight (e.g. the score of a review)



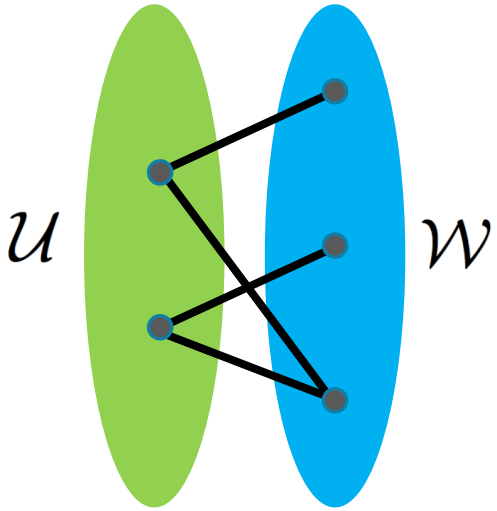


BIPARTITE GRAPHS

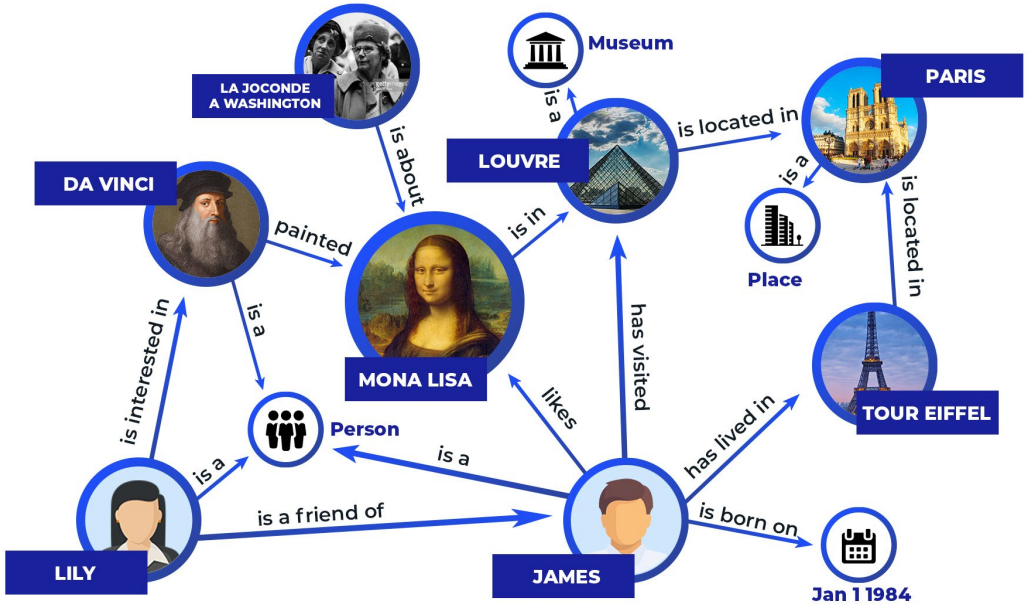
A **bipartite graph** is a graph with two sets of nodes, U and W.

Each edge connects a node from U and a node from W.

Example: (**Review graph**) U as the set of users, W as products, and edges representing a review by a user for a product.



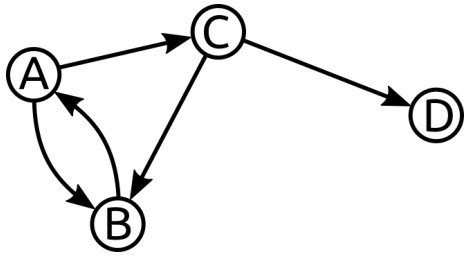
HETEROGENEOUS GRAPH



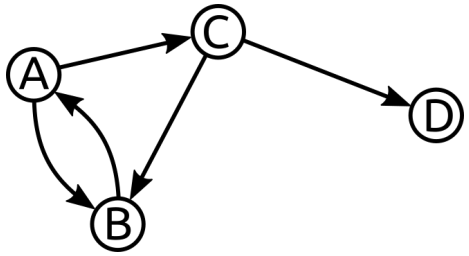
A **heterogeneous graph** has multiple types of nodes and / or edges

Example: **knowledge graphs** can have many node types (Person, Place, Country, …) and many different types of edges (“Is a”, “Is located in”, “Is born on”, …)

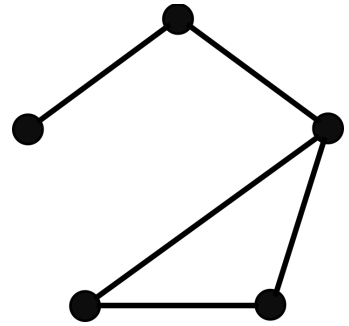
CYCLIC GRAPHS, MULTIGRAPHS, SPARSE GRAPHS



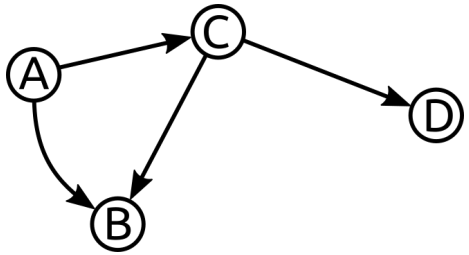
Cyclic Graph
vs.
(Directed) Acyclic Graph

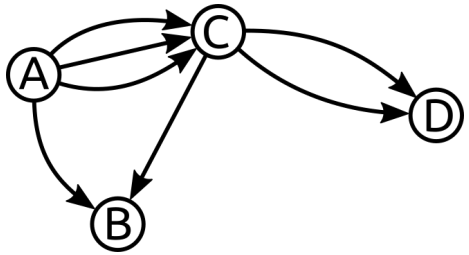


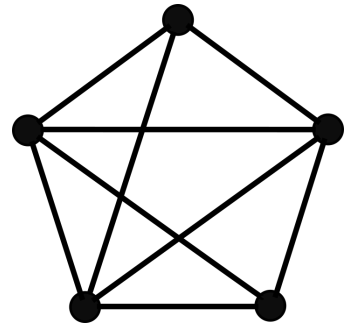
(Simple) Graph
vs.
Multigraph



Sparse Graph
vs.
Dense Graph







SUBGRAPHS AND NEIGHBORHOODS

A **subgraph** is a graph formed by a subset of the nodes and edges of a graph.

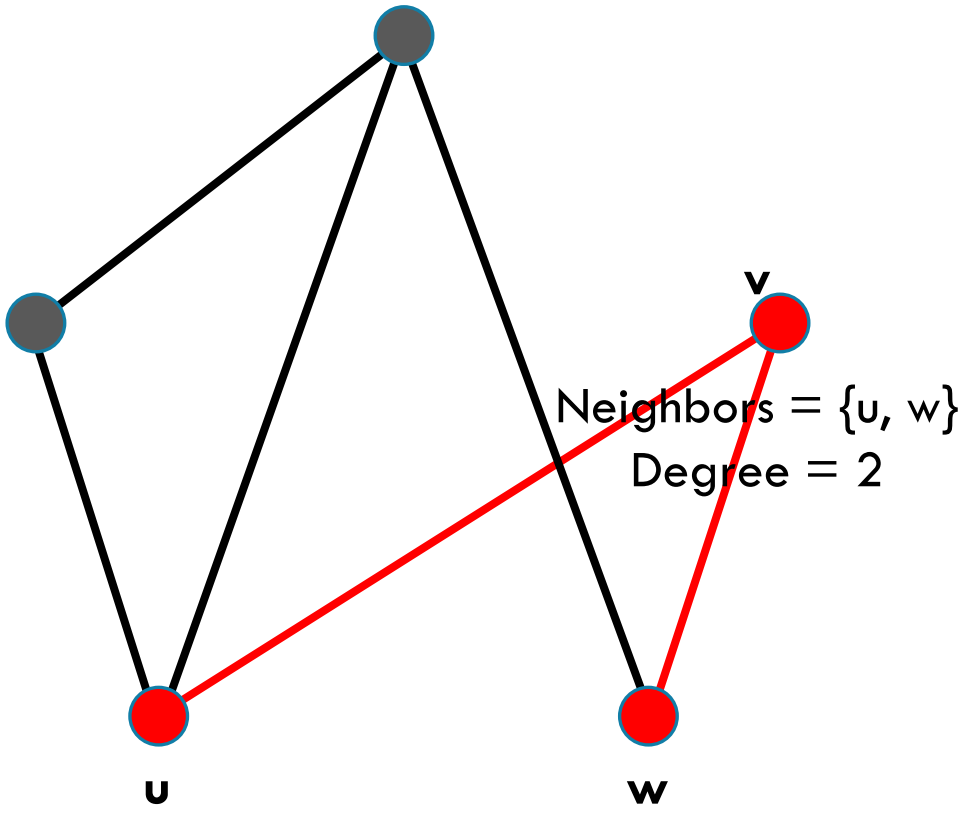
The **neighbors** of a node v are the set of nodes which have edges with v.

- (For directed graphs: **in-neighbors** have edges to v; **out-neighbors** have edges from v)

The **neighborhood** N(v) is a subgraph formed by the neighbors of the node v (and the node v itself[1], and the edges joining this set of nodes).

The **degree** of a node is its number of neighbors.

- (For directed graphs: **in-degree** is the number of in-neighbors; **out-degree** is the number of out-neighbors)



1: the "closed neighborhood" includes the node v itself; the "open neighborhood" excludes it.

ADJACENCY MATRIX

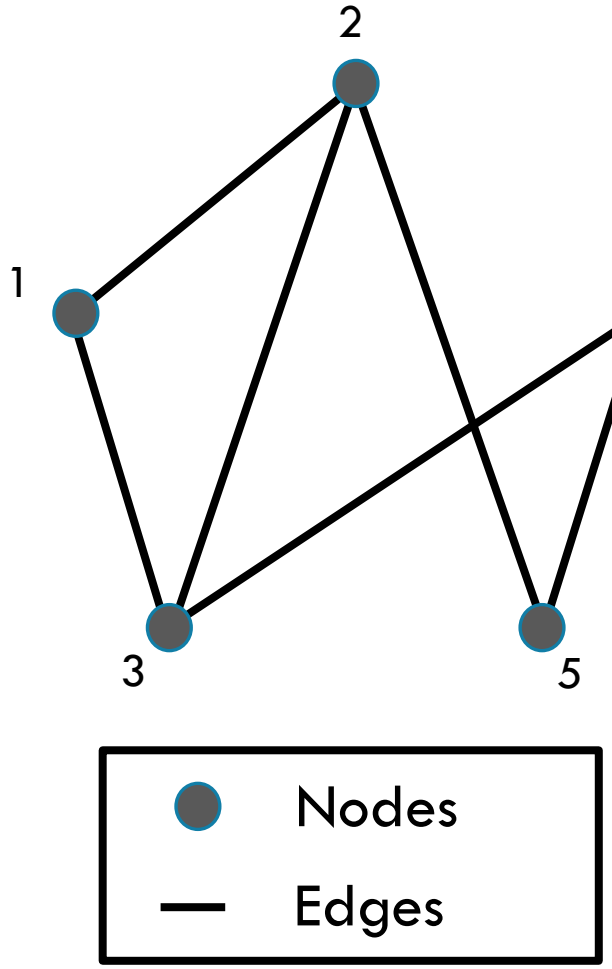
Binary matrix **A** representing a graph

Each row and column represents a node

Each entry is 1 if there is an edge from one node to the other; 0 otherwise

A is symmetric for undirected graphs, but not for directed graphs

For weighted graphs, **A** need not be a binary matrix



	1	**2**	**3**	**4**	**5**
1	0	1	1	0	1
2	1	0	1	0	0
3	1	1	0	1	0
4	0	0	1	0	1
5	1	0	0	1	0

CENTRALITY MEASURES

Centrality measures are ways to quantify the ‘importance’ of each node

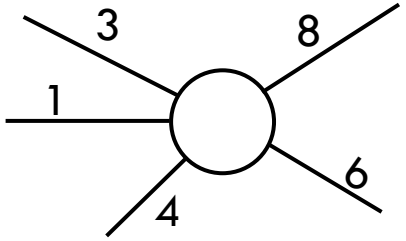
Different measures give different ‘flavors’ of importance

DEGREE CENTRALITY

Undirected graph

Sum of weights of connected edges

$$c_d(v_i) = \sum_{v_j \in V} A[i,j]$$

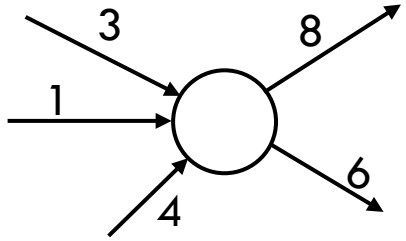


$$c_d(v) = 22$$

Directed graph

Sum of weights of incoming edges

$$c_{d\_in}(v_i) = \sum_{v_j \in V} A[j,i]$$



$$c_{d\_in}(v) = 8$$

Sum of weights of outgoing edges

$$c_{d\_out}(v_i) = \sum_{v_j \in V} A[i,j]$$

$$c_{d\_out}(v) = 14$$

Pro: easy and fast to calculate

Con: only considers the node's immediate neighborhood, ignoring the rest of the graph

CLOSENESS CENTRALITY

Intuition: a node is central if its distance to most other nodes is small

Closeness centrality is the *reciprocal of the node's average distance to other nodes*

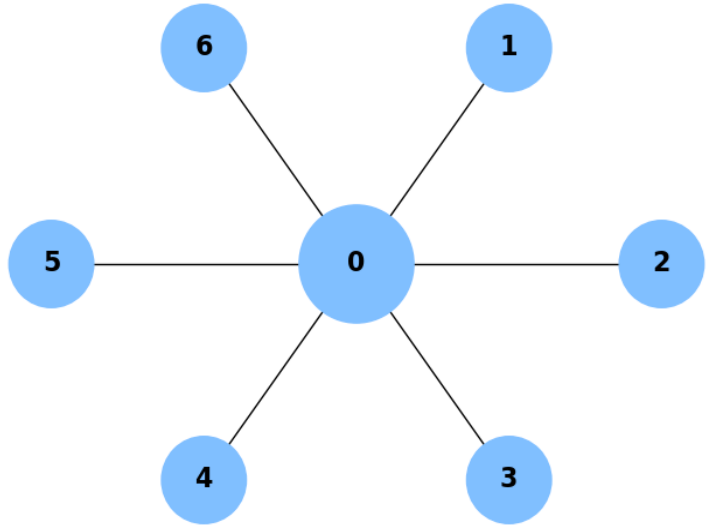
For directed graphs, we can either consider incoming or outgoing edges for calculating distances

n — number of nodes

$$C(u) = \frac{n-1}{\sum_{v=1}^{n-1} d(v,u)}$$

d(*v*,*u*) — length of shortest path from v to u

Note: For directed graphs, this definition calculates a node's closeness using its incoming edges — more common case.
To consider outgoing edges, *d*(*v*,*u*) becomes *d*(*u*,*v*).



BETWEENNESS CENTRALITY

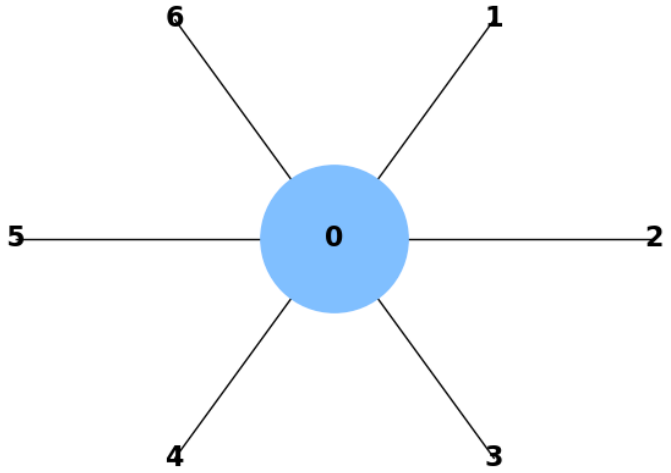
Intuition: node v is central if many shortest paths between other node pairs pass through v

- Removing such nodes would cause the most "disruption" to the graph
- Directly applicable to both directed/undirected and weighted/unweighted graphs

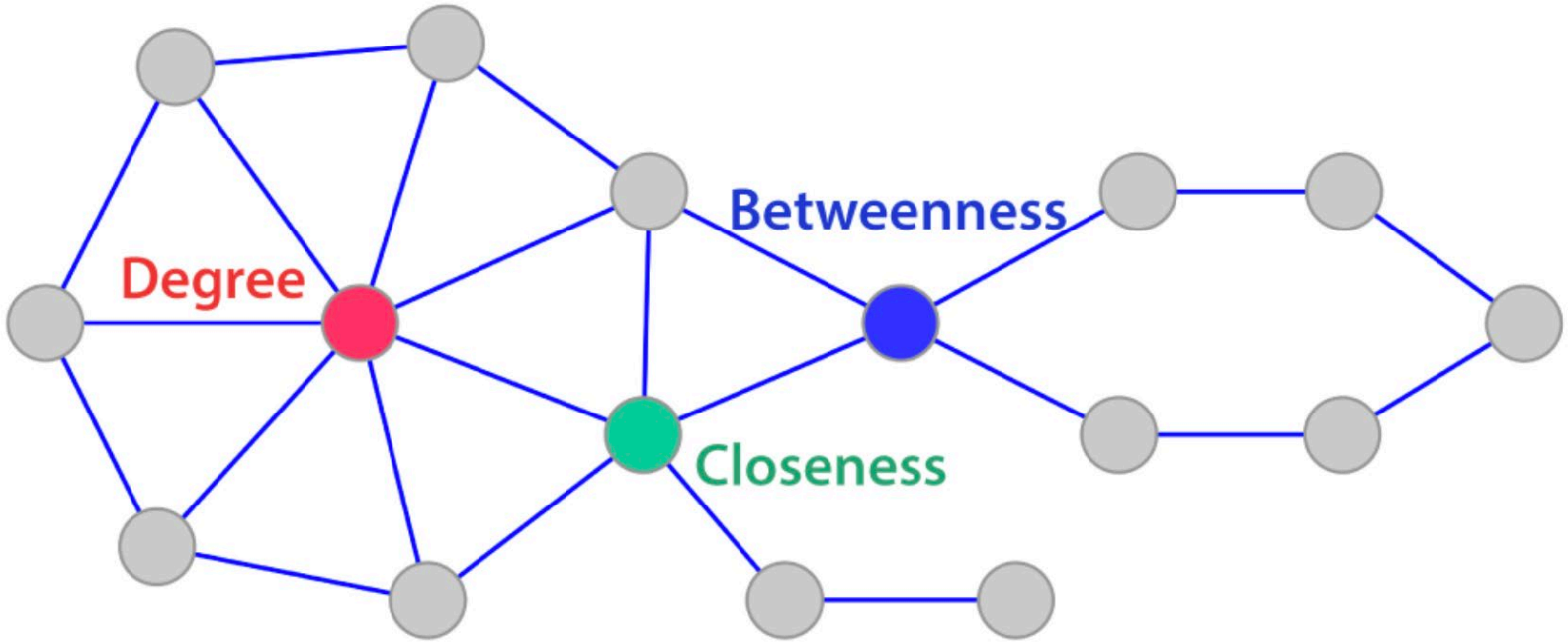
$$c_b(v) = \sum_{s,t \in V; s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

number of shortest paths from *s* to *t* passing through node *v*

total number of shortest paths from *s* to *t*



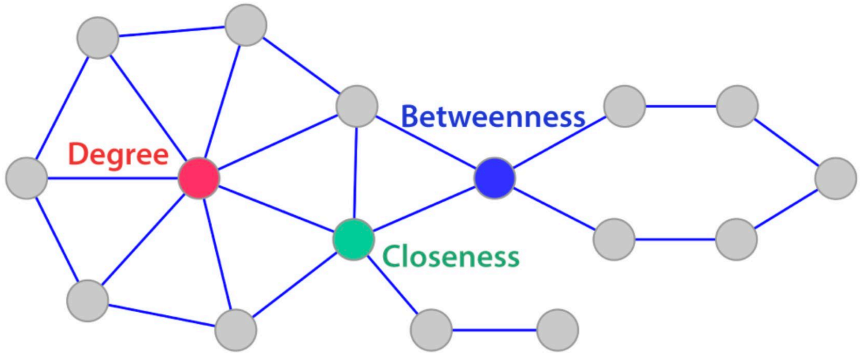
COMPARISON BETWEEN CENTRALITY MEASURES



DISCUSSION: CENTRALITY MEASURES

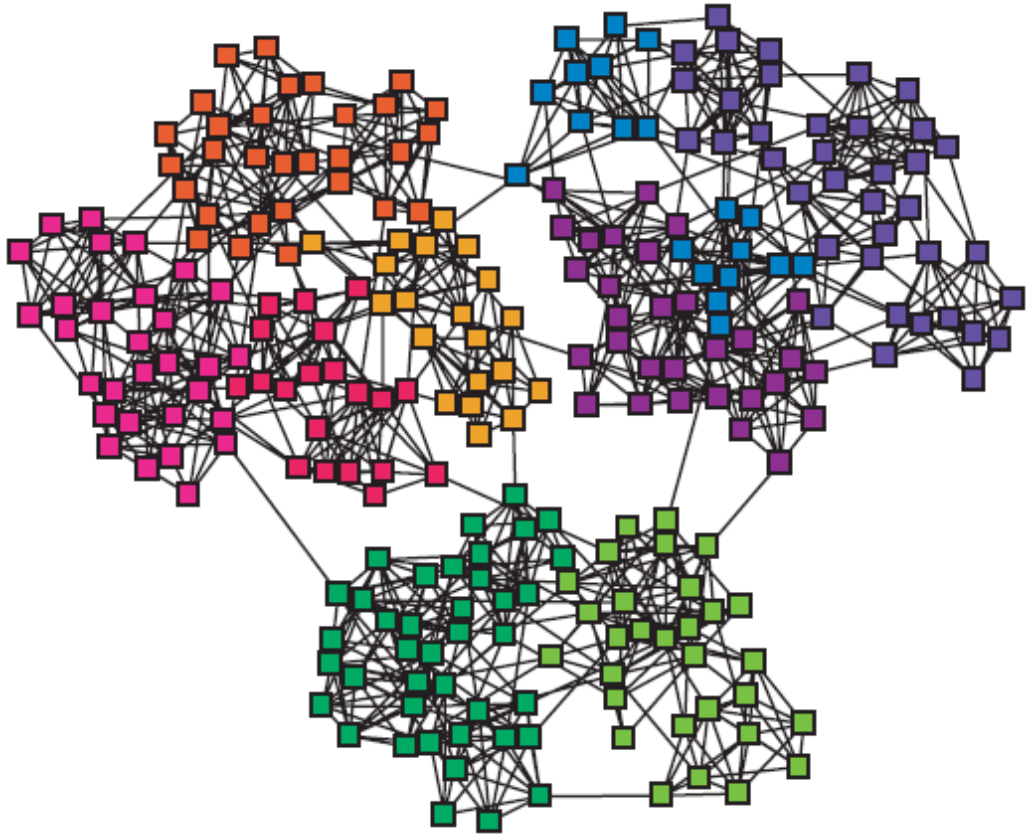
Different centrality measures have very different meanings and interpretations. Choose the one that makes the most sense for your use-case.

Time complexity: betweenness and closeness centrality are relatively slow: $O(|V| \cdot |E|)$ time, which is faster than the naive approach (of computing all shortest paths), but still often slow in practice

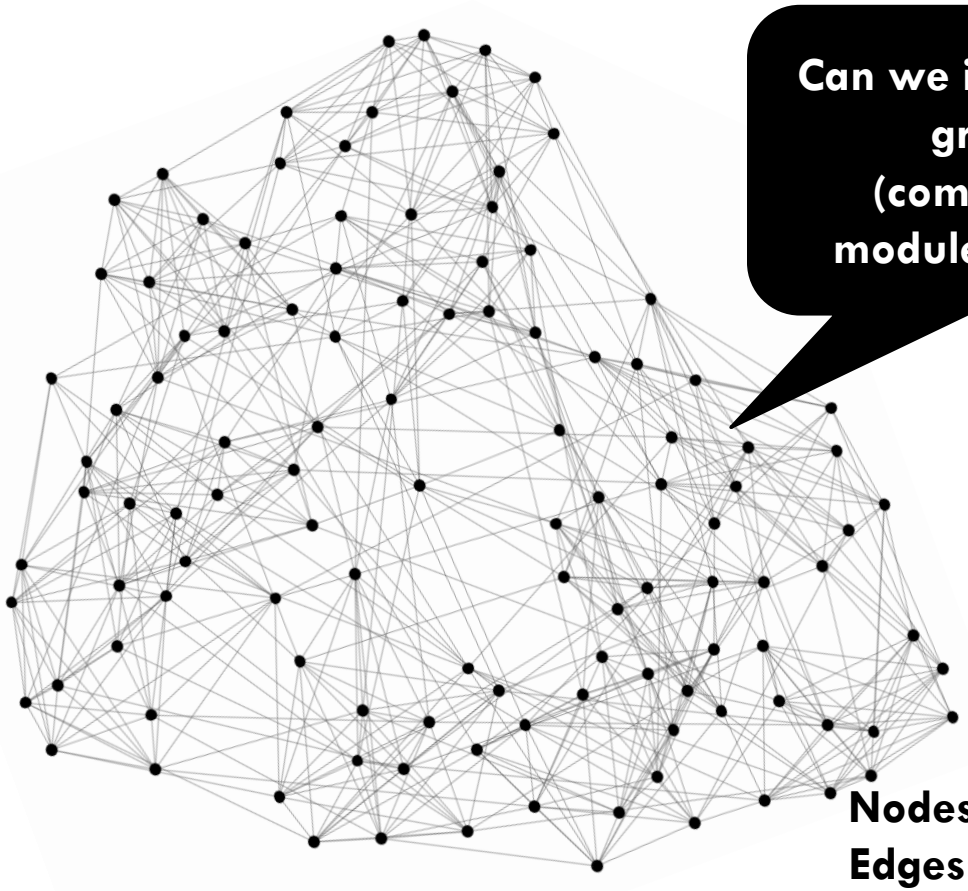


GRAPH MINING OVERVIEW

1. Introduction
2. Basic Concepts
3. Community Detection
   - Betweenness-based: Girvan Newman
   - Modularity Maximization
4. Classification Problems on Graphs



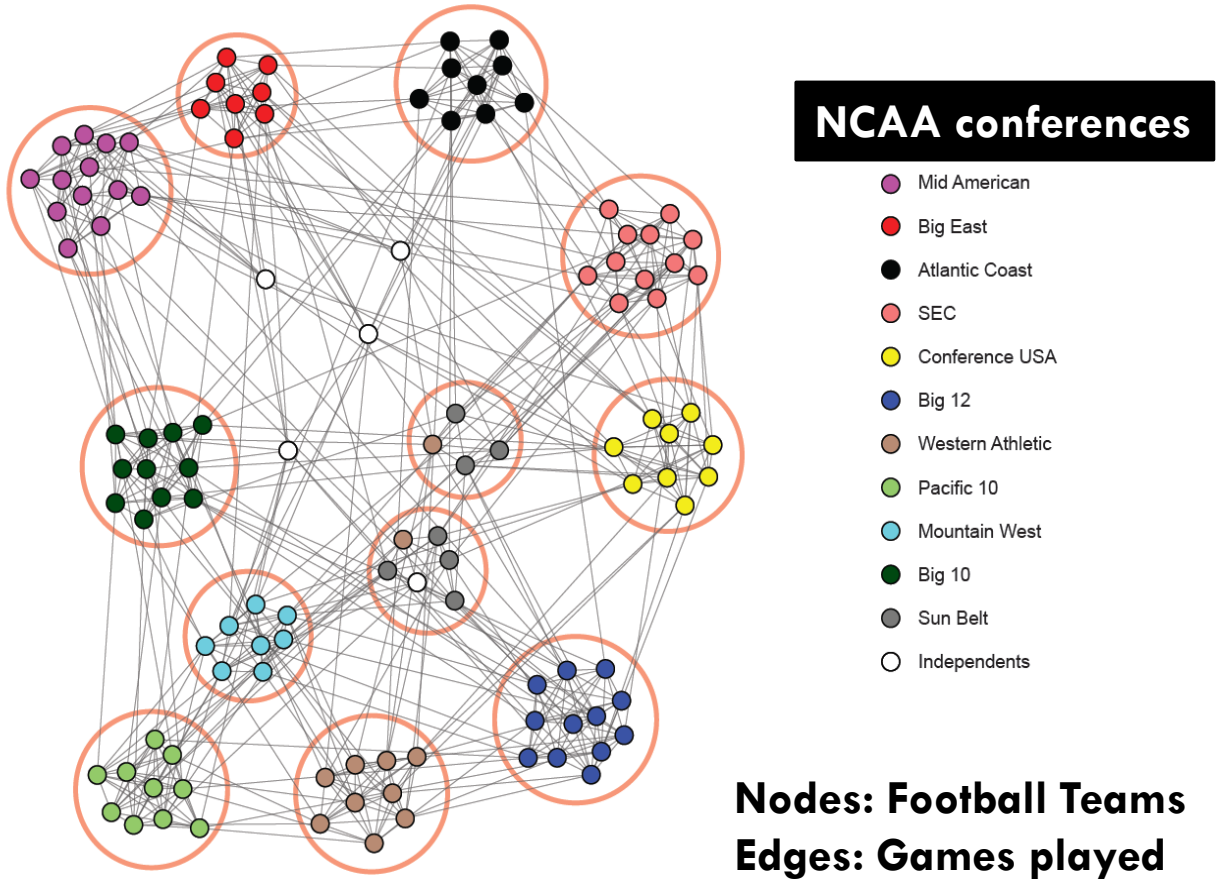
MOTIVATION: COMMUNITY DETECTION



Can we identify node groups? (communities, modules, clusters)

Nodes: Football Teams
Edges: Games played

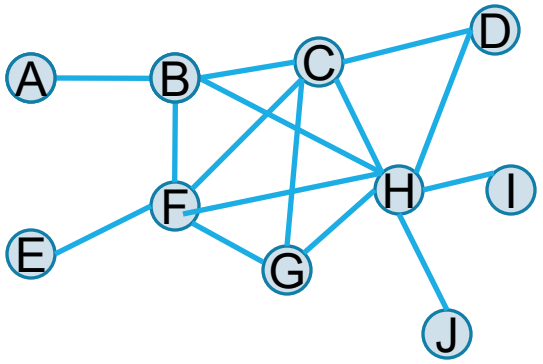
MOTIVATION: COMMUNITY DETECTION



Nodes: Football Teams
Edges: Games played

FINDING COMMUNITIES IN LARGE GRAPH

Many social networks or interaction graphs can be represented as an adjacency matrix consisting of 1's and 0's. Cluster the nodes based on Jaccard distance

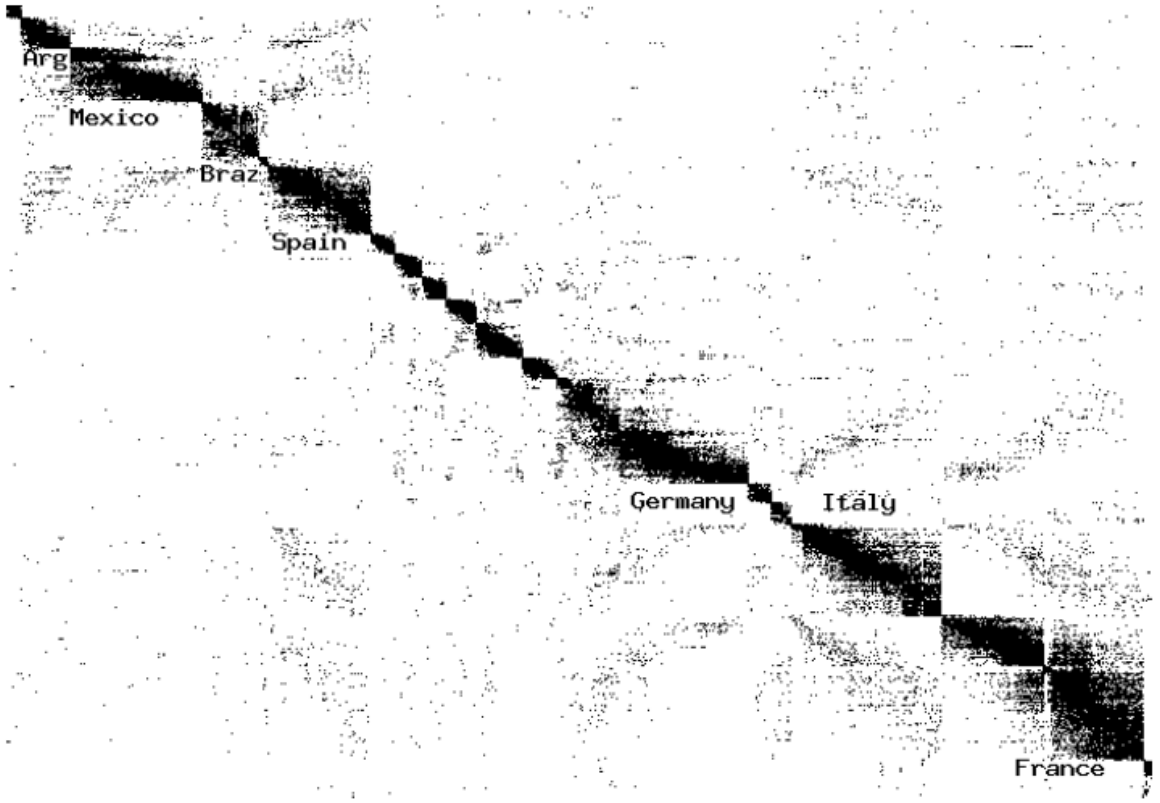


	A	B	C	D	E	F	G	H	I	J
A	0	1	0	0	0	0	0	0	0	0
B	1	0	1	0	0	1	1	1	0	0
C	0	1	0	1	0	1	1	1	0	0
D	0	0	1	0	0	0	0	1	0	0
E	0	0	0	0	0	1	0	0	0	0
F	0	1	1	0	1	0	1	1	0	0
G	0	1	1	0	0	1	0	1	0	0
H	0	1	1	1	0	1	1	0	1	1
I	0	0	0	0	0	0	0	1	0	0
J	0	0	0	0	0	0	0	1	0	0

EXAMPLE: MOVIES-TO-ACTORS GRAPH (BIPARTITE)

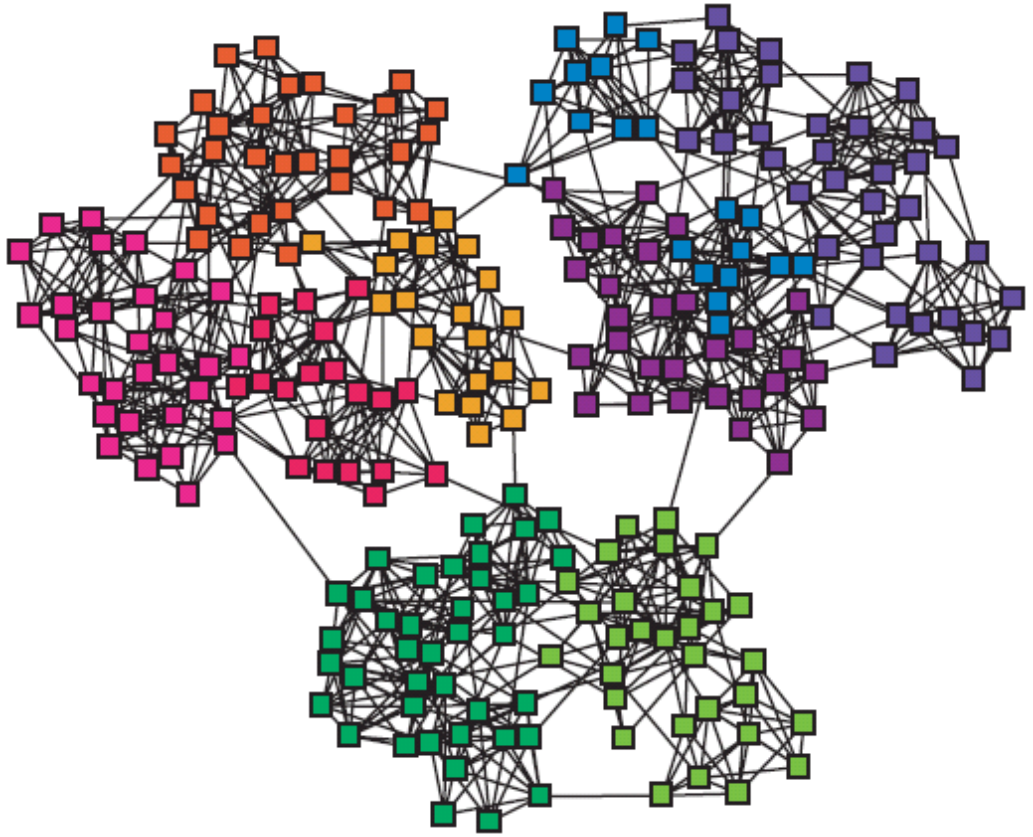
Actors

Movies



GRAPH MINING OVERVIEW

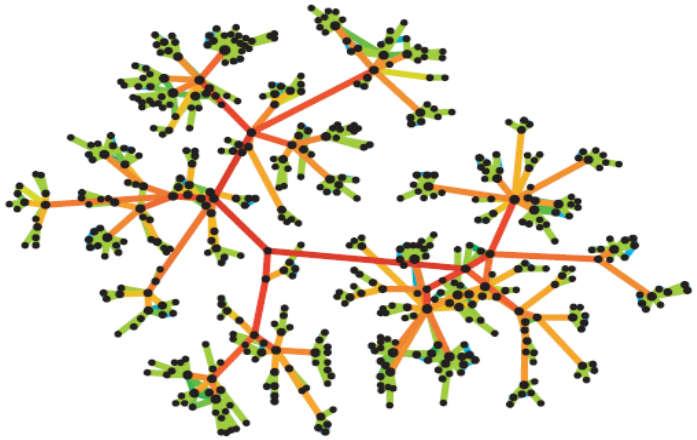
1. Introduction
2. Basic Concepts
3. Community Detection
   - Betweenness-based: Girvan Newman
   - Modularity Maximization
4. Classification Problems on Graphs



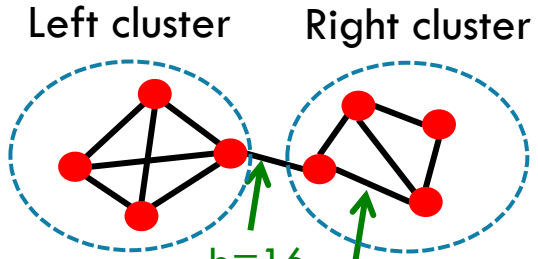
EDGE BETWEENNESS

Edge betweenness: Number of shortest paths passing over the edge

Intuition: high betweenness edges are the edges that act as 'bridges' between different parts of the graph (the red edges in the graph below):



Edge betweenness (red = higher)



(note: 2 "tied" shortest paths are counted as ½ each)

E.g. this edge is on the shortest path for all 4 x 4 = 16 of the paths from any point in the left cluster to any point in the right cluster

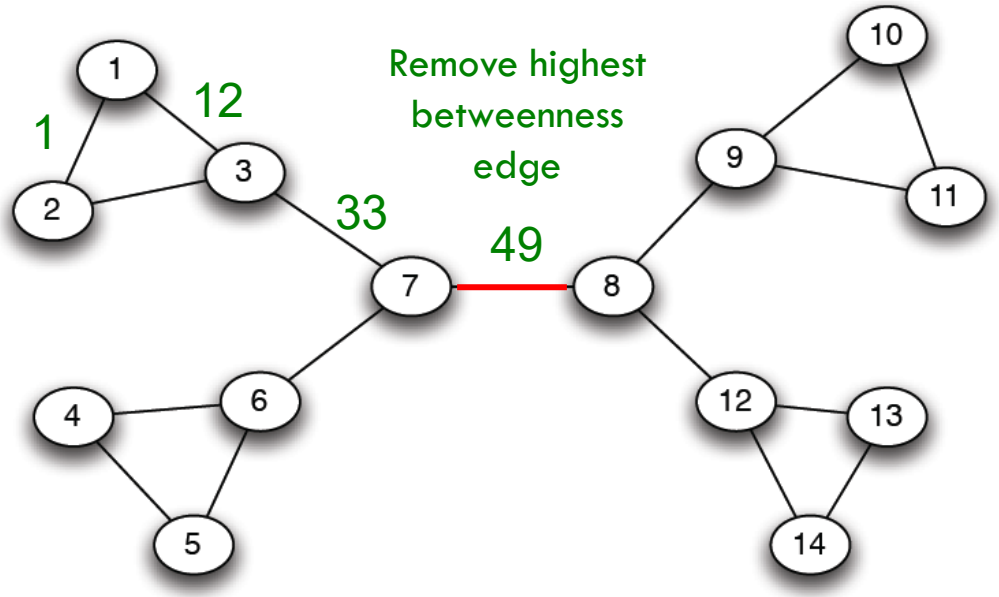
GIRVAN NEWMAN ALGORITHM

Divisive **hierarchical clustering** based on the notion of edge **betweenness**:

Girvan-Newman Algorithm:

- **Repeat until no edges are left:**
  - Calculate betweenness of edges
  - Remove edge with highest **betweenness**
- Gives a hierarchical decomposition of the network

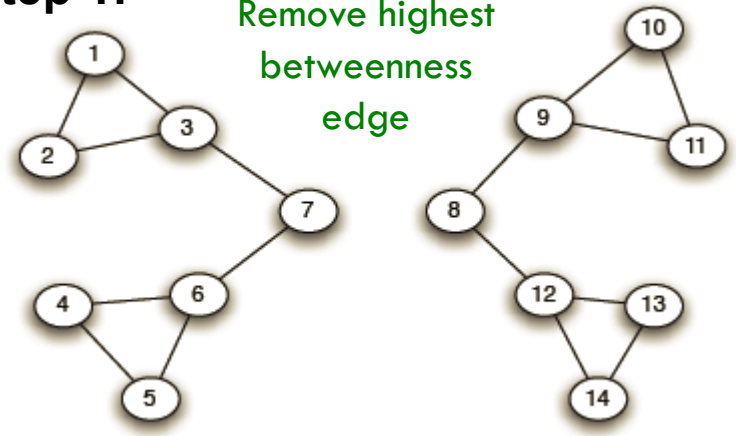
GIRVAN NEWMAN: EXAMPLE



GIRVAN NEWMAN: EXAMPLE

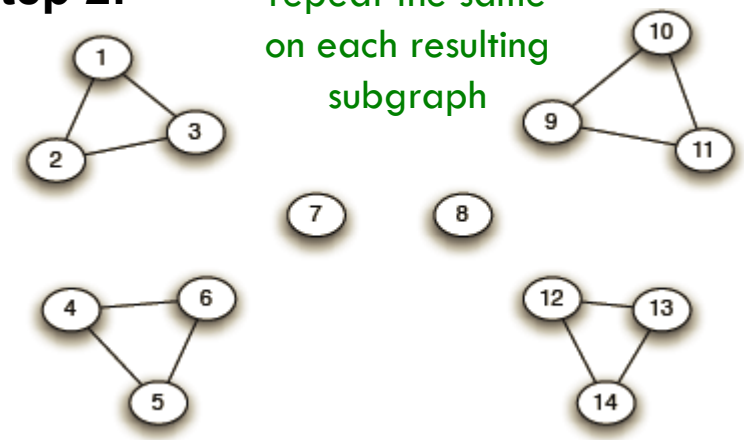
Step 1:

Remove highest betweenness edge



Step 2:

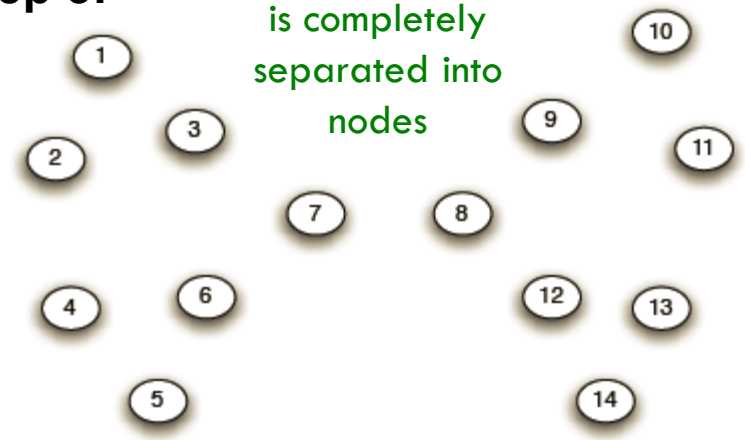
Recursively repeat the same on each resulting subgraph



Note: Need to re-compute betweenness at every step

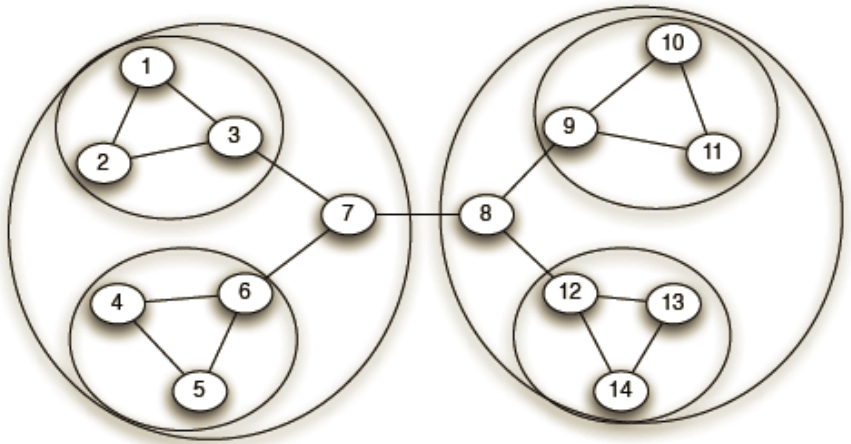
Step 3:

…until the graph is completely separated into nodes

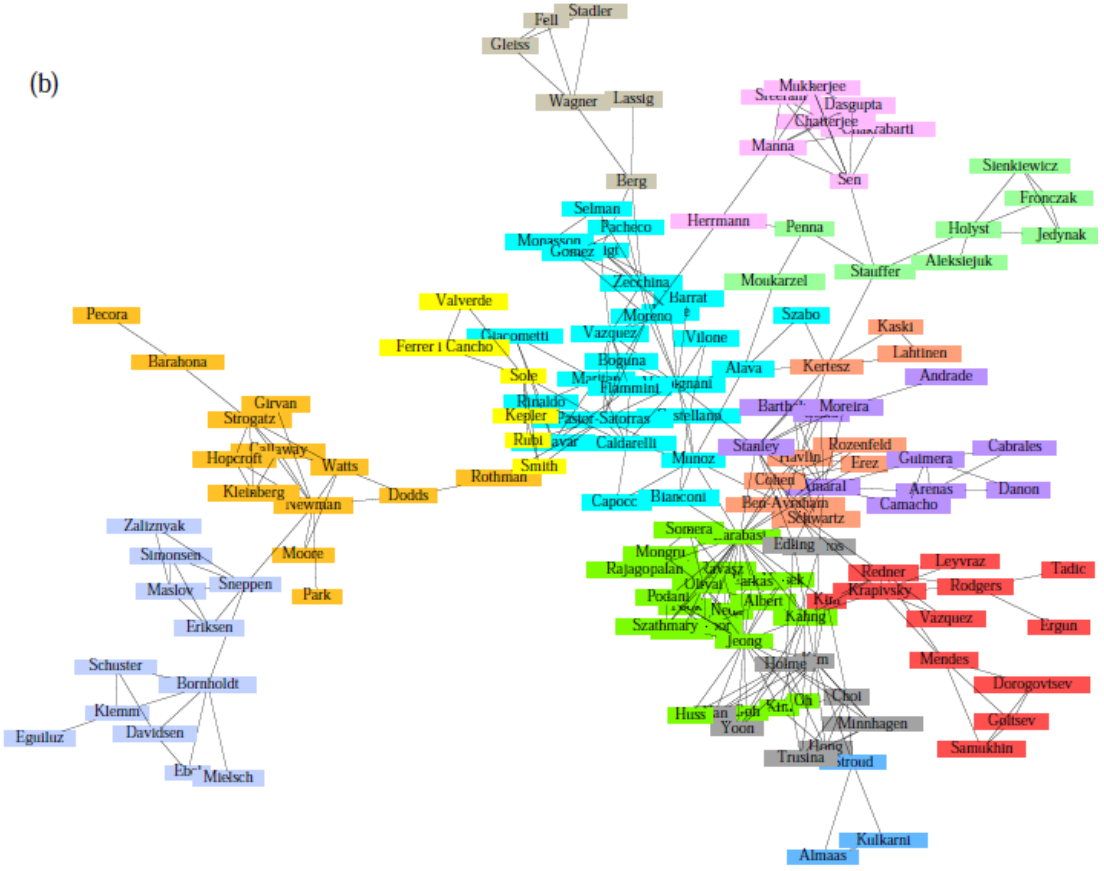


Just like in hierarchical clustering, this produces a hierarchy

Hierarchical network decomposition:



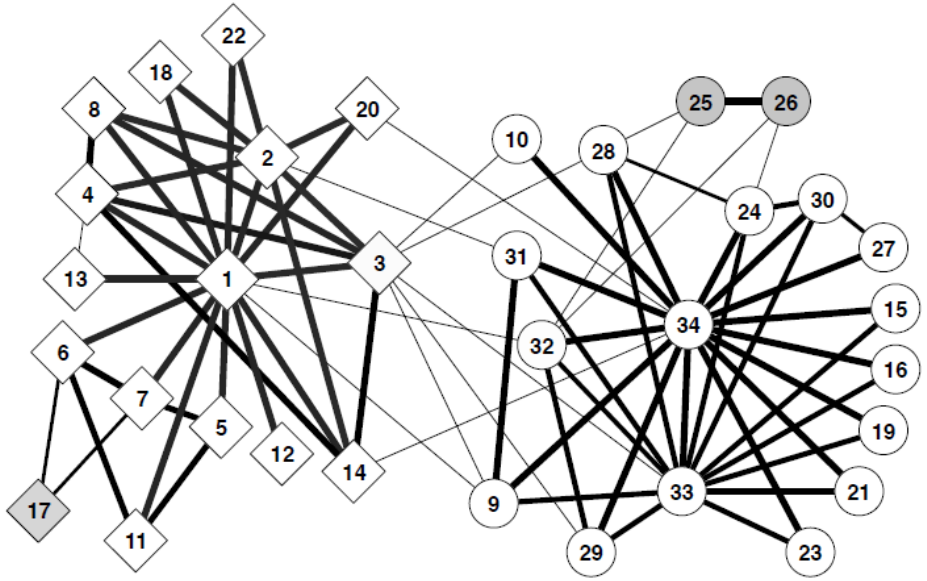
GIRVAN NEWMAN: EXAMPLE OUTPUT

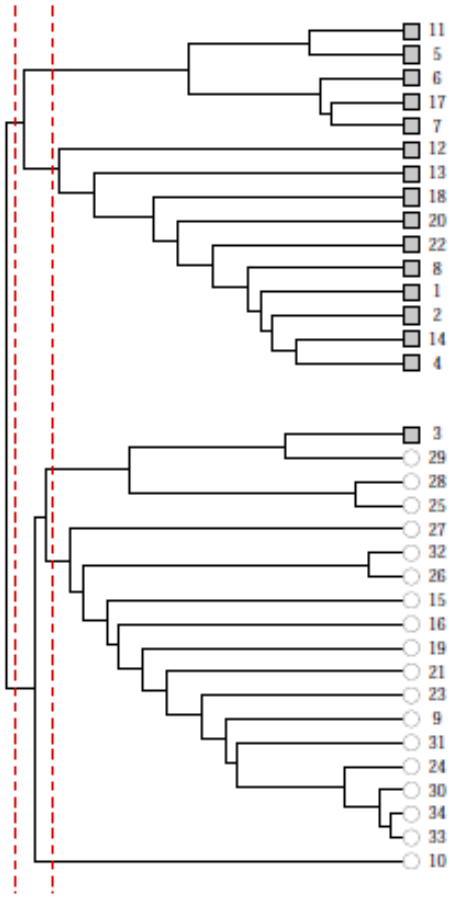


Communities in physics collaboration network

GIRVAN NEWMAN: EXAMPLE OUTPUT

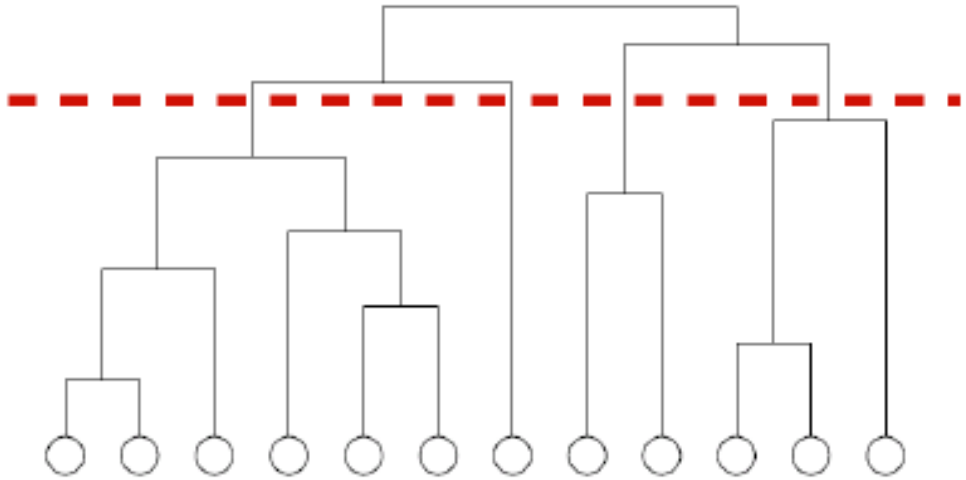
Zachary's Karate Club





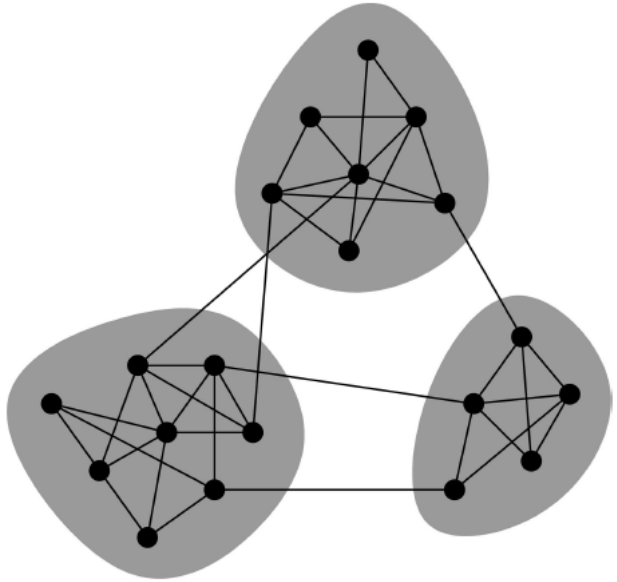
GIRVAN-NEWMAN: SELECTING NO. OF CLUSTERS

To select the no. of clusters, we need to find the right level to cut at (similar to in hierarchical clustering)
This requires a **quality measure** to decide which clustering is best



MODULARITY

Communities: sets of **tightly connected nodes**



Define: **Modularity Q**

- A measure of how well a network is partitioned into communities
- Given a partitioning of the network into groups $s \in S$:

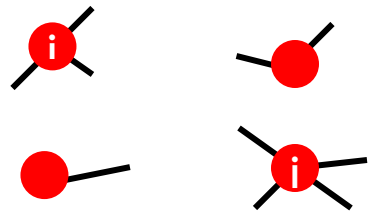
$$Q \propto \sum_{s \in S} [\ (\#\ \text{edges within group } s) - (\text{expected } \#\ \text{edges within group } s)\ ]$$

Need a null model!

NULL MODEL: CONFIGURATION MODEL

Given real G on n nodes and m edges, construct rewired network G'

- Same degree distribution but random connections
- Consider G' as a multigraph
- **The expected number of edges between nodes i and j** of degrees k_i and k_j equals to: $k_i \cdot \frac{k_j}{2m} = \frac{k_i k_j}{2m}$



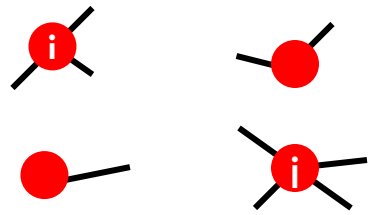
MODULARITY

Modularity of partitioning S of graph G:

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

Normalizing cost.: -1<Q<1

A_{ij} = 1 if i → j,
0 else

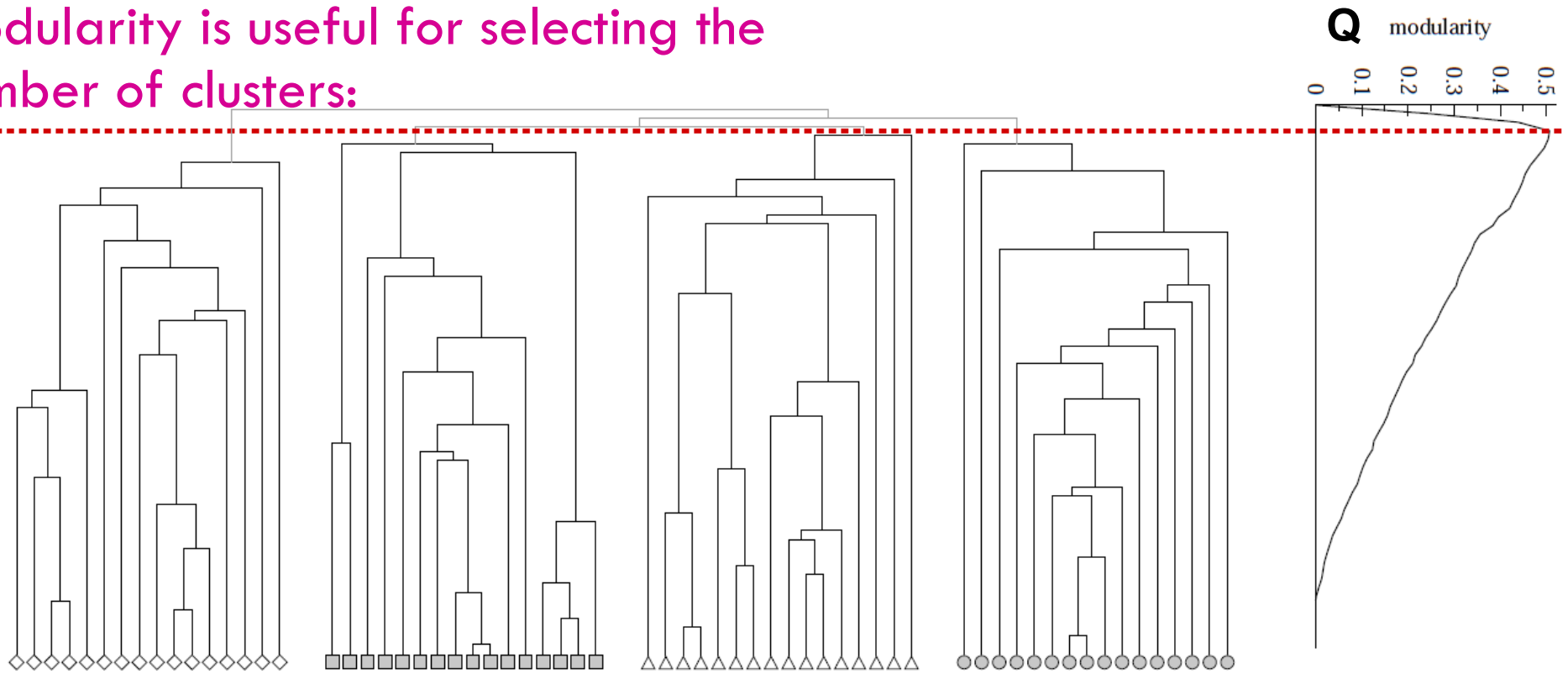


Modularity values take range [−1,1]

- It is positive if the number of edges within groups exceeds the expected number
- **0.3-0.7<Q** means significant community structure

MODULARITY: NUMBER OF CLUSTERS

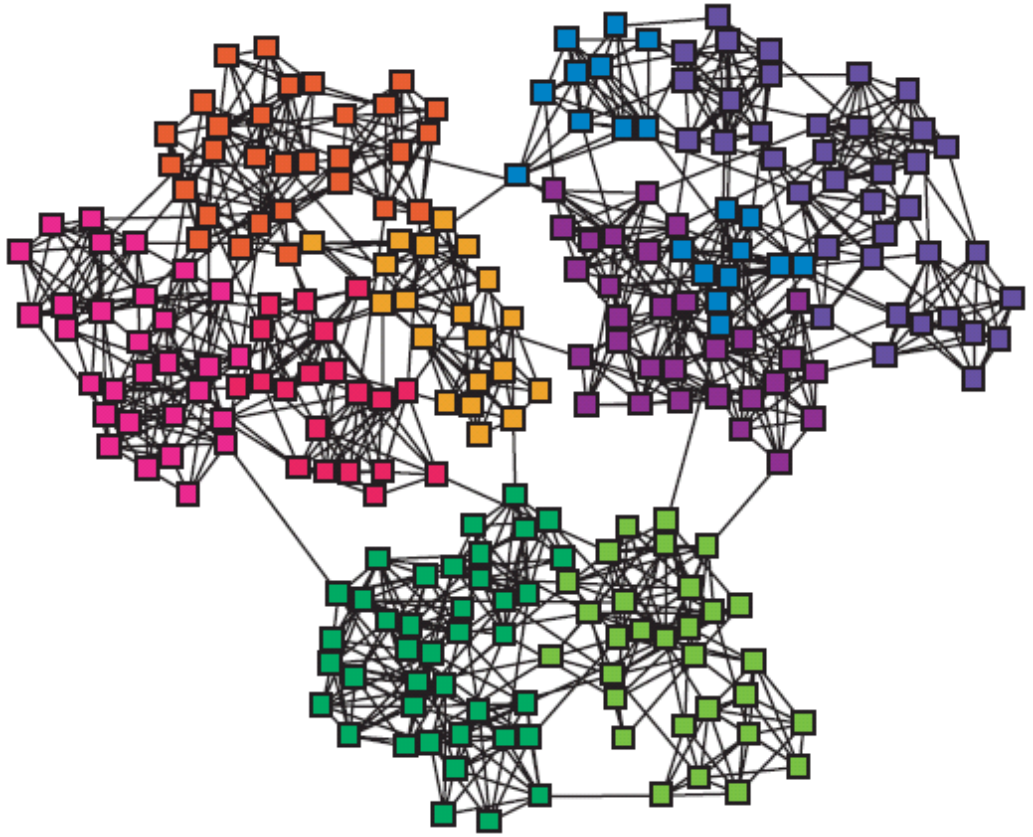
Modularity is useful for selecting the number of clusters:



Another idea: we could just optimize modularity directly!

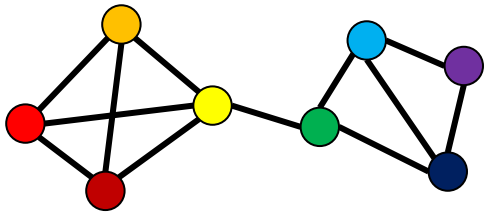
GRAPH MINING OVERVIEW

1. Introduction
2. Basic Concepts
3. Community Detection
   - Betweenness-based: Girvan Newman
   - Modularity Maximization
4. Classification Problems on Graphs



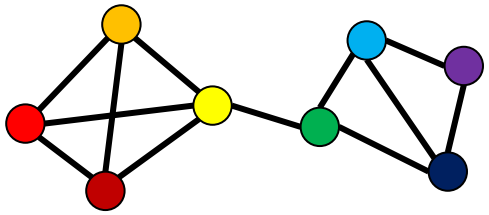
OPTIMIZING MODULARITY: LOUVAIN ALGORITHM (SKETCH)

1. Each node starts in its own community



OPTIMIZING MODULARITY: LOUVAIN ALGORITHM (SKETCH)

1. Each node starts in its own community
2. Repeat until convergence:
   - For each node i, compute the change in modularity if we move node i to each other community

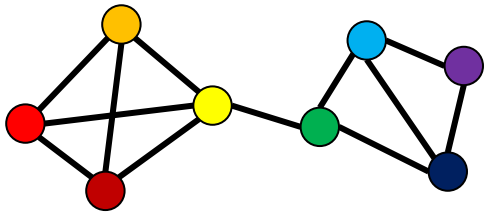


i

If moved to orange community: +0.1
If moved to blue community: -0.2
…

OPTIMIZING MODULARITY: LOUVAIN ALGORITHM (SKETCH)

1. Each node starts in its own community
2. Repeat until convergence:
   - For each node i, compute the change in modularity if we move node i to each other community
   - Move i to the community resulting in the highest modularity



i

Highest modularity

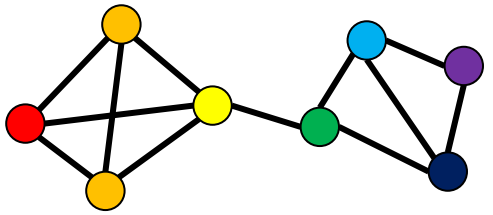
If moved to orange community: +0.1

If moved to blue community: -0.2

…

OPTIMIZING MODULARITY: LOUVAIN ALGORITHM (SKETCH)

1. Each node starts in its own community
2. Repeat until convergence:
   - For each node i, compute the change in modularity if we move node i to each other community
   - Move i to the community resulting in the highest modularity



i

Highest modularity

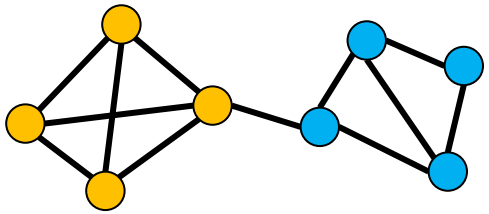
If moved to orange community: +0.1

If moved to blue community: -0.2

…

OPTIMIZING MODULARITY: LOUVAIN ALGORITHM (SKETCH)

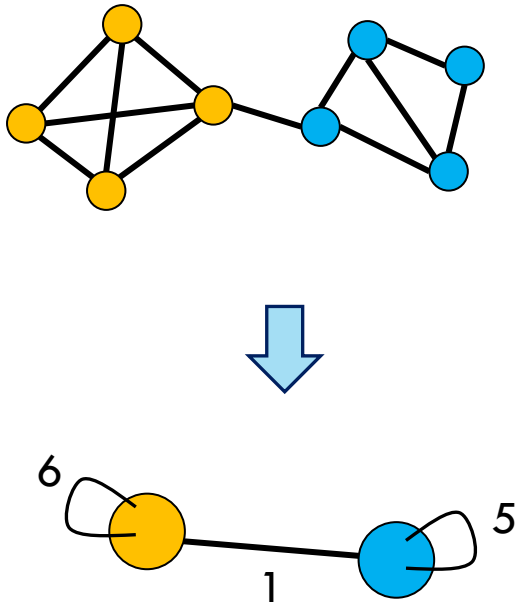
1. Each node starts in its own community
2. Repeat until convergence:
   - For each node i, compute the change in modularity if we move node i to each other community
   - Move i to the community resulting in the highest modularity



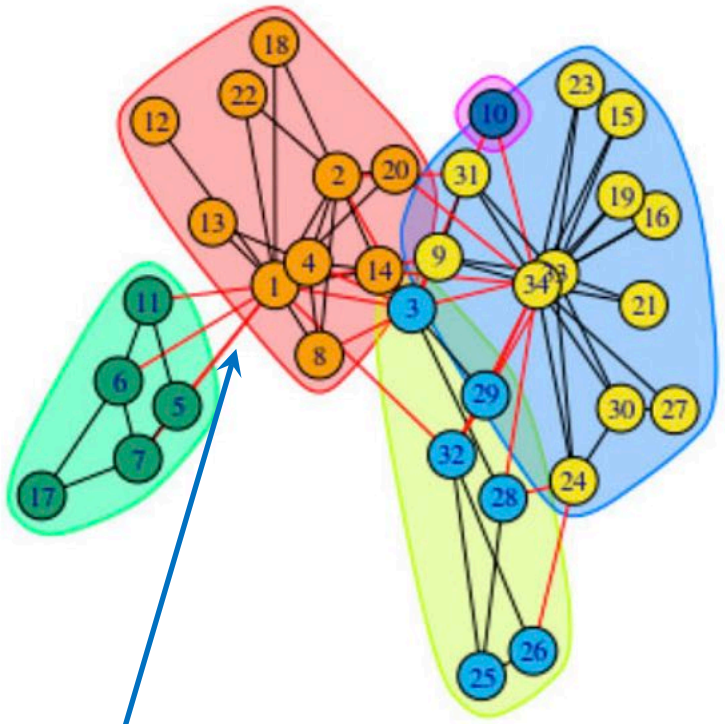
(Do the same for all other nodes)

OPTIMIZING MODULARITY: LOUVAIN ALGORITHM (SKETCH)

1. Each node starts in its own community
2. Repeat until convergence:
   - For each node i, compute the change in modularity if we move node i to each other community
   - Move i to the community resulting in the highest modularity
   - Collapse each community into a "supernode" and continue the algorithm

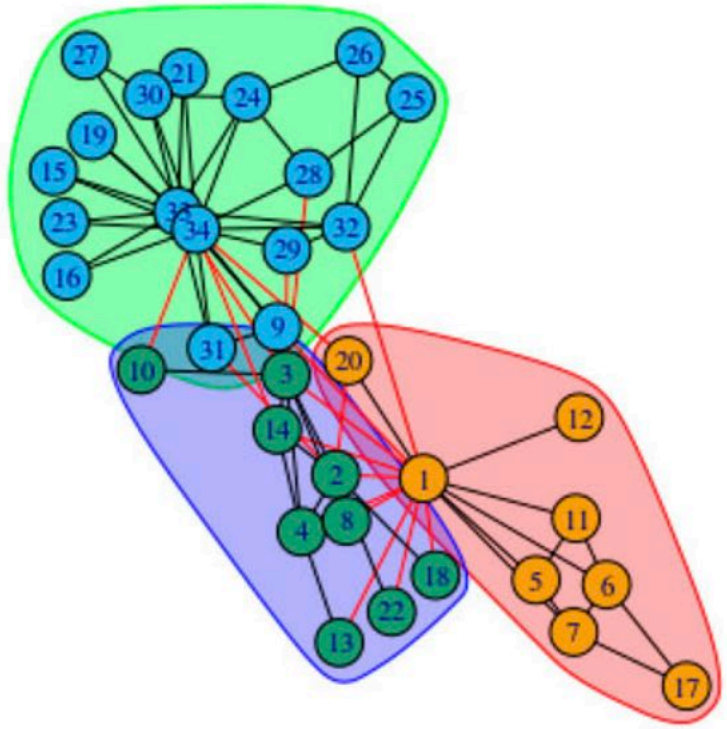


COMPARISON: GIRVAN-NEWMAN VS MODULARITY



Girvan-Newman

Prefers to "break" high betweenness edges

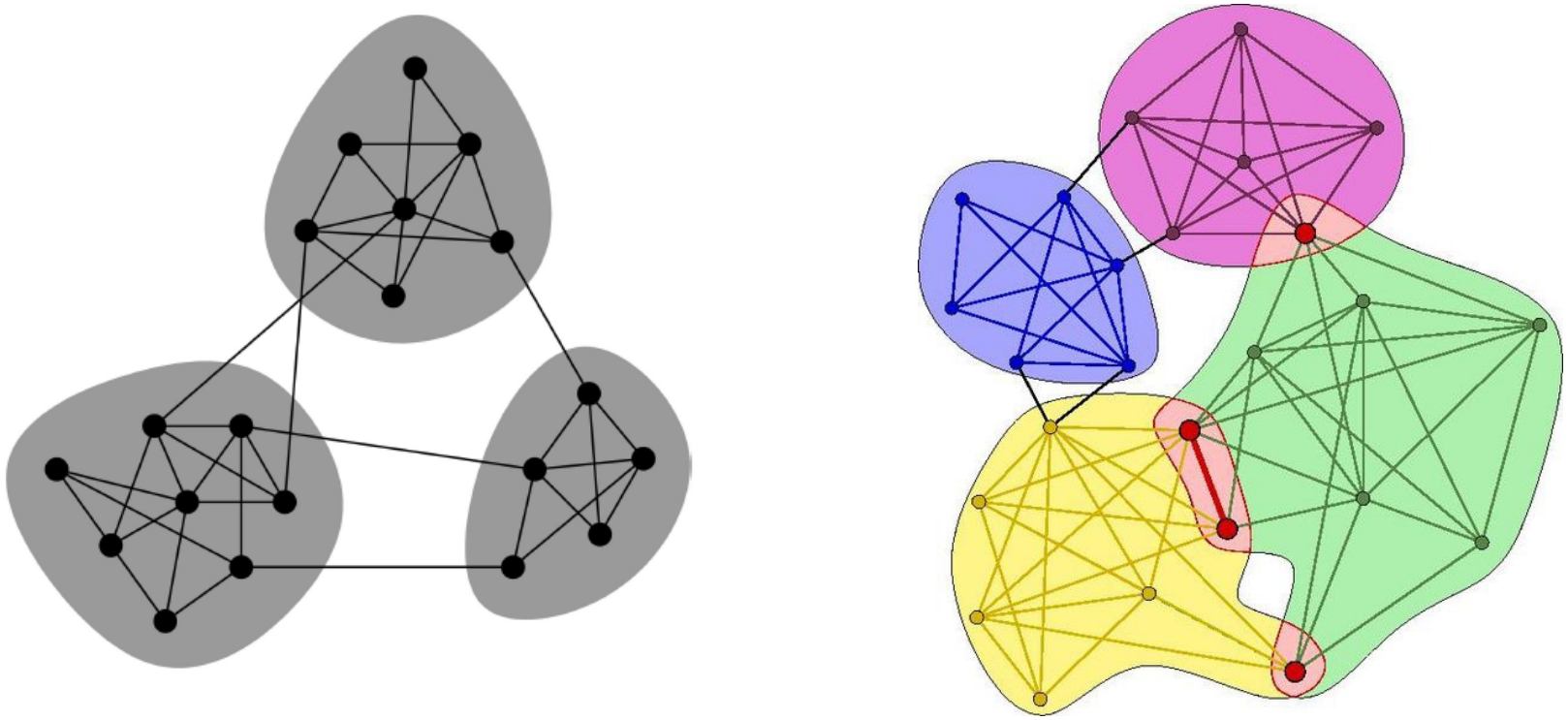


Modularity Maximization (Louvain)

Treats all edges equally

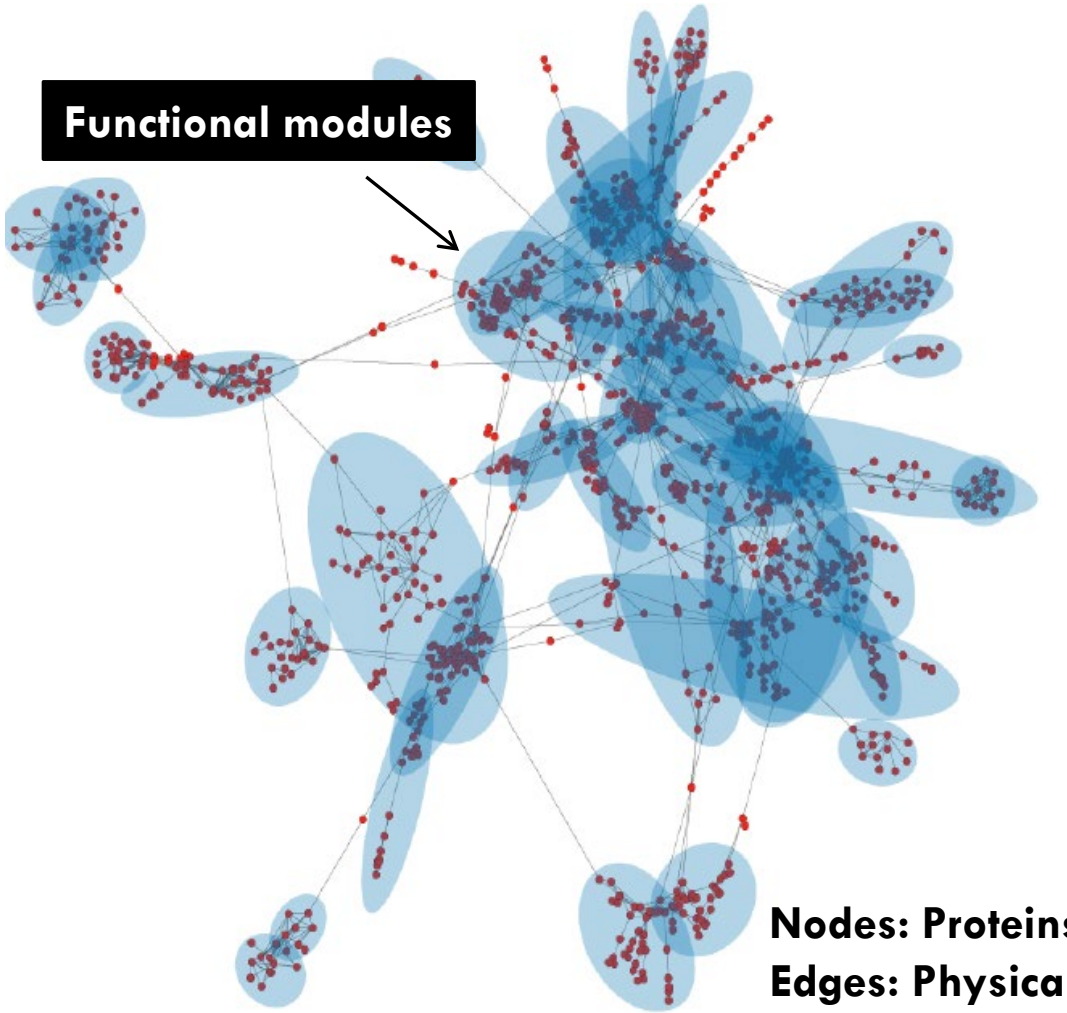
Maria Fiscarelli, Antonio, et al. "A vertex-similarity clustering algorithm for community detection." *Journal of Information and Telecommunication* 4.1 (2020): 36-50.

VARIANTS OF COMMUNITY DETECTION: OVERLAPPING COMMUNITIES



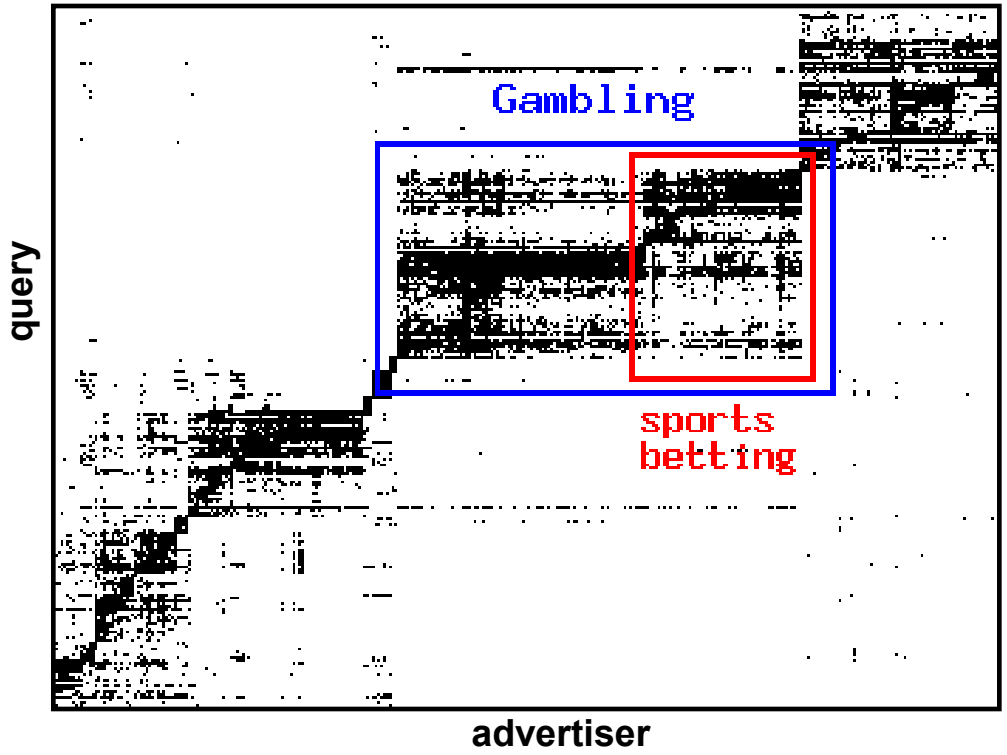
Non-overlapping vs. overlapping communities

EXAMPLE: PROTEIN-PROTEIN INTERACTIONS



Nodes: Proteins
Edges: Physical interactions

VARIANTS: BICLUSTERING (COMMUNITY DETECTION FOR BIPARTITE GRAPHS)

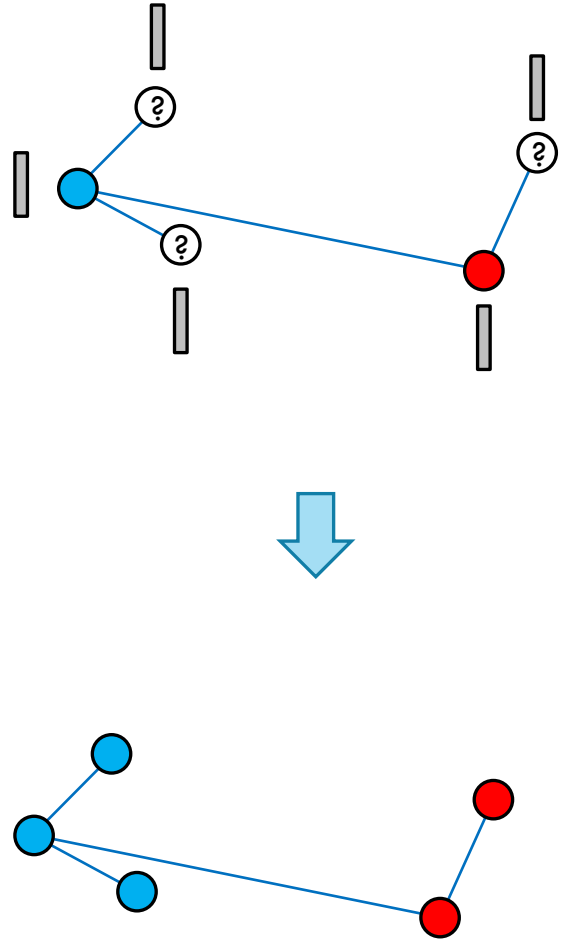


Goal: Groups the rows and columns into subgroups, such that most of the edges lie long a few "subrectangles"

[Andersen, Lang: Communities from seed sets, 2006]

GRAPH MINING OVERVIEW

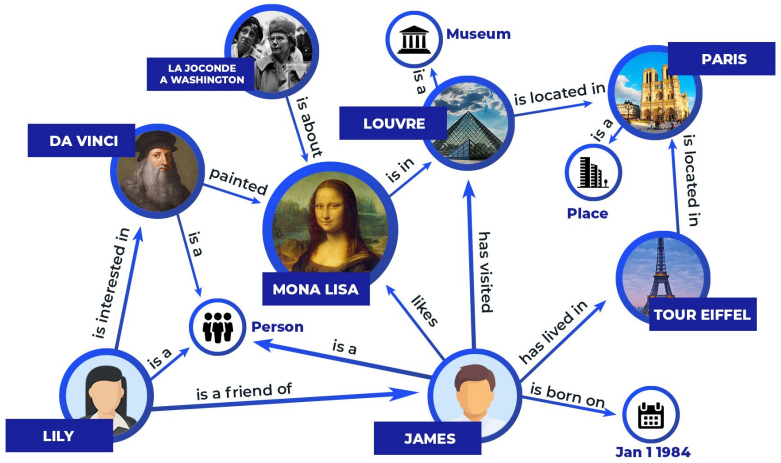
1. Introduction
2. Basic Concepts
3. Community Detection
   - Betweenness-based: Girvan Newman
   - Modularity Maximization
4. Classification Problems on Graphs

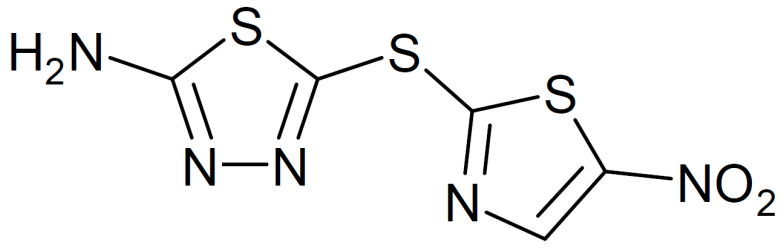


OVERALL GOAL

Many modern neural networks are designed for sequences (e.g. text, time series) and grids (e.g. images, videos)

How can we generalize them beyond sequences and grids, to graphs, which can model objects and relationships more generally?





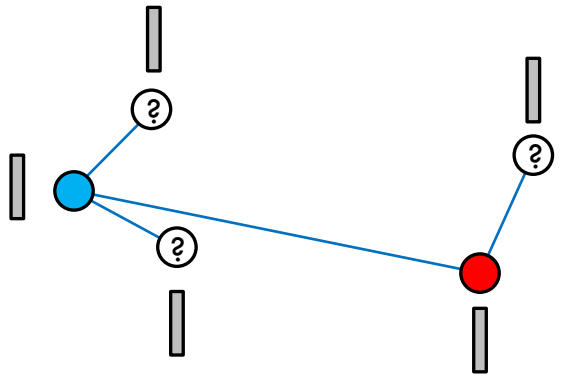
GOAL: NODE CLASSIFICATION

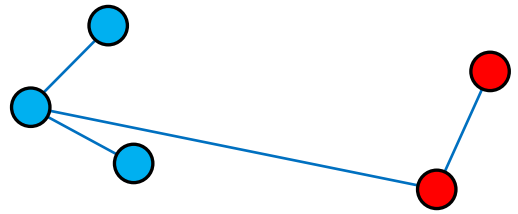
Given:

- Graph, with adjacency matrix A
- Node feature vectors x_i (for node i)
- Some labelled nodes y_i

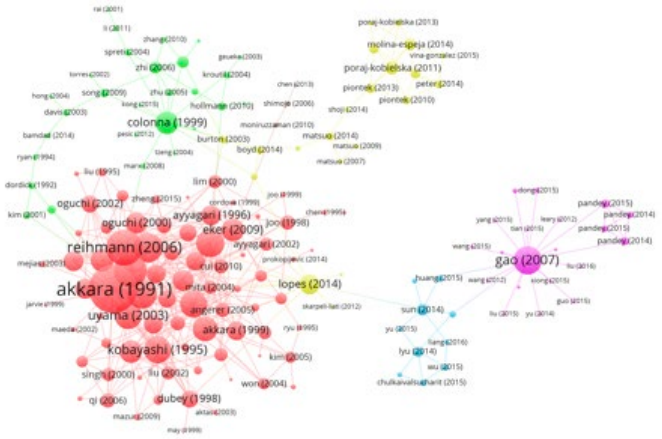
Output:

- Labels for unlabelled nodes

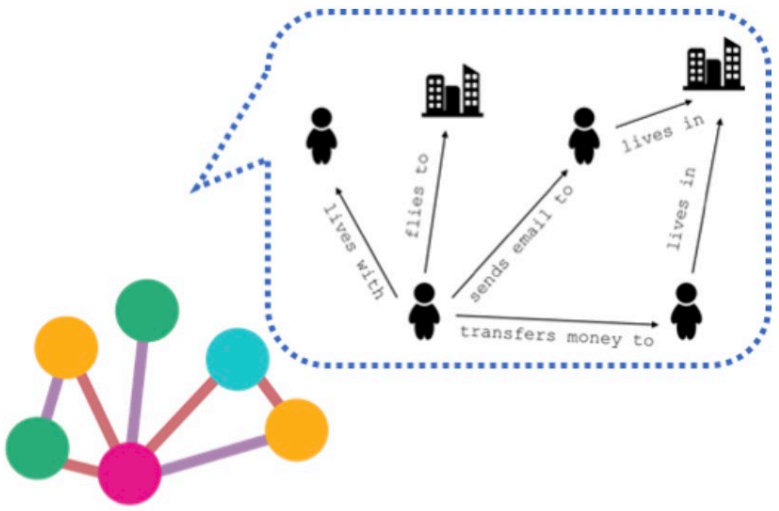




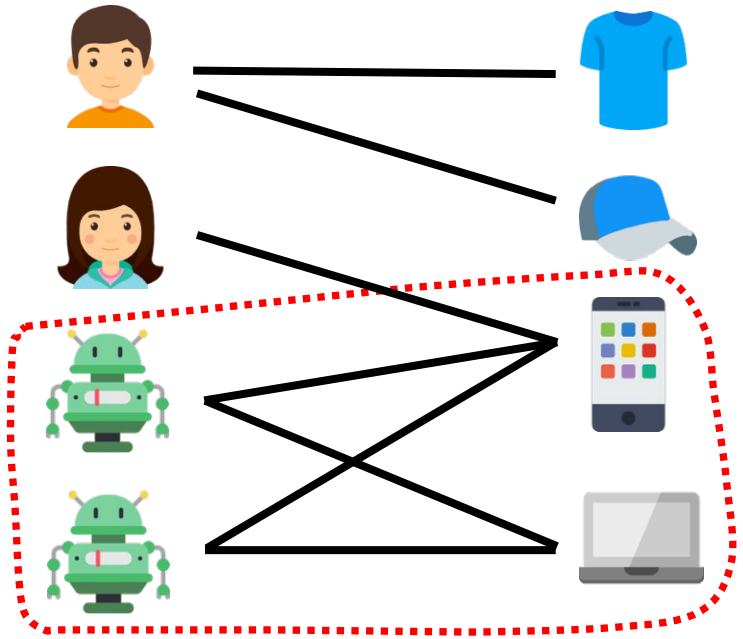
NODE PREDICTION: EXAMPLES



Citation networks



Social networks



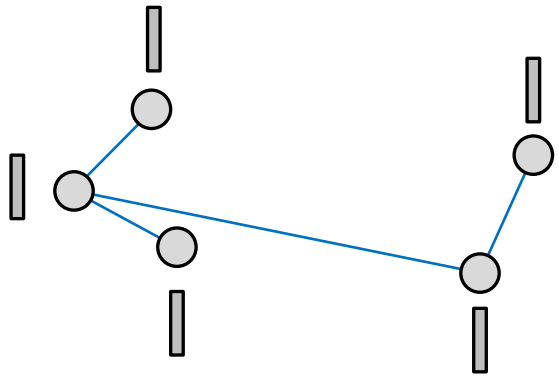
Online commerce

https://www.kdnuggets.com/2019/08/neighbours-machine-learning-graphs.html

GOAL: LINK PREDICTION

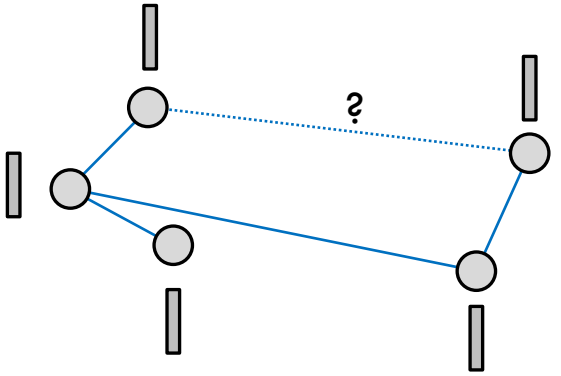
Given:

- Graph, with adjacency matrix A
- Node feature vectors x_i (for node i)

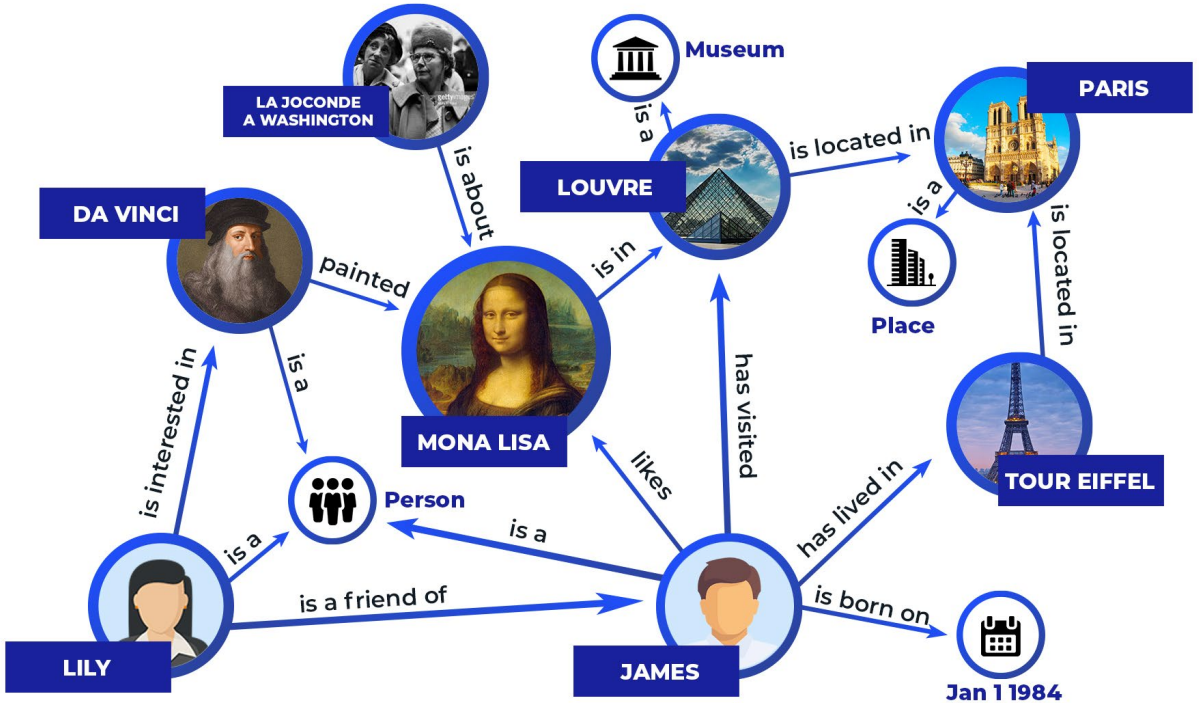


Output:

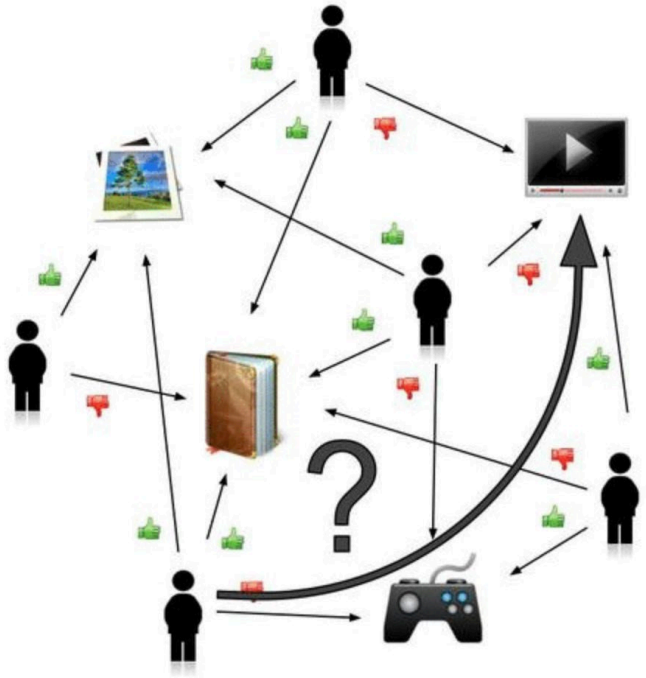
- Predict whether a given link will be created in future



LINK PREDICTION: APPLICATIONS



Knowledge graph completion



Recommender systems

https://yashuseth.blog/2019/10/08/introduction-question-answering-knowledge-graphs-kgqa/
https://commons.wikimedia.org

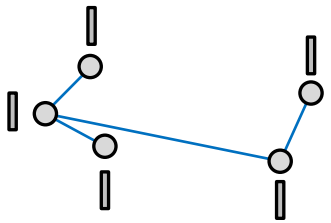
GOAL: GRAPH PREDICTION

Given:

- Graphs, with adjacency matrices A_i
- Labels for graphs, y_i

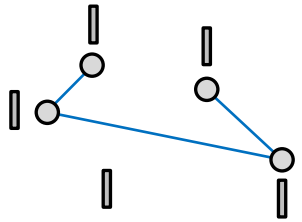
Output:

- Predict label for unlabeled graphs

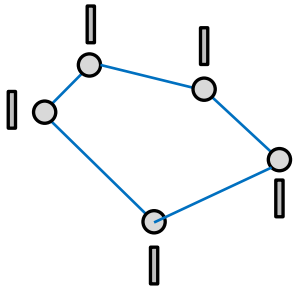


Label

0



1

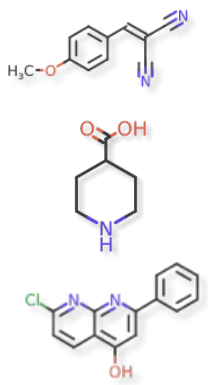


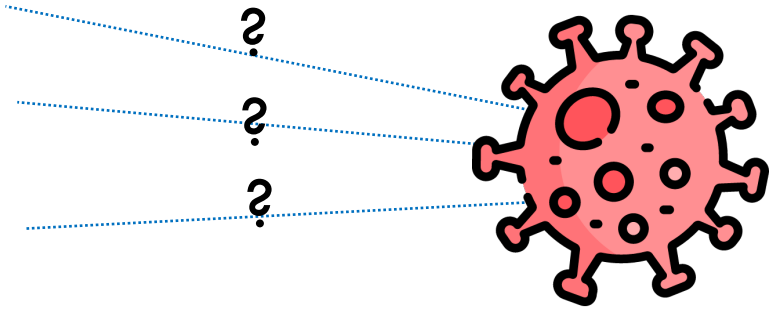
?

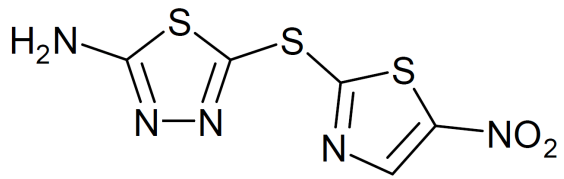
GRAPH PREDICTION EXAMPLE: DRUG REPURPOSING

Given a large database of drugs, we can predict which will interact with proteins associated with a given disease

Database of drug molecules ("Drug Repurposing Hub")







Halicin

Early successes: e.g. [1] reportedly discovered a new type of antibiotic ("halicin") based on this approach which was effective against a "wide spectrum of pathogens" in tests on mice, including "pan-resistant" strains for which antibiotics are urgently required

Stokes, Jonathan M., et al. "A deep learning approach to antibiotic discovery." *Cell* 180.4 (2020): 688-702.