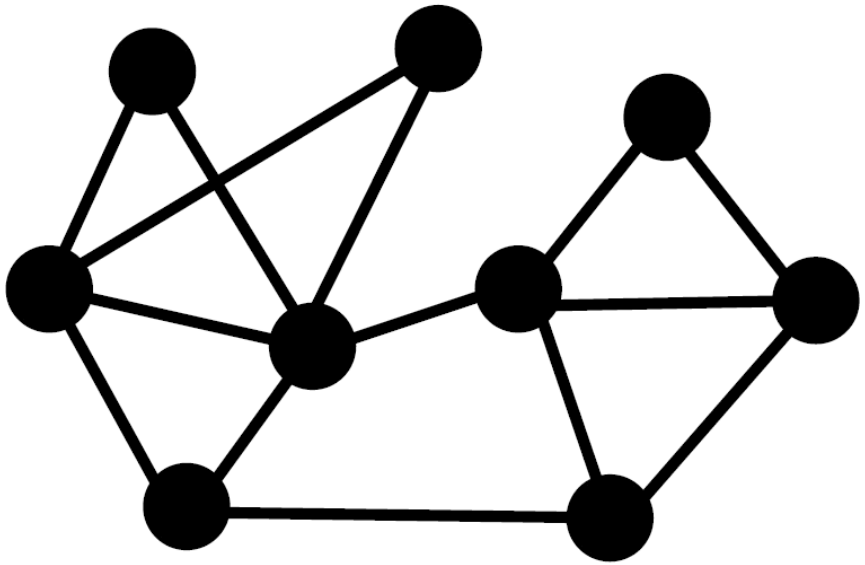
DS 503: Advanced Data Analytics

Lecture 2: Graph Data

Gagan Raj Gupta

Why Graphs

Graphs are a general language for describing and analyzing entities with relations/interactions



G = (V,E)
V: Set of Vertices/Nodes
E: Set of Edges (u, v) $u, v \in V$

Examples

Image credit: Medium

Social Networks

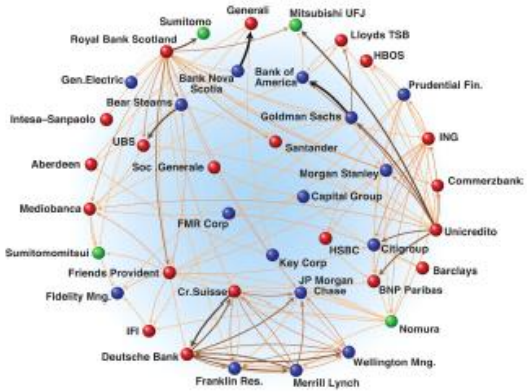
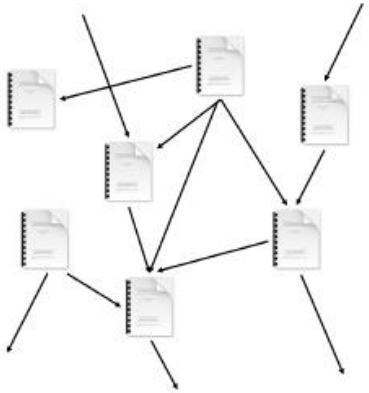


Image credit: Science

Economic Networks

Image credit: Lumen Learning

Communication Networks



Citation Networks

Image credit: Missoula Current News

Internet

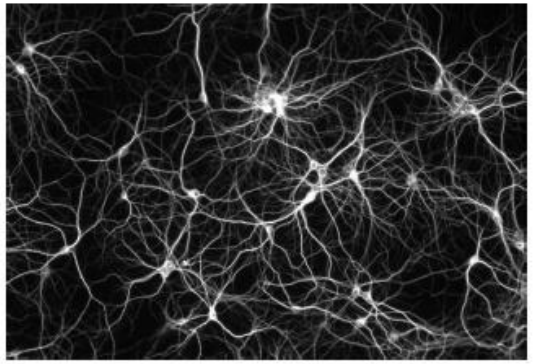


Image credit: The Conversation

Networks of Neurons

Examples

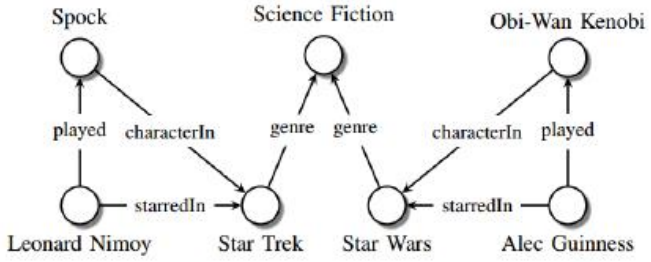


Image credit: Maximilian Nickel et al

Knowledge Graphs

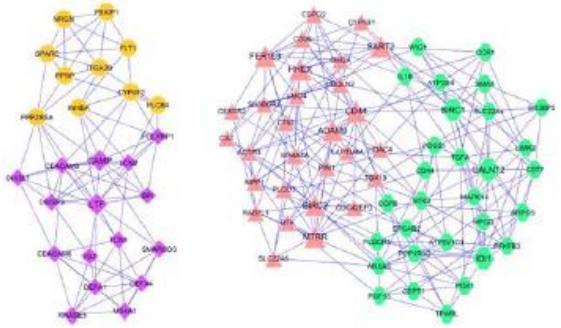


Image credit: ese.wustl.edu

Regulatory Networks

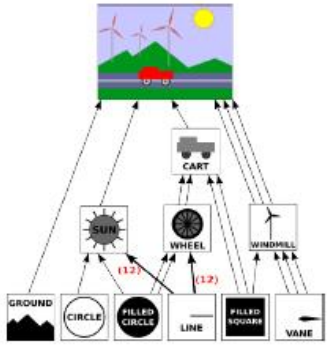


Image credit: math.hws.edu

Scene Graphs

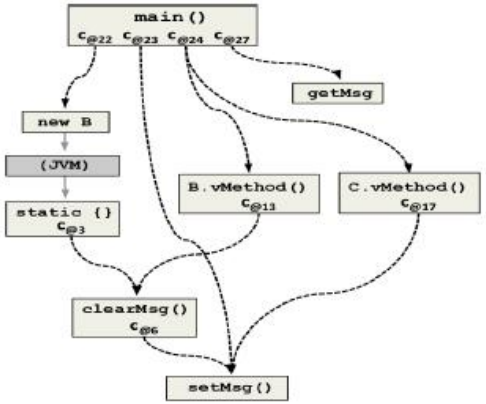


Image credit: ResearchGate

Code Graphs

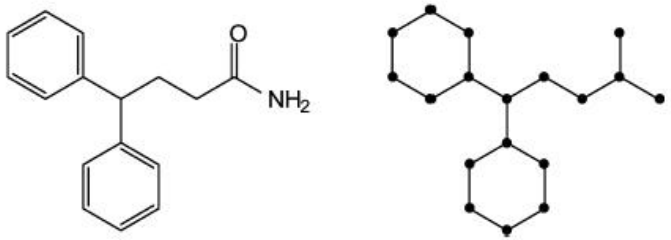


Image credit: MDPI

Molecules

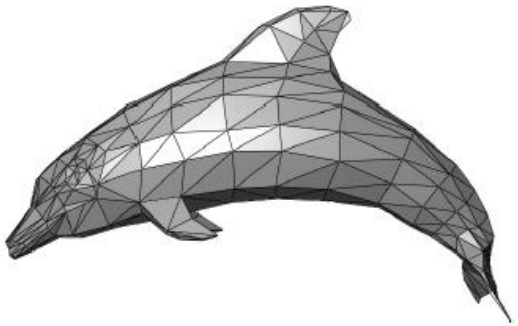
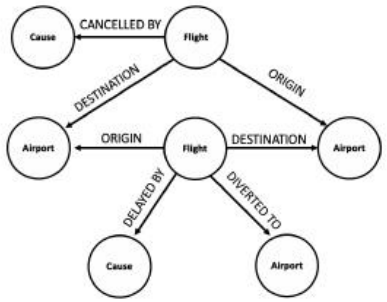


Image credit: Wikipedia

3D Shapes

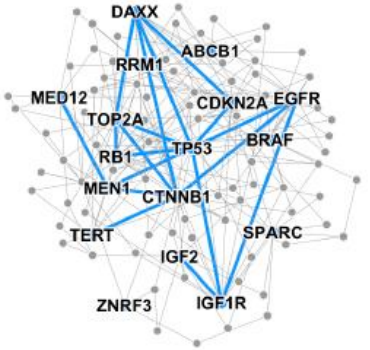
Examples



Event Graphs

Image credit: SalientNetworks

Computer Networks



Disease Pathways

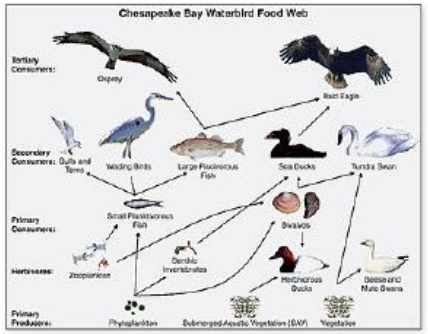


Image credit: Wikipedia

Food Webs

Image credit: Pinterest

Particle Networks

Image credit: visitlondon.com

Underground Networks

Networks and Graphs

- **Graphs (as a representation):**
- **Information/knowledge** are organized and linked
- **Software** can be represented as a graph
- **Similarity networks:** Connect similar data points
- **Relational structures:** Molecules, Scene graphs, 3D shapes, Particle-based physics simulations

Networks (also known as Natural Graphs):

- **Social networks:**
  - **Society** is a collection of 7+ billion individuals
- **Communication and transactions:**
  - Electronic devices, phone calls, financial transactions
- **Biomedicine:**
  - Interactions between **genes/proteins** regulate life
- **Brain connections:**

Sometimes the distinction between networks & graphs is blurred

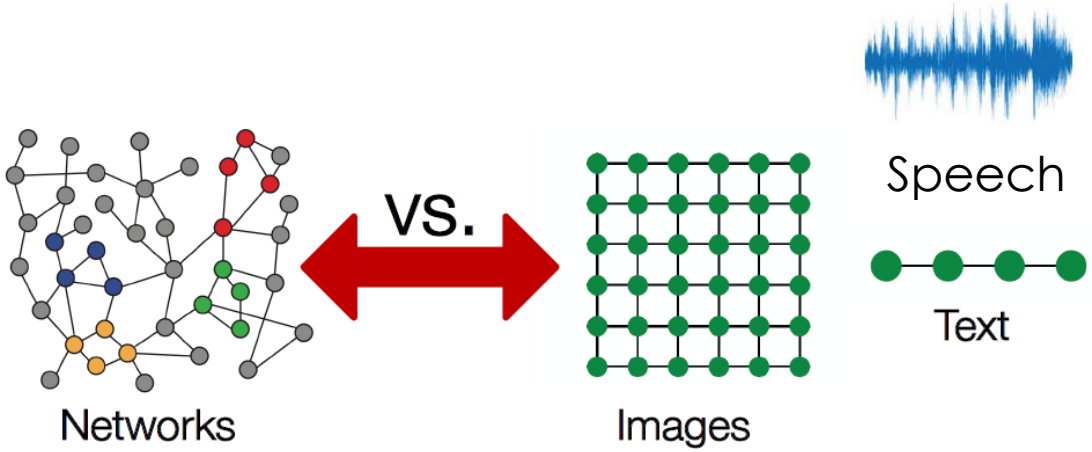
Graphs and Relational Data

Main question:

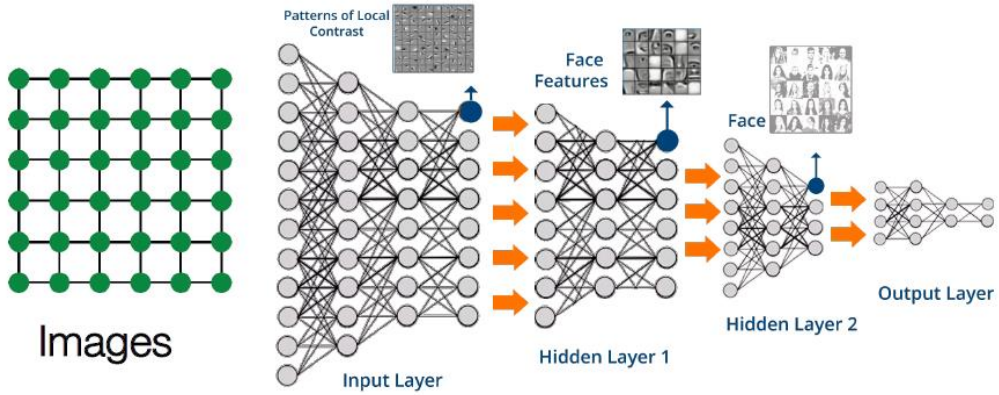
How do we take advantage of relational structure for better prediction?

- Complex domains have a rich relational structure, which can be represented as a **relational graph**
- **By explicitly modeling relationships we achieve better performance!**

Graph structured data



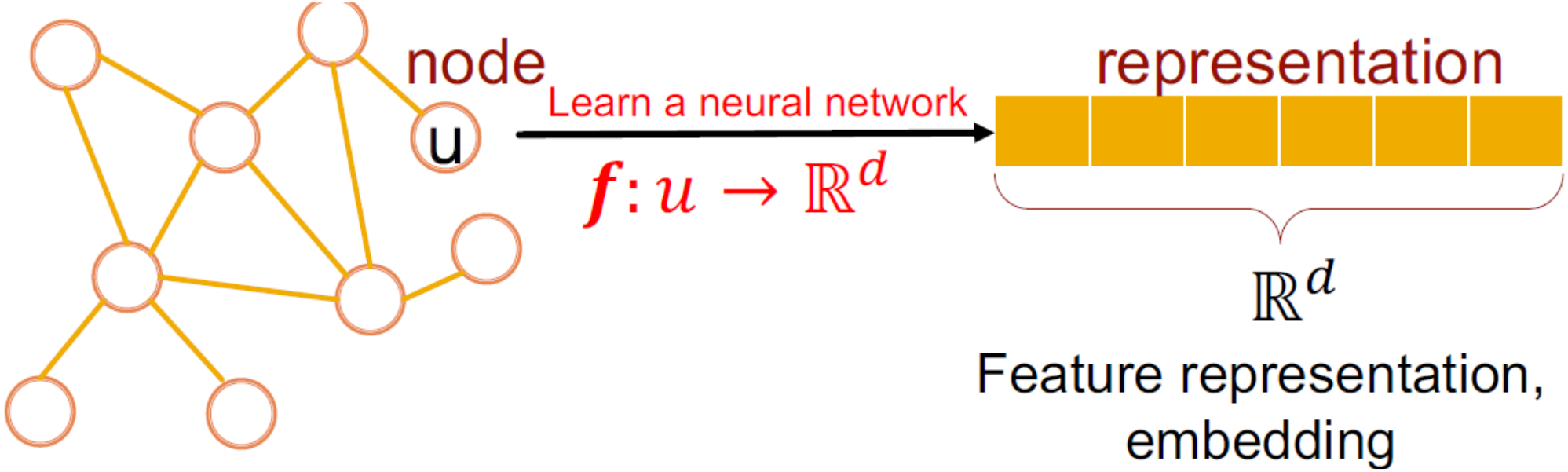
Modern ML Toolbox



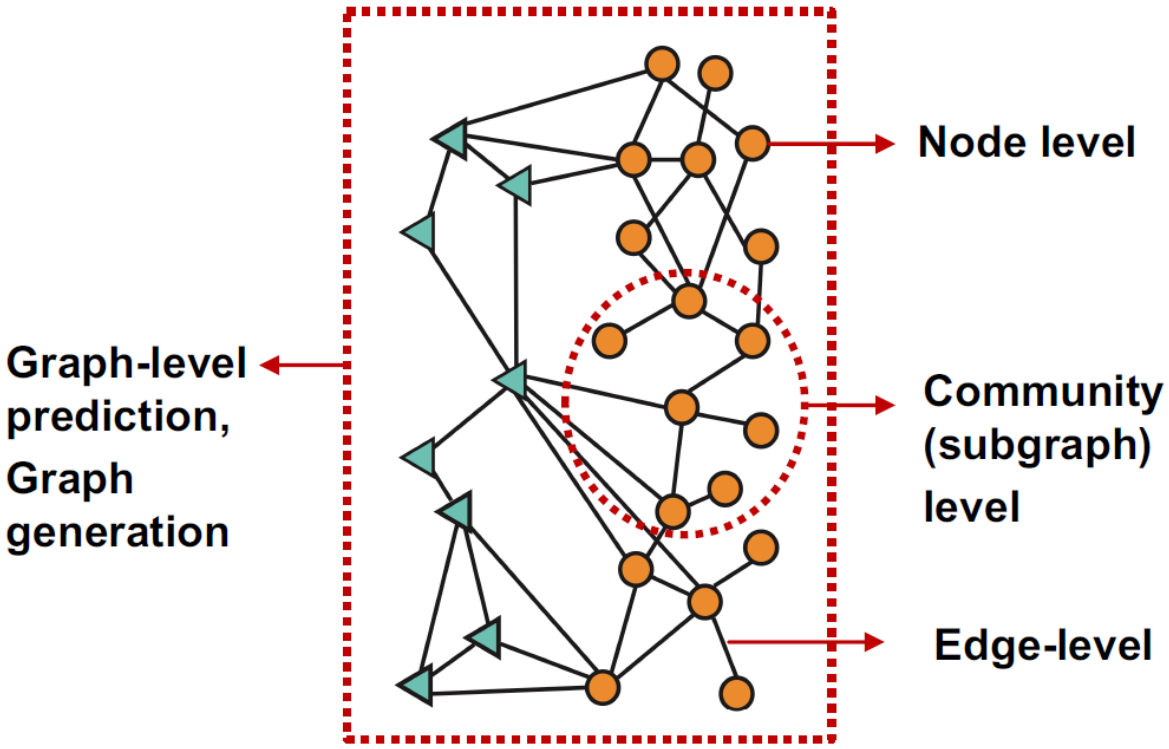
- Modern deep learning toolbox is designed for sequences and grids.
- Graph data {V, E} is far more complex with arbitrary size and complex topological structure (i.e. no spatial locality like grids).
- There is no fixed ordering or reference point. Even simple looking problem of comparing two graphs for equality is NP hard.
- Graph data is often dynamic and have multi-modal features.
- Eg. 3D meshes, social networks, telecommunication, networks, biological networks or brain connectomes.

Representation Learning

- (Supervised) Machine Learning Lifecycle: Retrain for every new feature. Every single time!
- Representation Learning – Automatically learn the features
- Map nodes to d-dimensional **embeddings** such that similar nodes in the network are embedded close together



Graph ML Tasks



- **Node classification**: Predict a property of a node
  - **Example:** Categorize online users / items
- **Link prediction**: Predict whether there are missing links between two nodes
  - **Example:** Knowledge graph completion
- **Graph classification**: Categorize different graphs
  - **Example:** Molecule property prediction
- **Clustering**: Detect if nodes form a community
  - **Example:** Social circle detection
- **Other tasks**:
  - **Graph generation**: Drug discovery
  - **Graph evolution**: Physical simulation

Defining a graph

- **How to build a graph:**
  - **What are nodes?**
  - **What are edges?**
- **Choice of the proper network representation of a given domain/problem determines our ability to use networks successfully:**
  - In some cases there is a unique, unambiguous representation
  - In other cases, the representation is by no means unique
  - The way you assign links will determine the nature of the question you can study

Graphs from Data Matrix

Many datasets that are not in the form of a graph can still be converted into one.

Let $\mathbf{D} = \{\mathbf{x}_i\}_{i=1}^n$ (with $\mathbf{x}_i \in \mathbb{R}^d$), be a dataset. Define a weighted graph $G = (V, E)$, with edge weight

$$w_{ij} = sim(\mathbf{x}_i, \mathbf{x}_j)$$

where $sim(\mathbf{x}_i, \mathbf{x}_j)$ denotes the similarity between points $\mathbf{x}_i$ and $\mathbf{x}_j$.

For instance, using the Gaussian similarity

$$w_{ij} = sim(\mathbf{x}_i, \mathbf{x}_j) = \exp\left\{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right\}$$

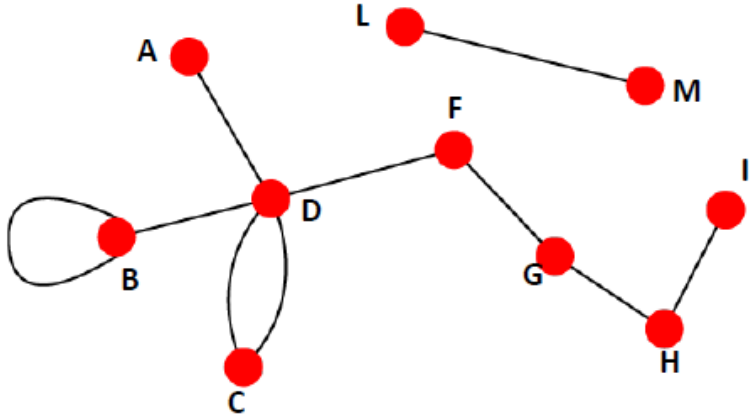
where σ is the spread parameter.

We can add a constraint that an edge exists between i and j only if w_{ij} exceeds a threshold

Directed vs Undirected Graphs

Undirected

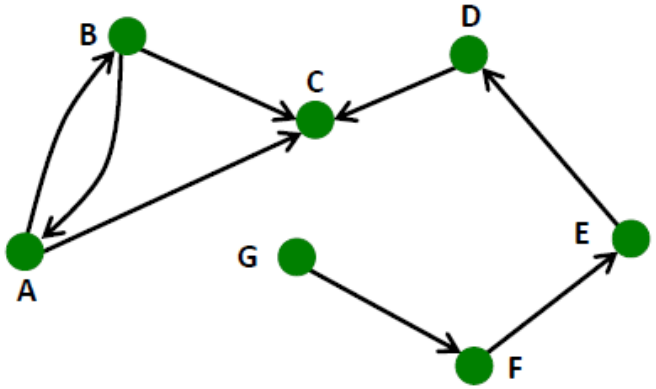
- **Links:** undirected (symmetrical, reciprocal)



- **Examples:**
  - Collaborations
  - Friendship on Facebook

Directed

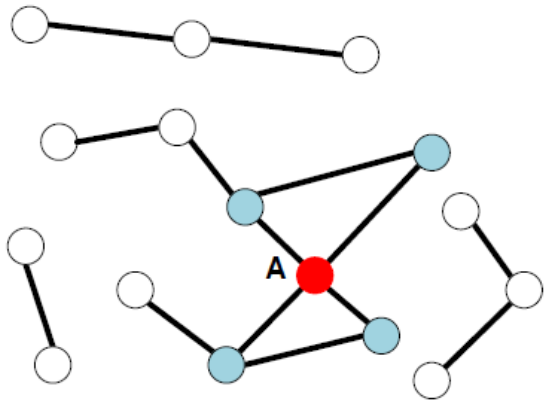
- **Links:** directed (arcs)



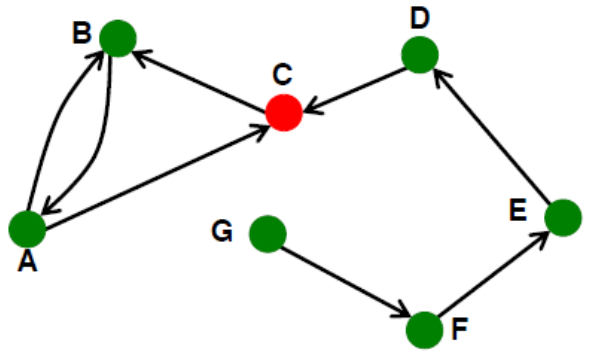
- **Examples:**
  - Phone calls
  - Following on Twitter

Node Degree

Undirected



Directed



Source: Node with $k^{in} = 0$
Sink: Node with $k^{out} = 0$

Node degree, k_i: the number of edges adjacent to node ***i***

$$k_A = 4$$

Avg. degree: $\bar{k} = \langle k \rangle = \frac{1}{N}\sum_{i=1}^{N} k_i = \frac{2E}{N}$

In directed networks we define an **in-degree** and **out-degree.**

The (total) degree of a node is the sum of in- and out-degrees.

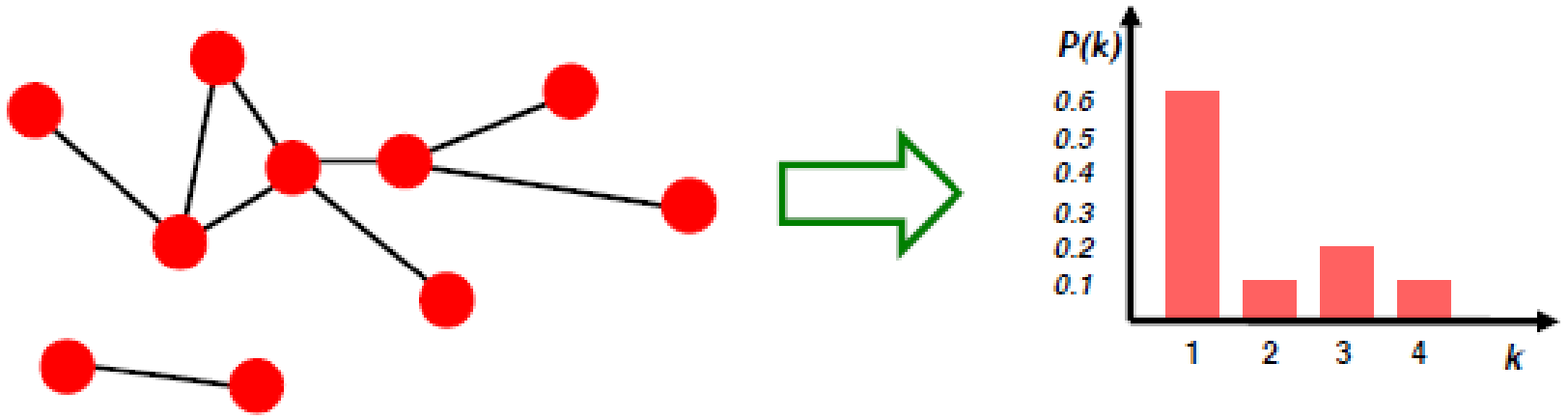
$$k_C^{in} = 2 \qquad k_C^{out} = 1 \qquad k_C = 3$$

$$\bar{k} = \frac{E}{N} \qquad \overline{k^{in}} = \overline{k^{out}}$$

Degree Distribution

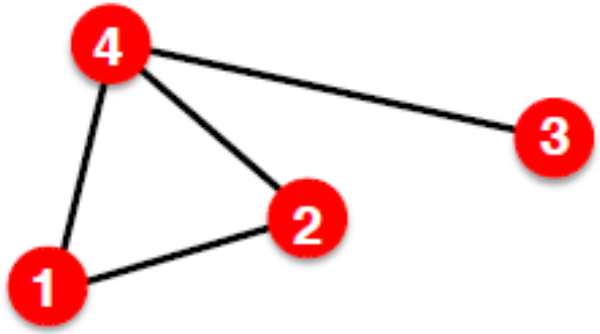
- The degree of a node $v_i \in V$ is denoted as $d(v_i)$ or d_i or k_i .
- The degree sequence of a graph is the list of the degrees of the nodes sorted in non-increasing order.
- Let N_k denote the number of vertices with degree k. The degree frequency distribution of a graph is given as $(N_0, N_1, \ldots, N_t)$ where t is the maximum degree for a node in G.
- Let X be a random variable denoting the degree of a node.
- The degree distribution of a graph gives the probability mass function f for X, given as f (0), f (1), . . . , f (t) where f (k) = P(X = k) = $\frac{N_k}{n}$ is the probability that a randomly chosen node has degree k.

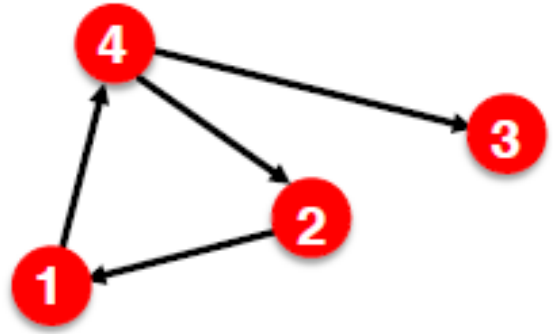
Degree distribution example



- The degree sequence of the graph is: (4, 3, 3, 2, 1, 1, 1, 1, 1, 1)
- The degree frequency distribution is: $(N_0, N_1, N_2, N_3, N_4) = (0,6,1,2,1)$
- The degree distribution is given as: $\big(f(0), f(1), f(2), f(3), f(4)\big) = (0,0.125,0.375,0.125,0.375)$
- Note: For directed graphs, we can have different in- and out- degree distributions

Representing Graphs: Adjacency Matrix





Some graphs may allow self-loops
Eg: Proteins, Hyperlinks

$A_{ij} = 1$ if there is a link from node *i* to node *j*

$A_{ij} = 0$ otherwise

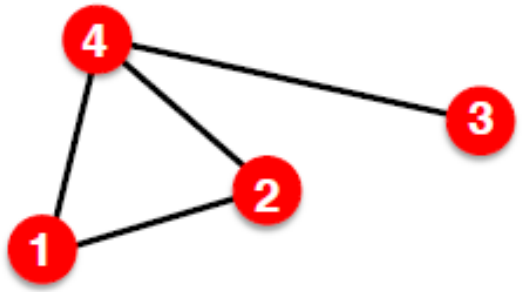
$$A = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

$$A = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

Note that for a directed graph (right) the matrix is not symmetric.

Adjacency Matrix

Undirected



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

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

$$k_i = \sum_{j=1}^{N} A_{ij}$$

Row Sum=Degree of i

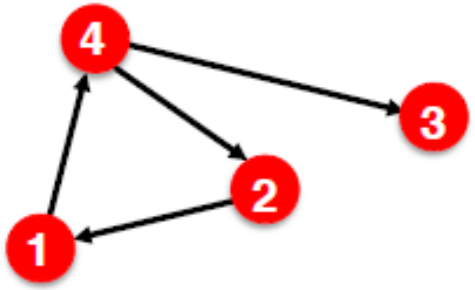
$$k_j = \sum_{i=1}^{N} A_{ij}$$

Col Sum=Degree of j

$$L = \frac{1}{2}\sum_{i=1}^{N} k_i = \frac{1}{2}\sum_{ij}^{N} A_{ij}$$

Total Number of Edges

Directed



$$A = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

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

$$k_i^{out} = \sum_{j=1}^{N} A_{ij}$$

Row Sum=Out-Degree of i

$$k_j^{in} = \sum_{i=1}^{N} A_{ij}$$

Col Sum=In-Degree of j

$$L = \sum_{i=1}^{N} k_i^{in} = \sum_{j=1}^{N} k_j^{out} = \sum_{i,j}^{N} A_{ij}$$

Total Number of Edges

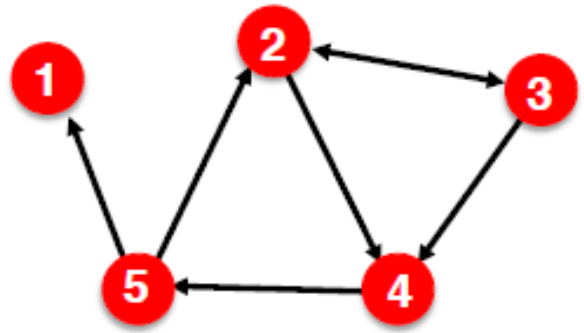
Adjacency Matrices are Sparse for Real World Graphs

- Most real world networks are sparse. $E \ll E_{max} = \frac{N}{2}(N-1)\ or\ \bar{k} \ll N-1$
- Consequence: Adjacency matrix is filled with zeros!
- Density of the matrix $\frac{E}{N^2}$: (WWW= 1.51×10^{-5} , MSN IM=$2.27x\ 10^{-8}$)

NETWORK	NODES	LINKS	DIRECTED/ UNDIRECTED	N	L	<k>
Internet	Routers	Internet connections	Undirected	192,244	609,066	6.33
WWW	Webpages	Links	Directed	325,729	1,497,134	4.60
Power Grid	Power plants, transformers	Cables	Undirected	4,941	6,594	2.67
Phone Calls	Subscribers	Calls	Directed	36,595	91,826	2.51
Email	Email Addresses	Emails	Directed	57,194	103,731	1.81
Science Collaboration	Scientists	Co-authorship	Undirected	23,133	93,439	8.08
Actor Network	Actors	Co-acting	Undirected	702,388	29,397,908	83.71
Citation Network	Paper	Citations	Directed	449,673	4,689,479	10.43
E. Coli Metabolism	Metabolites	Chemical reactions	Directed	1,039	5,802	5.58
Protein Interactions	Proteins	Binding interactions	Undirected	2,018	2,930	2.90

Representing Graphs : Edge List

- For large sparse graphs, it is more efficient to represent the graphs as a list of edges
- Allows us to quickly retrieve all the neighbors of a given node
  - 1:
  - 2: 3,4
  - 3: 2,4
  - 4: 5
  - 5: 1,2

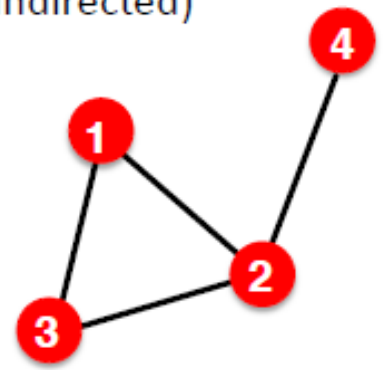


Node and Edge Attributes

- **Possible options:**
  - Weight (*e.g.*, frequency of communication)
  - Ranking (best friend, second best friend…)
  - Type (friend, relative, co-worker)
  - Sign: Friend vs. Foe, Trust vs. Distrust
  - Properties depending on the structure of the rest of the graph: Number of common friends

More Types of Graphs

- **Unweighted**
  (undirected)



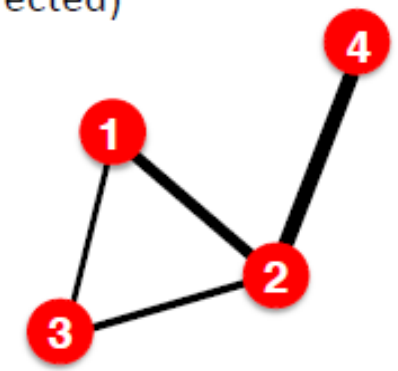
$$A_{ij} = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

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

$$E = \frac{1}{2}\sum_{i,j=1}^{N} A_{ij} \qquad \bar{k} = \frac{2E}{N}$$

Examples: Friendship, Hyperlink

- **Weighted**
  (undirected)



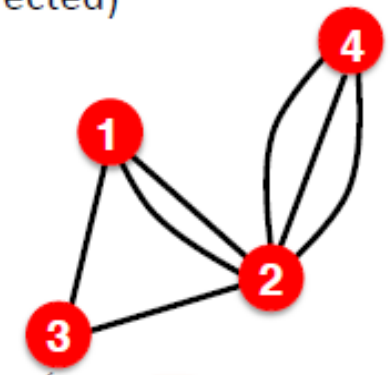
$$A_{ij} = \begin{pmatrix} 0 & 2 & 0.5 & 0 \\ 2 & 0 & 1 & 4 \\ 0.5 & 1 & 0 & 0 \\ 0 & 4 & 0 & 0 \end{pmatrix}$$

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

$$E = \frac{1}{2}\sum_{i,j=1}^{N} nonzero(A_{ij}) \qquad \bar{k} = \frac{2E}{N}$$

Examples: Collaboration, Internet, Roads

- **Multigraph**
  (undirected)



$$A_{ij} = \begin{pmatrix} 0 & 2 & 1 & 0 \\ 2 & 0 & 1 & 3 \\ 1 & 1 & 0 & 0 \\ 0 & 3 & 0 & 0 \end{pmatrix}$$

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

$$E = \frac{1}{2}\sum_{i,j=1}^{N} nonzero(A_{ij}) \qquad \bar{k} = \frac{2E}{N}$$

Examples: Communication, Collaboration

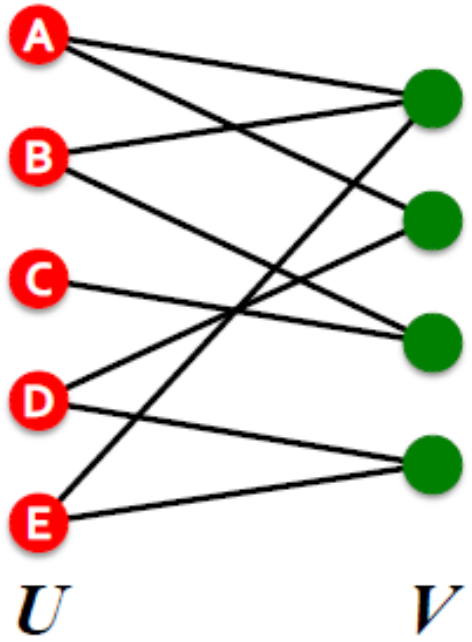
Question: Calculate the formula for E if the graph has self-loops

Path and Distance

- **Walk:** in a G between nodes x and y is an ordered sequence of vertices, starting at x and ending at y, $x = v_0, v_1, \dots, v_{t-1}, v_t = y$, such that there is an edge between every pair of consecutive vertices, that is, $(v_{i-1}, v_i) \in E, \forall\, i = 1,2, \dots, t$
  - Length of the walk, t = hops = number of edges along the walk
- **Path:** is a walk with distinct vertices (with the exception of the start and end vertices, which are called cycles).
  - A path of minimum length between nodes x and y is called a shortest path
  - Length of the shortest path = distance between x and y, denoted as d(x, y) or h_{xy}.
  - If no path exists between the two nodes, the distance is assumed to be d(x, y) = ∞.
- Two nodes x and y are **connected** if there exists a path between them.
- In directed graphs, paths need to follow edge directions: $\mathrm{d}(\mathrm{x}, \mathrm{y}) \neq d(y, x)$

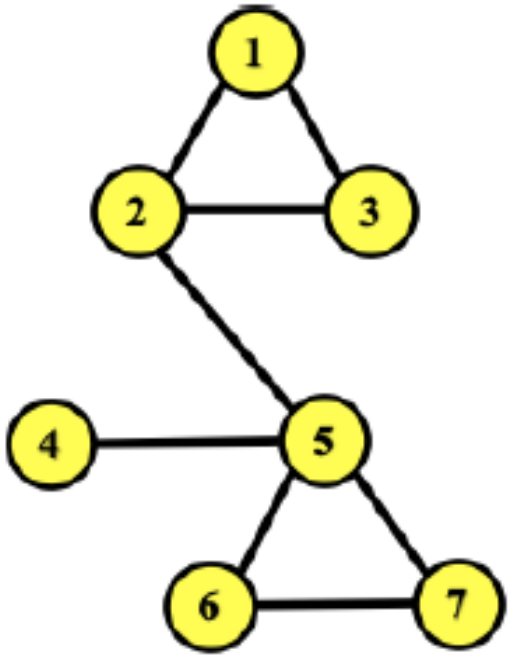
Bipartite Graph

- **Bipartite graph** is a graph whose nodes can be divided into two disjoint sets ***U*** and ***V*** such that every link connects a node in ***U*** to one in ***V***; that is, ***U*** and ***V*** are **independent sets**
- **Examples:**
  - Authors-to-Papers (they authored)
  - Actors-to-Movies (they appeared in)
  - Users-to-Movies (they rated)
  - Recipes-to-Ingredients (they contain)
- **"Folded" networks:**
  - Author collaboration networks
  - Movie co-rating networks



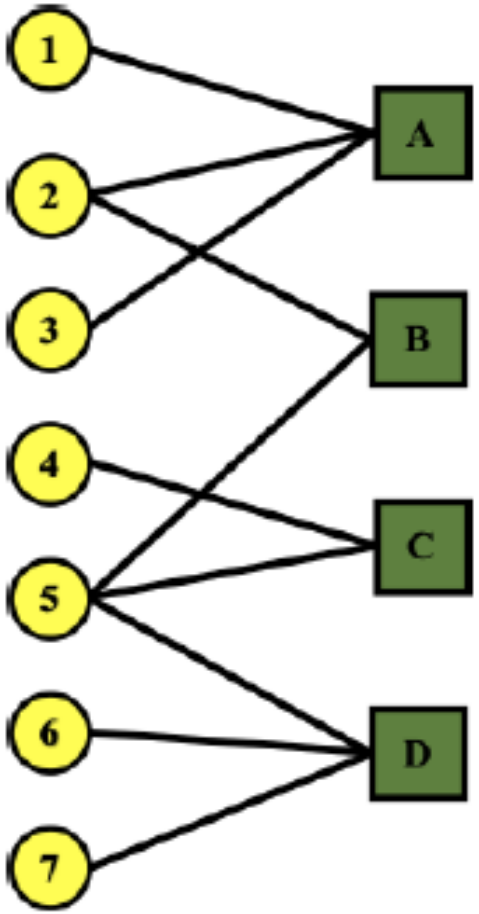
Folded/Projected Bipartite Graphs

Projection U

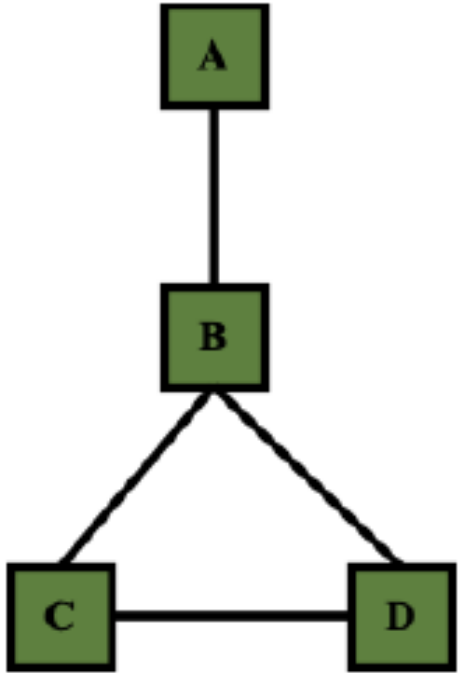


1 is connected to 2 via A
1 is connected to 5 via A,2,B

U V



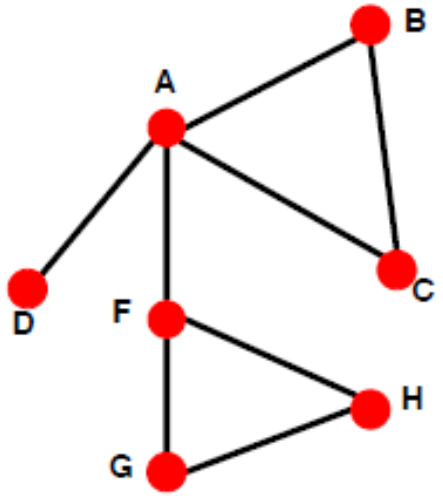
Projection V



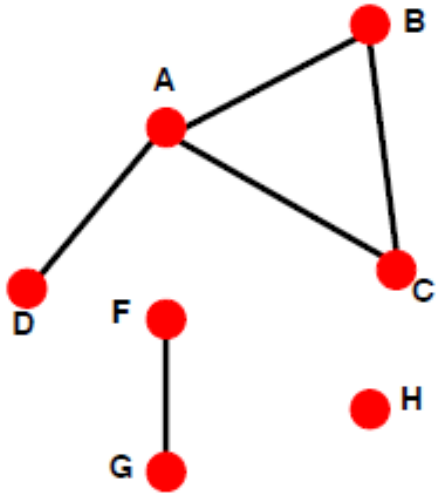
A is connected to B via 2
A is connected to D via 2,B,5

Connectivity of Undirected Graphs

- **Connected (undirected) graph:**
  - Any two vertices can be joined by a path
- A disconnected graph is made up by two or more connected components



Connected graph



Dis-connected graph

Largest Component:
Giant Component

Isolated node (node H)

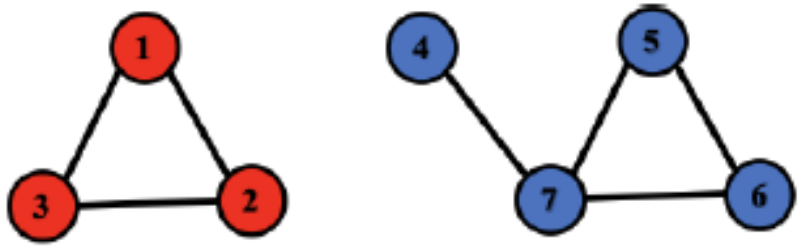
Size of the largest connected component (Giant component)

- Largest set where any two vertices can be joined by a path

Connectivity: Example

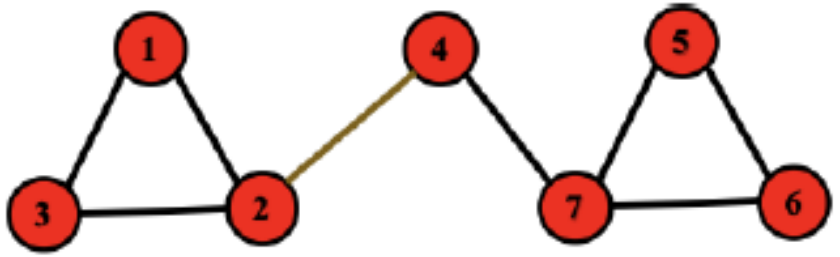
The adjacency matrix of a network with several components can be written in a block- diagonal form, so that nonzero elements are confined to squares, with all other elements being zero:

Disconnected



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

Connected

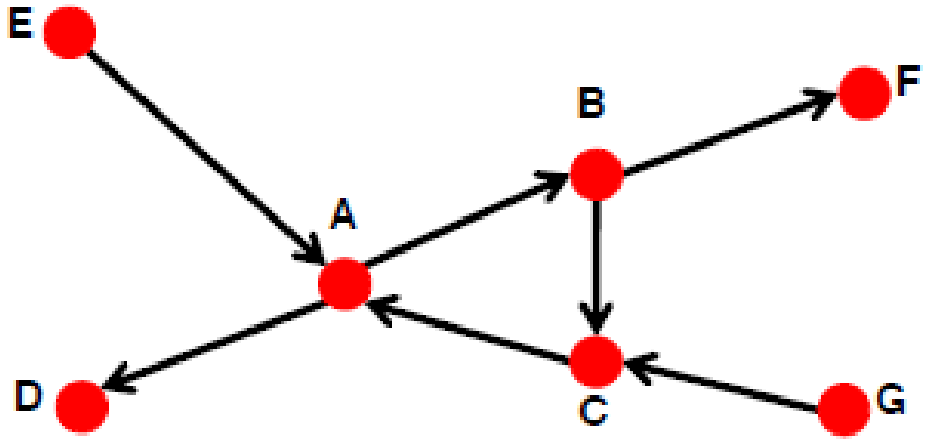


$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{pmatrix}$$

Connected Components can be determined using BFS.
Start from a random node and perform BFS.
Label node visits.
All nodes that can be visited are in the connected component.
Otherwise, find an unvisited node and start BFS again.

Connectivity of Directed Graphs

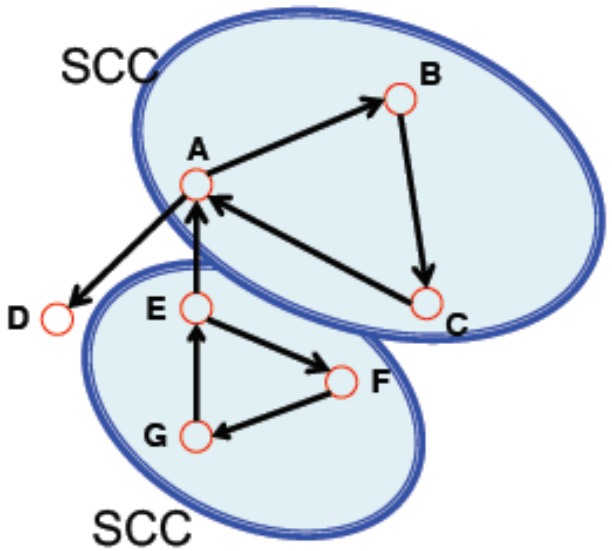
- **Strongly connected directed graph**
  - has a path from each node to every other node and vice versa (e.g., A-B path and B-A path)
- **Weakly connected directed graph**
  - is connected if we disregard the edge directions

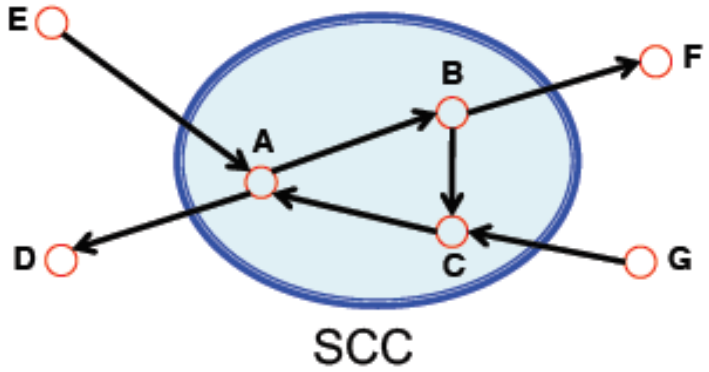


Graph on the left is connected but not strongly connected
(e.g., there is no way to get from F to G by following the edge directions).

Connectivity of Directed Graphs

- **Strongly connected components (SCCs)** can be identified, but not every node is part of a nontrivial strongly connected component.
- In-component: nodes that can reach the SCC,
- Out-component: nodes that can be reached from the SCC.





Network Diameter

- **Eccentricity of a node v_i: $e(v_i)$ =** Max distance (shortest path) from any other node v_j in the graph
- **Radius of a connected graph:** Minimum eccentricity of any node, $\min_i e(v_i)$
- **Diameter:** The maximum distance between any pair of nodes in a graph, $\max_i e(v_i)$
  - Effective diameter: min number of hops in which a large fraction (e.g. 90%) of all connected pair of nodes can reach each other
- **Average path length** for a connected graph or a strongly connected directed graph
  - $\bar{h} = \frac{1}{2E_{max}} \sum_{i,j \neq i} h_{ij}$
    - h_{ij} is the distance from node i to j
    - E_{max} is the max number of edges (total number of node pairs) = $\frac{n(n-1)}{2}$
  - Many times we compute the average only over the connected pairs of nodes (that is, we ignore "infinite" length paths)
  - Note that this measure also applies to (strongly) connected components of a graph

Centrality Analysis

A centrality is a function $c: V \to \mathbb{R}$, that induces a ranking on V.

Degree Centrality: The simplest notion of centrality is the degree d_i of a vertex v_i – the higher the degree, the more important or central the vertex.

Eccentricity Centrality: Eccentricity centrality is defined as:

$$c(v_i) = \frac{1}{e(v_i)} = \frac{1}{\max_j \{d(v_i, v_j)\}}$$

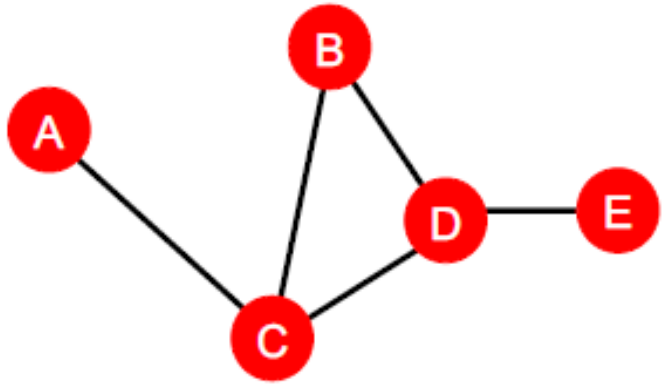
The less eccentric a node is, the more central it is.

Closeness Centrality: closeness centrality uses the sum of all the distances to rank how central a node is

$$c(v_i) = \frac{1}{\sum_j d(v_i, v_j)}$$

Betweenness Centrality

- A node is important if it lies on many shortest paths between other nodes.
- $c_v = \sum_{s \neq v \neq t} \frac{\#\ shortest\ paths\ between\ s\ and\ t\ that\ contain\ v}{\#\ shortest\ paths\ between\ s\ and\ t}$



$c_A = c_B = c_E = 0$

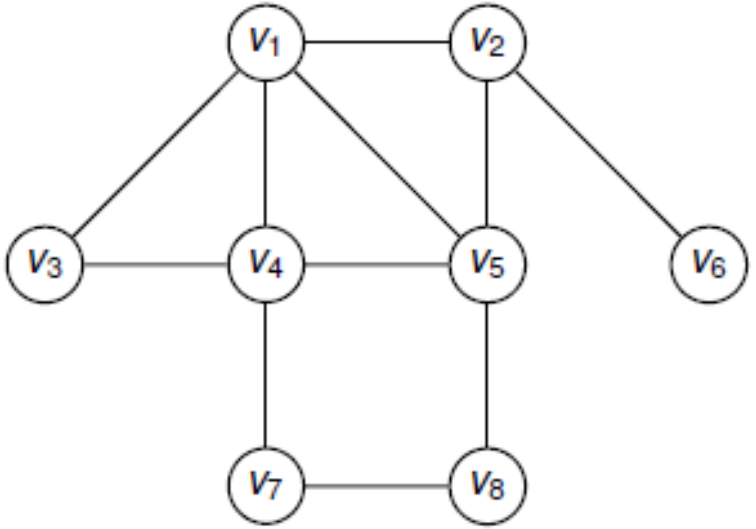
$c_C = 3$

(A-C-B, A-C-D, A-C-D-E)

$c_D = 3$

(A-C-D-E, B-D-E, C-D-E)

Example



Centrality	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8
Degree	4	3	2	4	4	1	2	2
Eccentricity $e(v_i)$	0.5 2	0.33 3	0.33 3	0.33 3	0.5 2	0.25 4	0.25 4	0.33 3
Closeness $\sum_j d(v_i, v_j)$	0.100 10	0.083 12	0.071 14	0.091 11	0.100 10	0.056 18	0.067 15	0.071 14
Betweenness	4.5	6	0	5	6.5	0	0.83	1.17

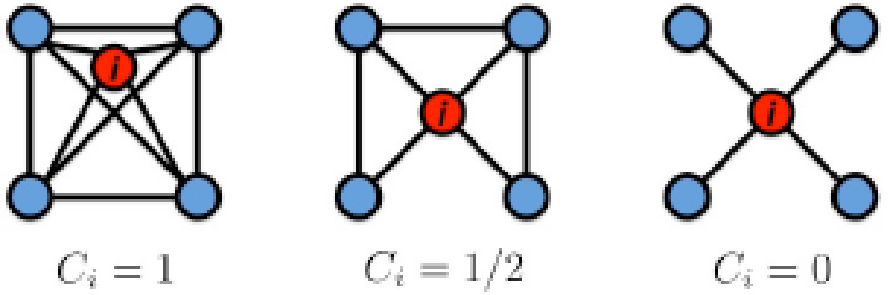
These can serve as node features in machine learning tasks

Prestige or Eigenvector Centrality

- A node v is important if surrounded by important neighboring nodes $u \in N(v)$.
- We model the centrality of node v as the sum of the centrality of neighboring nodes: $c_v = \frac{1}{\lambda}\sum_{u \in N(v)} c_u$
- Notice that the above equation models centrality in a recursive manner. How do we solve it?
- By rewriting the equation in matrix form, $\lambda\, c = A\, c$, where A is adjacency matric and c is the centrality vector
- We see that centrality is the **eigenvector**!
- The largest eigenvalue λ_{max} is always positive and unique (by Perron-Frobenius Theorem).
- The leading eigenvector c_{max} is used for centrality.

Clustering Coefficient

- **Clustering coefficient (for undirected graphs):**
  - How connected are i's neighbors to each other?
  - Node i with degree k_i
  - $C_i \in [0,1]$
  - $C_i = \frac{2e_i}{k_i(k_i-1)}$

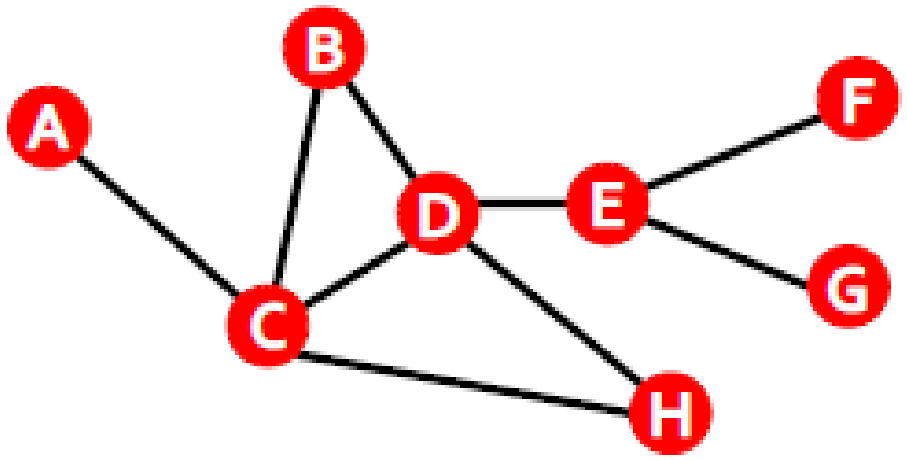


$C_i = 1$ $C_i = 1/2$ $C_i = 0$

where e_i is the number of edges between the neighbors of node i

- **Average clustering coefficient of the Graph:** $C = \frac{1}{N}\sum_{i=1}^{N} C_i$
  - Clustering coefficient is undefined (or defined to be 0) for nodes with degree 0 or 1

Clustering Coefficient Example



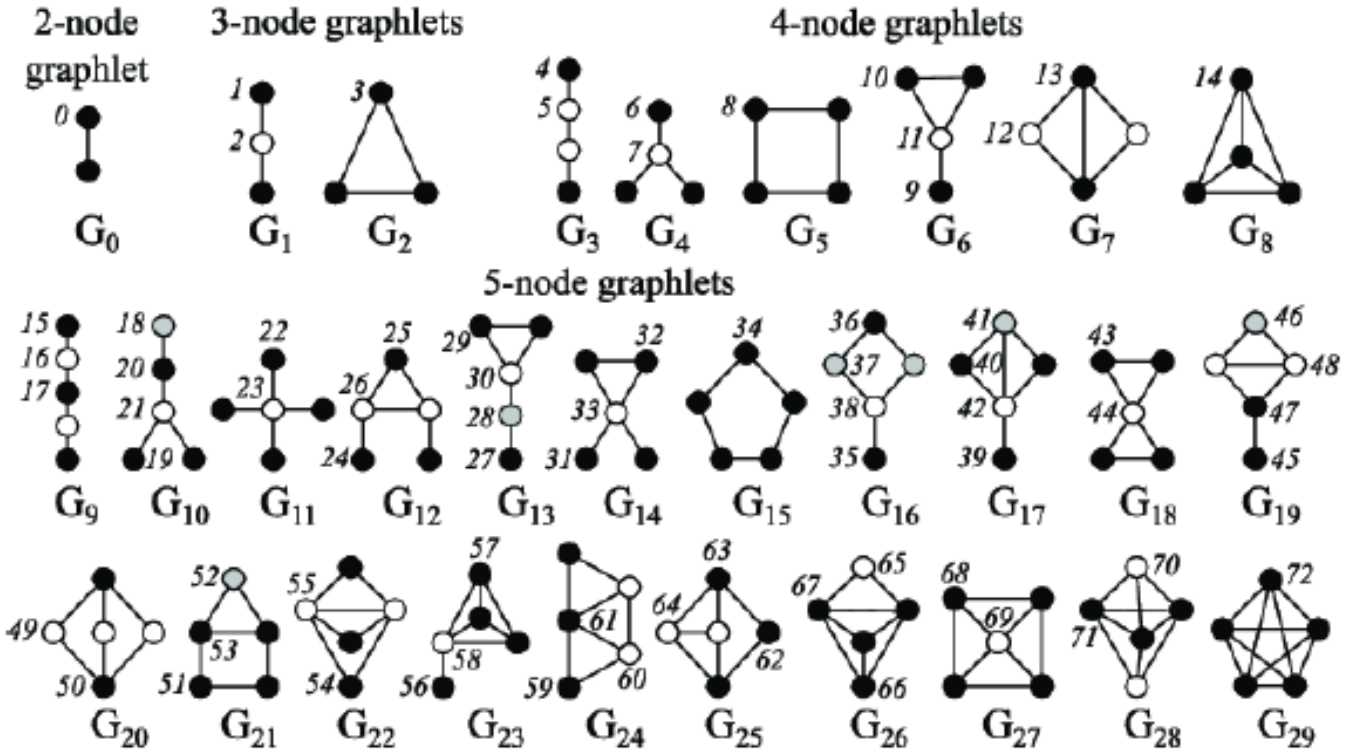
$k_B=2,\ e_B=1,\ C_B=2/2=1$

$k_D=4,\ e_D=2,\ C_D=4/12=1/3$

Question: Compute clustering coefficient for every node and determine the average clustering coefficient for the graph.

Graphlets

- **Observation:** Clustering coefficient counts the #(triangles) in the ego-network
- We can generalize the above by counting #(pre-specified subgraphs, i.e., **graphlets**).
- Graphlets are rooted connected non-isomorphic subgraphs:



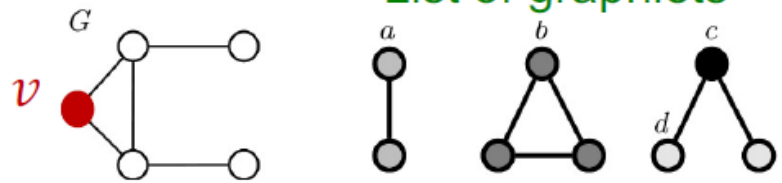
Graphlet Degree Vector (GDV): Graphlet-base features for nodes

- **Degree** counts **#(edges)** that a node touches
- **Clustering coefficient** counts **#(triangles)** that a node touches
- **GDV** counts **#(graphlets)** rooted at a given node

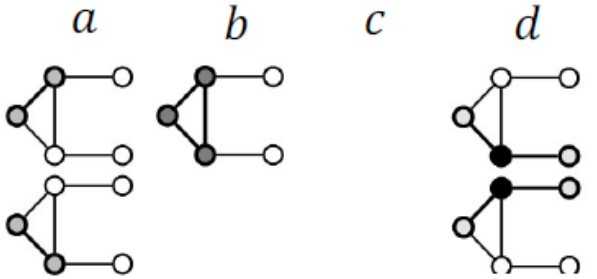
Graphlets

- Considering graphlets on 2 to 5 nodes we get:
- **Vector of 73 coordinates** is a signature of a node that describes the topology of node's neighborhood
- Captures its interconnectivities out to a **distance of 4** hops
- Graphlet degree vector provides a measure of a **node's local network topology**:
- Comparing vectors of two nodes provides a more detailed measure of local topological similarity than node degrees or clustering coefficient.

- **Example:**



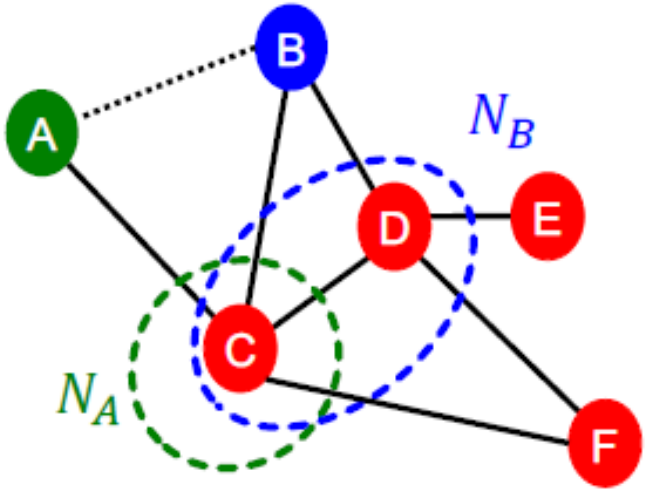
Graphlet instances:



GDV of node v:
a, b, c, d
[2,1,0,2]

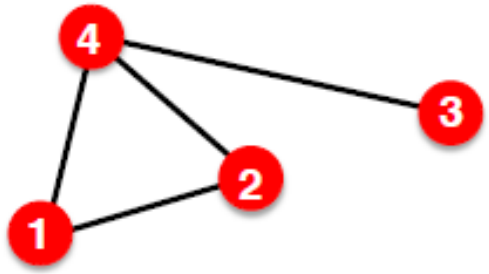
Link Feature: Local Neighborhood Overlap

- **Captures # neighboring nodes shared between two nodes u and v**
- **Number of Common neighbors:** $|N(u) \cap N(v)|$
  - Example: $|N(A) \cap N(B)| = |\{C\}| = 1$
- **Jaccard's coefficient:** $\frac{|N(u) \cap N(v)|}{|N(u) \cup N(v)|}$
  - Example: $\frac{|N(A) \cap N(B)|}{|N(A) \cup N(B)|} = \frac{1}{2}$
- **Adamic-Adar index:** $\sum_{s \in N(u) \cap N(v)} \frac{1}{log(k_s)}$
  - Example: $\frac{1}{\log(k_C)} = \frac{1}{\log(4)}$



Global Neighborhood Overlap

- Katz Index: use the paths of all lengths between a given pair of nodes
  - These can be computed using powers of the graph adjacency matrix
- Recall: $A_{uv} = 1$ if $u \in N(v)$
- Let P_{uv}^K= # of paths of length K between u and v
  - $P^K = A^K$



Node 1's neighbors

#paths of length 1 between Node 1's neighbors and Node 2

$$P_{12}^{(2)} = A_{12}^2$$

$$A^2 = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix} \times \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 2 & 1 & 1 & 1 \\ 1 & 2 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 3 \end{pmatrix}$$

Power of adjacency

Global Neighborhood Overlap

- **Katz index** between v_1 and v_2 is calculated as

Sum over *all path lengths*

$$S_{v_1 v_2} = \sum_{l=1}^{\infty} \beta^l \, A^l_{v_1 v_2}$$

#paths of length l between v_1 and v_2

$0 < \beta < 1$: discount factor

- Katz index matrix is computed in closed-form:

$$\boldsymbol{S} = \sum_{i=1}^{\infty} \beta^i \boldsymbol{A}^i = \underbrace{(\boldsymbol{I} - \beta \boldsymbol{A})^{-1}}_{= \sum_{i=0}^{\infty} \beta^i \boldsymbol{A}^i} - \boldsymbol{I},$$

by geometric series of matrices

Summary: Key properties of the network

Property	Notation
Degree Distribution	P(k)
Path Length	H
Clustering Coefficient	C
Connected Components	s
Node centrality measures	
Graphlet count vector	
Katz Index matrix	S

Node features can be useful for predicting influential nodes in a graph for:

- Predicting celebrity users in a social network
- Targeted promotions/advertising
- Recommendations
- Predicting protein functionality in a protein-protein interaction network