

1.Introduction, Structure of Graphs

Anil

Graph is a very general data structure that describes many real life datasets like social networks, economic networks, communication networks, knowledge graphs, drug interactions, atoms in a molecule etc. They are a general language for describing complex systems of interacting entities. Usually in machine learning, we train on features describing entities, but we don't take advantage of how these entities are interacting with each other. By explicitly modeling these relationships we can achieve better performance. It will help us design better models in drug design, recommendation systems and AI reasoning.

Given a network we might be having the following problems.

Predicting the class of different entities/nodes **Node Classification**. Predict whether two nodes that are not connected now might connect in future **Link Prediction**. Identifying communities with in the network **Community Detection** or measuring similarity of two nodes or networks **Network Similarity** and etc.

Applications

For example, given a social network we might be interested in identifying the communities with in it, and knowing what communities a given person's ego-network has access to and etc..

If a blackout happens in a major power grid network, we need to understand how to minimize its impact. Develop quantitative tools to assess interplay between network structure and the dynamical processes on the network to make it more robust to failures.

We should be able to figure out entity embeddings not just taking into account its own features, but how it fits in the network and its neighbourhood. So that we can use those embeddings in whatever downstream machine learning task.

Understanding polarization on social media using retweet networks or fighting misinformation. Making use of the graph structure for example gave us much better prediction accuracies in the task of identifying hoax articles on wikipedia. You can read more about it in the paper [Disinformation on the Web: Impact, Characteristics and Detection of Wikipedia Hoaxes](#). They formed a network based on how a given wiki article links to other articles on wikipedia. They were able to predict with 86% accuracy, where humans were able to predict with 66% accuracy.

There are applications where you need to be able to predict virality and understand information cascades in social networks. If you are a social media marketer, you need to know which influencers will give you best results.

You can read about how linkedin made use of networks for product adoption using invitation cascades. 60-90% of the linkedin users signed up due to an invitation from another user. [Global Diffusion via Cascading Invitations: Structure, Growth and Homophily](#)

In case of Biomedicine, we need to understand Protein-Protein interaction networks. Or how when only side effects of a single drug usage were known in patients, how taking multiple drugs will affect them, as it's not tractable to do trials of how each drug will interact with every other drug in a patient with multiple diseases or conditions. This is an important problem as almost half of the people above 70 years of age take more than 5 drugs.

▼ Structure of Networks

In this section we will learn about the basics of networks/graphs and define some of its basic properties and also an introduction to common graph related terms/nomenclature.

A network is a collection of objects where some pairs of objects are connected by links.

Objects: nodes, vertices N

Interactions: links, edges E

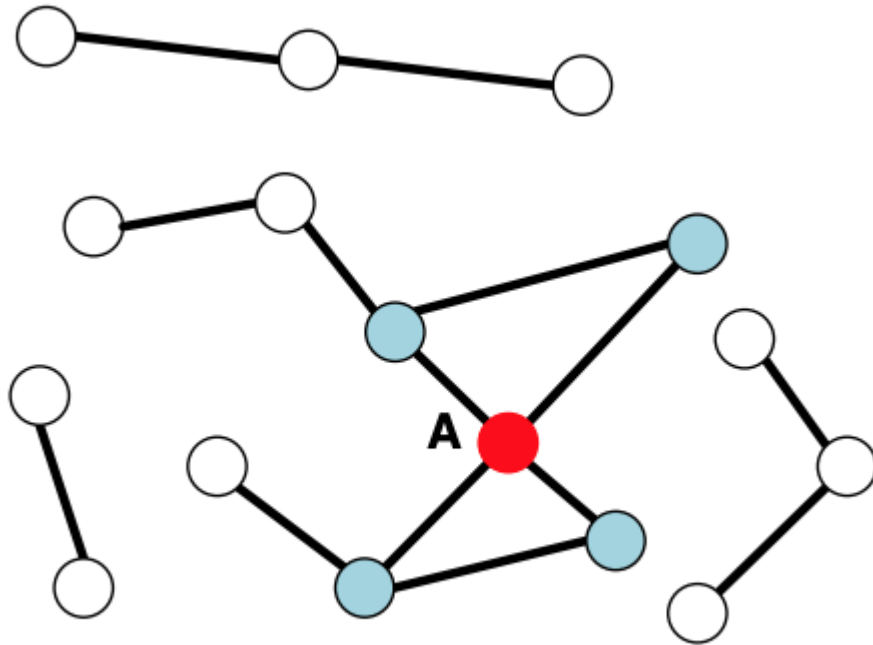
System: network, graph $G(N, E)$

You build a graph or network by defining what are nodes and what are edges. The choice of proper network representation of a given domain/problem determines our ability to use networks. Sometimes the choice of nodes and edges will be unique and unambiguous, but in some cases it may not be unique and the way you assign these will determine the nature of the problem you can study.

Networks can be **directed** or **undirected**. A **directed** graph has edges whose direction matters. Where as an **undirected** graph has reciprocal edges. Examples of undirected edges are friendship on facebook, collaborations and etc. Where as follower network on twitter is a directed one.

Node Degree.

Node degree for a node in an undirected graph is defined as the number of edges adjacent to node i . Number of neighbours to a given node is the degree of that node.

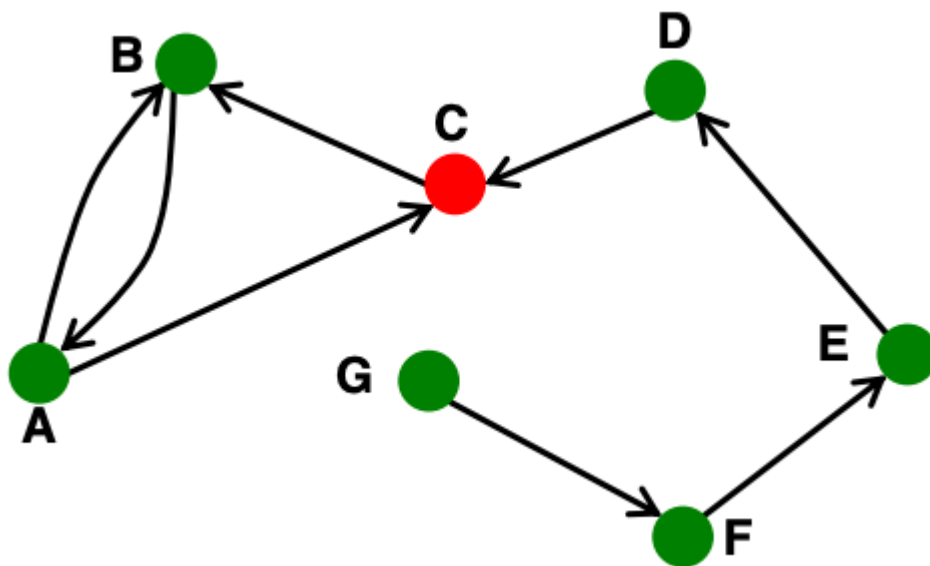


In the above undirected graph, for node A the degree $k_A = 4$

Average Node Degree in an undirected graph $\bar{k} = \langle k \rangle =$

$$\frac{1}{N} \sum_{i=1}^N k_i = \frac{2E}{N}$$

In case of a directed graph we define **in-degree** and **out-degree** of a node. The total degree of a node is the sum of both.



In the above directed graph for node C, $k_C^{\text{in}} = 2$, $k_C^{\text{out}} = 1$ and $k_C = 3$

Average degree in case of a directed graph is $\bar{k} = \frac{E}{N}$

If the **in-degree** of a node in a directed graph is zero, its called a source node and if the **out-degree** is zero its called a sink node.

Example graphs and their respective avg degrees.

Network	Nodes	Links	Category	N	E	\bar{k}
Internet	Routers	Connections	Undirected	192244	609066	6.33
WWW	Webpages	Links	Directed	325729	1497134	4.60
Power Grid	Power plants, transformers	Cables	Undirected	4941	6594	2.67
Phone Calls	Subscribers	Calls	Directed	36595	91826	2.51
Email	email addresses	emails	Directed	57194	103731	1.81
Science Collaborations	Scientists	Co-authorship	Undirected	23133	93439	8.08
Actor Network	Actors	Co-acting	Undirected	702388	29397908	83.71
Citation Network	Paper	Citations	Directed	449637	4689479	10.43
Protein Interactions	Proteins	Binding Interactions	Undirected	2018	2930	2.90

In the above table, you can see what the average degree for various graph datasets is.



Complete Graph

If every node is connected to every other node available in a graph then that graph is called a **complete graph**. The maximum number of edges in an undirected graph on N nodes is

$$E_{\max} = \binom{N}{2} = \frac{N(N-1)}{2}$$

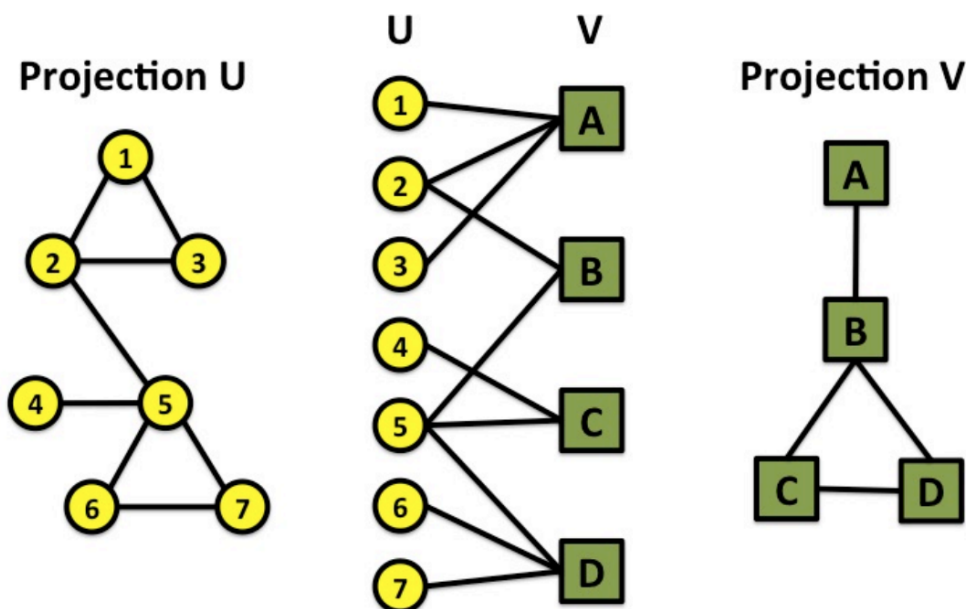
An undirected graph with $E = E_{\max}$ is called a complete graph, and

Bipartite Graph

Bipartite graph is a graph whose nodes can be divided into two disjoint sets U and V such that every edge connects a node from U to a node from V . There are no edges that connect nodes within either U or V . An example bipartite graph can be Authors to Papers. Where the edges represent authorship. Other examples are Actors-to-movies, Users-to-Movies, Recipes-to-ingredients and etc.

You can fold or do a projection on a bipartite graph. From the Author-to-Papers bipartite graph, where authors are one kind of nodes and papers are a different kind of nodes, you can do a projection on paper nodes to get an Author collaboration network with just the nodes of kind Authors. Where two author nodes that worked on the same paper are connected with an edge.

Below image shows how you can do a projection on a small bipartite graph.



Say for example you have co-actor bipartite network of nodes containing actors and the movies they acted in. Your goal is to

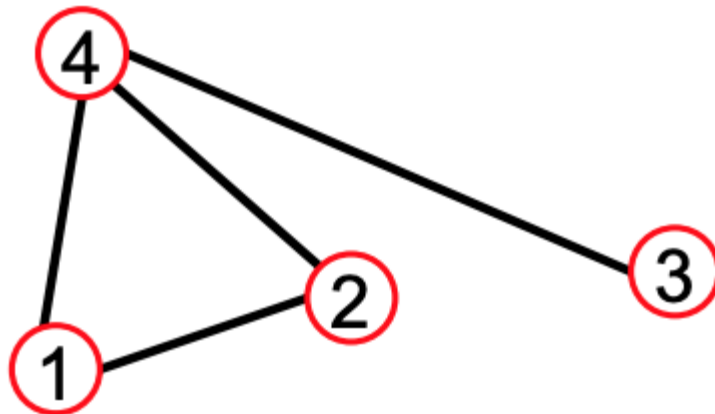
identify movies that are similar. One way of doing it is, create a vector of length actors and then have 1's if a given actor is in that movie and 0s if the actor is not in it. After we have all the movie vectors, we can use cosine similarity or other similarity metrics to identify movies that are similar. Instead of doing this if we fold the bipartite graph removing the actor nodes and creating edges between movies in place of them. We get a new graph and we can cluster this folded graph and identify movies similar to each other.

Adjacency Matrix

One way we can represent graphs is using Adjacency Matrix A , its a $N \times N$ matrix where every element represents the presence of an edge. $A_{ij} = 1$ if there is an edge between nodes i and j . $A_{ij} = 0$ if there is no edge between i and j .

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

$A_{\text{undirected}}$ is a sample adjacency matrix for the undirected graph below



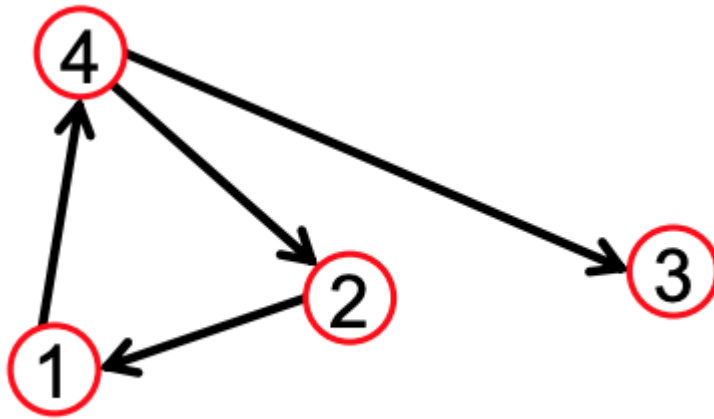
. The adjacency matrix of an undirected graph is symmetric.

For an undirected graph

$$A_{ij} = A_{ji}, \quad k_i = \sum_{j=1}^N A_{ij}, \quad E = \frac{1}{2} \sum_{i=1}^N k_i = \frac{1}{2} \sum_{ij} A_{ij}$$

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

A_{directed} is a sample adjacency matrix for the directed graph below



For a directed graph

$$A_{ij} \neq A_{ji} \quad k_i^{\text{out}} = \sum_{j=1}^N A_{ij}, \quad k_i^{\text{in}} = \sum_{j=1}^N A_{ji},$$

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

For most real world networks are sparse $E \ll E_{\text{max}}$ or $\bar{k} \ll N$. As we saw in the table above, the average node degree is in the order of 10 even for networks with nodes in the orders of $1e6$

So to take advantage of this, we can also represent graphs as just a list of its edges. We also call it edge list. Most python graph libraries make use of this. They are easier to work with even if the graph is very large but sparse. And allows us to quickly retrieve all the neighbours of a given node. For the above undirected graph the edge list will be (1, 4), (2, 1), (4, 2), (4, 3)

Edge Attributes

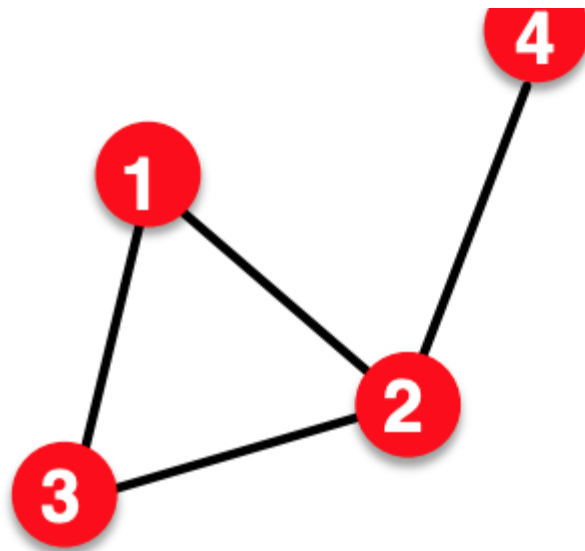
Edges of a graph can have various attributes, like

- weight of the connection between two nodes(e.g frequency of the communication)
- Ranking of the connection(e.g best friend, second best friend)
- type of the connection (e.g friend, coworker, relative)
- sign (e.g friend vs foe, trust vs distrust)
- properties depending on the neighbourhood of the node, rest of the graph structure (e.g number of common friends)

You can replace the 1s in the adjacency matrix with the respective weights.

Types of graphs.

Unweighted



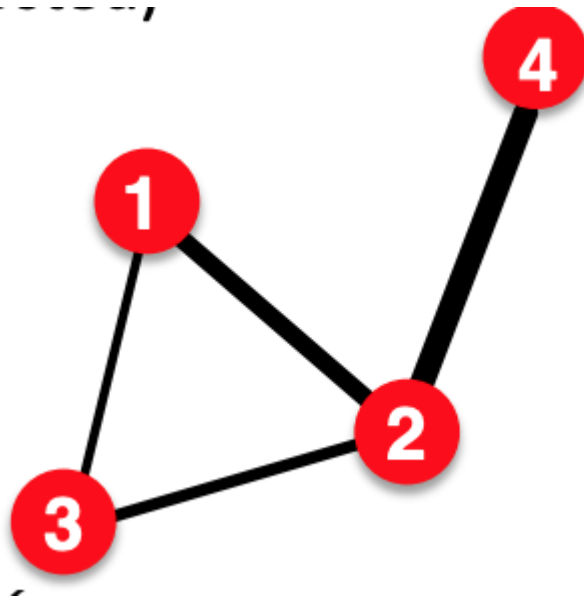
$$A_{\text{unweighted}} = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

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

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

Examples: Friendship, Hyperlinks

Weighted



Weighted graphs are graphs with unequal edge weights. Instead of 1s we use the edge weight in the Adjacency matrix like below.

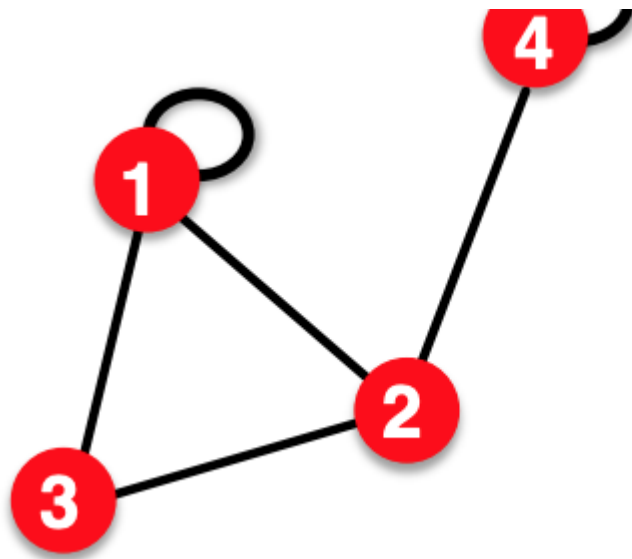
$$A_{\text{weighted}} = \begin{pmatrix} 0 & 2 & 0.5 & 0 \\ 2 & 0 & 1 & 4 \\ 0.5 & 1 & 0 & 0 \\ 0 & 4 & 0 & 0 \end{pmatrix}$$

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

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

Examples: Collaboration, Internet, Roads.

Self-edges(self-loops)



Graphs where some nodes have edges to itself. In this kind of graph, the diagonal of the matrix has non-zeros.

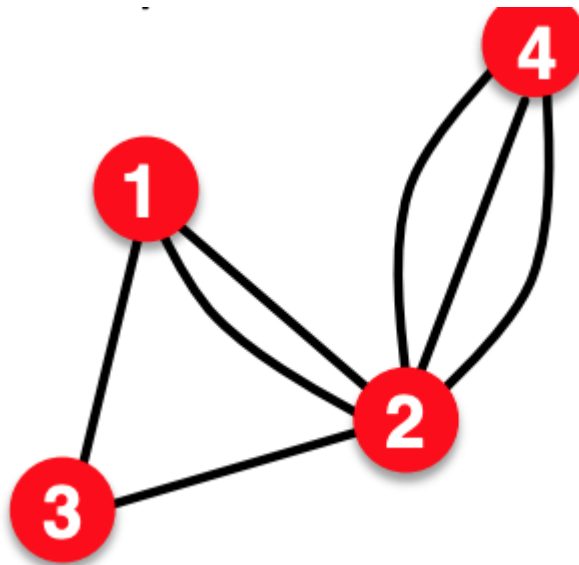
$$A_{\text{self-loops}} = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}$$

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

$$E = \frac{1}{2} \sum_{i,j=1, i \neq j}^N A_{ij} + \sum_{i=1}^N A_{ii}$$

Examples: Proteins, Hyperlinks

Multigraph



Graphs where the same nodes have multiple edges.

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

$$A_{ii} = 0, \quad A_{ij} = A_{ji} \quad \forall i, j \in \{1, 2, \dots, N\}$$

Connectivity

Undirected

A connected graph is a graph where there exists a path for each node to reach every other node. Not all nodes in a graph can reach each other in most cases. So most graphs are disconnected. And a disconnected graph is made up of more than 1 connected components.

Largest Connected component or the largest set of nodes that can reach each other in a graph is also called its **Giant Component**. And nodes that can't reach any other node are called **Isolated nodes**.

By erasing a single edge, if we make a connected graph disconnected, we call that edge **bridge edge**. And by erasing a single node, if we make a connected graph disconnected, we call that node **articulation node**.

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

Here is the adjacency matrix for a graph with two components. The first 4 nodes have no edge linking the last 3 nodes. The squares made up of dots in the below matrices are two different components in the same graph.

$$A_{\text{disconnectedgraph}} = \begin{pmatrix} . & . & . & 0 & 0 & 0 & 0 \\ . & . & . & 0 & 0 & 0 & 0 \\ . & . & . & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & . & . & . & . \\ 0 & 0 & 0 & . & . & . & . \\ 0 & 0 & 0 & . & . & . & . \end{pmatrix}$$

We can add a bridge edge between node 2 and node 4 like below to make the graph connected. Look at the 1's in the below adjacency matrix.

$$A_{\text{connectedgraph}} = \begin{pmatrix} . & . & . & 0 & 0 & 0 & 0 \\ . & . & . & 1 & 0 & 0 & 0 \\ . & . & . & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & . & . & . & . \\ 0 & 0 & 0 & . & . & . & . \\ 0 & 0 & 0 & . & . & . & . \end{pmatrix}$$

We make use of the block-diagonal form of disconnected components, to batch unrelated graphs, when we are training graph neural networks and etc. Say for example we are training a graph classification algorithm. Instead of passing all the graphs separately through our neural network, we batch unrelated graphs as disconnected components of the same big graph to speed up the training process. There is no way for information or messages to pass from one unrelated graph to another, as there are no edges connecting them.

Directed Graph

Strongly connected directed graph: Every node has a path from every other node and vice versa.

Weakly connected directed graph: If we disregard the edge directions, and the graph becomes a connected graph, we call it weakly connected.

In case of a directed graph, once we identify the Strongly Connected Graph(SCC), the nodes that can reach the SCC are called **In-component** nodes and the nodes that can be reached from the SCC

are called **Out-component** nodes. A node cannot be both in-


▼ Chapters

1. [Introduction, Structure of Graphs](#)
2. [Properties of Networks and Random Graph Models](#)
3. [Motifs and Structural Roles in Networks](#)
4. [Community Structure in Networks](#)
5. [Spectral Clustering](#)
6. [Message Passing and Node Classification](#)
7. [Graph Representation Learning](#)
8. [Graph Neural Networks](#)
9. [Graph Neural Networks - Pytorch Geometric](#)
10. [Deep Generative Models for Graphs](#)
11. [Link Analysis: PageRank](#)
12. [Network Effects and Cascading Behaviour](#)
13. [Probabilistic Contagion and Models of Influence](#)
14. [Influence Maximization in Networks](#)
15. [Outbreak Detection in Networks](#)
16. [Network Evolution](#)
17. [Reasoning over Knowledge Graphs](#)
18. [Limitations of Graph Neural Networks](#)
19. [Applications of Graph Neural Networks](#)

▼ Links and Readings

- [link to pdf](#)
- [Video Lecture](#)
- P. Erdos, A. Renyi. [On Random Graphs](#) I. Publ. Math. Debrecen, 1959.

- P. Erdos, A. Renyi. [On the evolution of random graphs](#). Magyar Tud. Akad. Mat. Kutato Int. Koezl., 1960.
- B. Bollobas. [Random Graphs](#). Cambridge University Press.
- M.E.J. Newman, S. H. Strogatz and D.J. Watts. [Random graphs with arbitrary degree distributions and their applications](#). Phys. Rev. E 64, 026118, 2001.
- R. Milo, N. Kashtan, S. Itzkovitz, M.E.J. Newman, U. Alon. [On the uniform generation of random graphs with prescribed degree sequences](#). Arxiv, 2004.
- D. Ellis. [The expansion of random regular graphs](#). Lecture notes from Algebraic methods in combinatorics, Cambridge University, 2011.
- S. Arora, S. Rao and U. Vazirani. [Expander Flows, Geometric Embeddings and Graph Partitioning](#). In proc. STOC '04, 2004

Created with  on Weights & Biases.

<https://wandb.ai/syllogismos/machine-learning-with-graphs/reports/1-Introduction-Structure-of-Graphs---VmIldzozNzU1NDU>