Network science / Graph mining

Tools & metrics for analyzing a connected world of data

Erwan Le Merrer[1]

[1]Inria, Rennes

ESIR, 2020

Outline

Outline

Graphs ?

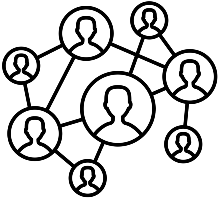


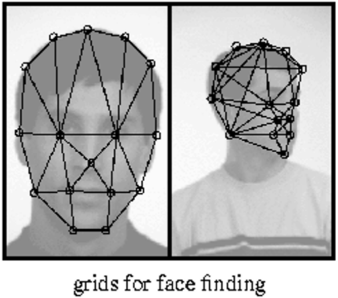
Figure: A graph: entities (nodes) and connections (edges)

- An abstraction for reasoning about characteristics of a relational data, networks...
- Sometimes called “network science”.

More precisely

- *Vertex*: a graph node, or entity.
- *Edge*: a link connecting two vertices.
- *Directed* and *undirected* graphs:
  - in directed graphs, edges have orientation (arrow end)

The omnipresence of graphs in applications



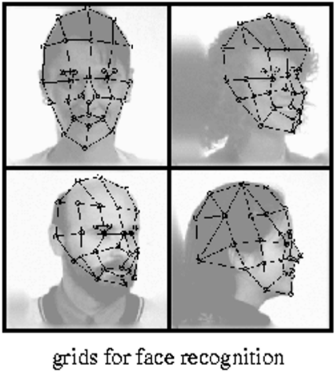


Figure: Graphs in computer vision

The omnipresence of graphs in applications

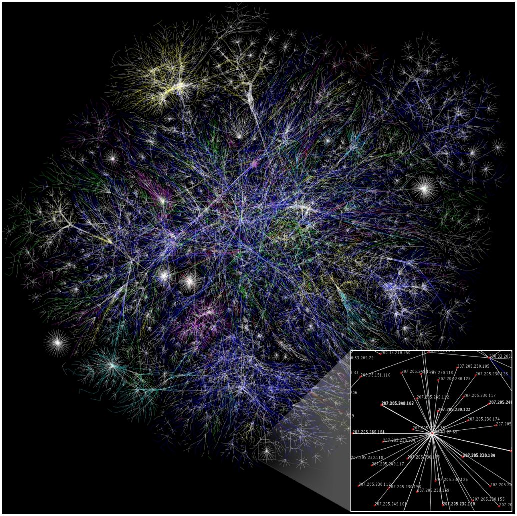


Figure: The Internet AS graph

The omnipresence of graphs in applications

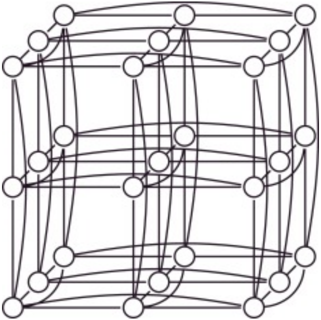


Figure: Interconnecting system-on-chips in a datacenter rack

- exemple use in social nets, epidemics...

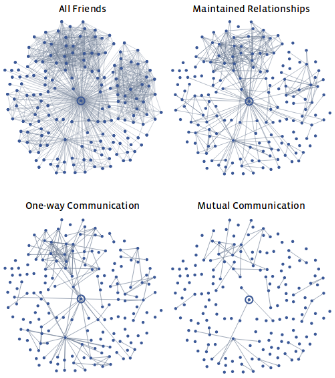


Figure 3.8: Four different views of a Facebook user's network neighborhood, showing the structure of links coresponding respectively to all declared friendships, maintained relationships, one-way communication, and reciprocal (i.e. mutual) communication. (Image from [286].)

Figure: From Networks, Crowds, and Markets: Reasoning about a Highly Connected World . By David Easley and Jon Kleinberg. Cambridge University Press, 2010.

e.g.: recommendations on YouTube

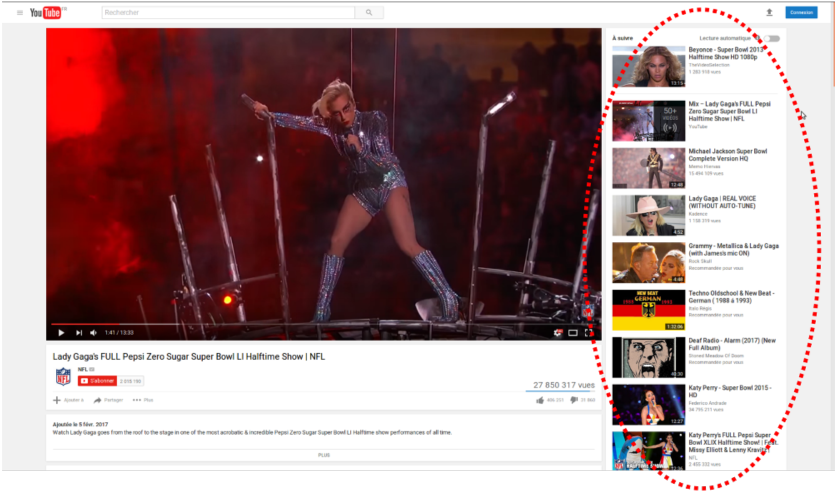
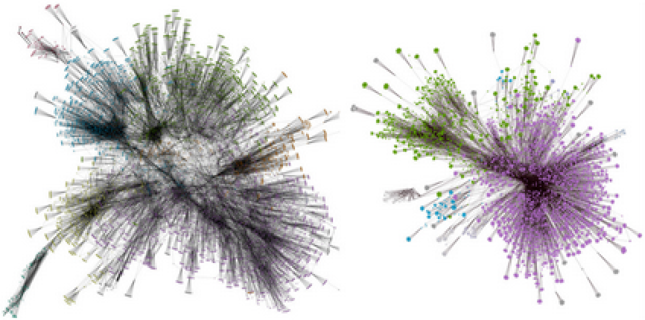


Figure: Recommandations: contextual, personalized?

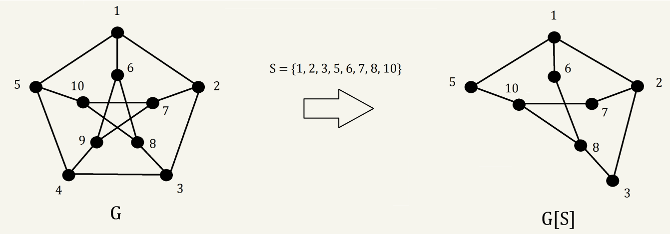


4-hops graphs from a YouTube video, new user (left) and returning user (ri

Figure: Blank profile vs. my recommandations

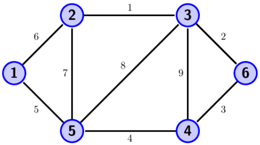
- *Directed* and *undirected* graphs:
  - in directed graphs, edges have orientation (arrow end)

- A *subgraph* of G: formed by a subset of vertices and edges from G.

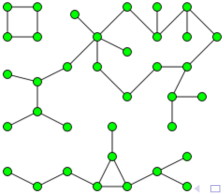


Most important notions (cont'd)

- Edge *weight*: value assigned as a label to an edge.
  - e.g., distance in km of a road from city 1 to 2.

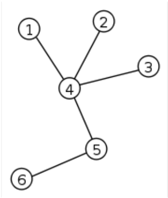


- Graph *connectivity:*
  - A graph is *connected* if there is a *path* btw any pair of vertices.
  - Otherwise, *connected components* are the subgraphs in which paths exist.

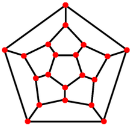


Most important notions (cont'd)

- A *cycle*: a path in which a vertex is reachable from itself.
    - Example of an *acyclic* connected graph: a *tree*



- A *planar* graph: can be draw without any edges crossing each other.



Special *topologies*

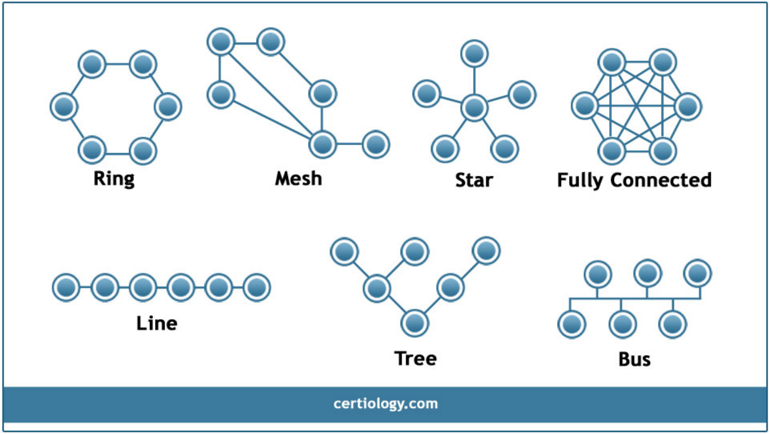
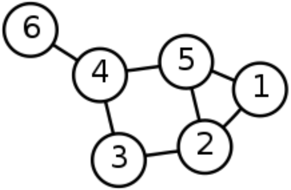


Figure: Graphs to remember, often used as illustrations

Outline

Basic notations



- $G(V,E)$: graph G with node set V, connected by edge set E.
  - $V = \{1,2,3,4,5,6\}$;
    $E = [[1,5],[1,2],[2,3],[2,5],[3,4],[4,5],[4,6]]$
- Number of nodes is $n = |V|$, edges is $m = |E|$.
- Neighbors of node i are set $\Gamma(i)$.
  - $\Gamma(1) = \{2,5\}$

Degree of a node

- The degree d_v of node v is equal to $|\Gamma(v)|$ (its number of neighbors).
- Degree span: $0 \leq d_v \leq n-1$ (if no self loops).
- *Degree distribution* P(d) is the probability distribution of each degree in the current graph:

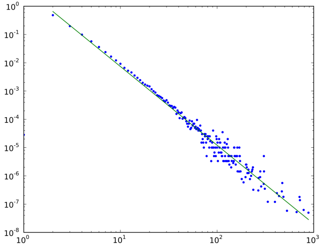


Figure: Degree distribution: x-axis is degree, y-axis is probability

- In(out)-degree of v: counts incoming(outgoing) edges only.

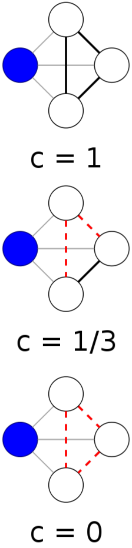
Clustering coefficient

- Every two nodes in a *clique* are neighbors.
- *Local clustering coefficient of a node i measures "how close are* $\Gamma(i)$ *from being a clique":*

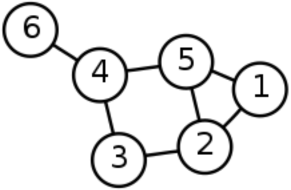
$$C_i = \frac{2|e_{jk} : v_j, v_k \in \Gamma(v_i), e_{jk} \in E|}{d_i(d_i - 1)}$$

- Average clustering coefficient:

$$\bar{C} = \frac{1}{n}\sum_{i=1}^{n} C_i$$

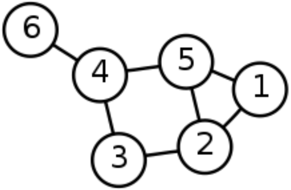


Path lengths



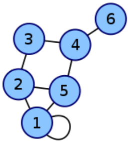
- *Path*: sequence of adjacent nodes connecting two nodes (if exists).
  - e.g., two paths btw 6 and 1: (4,5,1) and (4,3,2,5,1).
  - One *hop*: one transition from a node to another.
- *Shortest path*: path of minimal cardinality.
  - Distance $dist(6,1) = |(4,5,1)| = 3$
- *Single-source shortest path* (SSSP): shortest paths from node i to all other nodes ($V \setminus i$).
- *All-pairs shortest paths (APSP)*: SSSP from $\forall i \in V$.

Diameter



- *Average path length*: average of all-pair shortest distances in the graph.
- *Diameter*: longest path of the APSP, i.e., greatest distance between any pair of vertices.
  - $diam(G) = |(4,5,1)| = 3$, starting at node 6.

Algebraic connectivity



- *Degree matrix D*: diagonal matrix containing the degree of each vertex.
- *Adjacency matrix A*: 1 if edge exists (2 for self-loop), 0 otherwise.
- *Laplacian matrix*: $L = D - A$.
- *Algebraic connectivity*: second-smallest eigenvalue of L.
  - $> 0 \iff$ graph is connected.
- Number of 0s as eigenvalues equals number of connected components.

$$\begin{pmatrix} 4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Figure: D

$$\begin{pmatrix} 2 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

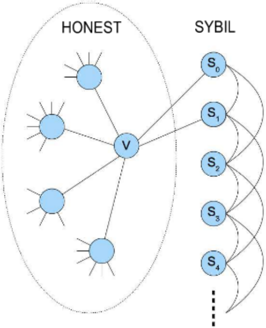
Figure: A

Conductance

- The conductance $\Phi(C)$ of a set C of vertices in a given graph G is the ratio between the number of edges going out from C and the number of edges inside C:

$$\Phi(C) = \frac{|cut(C)|}{vol(C)},$$

where $vol(C)$, is the sum of the degrees of the vertices in C.

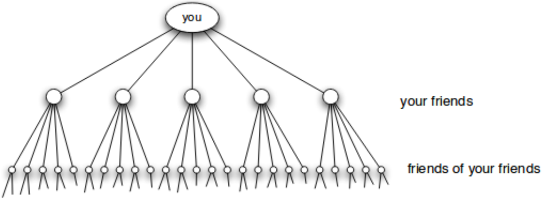


- *Expansion* of G: mean number of nodes that are reached in h hops from all nodes:

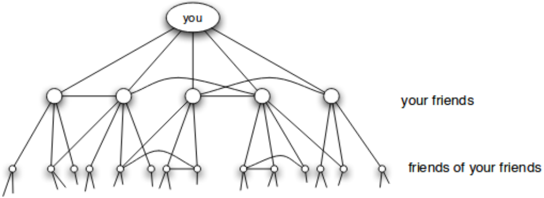
$$e_G(h) = \frac{1}{n^2} \sum_{v \in V} |C_v(h)|,$$

with $C_v(h)$ the set of reachable nodes from v in h hops.

Expansion - example



(a) *Pure exponential growth produces a small world*



(b) *Triadic closure reduces the growth rate*

Figure: Expansion in a social network

Resilience

- Measures the robustness of a graph:

$$r_G(h) = \frac{1}{|E|} \sum_{v \in V} I(v, |C_v(h)|),$$

with $I(v, |C_v(h)|)$ the number of edges that need to be removed to split $C_v(h)$ into 2 sets (of roughly the same size). h: distance (hops).

Outline

Adjacency list or edge list representations

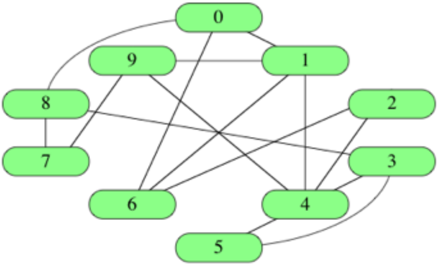


Figure: A graph, to load for analysis

Edge list:

[[0,1], [0,6], [0,8], [1,4], [1,6], [1,9], [2,4], [2,6], [3,4], [3,5], [3,8],[4,5], [4,9], [7,8], [7,9]]

$O(|V|)$ access time to find an edge, but $O(|E|)$ space in memory.

Adjacency list:

[[1, 6, 8], [0, 4, 6, 9], [4, 6], [4, 5, 8], [1, 2, 3, 5, 9], [3, 4], [0, 1, 2], [8, 9], [0, 3, 7], [1, 4, 7]]

$O(1)$ access time to vertex , but $O(|V|)$ to access a given edge.

Image[1]

Matrix representation

	0	1	2	3	4	5	6	7	8	9
0	0	1	0	0	0	0	1	0	1	0
1	1	0	0	0	1	0	1	0	0	1
2	0	0	0	0	1	0	1	0	0	0
3	0	0	0	0	1	1	0	0	1	0
4	0	1	1	1	0	1	0	0	0	1
5	0	0	0	1	1	0	0	0	0	0
6	1	1	1	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	1	1
8	1	0	0	1	0	0	0	1	0	0
9	0	1	0	0	1	0	0	1	0	0

Figure: Matrix representation of previous graph

Find edge presence in $O(1)$ time, but $\Theta(V^2)$ space in memory.
1's to be replaced by edge weigths for weighted graphs.

Example tool families for manipulating graphs

GraphX

Figure: For massive graphs (cannot fit into on server's memory)

X – Stream

Figure: Big graph processing on a single machine

neo4j

Figure: For a database-like handling of graphs

NetworkX

Figure: Prototyping in Python, lots of contributions