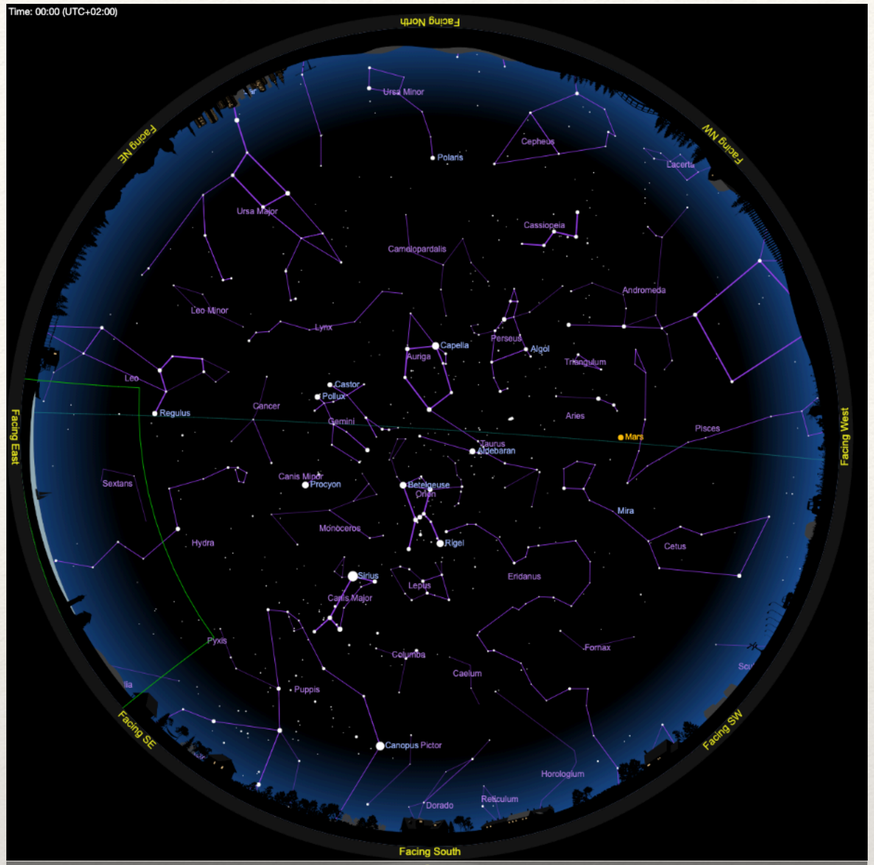


Complex Networks Analysis

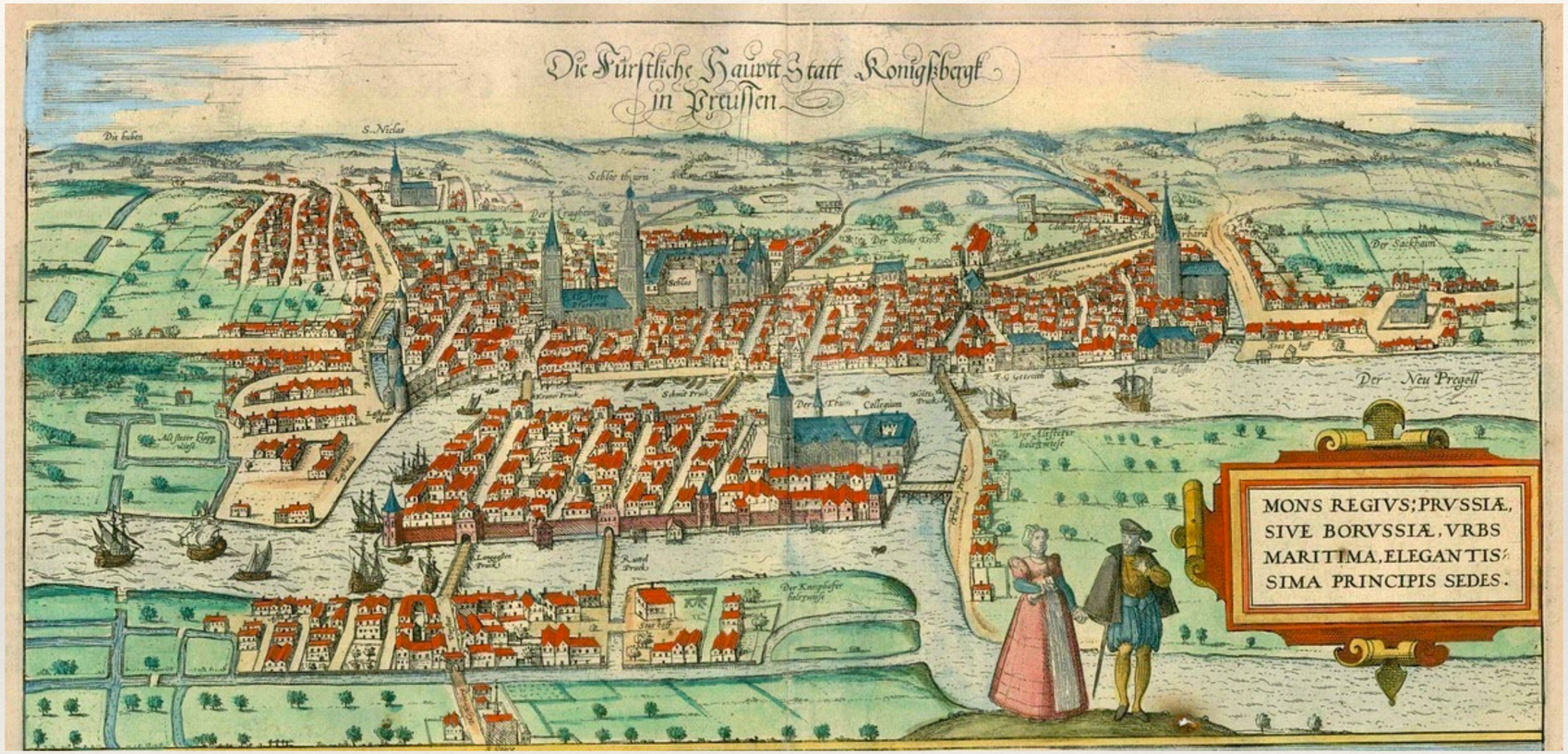
Segev Arbiv

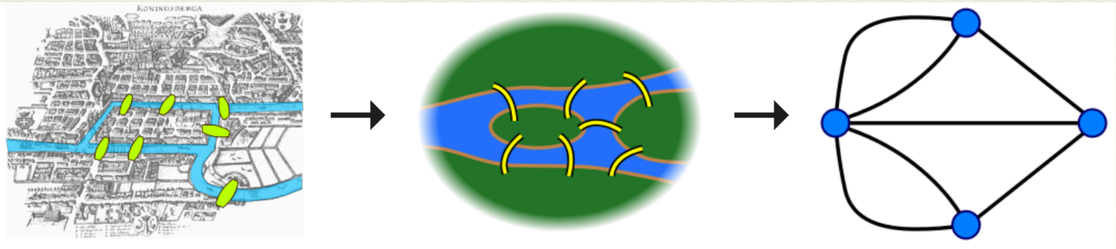
segevarbiv@gmail.com

Agenda

- Introduction
  - Applications - graphs around us
  - Basic definitions and statistical characterization
- Centrality measurements
- Community detection
- Link prediction - real life application
- Intro to Graph Representation Learning

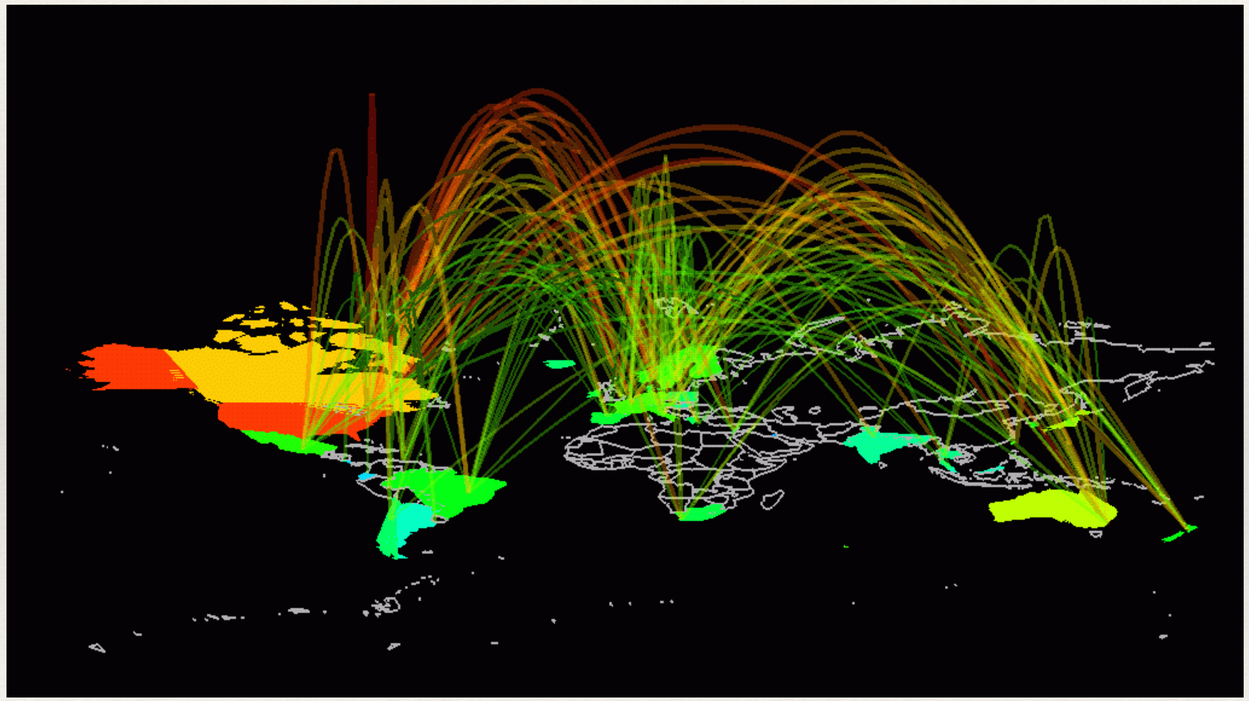
Intro

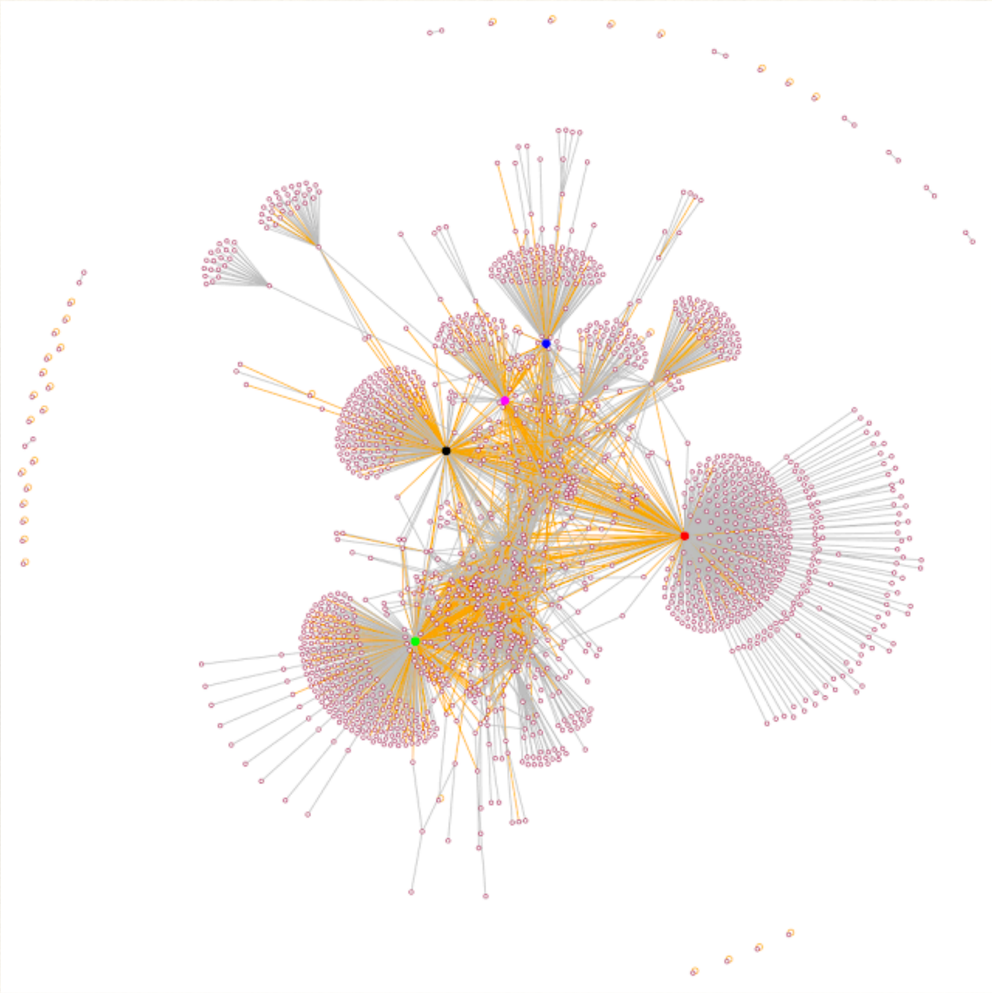




Segev Arbiv

Applications
Communication



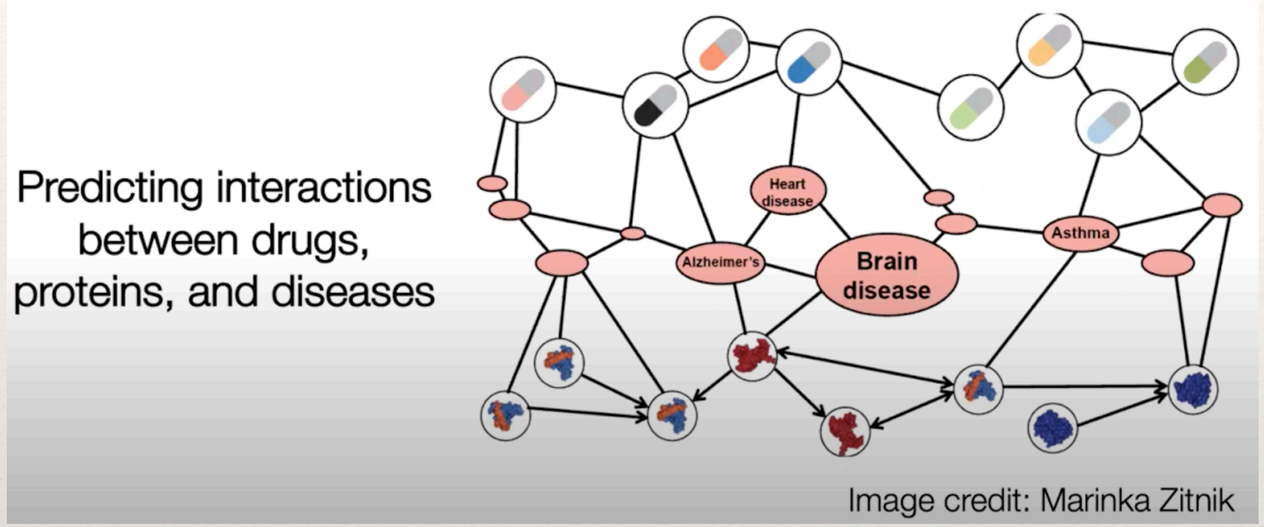


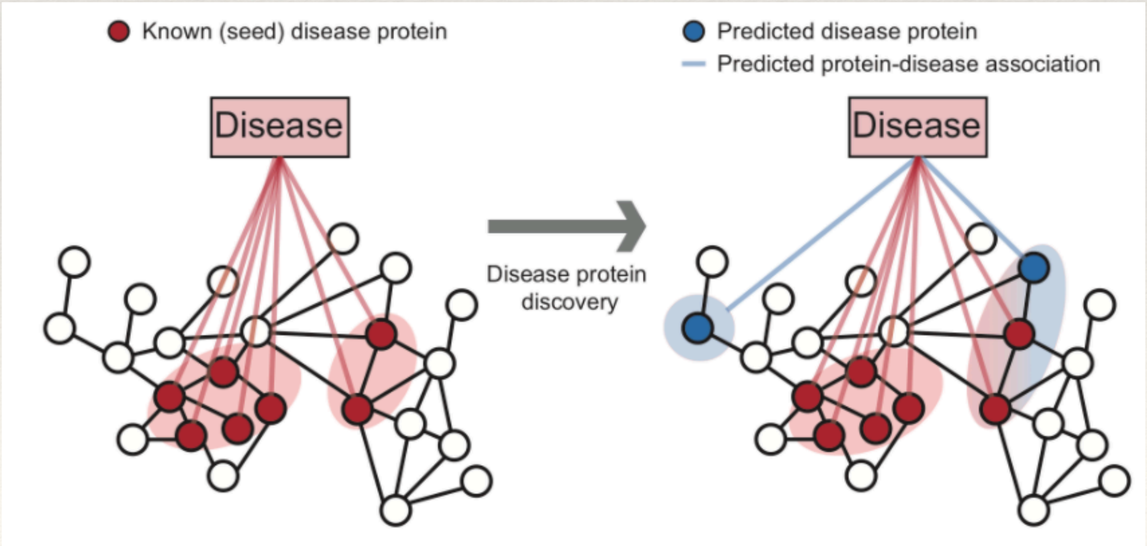
DNC leaked emails

Applications
Transportation

http://www.martingrandjean.ch/connected-world-air-traffic-network/

Applications
Bio-Medical

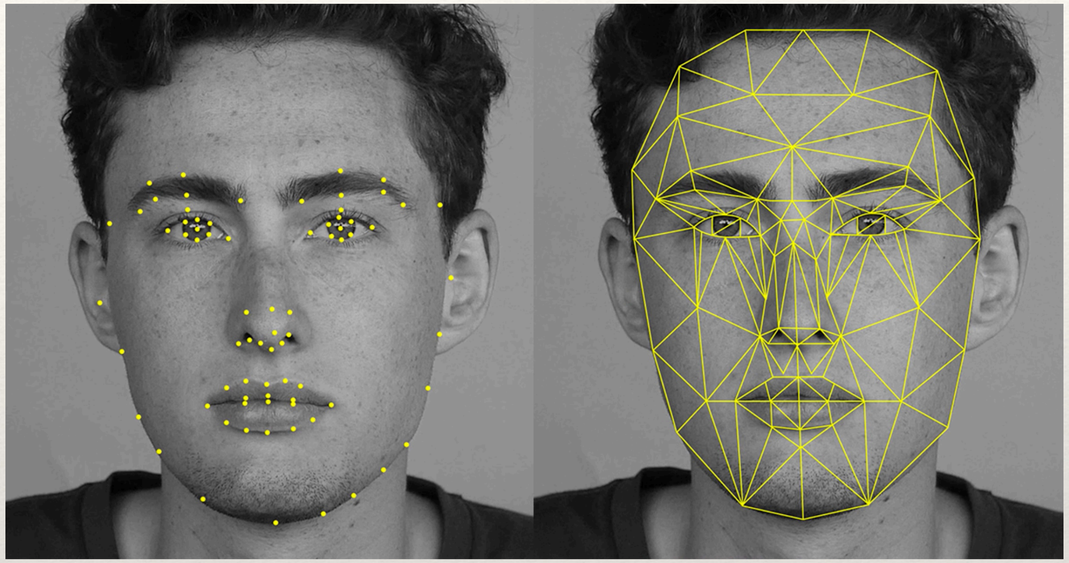




Large-scale analysis of disease pathways

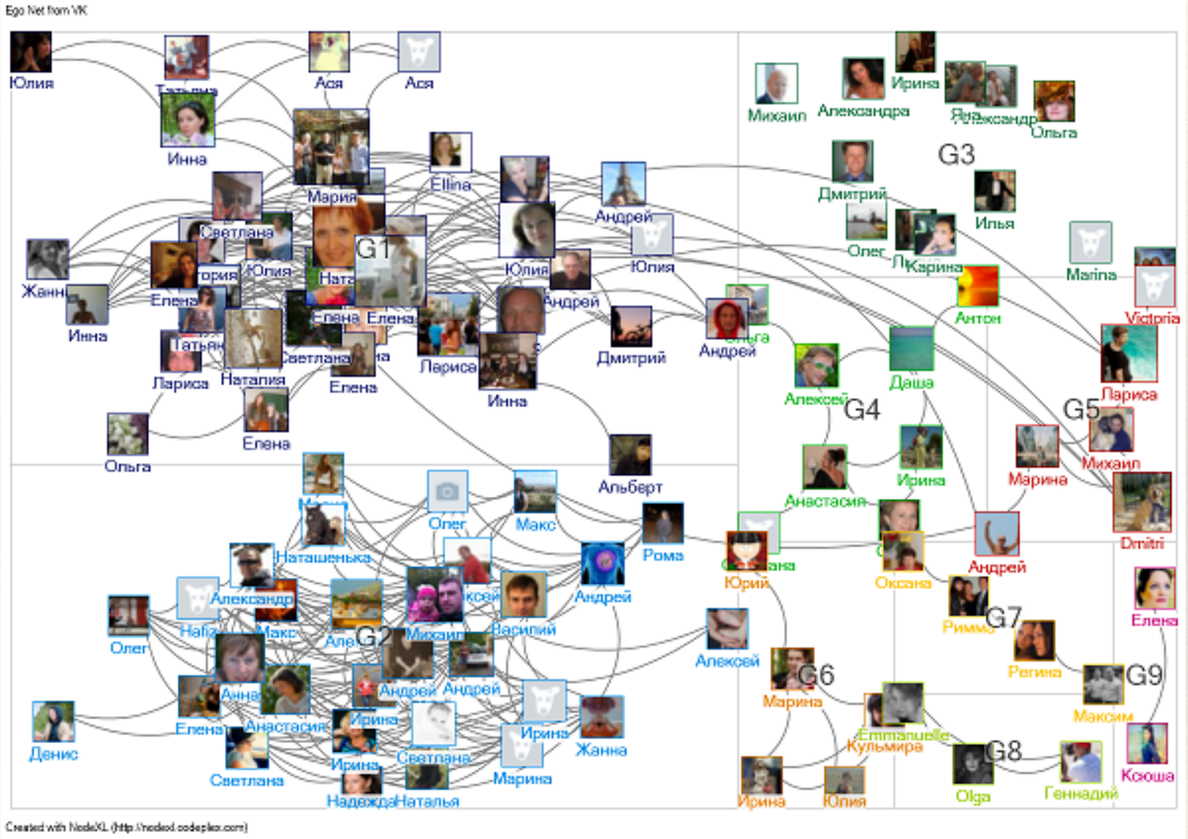
Applications
Image Processing

AlphaPose

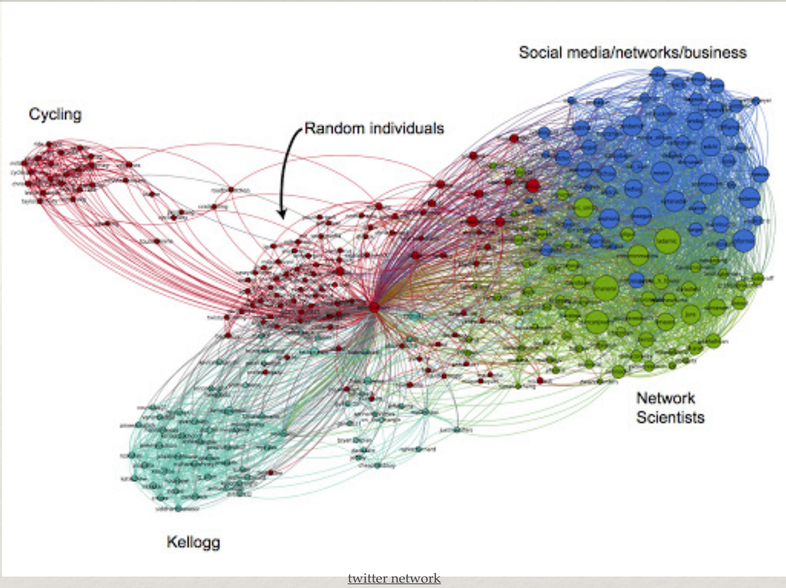


https://petapixel.com/

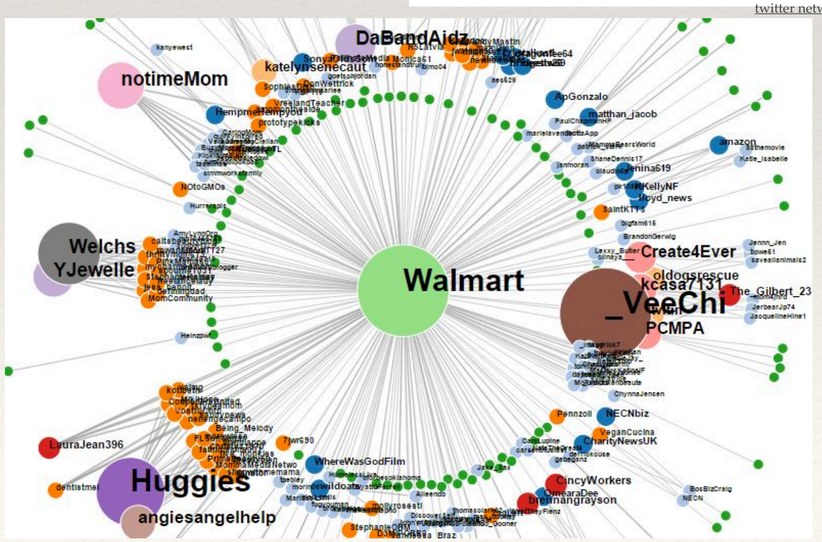
Applications
Social



VK



twitter network



https://www.datasciencecentral.com/profiles/blogs/visualize-your-social-media-analytics

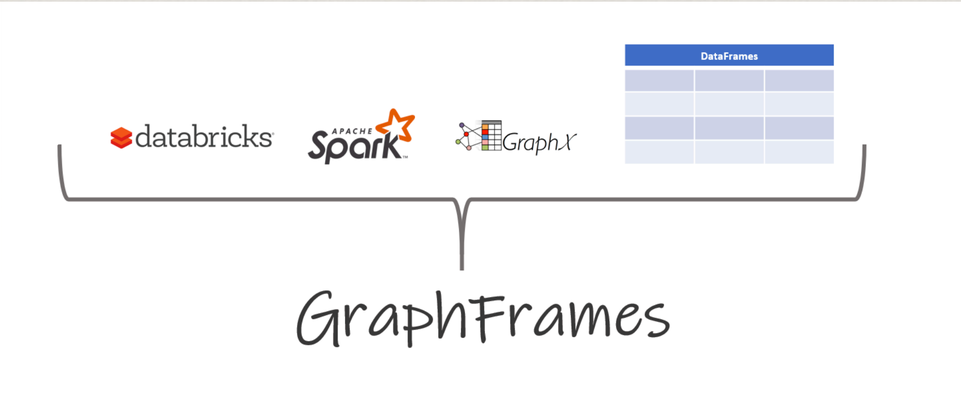
Code-wise

- To address these various applications, there are many graph libraries:

NetworkX
Network Analysis in Python

neo4j

SNAP



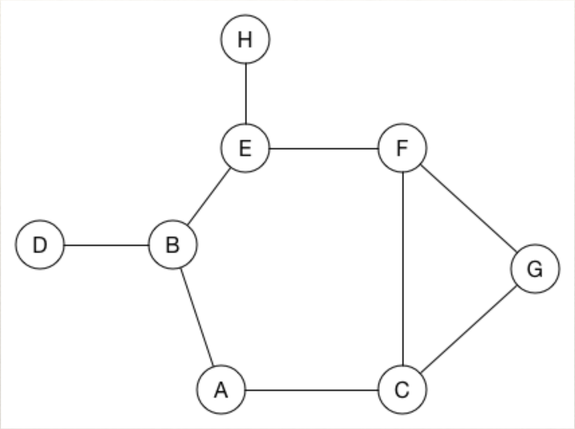
Basic graph definitions

- A graph $G = (V, E)$ is a set of V vertices and E edges
- $e_{ij} \in E$ — an edge between vertices v_i and v_j
  - or, alternatively, (i, j)
- A graph can be either undirected ($e_{ij} \iff e_{ji}$) or directed
- An edge can be associated with a weight $w(e)$
  - In most of the use cases, assume all weights equal to 1

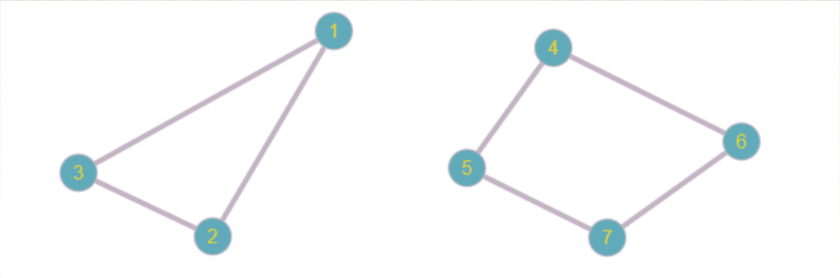
Basic graph definitions

- A path is a sequence of vertices with edges between consecutive ones

$$\{v_1, v_2, \cdots, v_k\} \,|\, \forall i \in [1, k-1] : e_{i,i+1} \in E$$



  - Not unique
  - *shortest path* — minimum #edges (*geodesic path*)
  - Connected graph — there exist a path between any two nodes in G.



Segev Arbiv

Basic graph definitions

- We can define the node neighbours

$$\Gamma(u) = \{v \mid e_{uv} \in E\}$$

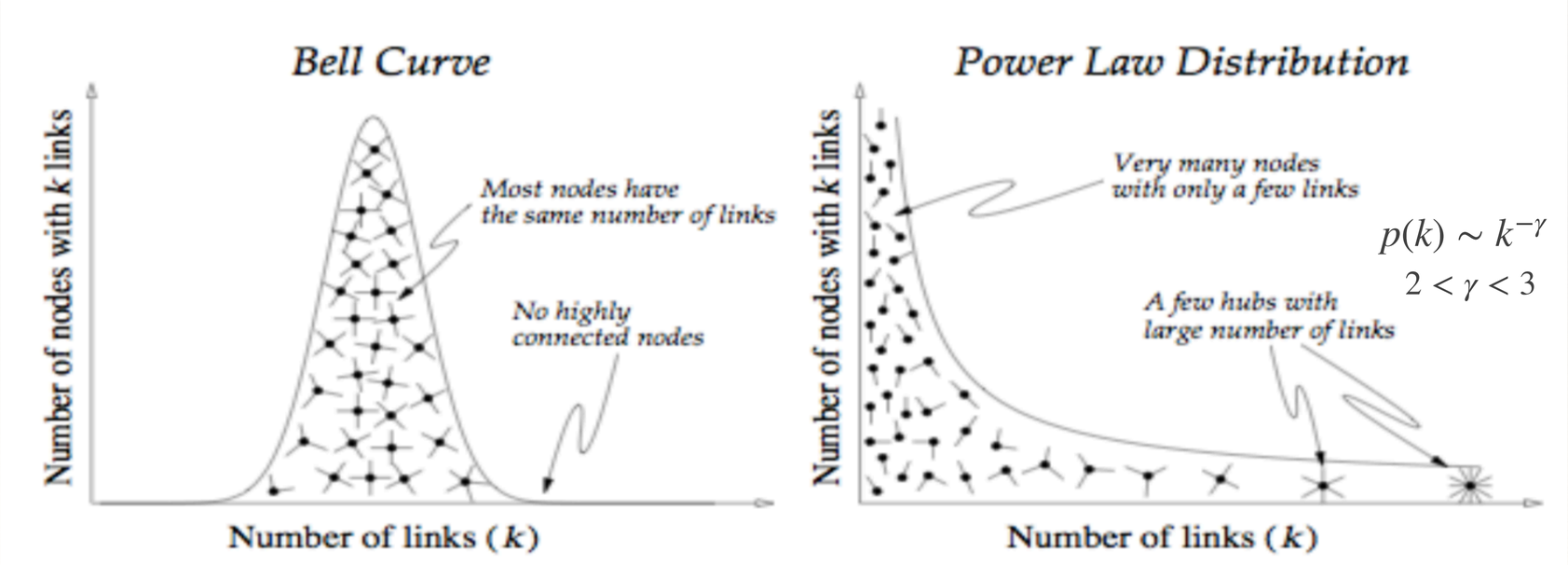
- And the node degree*

$$D(v) \triangleq k_v = \left|\Gamma(v)\right|$$

Basic graph definitions
Degree distribution

Network	Size	$\langle k \rangle$
WWW	4×10^7	7
Movie actors*	212, 250	28.78
Coauthors, math*	70, 975	3.9
Protein, S. cerev.*	1870	2.39

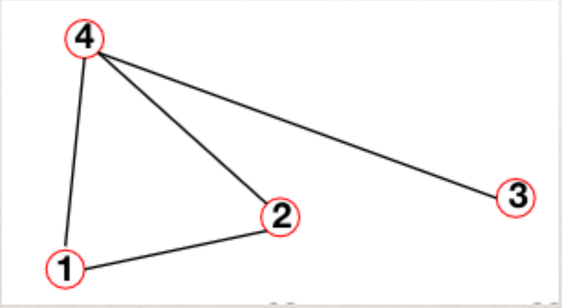
Albert, Barabasi



Diameter of the WWW

Basic graph definitions

- The most common way to represent a graph: the *Adjacency Matrix* $A \in R^{V \times V}$.
  $A_{ij} = 1 \iff e_{ij} \in E$

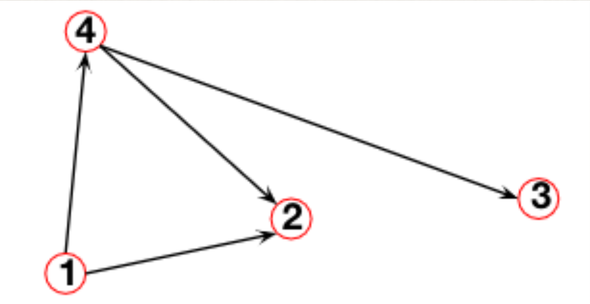


Undirected graph

$$A_{ij} = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

$$A_{ij} = A_{ji} \quad A_{ii} = 0$$

$$k_i = \sum_j A_{ij}$$



Directed graph

$$A_{ij} = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

$$A_{ij} \neq A_{ji} \quad A_{ii} = 0$$

$$k_i^{out} = \sum_j A_{ij}$$

$$k_i^{in} = \sum_j A_{ji}$$

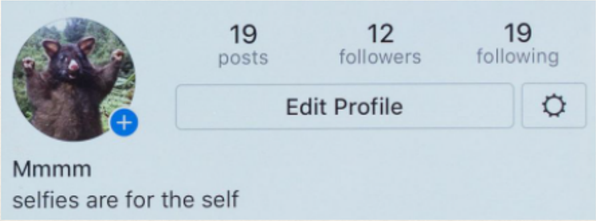
Segev Arbiv

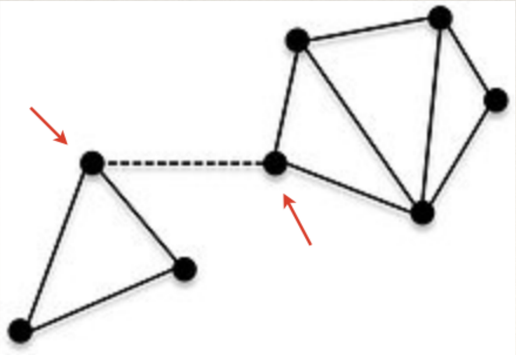
Centrality measurements

Segev Arbiv

Centrality

- How can we quantify the *importance* of a node?
  - #connections (in/out - followers/follows)
  - Central node is one that has central neighbours
  - Removing a node disconnect the graph
  - High number of paths passing through the node (Google, Ad Networks)
  - etc.



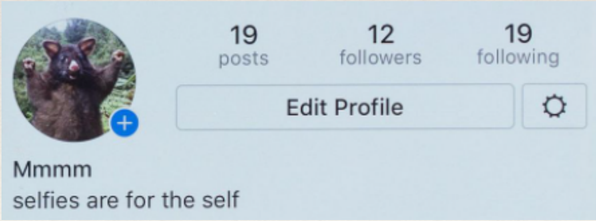


Centrality
Degree connectivity

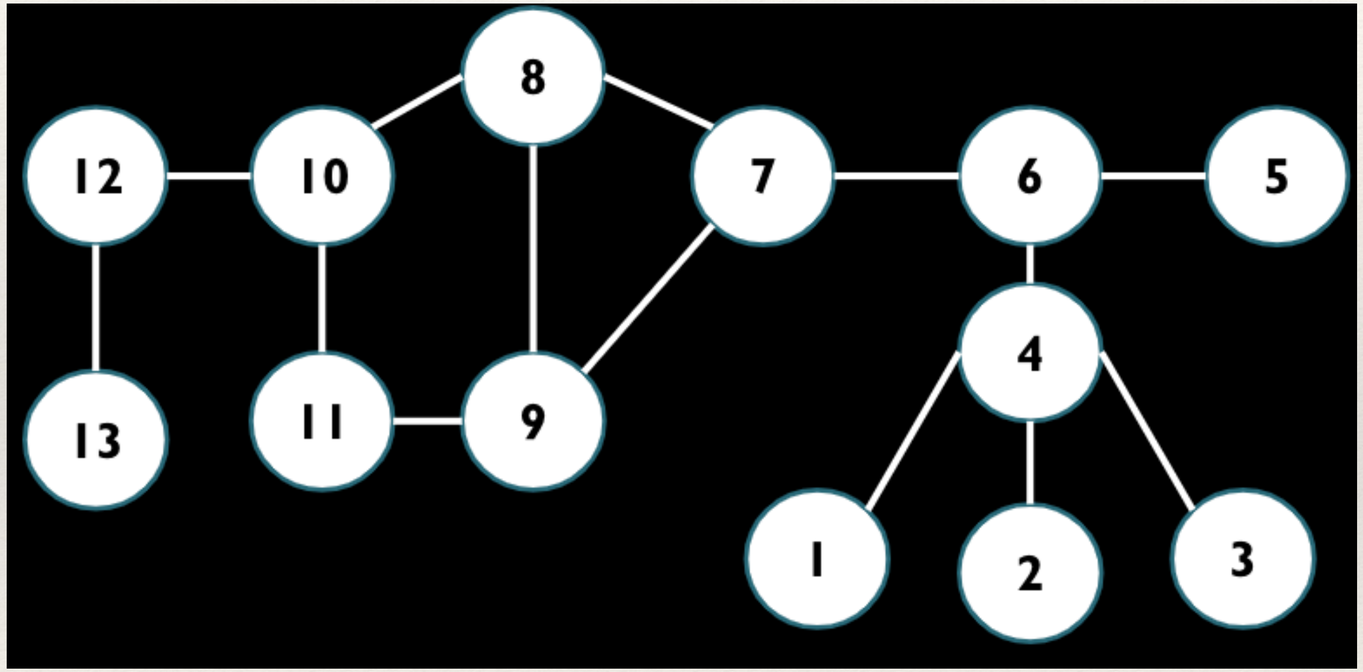
- Degree connectivity defined as the number of neighbours a node has

$$D(v) \triangleq k_v = \left|\Gamma(v)\right|$$

- Alternative definitions consider also:
  - The strength (weight) of v's connections
  - The direction of v's connections



Centrality
Degree connectivity



Node	Degree
4	4
6	3
7	3
8	3
9	3
10	3
11	2
12	2

networkx.algorithms.centrality.degree_centrality(G)

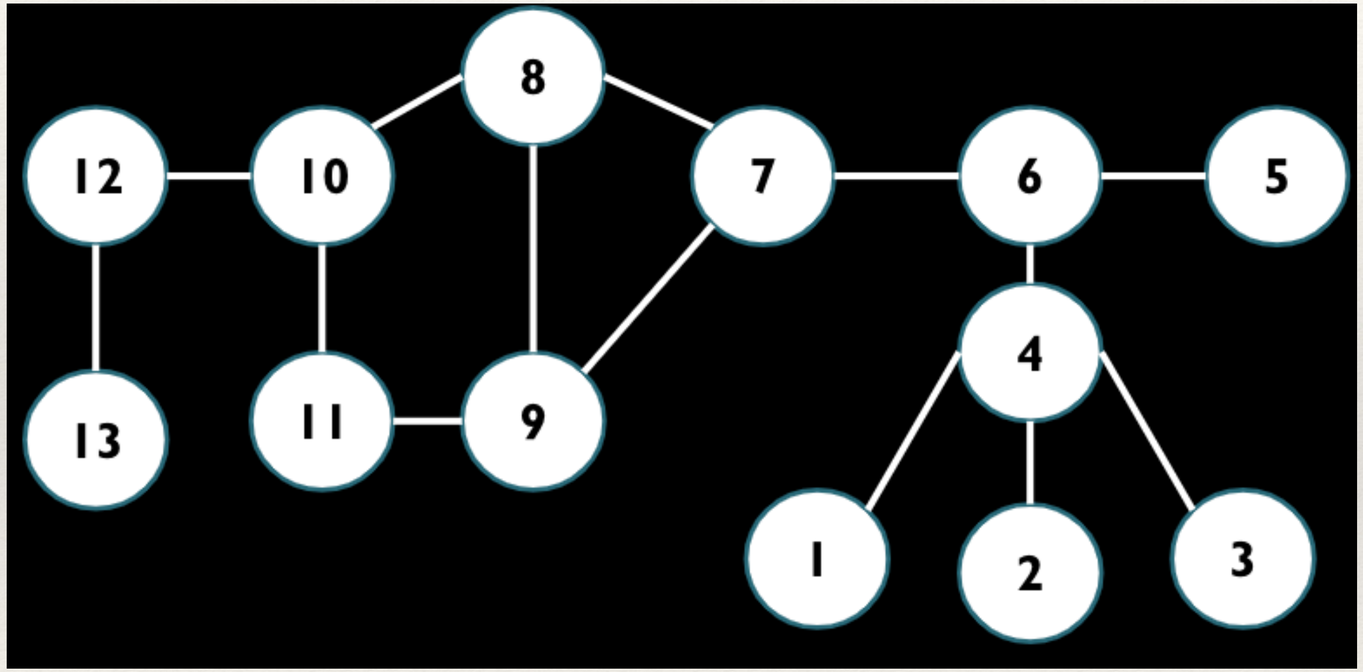
Centrality
Closeness

- Farther vertices are “harder” to reach
- uses the geodesic distance (#edges) between 2 vertices

$$C(v) = \frac{n-1}{\sum_u dist(u,v)}$$

- Only defined for connected graphs
- Normalization (optional) allow us to compare metric between graphs

Centrality
Closeness



Node	Degree	1/Closeness
4	4	33
6	3	28
7	3	27
8	3	30
9	3	32
10	3	35
11	2	…
12	2	…

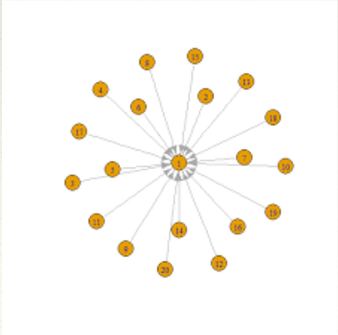
networkx.algorithms.centrality.closeness_centrality(G)

Centrality
Betweenness

- Measures the extent to which a node lies between all others in the network - its ability to "control" the information passed between parts of graph.

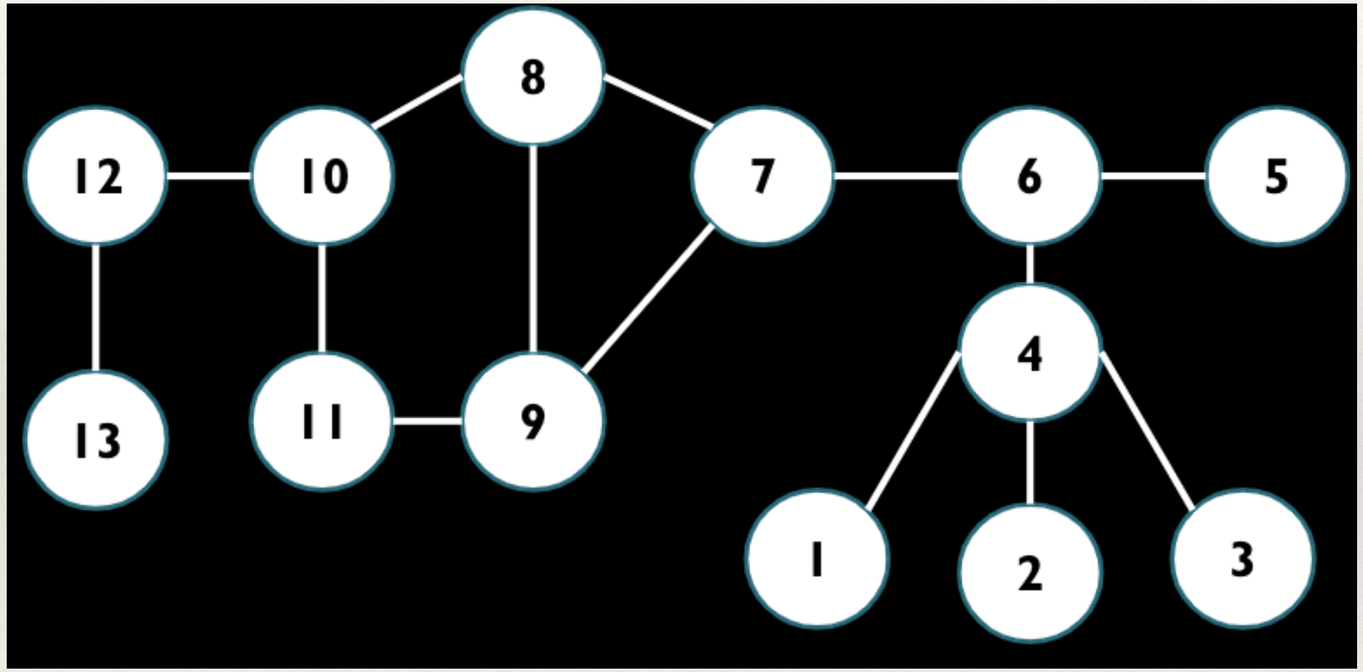
$$B(v) = \sum_{u,w \in V - \{v\}} \frac{\sigma_{uw}(v)}{\sigma_{uw}}$$

  - σ_{uw} - the number of shortest paths between u and w.
  - $\sigma_{uw}(v)$ - the number of shortest paths between u and w, that pass through v.
- How to compare to other graphs?
  - Normalize by $\frac{(n-1)(n-2)}{2}$ (Freeman)



Segev Arbiv

Centrality
Betweenness



Node	Degre	1/Closeness	Betweenness
4	4	33	60
6	3	28	78
7	3	27	72
8	3	30	43
9	3	32	15
10	3	35	41
11	2	…	…
12	2	…	…

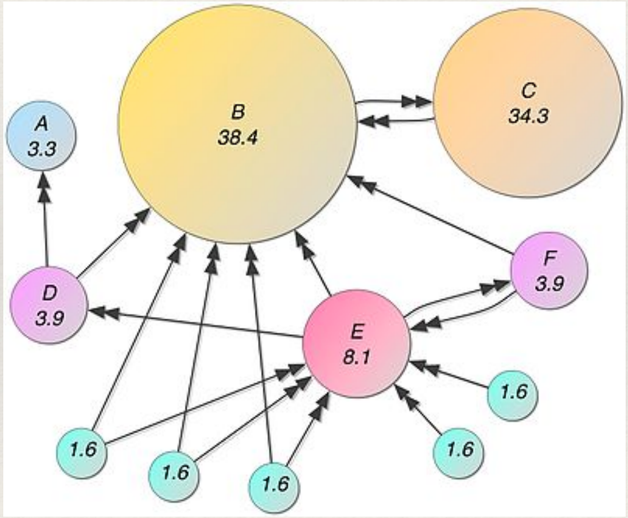
networkx.algorithms.centrality.betweenness_centrality(G)

Centrality
Example - simplified PageRank (Brin, Page)

- The way Google rank(ed) their Search Engine Results Page (SERP)
  - Now incorporates many additional various contexts
- Relies mainly on the *backlinks* to a website.

$$PR(v) = \sum_{u \in \Gamma(v)} \frac{PR(u)}{L(u)}$$

  - Where $L(u)$ is the #outgoing links (edges) from u.
- Iterative process

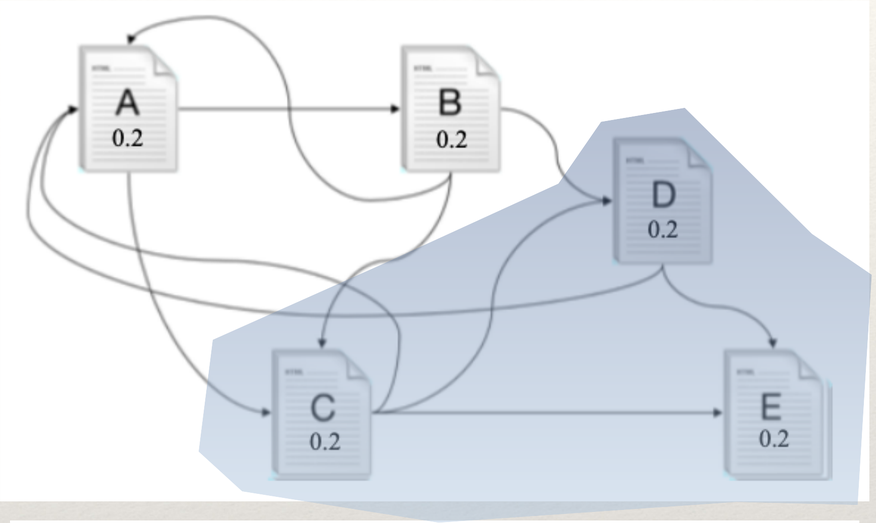


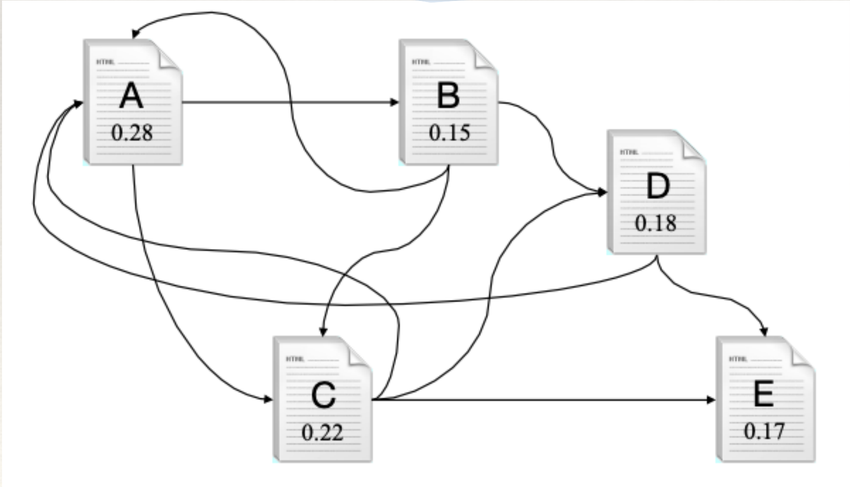
Centrality
Example - simplified PageRank (Brin, Page)

- PageRanks are initialized uniformly.
  - In the current example, page *E* has 2 incomings links: one from *C*, one from *D*.
  - Page *C* contributes 1/3 of its PR to *E*, as it has 3 outgoing links, from which 1 is to *E*.
    Similarly, *D* offers 1/2 of its PR to *E*.
  - Hence, the new PR(E) is:

$$PR(E) = \frac{PR(C)}{3} + \frac{PR(D)}{2} = \frac{0.2}{3} + \frac{0.2}{2} \approx 0.17$$

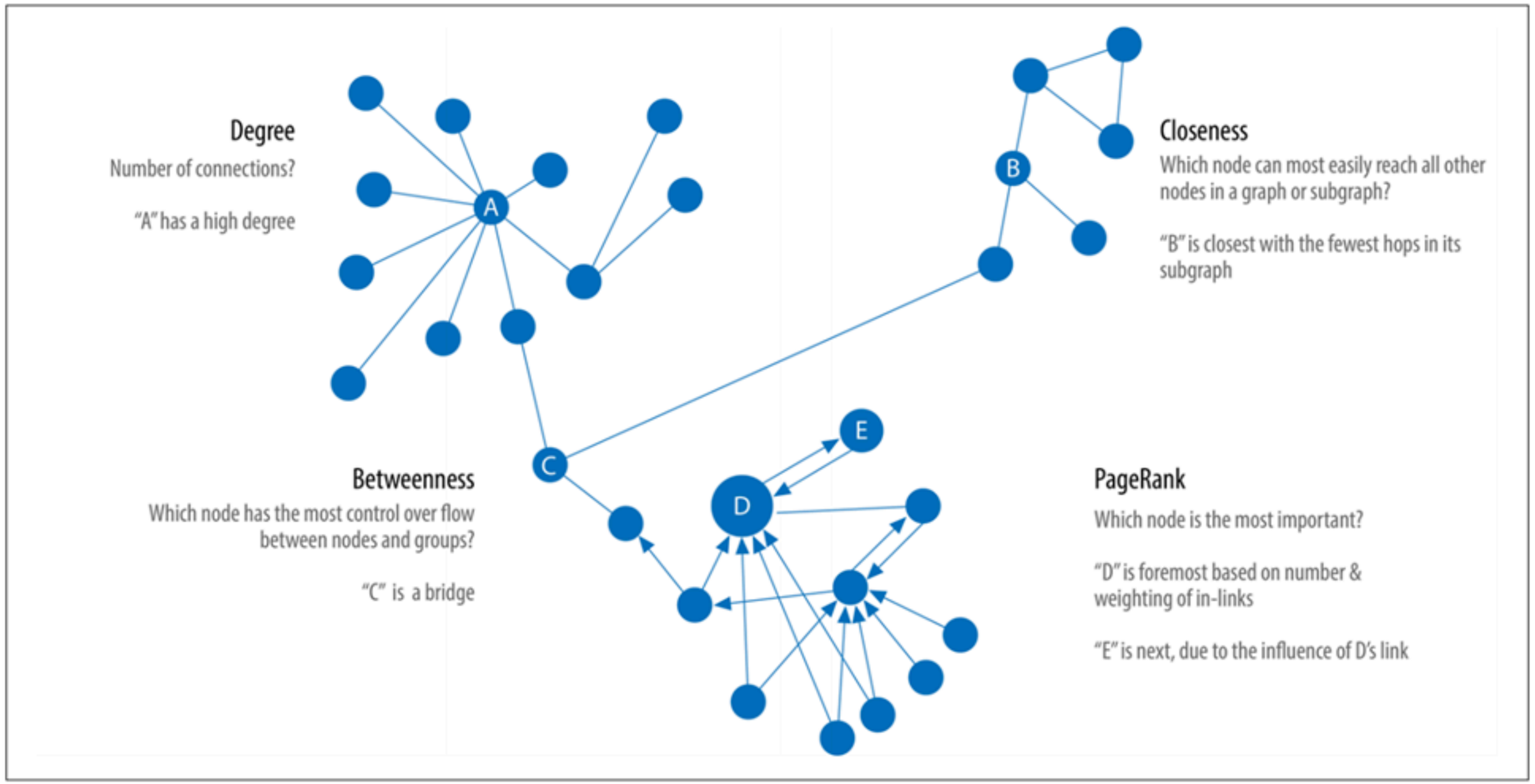
- Repeat the process until the PR's stabilises*





https://web.stanford.edu/class/cs54n/handouts/24-GooglePageRankAlgorithm.pdf

Centrality


Degree
Number of connections?
"A" has a high degree
A
Closeness
Which node can most easily reach all other nodes in a graph or subgraph?
"B" is closest with the fewest hops in its subgraph
B
Betweenness
Which node has the most control over flow between nodes and groups?
"C" is a bridge
C
E
D
PageRank
Which node is the most important?
"D" is foremost based on number & weighting of in-links
"E" is next, due to the influence of D's link

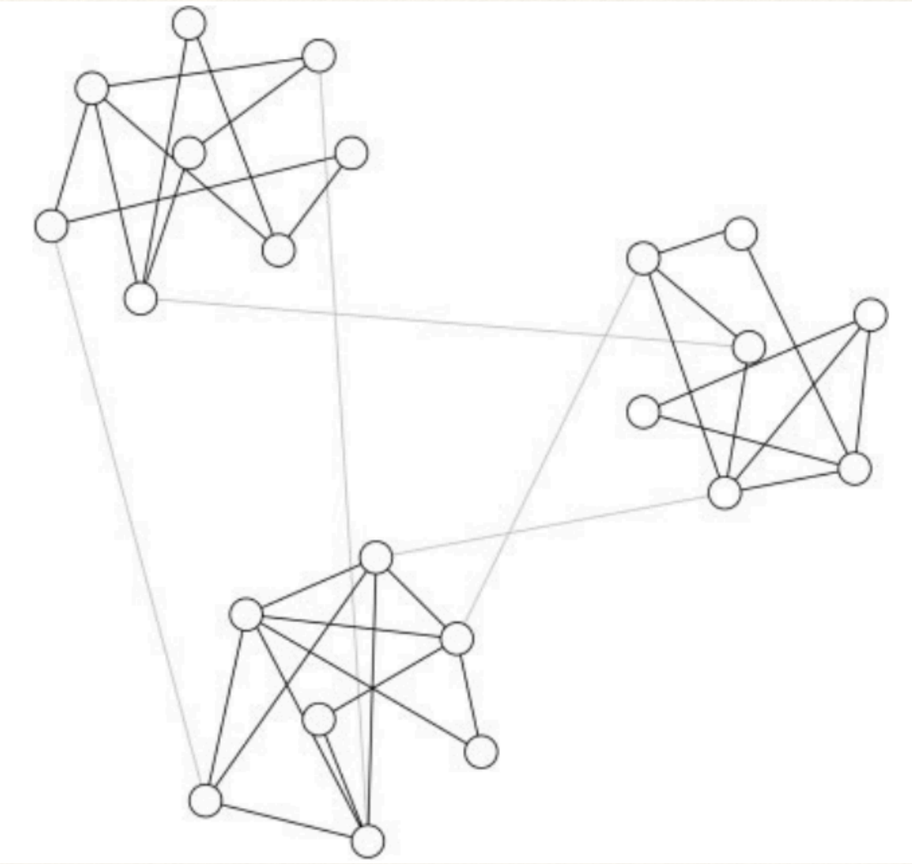
Community detection

Community detection

What is a community?

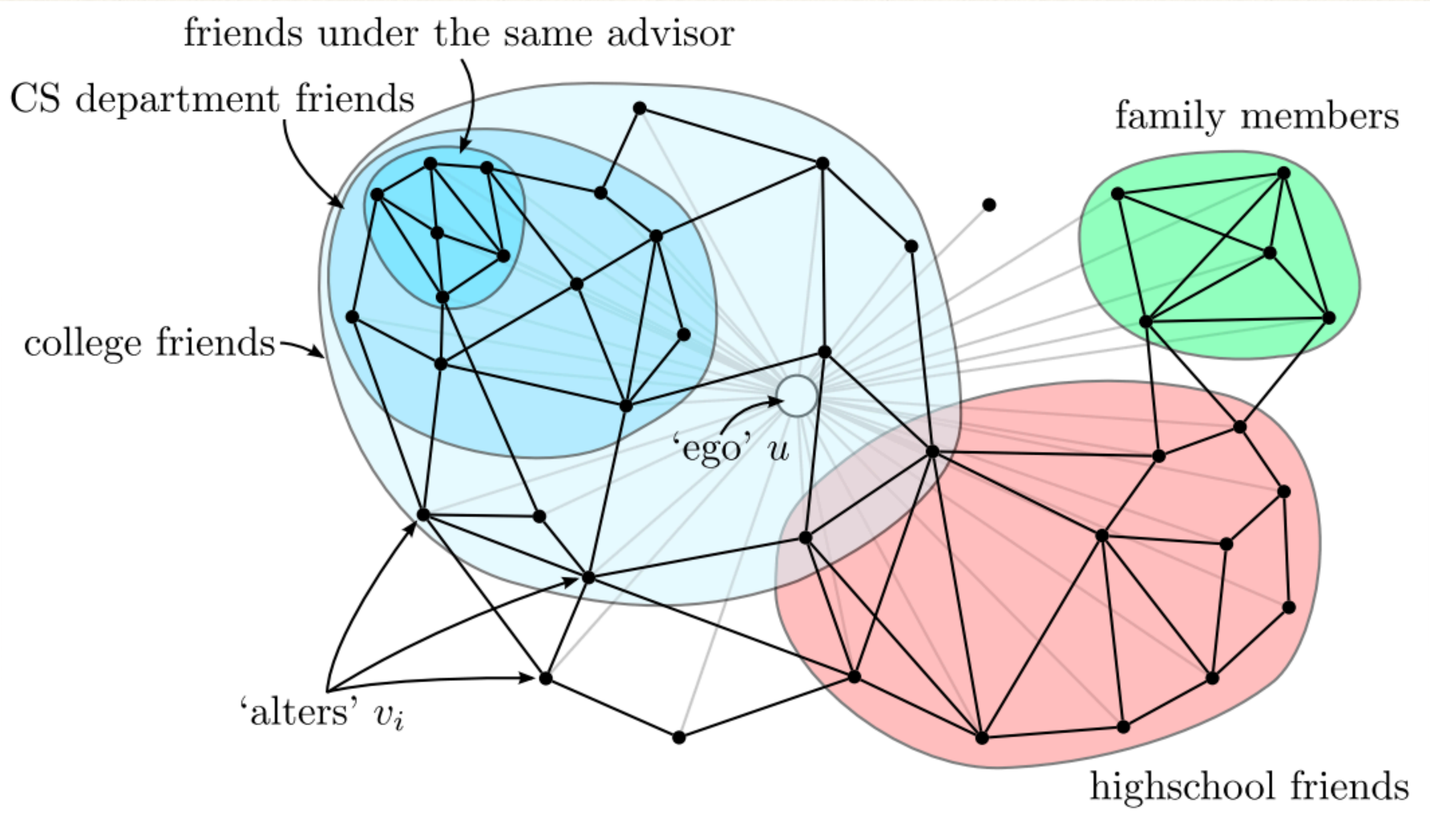
- Society definition - social unit of any size that share common values.
  - Facebook friends, colleagues etc.
  - Shopping - customer with similar interest
  - Citations - papers related to the same research field

- A subset of vertices with **dense connections internally**, and sparse connections to the rest of the graph.
  - A path exists between all the edges in the community
  - When all possible edges exists, the subset is called a **clique**.



Girvan & Newman [2002]

Community detection

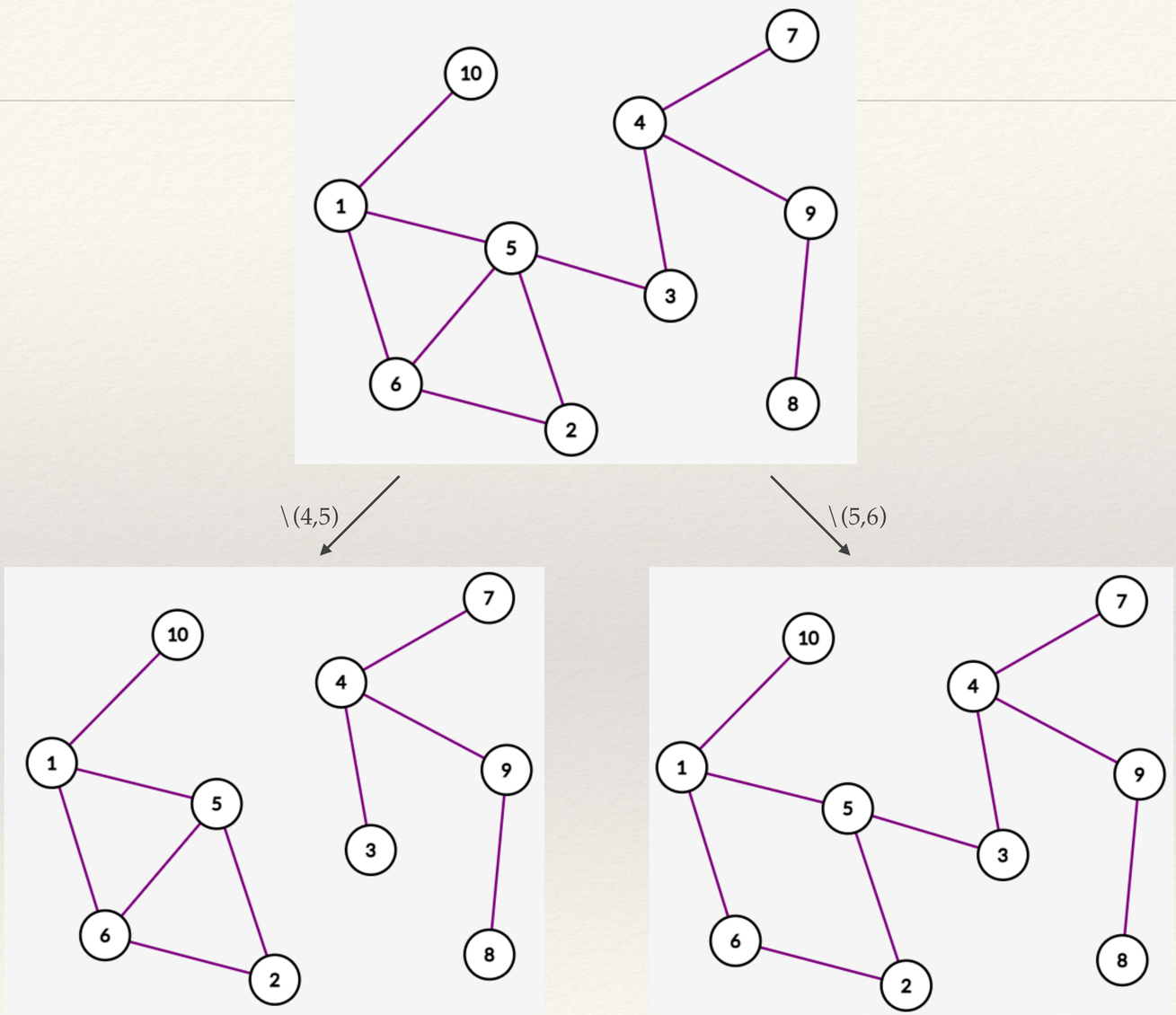


Discovering social circles in Ego networks

Community detection
Girvan-Newman algorithm

- consider the generalization of the betweenness centrality, now for edges:
  - *edge-betweenness-centrality*

$$B(e) = \sum_{u,w \in V} \frac{\sigma_{uw}(e)}{\sigma_{uw}}$$



Segev Arbiv

Community detection
Girvan-Newman algorithm

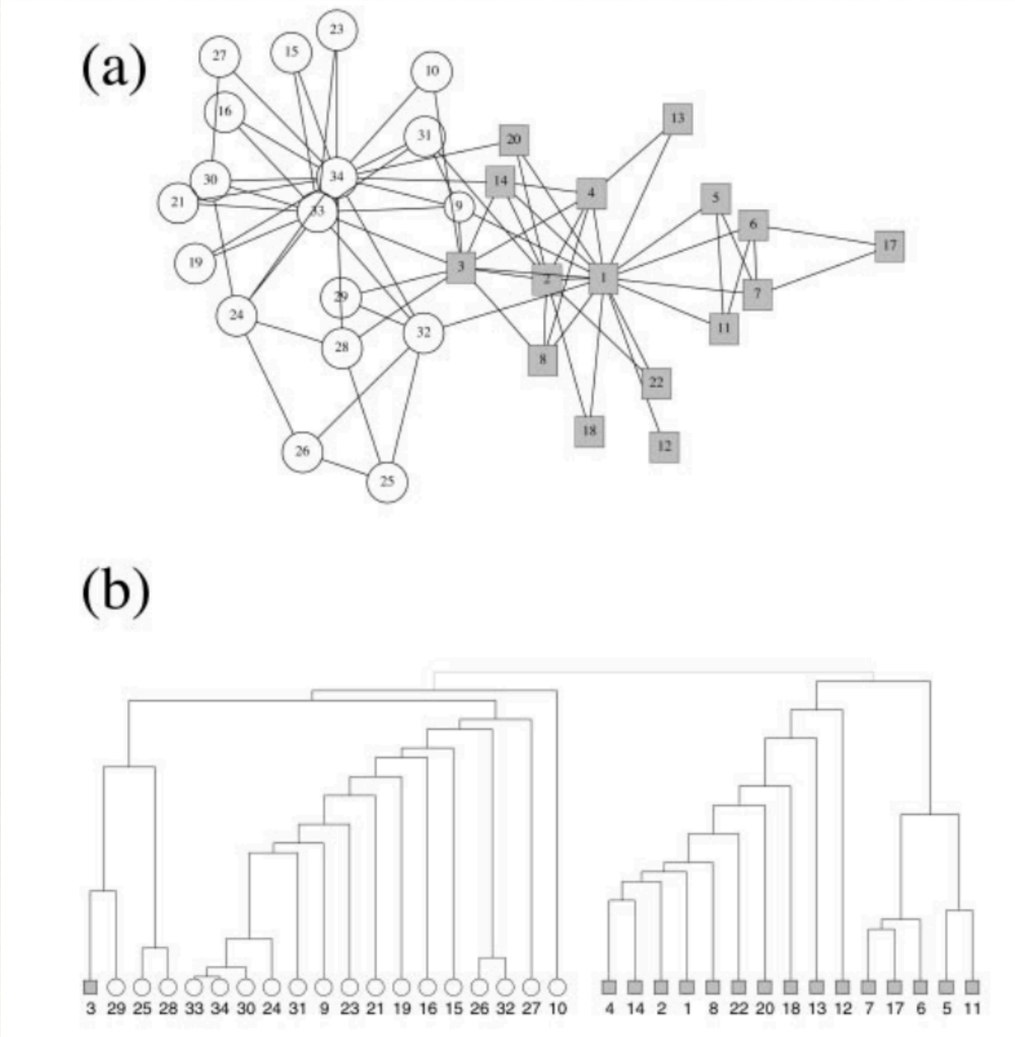
Calculate the *betweenness* for all the edges in the network

- Until no edges remain:
  - Remove the edge with the highest *betweenness*
  - Recalculate *betweenness* for all edges affected by the removal

networkx.algorithms.community.centrality.girvan_newman(G)

Community detection
Girvan-Newman algorithm

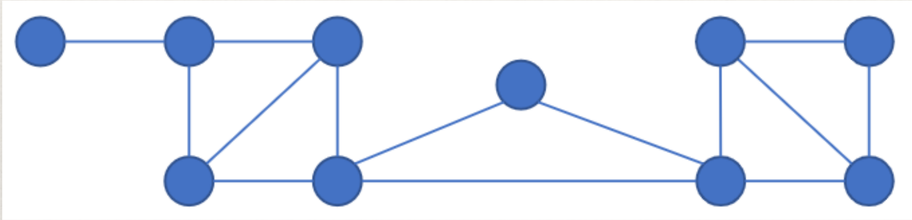
By iteratively repeating the process, we can find hierarchies of *connected components* (communities) in the graph.

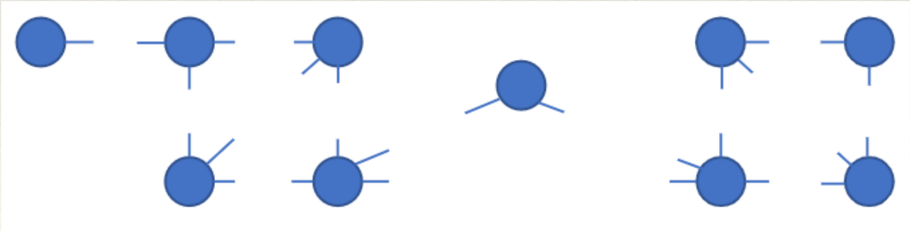


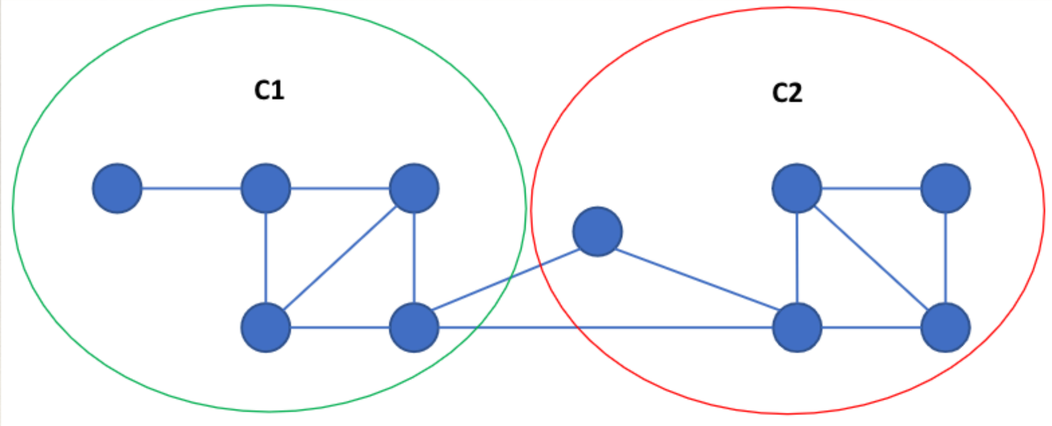
https://www.pnas.org/content/pnas/

Community detection
Modularity

- Modularity is used to compare the quality of the partitions obtained by different methods.







?

How significant this community structure is?

Community detection
Modularity

- Mainly, it measures the **difference from random edge generation**.

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

m -#edges in graph

Is there an edge between *i* and *j*

Probability of a random edge between *i* and *j*

Only consider nodes in the same community

- A signal for a "good" community is if the number of edges **within groups is significantly more than expected.**
  - $Q \in [-1,1]$

Community detection
Bottom up greedy modularity maximization

For a graph with n vertices and m edges:

- Define n singleton communities
- While there is more then one community:
  - For each edge m,
    - calculate ΔQ after adding m
  - connect the 2 communities with the highest $\Delta Q(>0)$

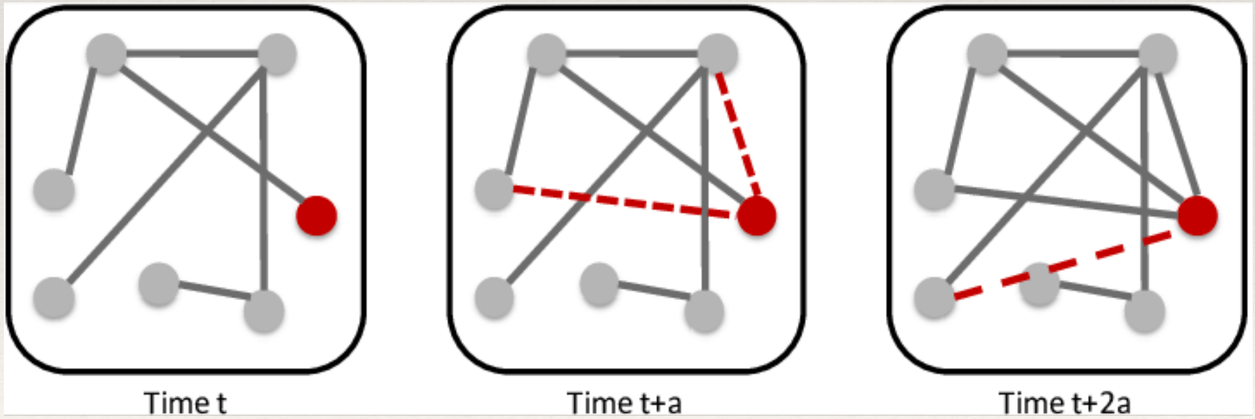
networkx.algorithms.community.modularity_max.greedy_modularity_communities(G)

Link prediction

Segev Arbiv

Link prediction

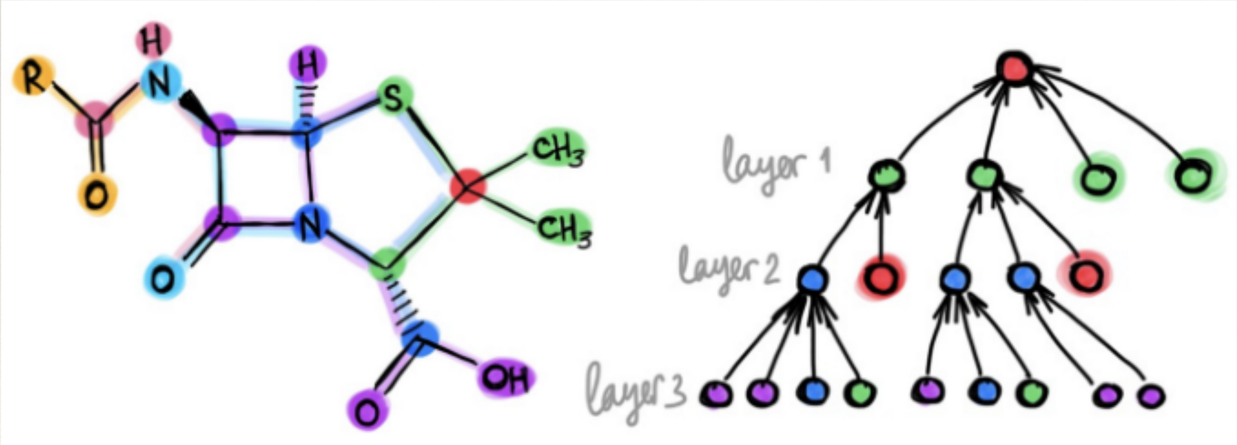
- One of the most popular topics
  - Online market targeting
  - Finding relationship between proteins
  - Predicting user-site interaction
  - etc.



Link prediction

- Real life use-case

Representation Learning

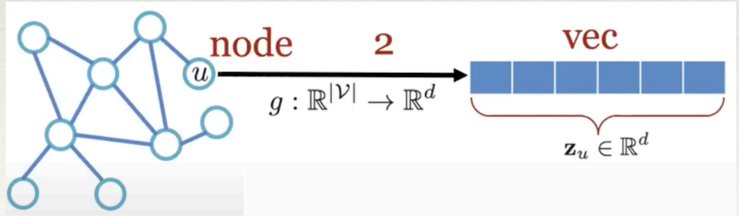


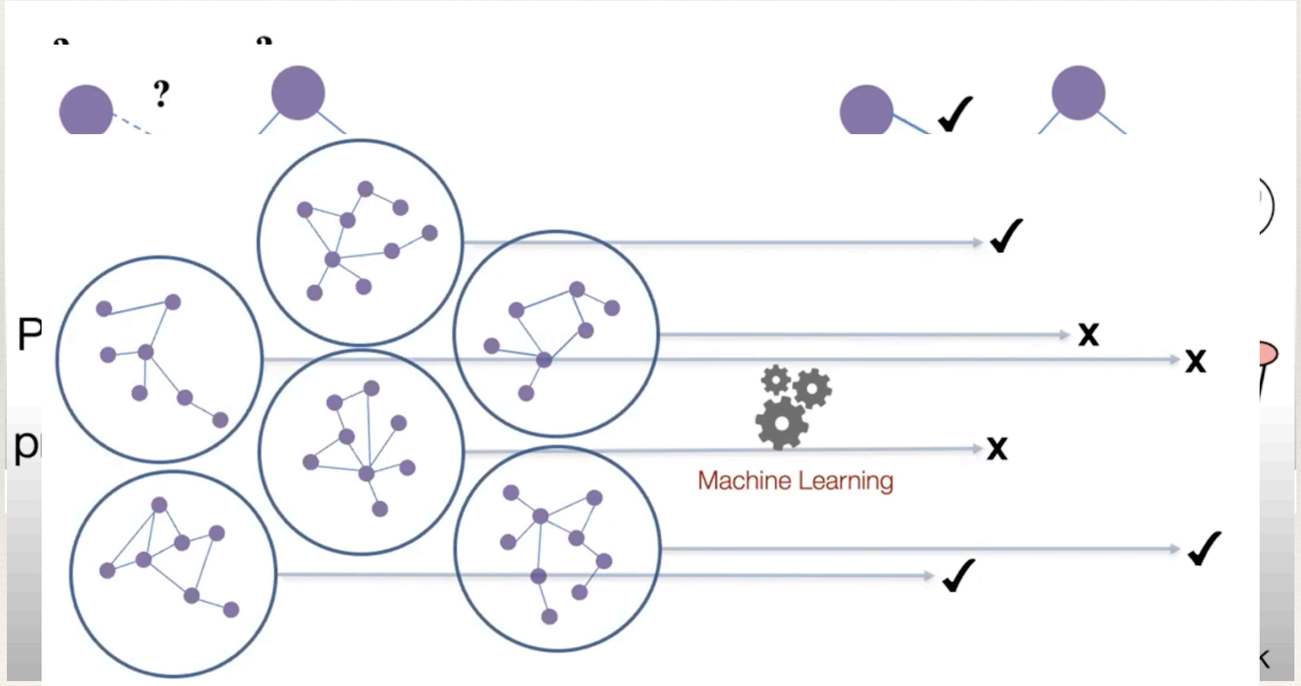
blog

Segev Arbiv

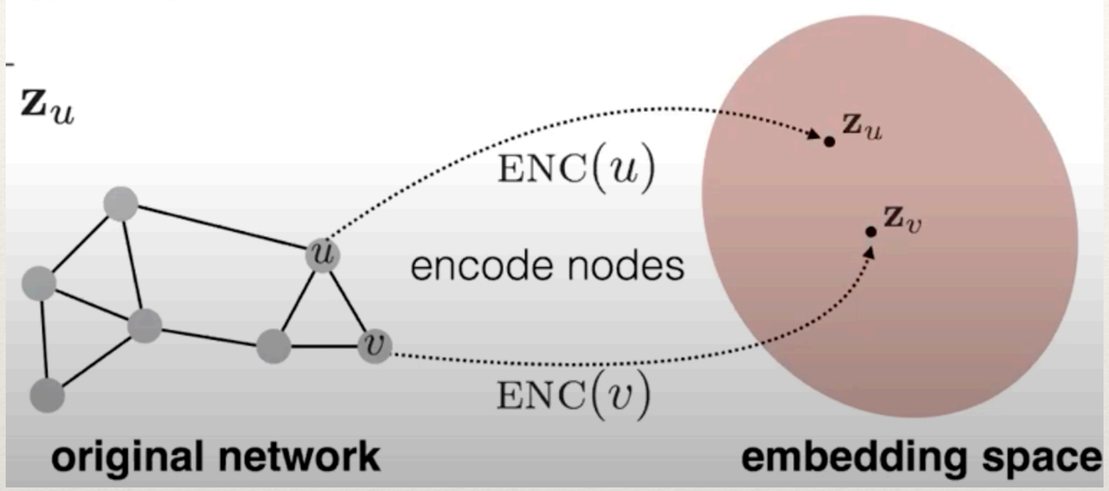
Representation Learning
Node embedding

- More recent approaches tries to embed nodes into meaningful numeric vectors. By generating useful node embeddings, we can tackle much of the problem we discussed. Mainly:
  - node classification
  - link prediction
  - graph classification





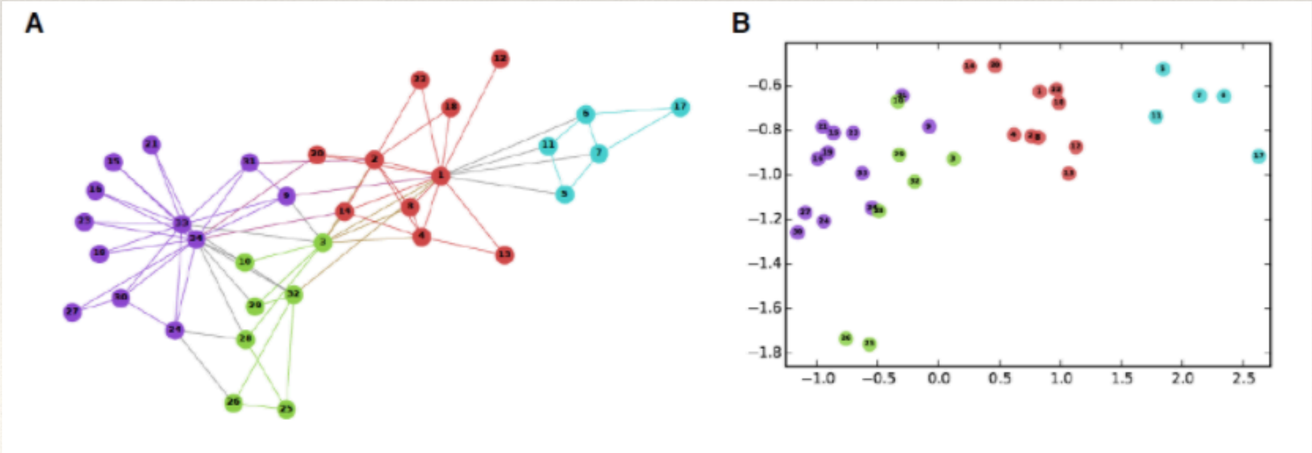
Representation Learning
Classic approach - Node embedding



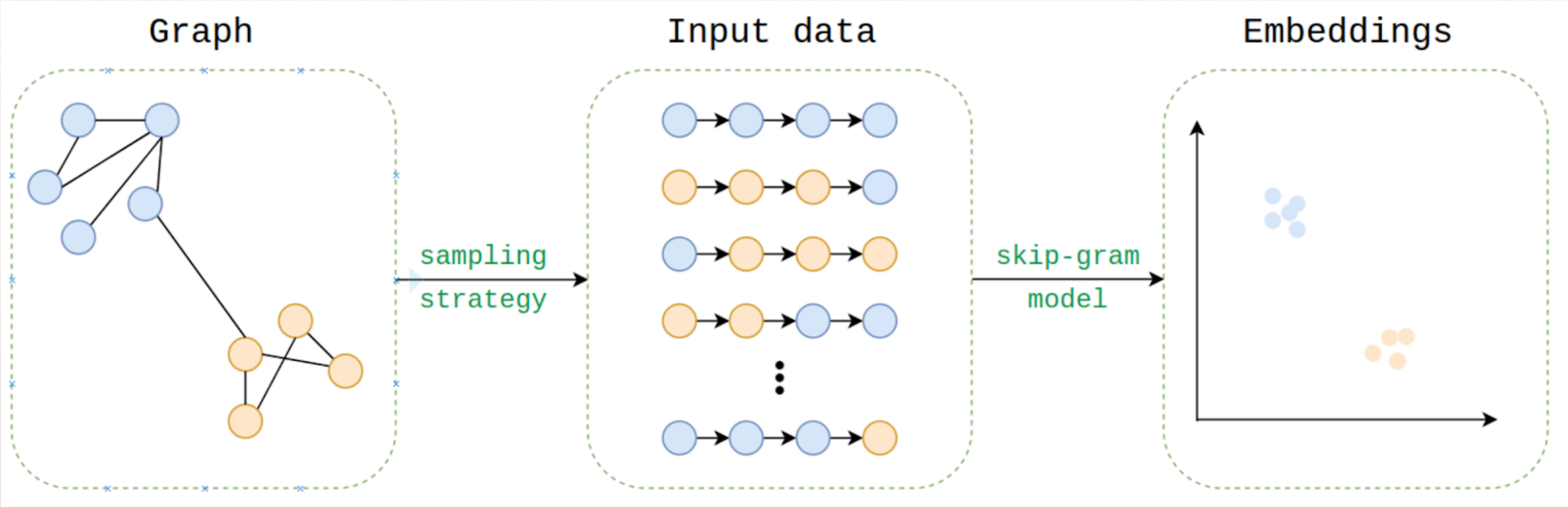
$$\text{similarity}(u, v) \approx \mathbf{z}_v^\top \mathbf{z}_u$$

$$\mathbf{z}_u^\top \mathbf{z}_v \approx \boxed{\mathbf{A}^k[u, v]}$$

$\mathbf{z}_u^\top \mathbf{z}_v \approx$ probability that u and v co-occur on a random walk over the network



Representation Learning
Random Walk approaches - DeepWalk, Node2Vec



Elior Cohen's blog

Using Skip-gram, tries to model $p_w(v_j \mid v_i)$ — the probability that v_j appear on a random walk started at v_i.

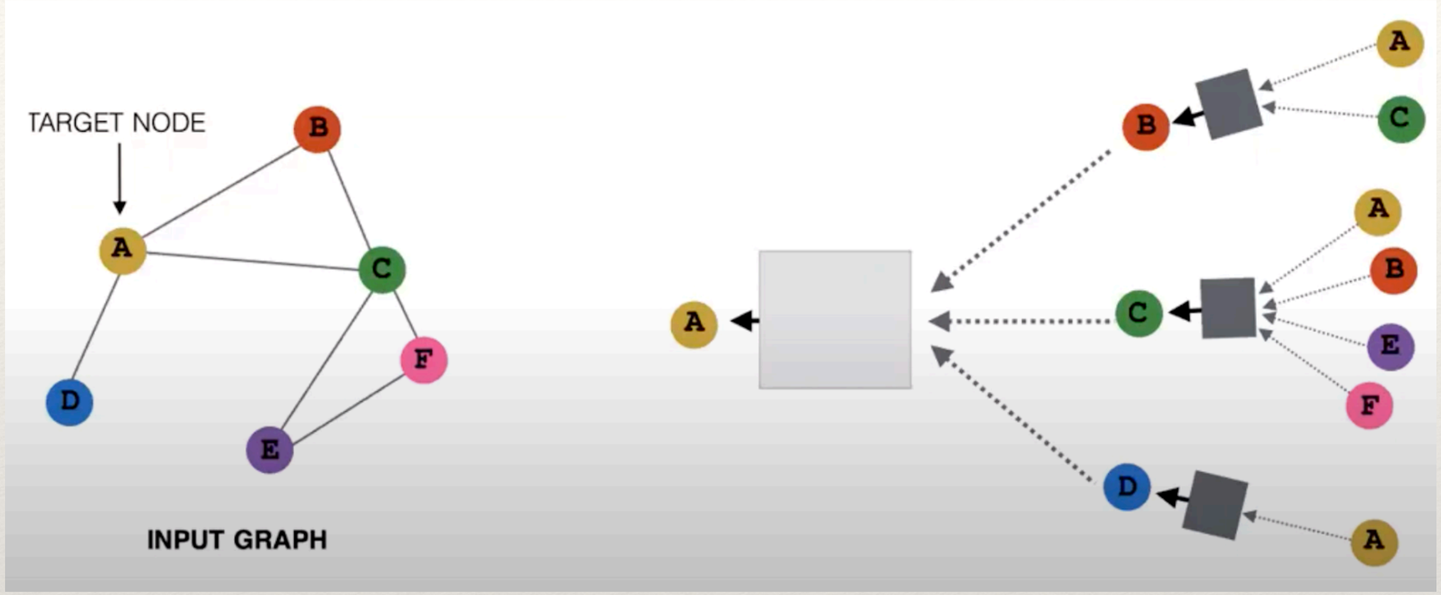
- $\mathcal{O}(V)$ parameters are needed
- Transductive - can't generate embeddings for new nodes
- do not incorporate node features

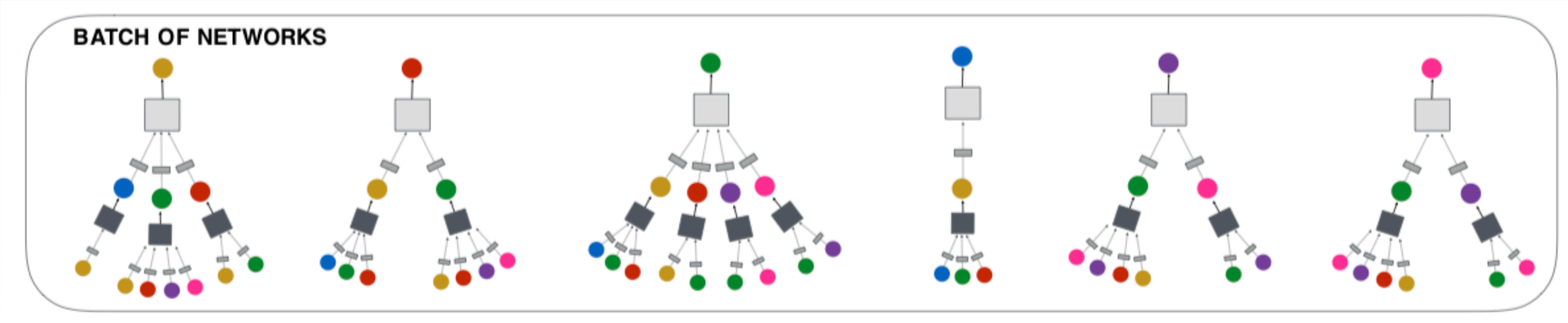
Representation Learning
Graph Neural Networks

- These approaches work well, but do not incorporate valuable information that we often posses - node features (age, gender, etc.)

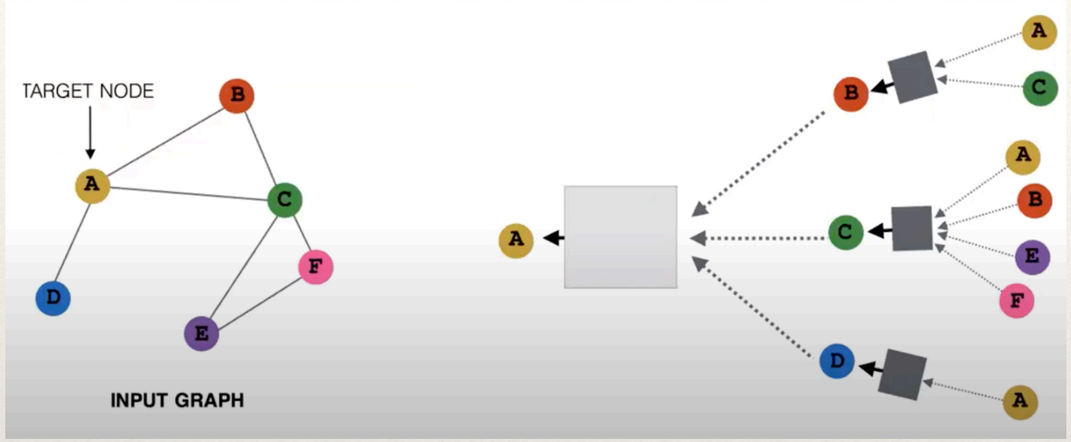
Traditional node embeddings	Graph neural networks
Encoder is just a shallow lookup from an embedding matrix	*Encoder is complex function of graph structure and node features*
$\text{ENC}(v) = \mathbf{Z}\mathbf{v}$	$\text{ENC}(v) = f(\mathbf{A}, \mathbf{X})$
$\mathbf{Z} \in \mathbb{R}^{d \times \lvert\mathcal{V}\rvert}$ Embedding matrix, each column is node embedding	$\mathbf{A} \in \mathbb{R}^{\lvert\mathcal{V}\rvert \times \lvert\mathcal{V}\rvert}$ adjacency matrix for the graph
$\mathbf{v} \in \mathbb{I}^{\lvert\mathcal{V}\rvert}$ indicator vector for node *v*	$\mathbf{X} \in \mathbb{R}^{\lvert\mathcal{V}\rvert \times d}$ matrix of node features

Representation Learning
Graph Neural Networks





Representation Learning
Graph Neural Networks



Update node embedding

$$\mathbf{h}_u^{(k+1)} = \mathrm{UPDATE}^{(k)}\left(\mathbf{h}_u^{(k)}, \mathrm{AGGREGATE}^{(k)}(\{\mathbf{h}_v^{(k)}, \forall v \in \mathcal{N}(u)\})\right)$$
$$= \mathrm{UPDATE}^{(k)}\left(\mathbf{h}_u^{(k)}, \mathbf{m}_{\mathcal{N}(u)}^{(k)}\right),$$

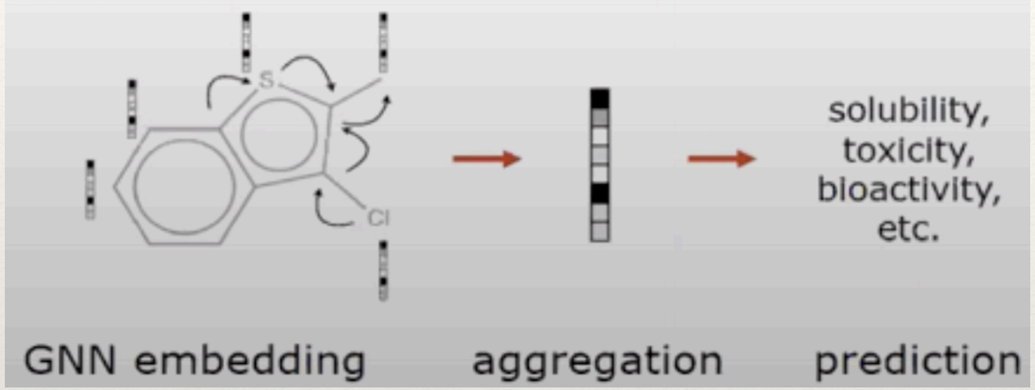
Information aggregated from neighboring nodes

(sometimes called the "message" from the neighbors)

Combined with the previous layer embedding of this node

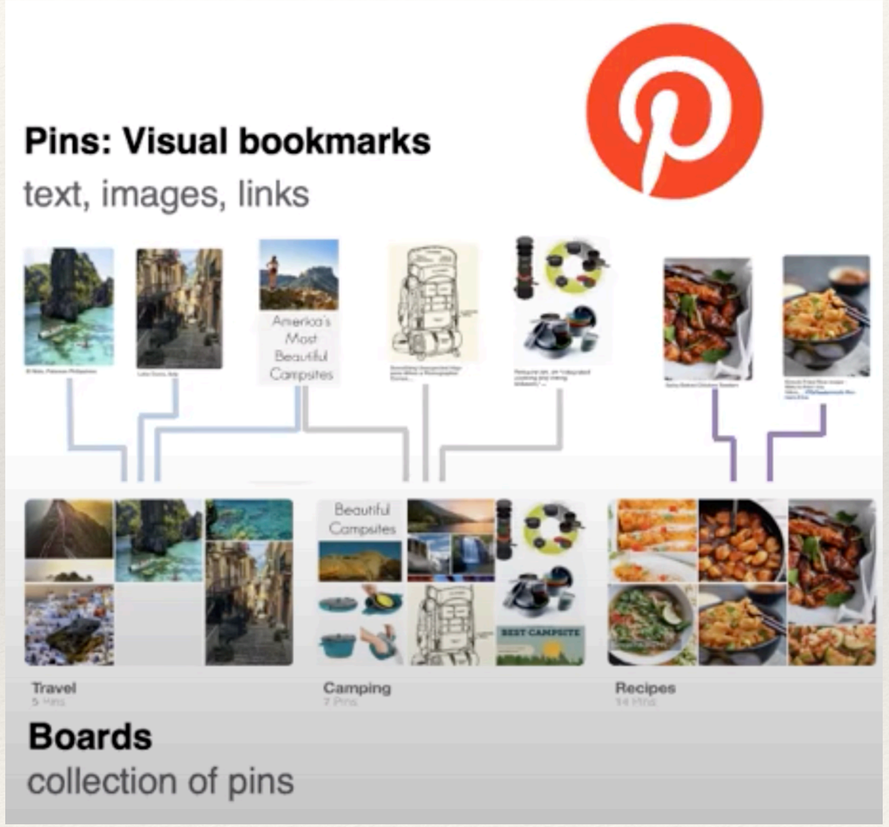
- #params ⫫ graph size
- Inductive - can infer embedding for unseen nodes
- Naturally incorporate node features

Representation Learning
Graph Neural Networks



DL approach for Antibiotic discovery

LambdaZero



PinSage

Thank You.

Hope you enjoyed (and learnt)!

segevarbiv@gmail.com