Introduction to Graphs

Contents

Real-world relationships → formal abstraction → machine learning

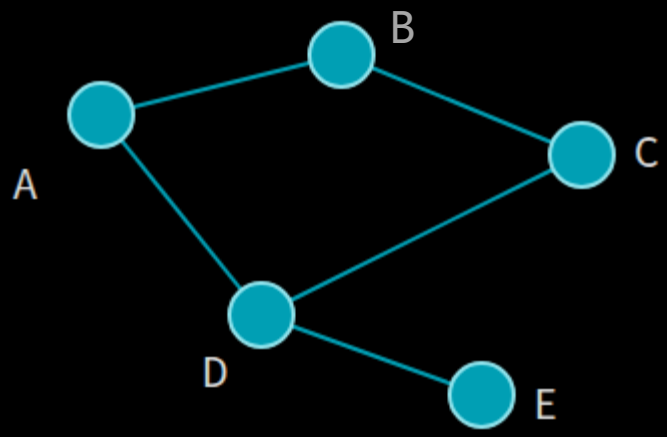
From Independent Data to Connected Data

Independent Samples

Customer A	
Customer B	
Customer C	
Customer D	

- Each row is treated separately
- Samples assumed i.i.d.
- Example: customer records

Connected Data

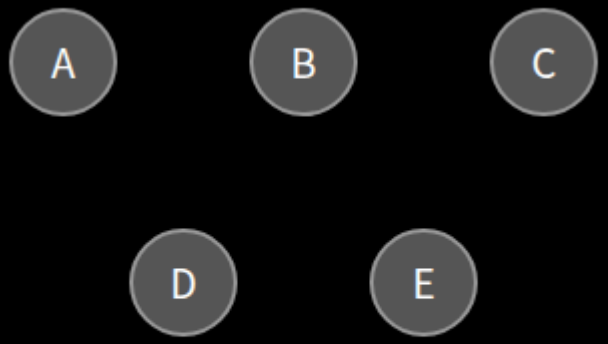


- Entities influence each other
- Relationships carry signal
- Example: shared transactions

In many problems, observations are not independent.

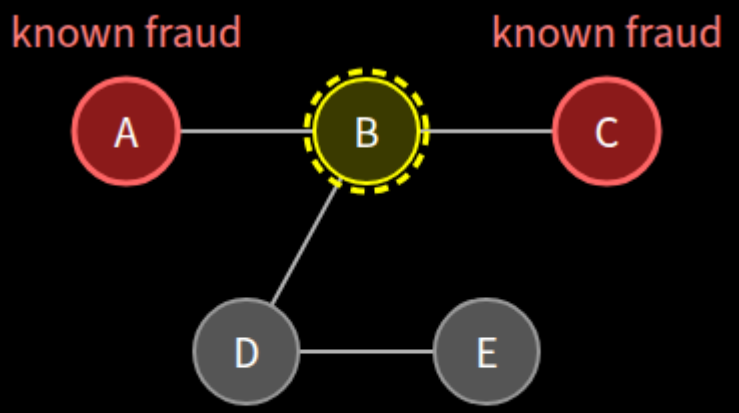
Relationships Carry Information

Attributes only



All accounts look equally normal

Attributes + Relationships



B becomes suspicious because of its neighbors

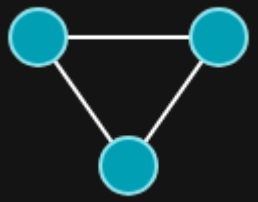
The relationship between entities can be as informative as the entities themselves.

Same idea: citation links reveal research communities.

Real-World Systems as Graphs

Real-world system → Graph abstraction

Social Network

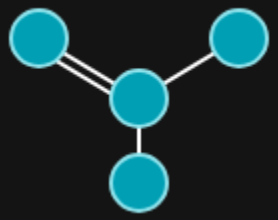


User → Node

Friendship → Edge

Undirected, one node type

Molecule

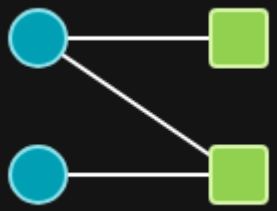


Atom → Node

Bond → Edge

Bond type is an edge attribute

Recommendation

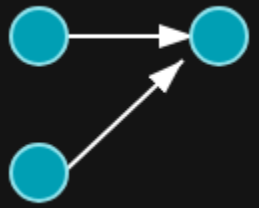


User / Item → Nodes

Interaction → Edge

Two node types (bipartite)

Citation Network



Paper → Node

Citation → Edge

Directed edges

What Is a Graph?

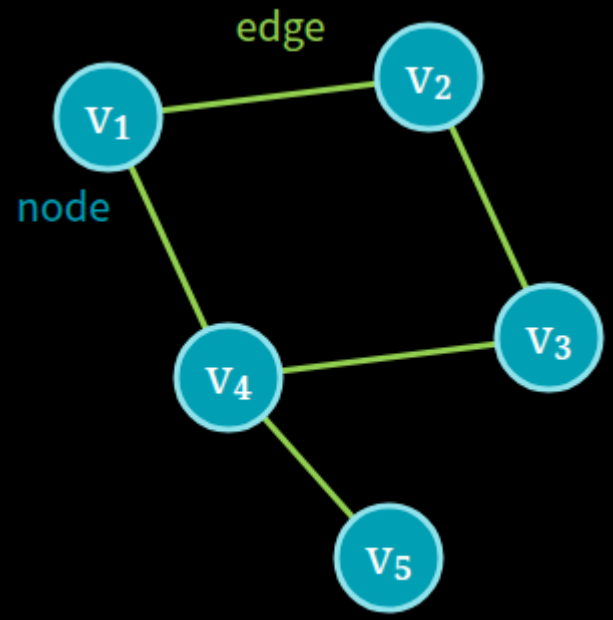
$$G = (V, E)$$

V : set of **nodes** (vertices)

E : set of **edges** connecting pairs of nodes

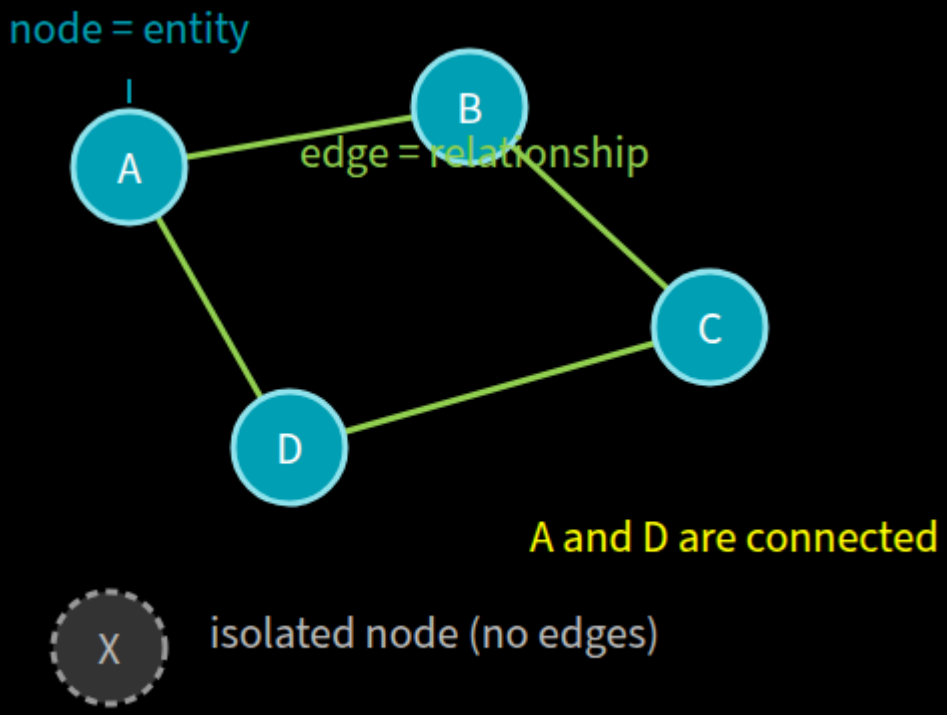
$V = \{v1, \ldots, vn\}$

$E \subseteq V \times V$



A graph is a set of entities and their links.

Nodes and Edges



Nodes

represent entities

Edges

represent relationships

relationships or

interactions

Person —follows→ Person

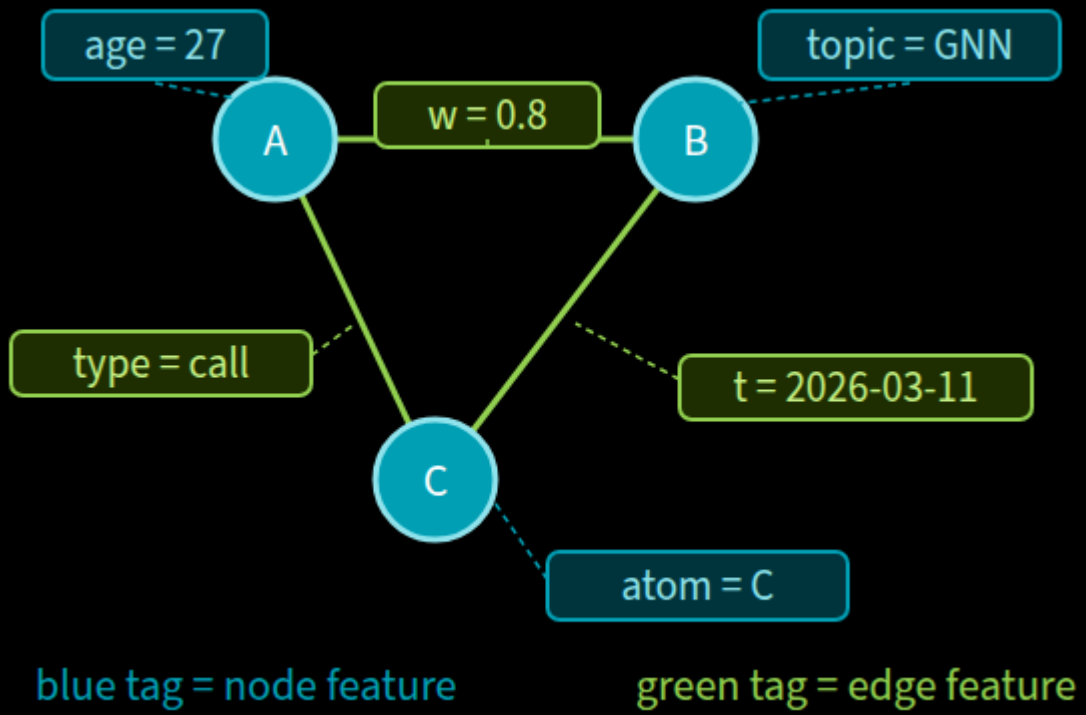
Atom —bonded to→ Atom

Paper —cites→ Paper

Choosing what becomes a node and what becomes an edge is the first modeling step.

Node and Edge Features

Graphs carry attributes in addition to connectivity.



Node features

- user age, interests
- atom type
- paper keywords

Edge features

- interaction type
- bond type
- transaction amount
- timestamp

Attributes attach to graph elements,
not to isolated rows.

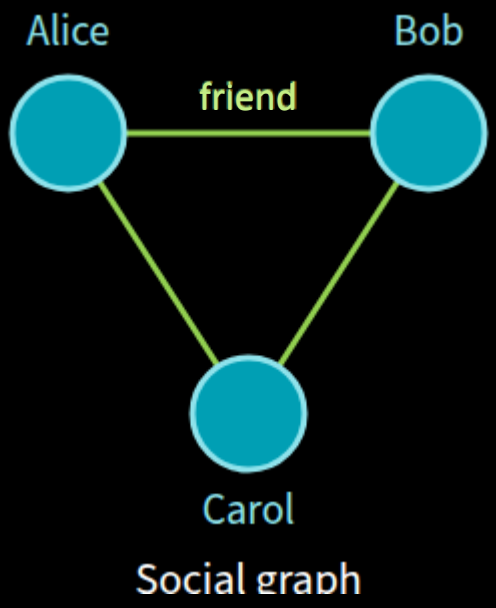
A Graph Is More Than Connectivity

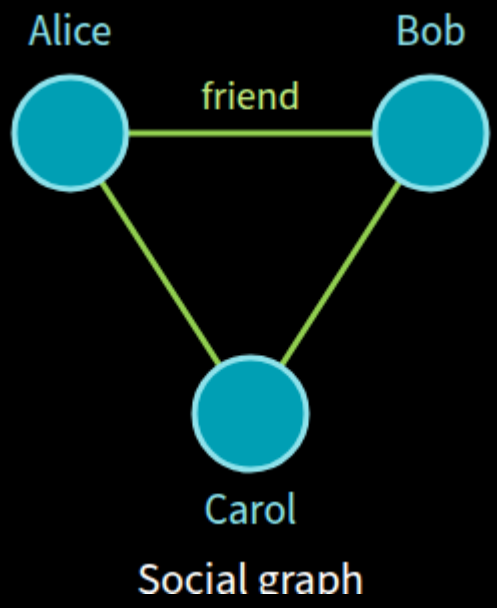
1. Structure
who connects to whom

2. Node information
attributes of entities

3. Edge information
attributes of relations

Same topology, different information

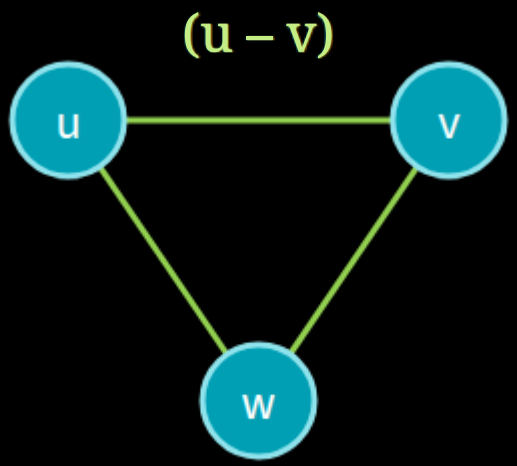




Graph data combines structure and attributes.

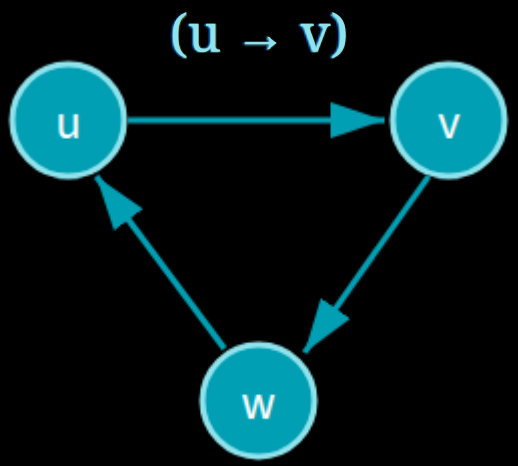
Directed vs. Undirected Graphs

Undirected



- Relationship is symmetric
- Both nodes play the same role
- Example: friendship

Directed

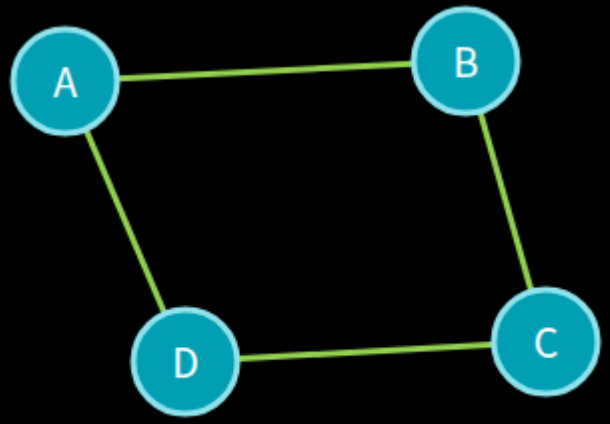


- Relationship has direction
- Source and target differ
- Example: following, citation

Same nodes, same links — direction changes the meaning.

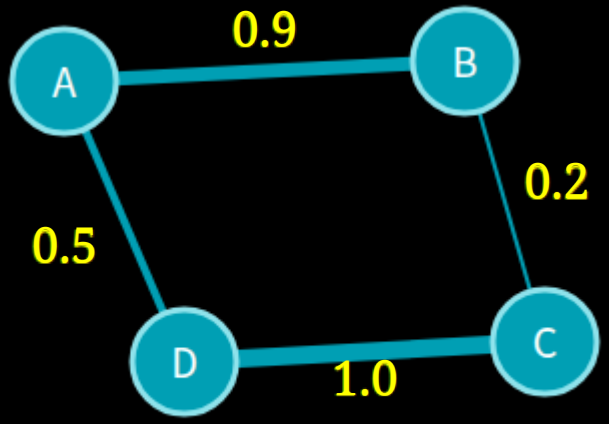
Weighted vs. Unweighted Graphs

Unweighted



Edge = does a relation exist?

Weighted



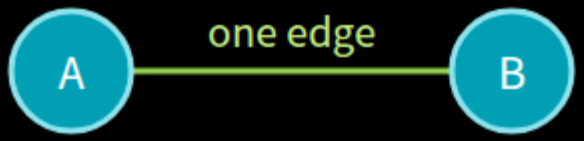
Edge also carries a value

- road distance
- number of interactions
- transaction amount
- similarity score

Weight expresses strength, cost, or intensity of a relation.

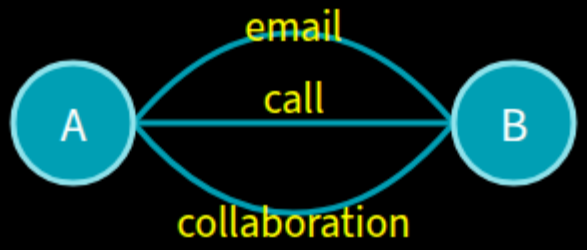
Simple Graphs vs. Multigraphs

Simple Graph



At most one edge per node pair

Multigraph



Multiple edges may connect the same pair

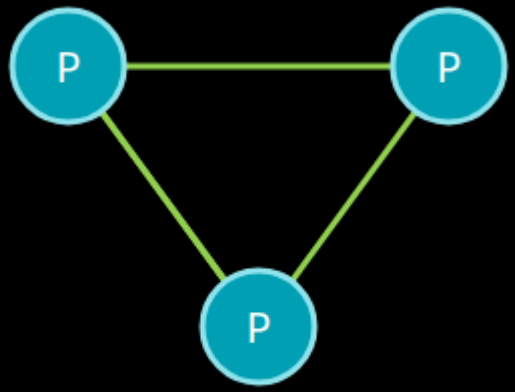
Why it matters

- Two people may interact through several distinct channels
- Each interaction can carry its own type and timestamp

Collapsing multiple relations into one edge discards information.

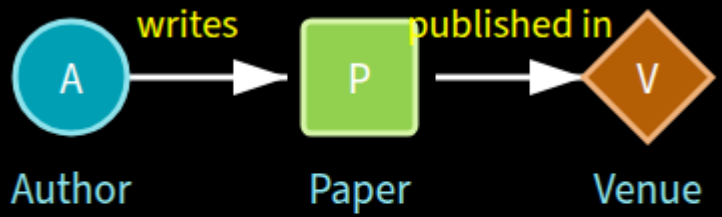
Homogeneous vs. Heterogeneous Graphs

Homogeneous



One node type: Paper
One edge type: cites

Heterogeneous



Multiple node types
Multiple edge types

Different shapes mark different node types.

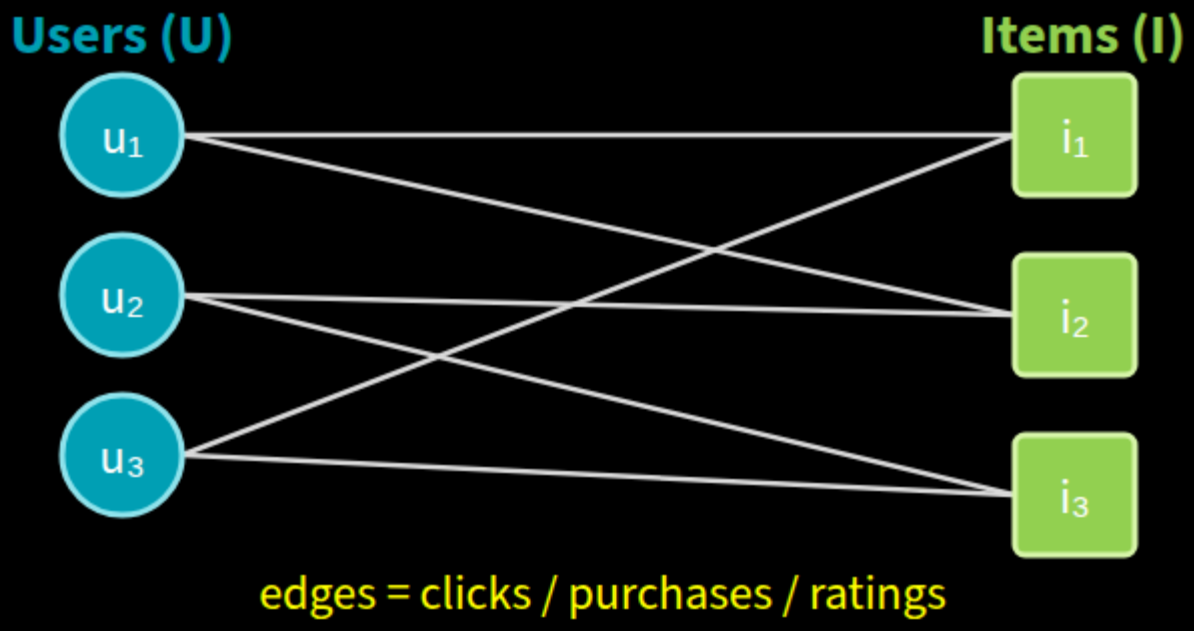
Node type and edge type become part of the data definition.

Most real-world systems are naturally heterogeneous.

Bipartite Graphs

Nodes split into two disjoint sets; edges only go across the sets.

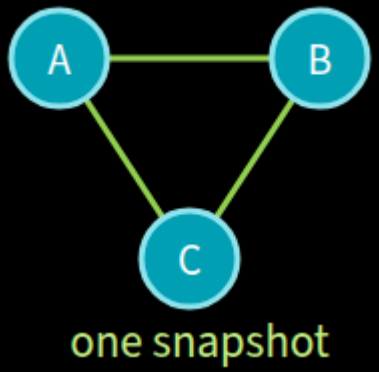
$$G = (U, I, E)$$



No user–user or item–item edges exist by construction.

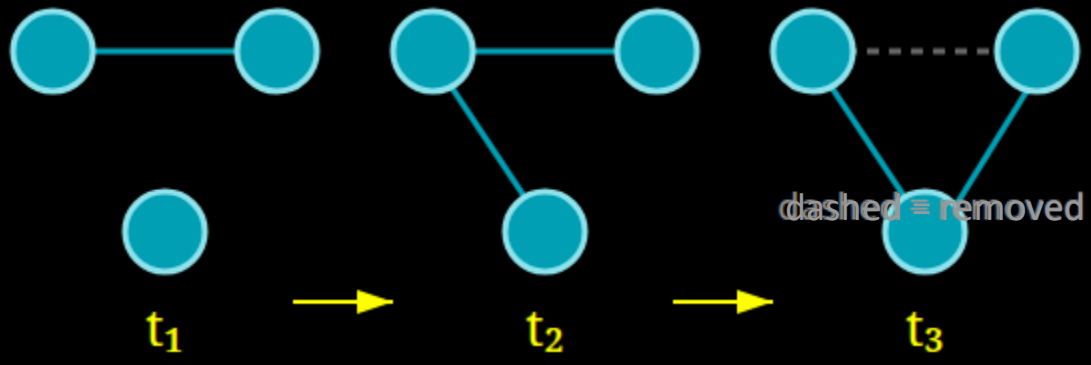
Static vs. Dynamic Graphs

Static graph



- Fixed set of nodes and edges
- No notion of time

Dynamic graph



- Nodes, edges, or attributes change
- Edges appear and disappear

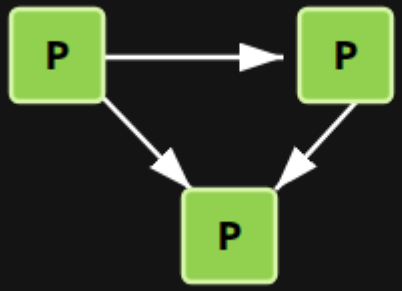
Examples

social interactions · transactions · communication networks

One Dataset, Different Graph Views

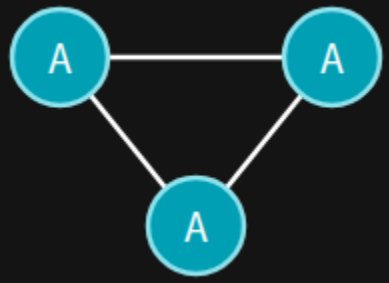
Academic ecosystem data

Paper – Paper (citation)



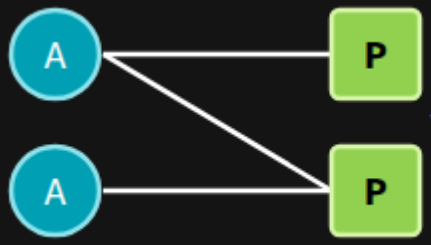
directed, homogeneous

Author – Author (collaboration)



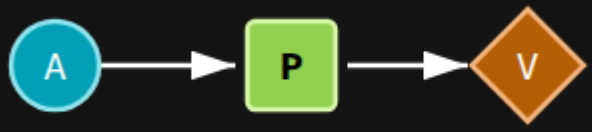
undirected, weighted

Author – Paper (bipartite)



two node sets

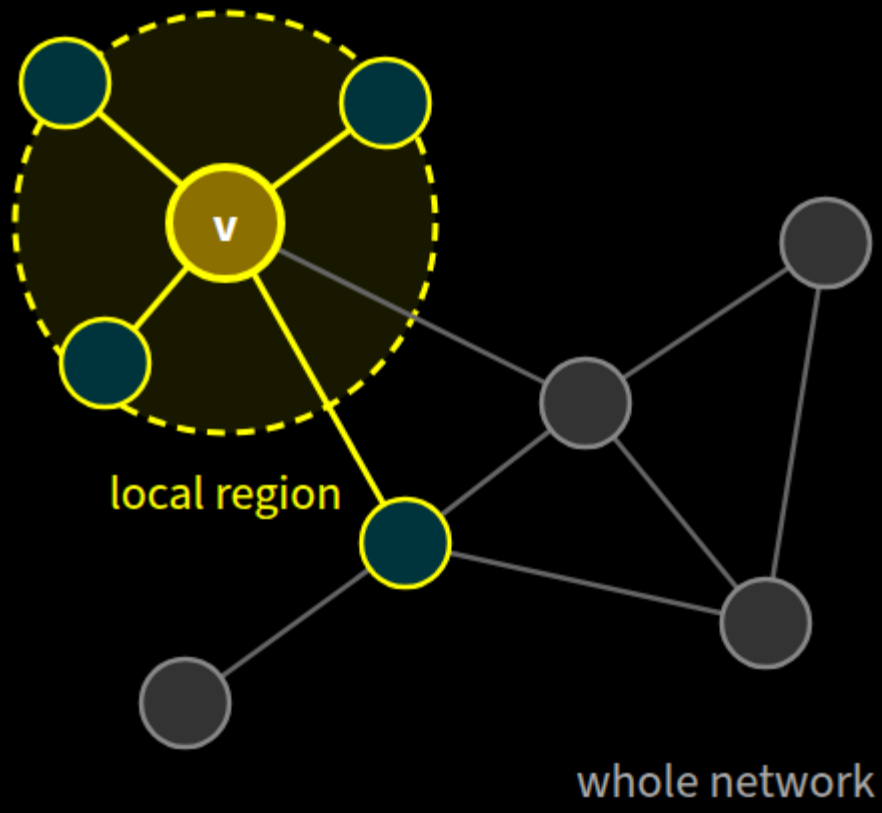
Author – Paper – Venue



heterogeneous, 3 node types

Choosing nodes and edges defines the learning problem.

Local vs. Global Structure



Local structure

- immediate neighborhood
- degree of a node
- nearby connectivity

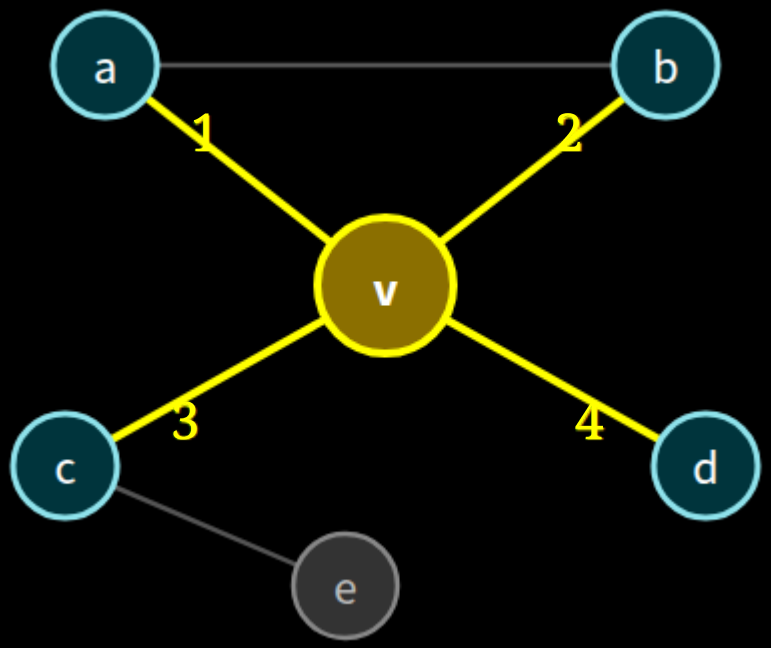
Global structure

- paths across network
- connected components
- overall organization

The next slides build the vocabulary for both views.

Node Degree

$$d(v) = |N(v)|$$

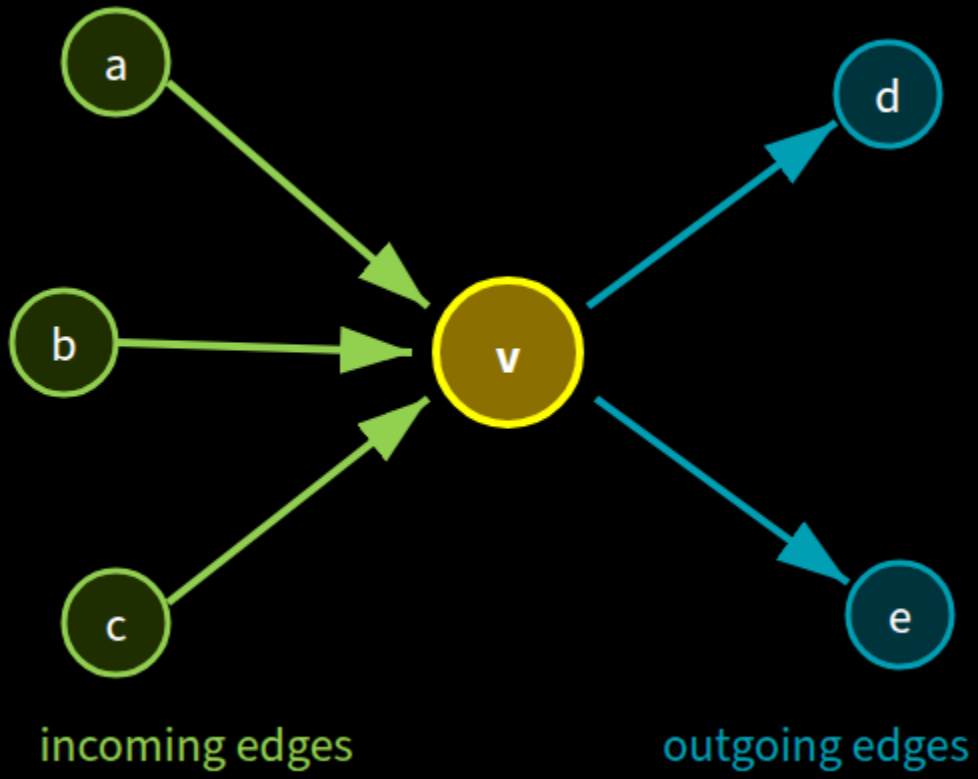


$$d(v) = 4$$

- Degree = number of adjacent adjacent neighbors
- Counted from incident edges
- High degree may mean a hub node

Degree is the simplest description of a node's local structure.

In-Degree and Out-Degree



In-degree
number of incoming edges
= 3

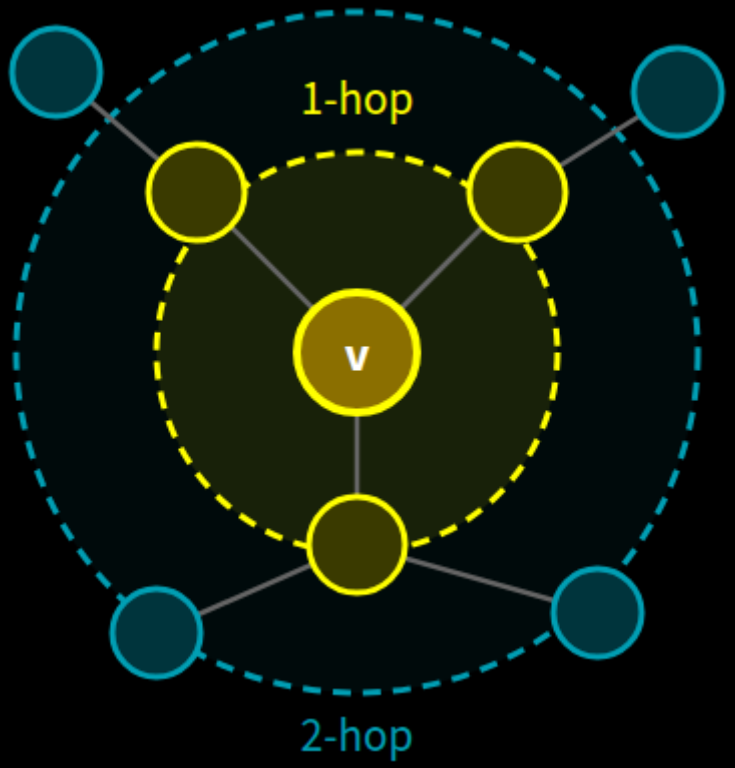
Out-degree
number of outgoing edges
= 2

followers → in
following → out

In directed graphs, a node has two different degrees.

Neighborhood

$$N(v) = \{ u \in V \mid (u, v) \in E \}$$



1-hop

nodes directly adjacent to node v

2-hop

neighbors of neighbors

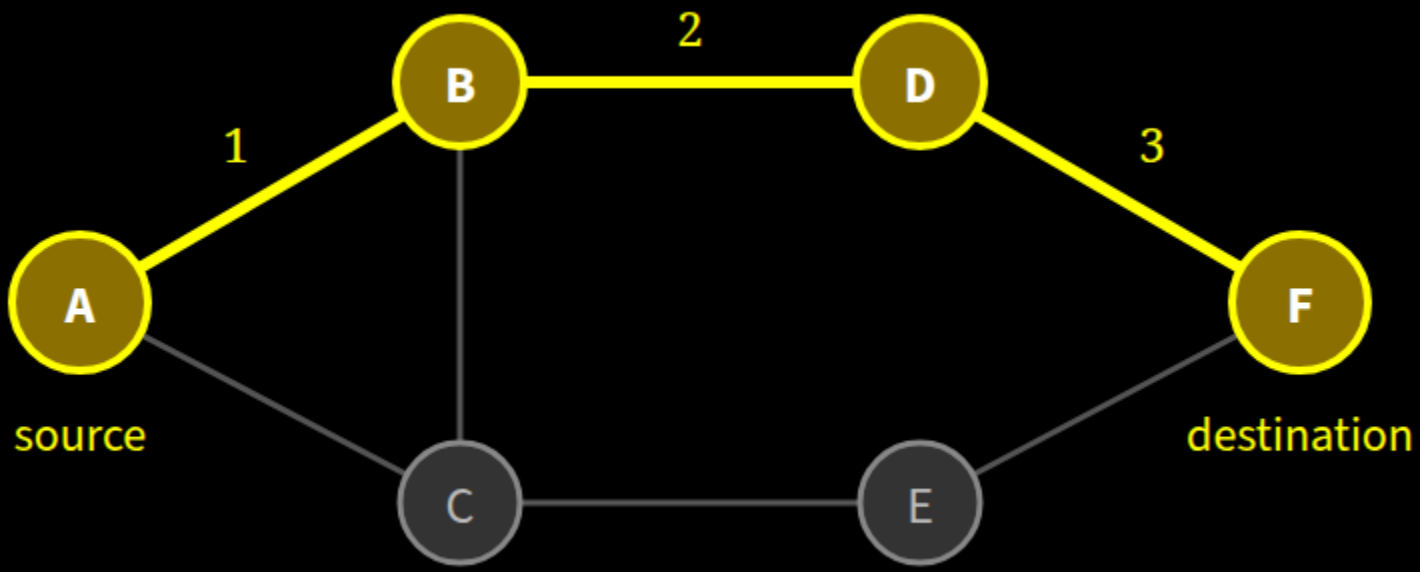
k-hop

within k steps

Local neighborhoods will later become the basic information context used by GNNs.

Paths in a Graph

A path is a sequence of nodes connected by edges.

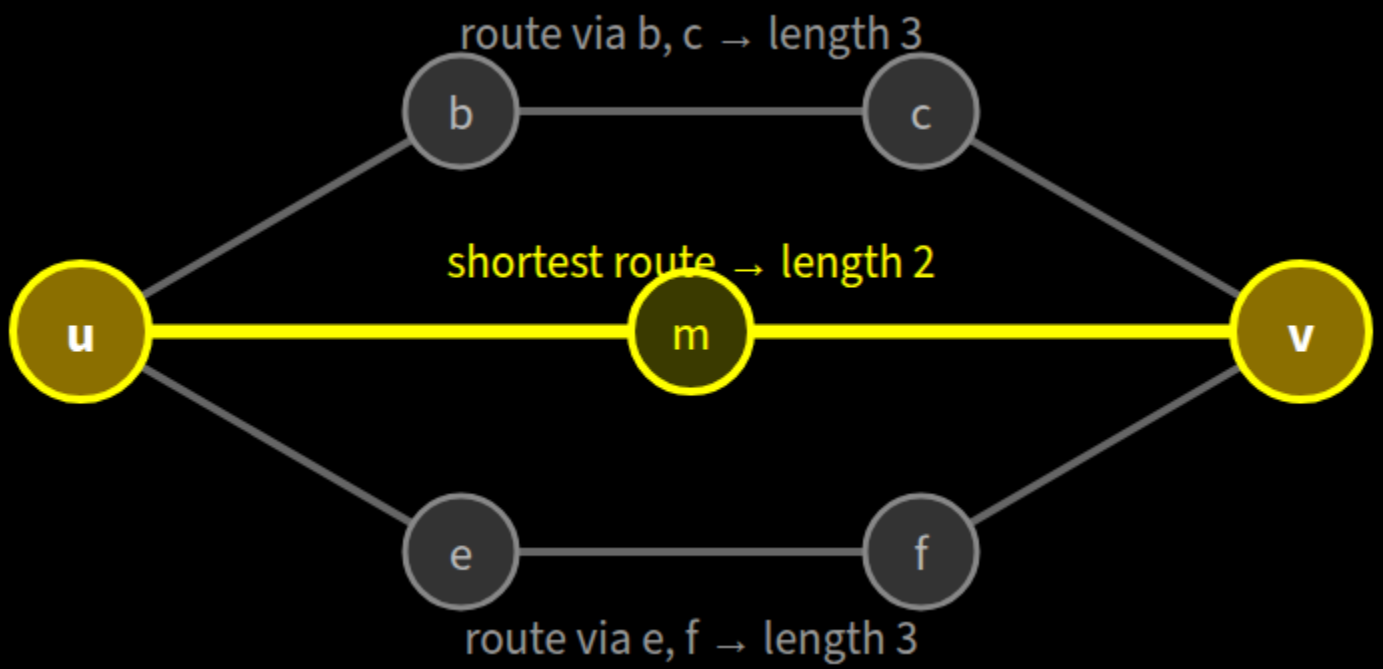


$$A \rightarrow B \rightarrow D \rightarrow F$$

Path length = number of edges traversed → 3

Shortest-Path Distance

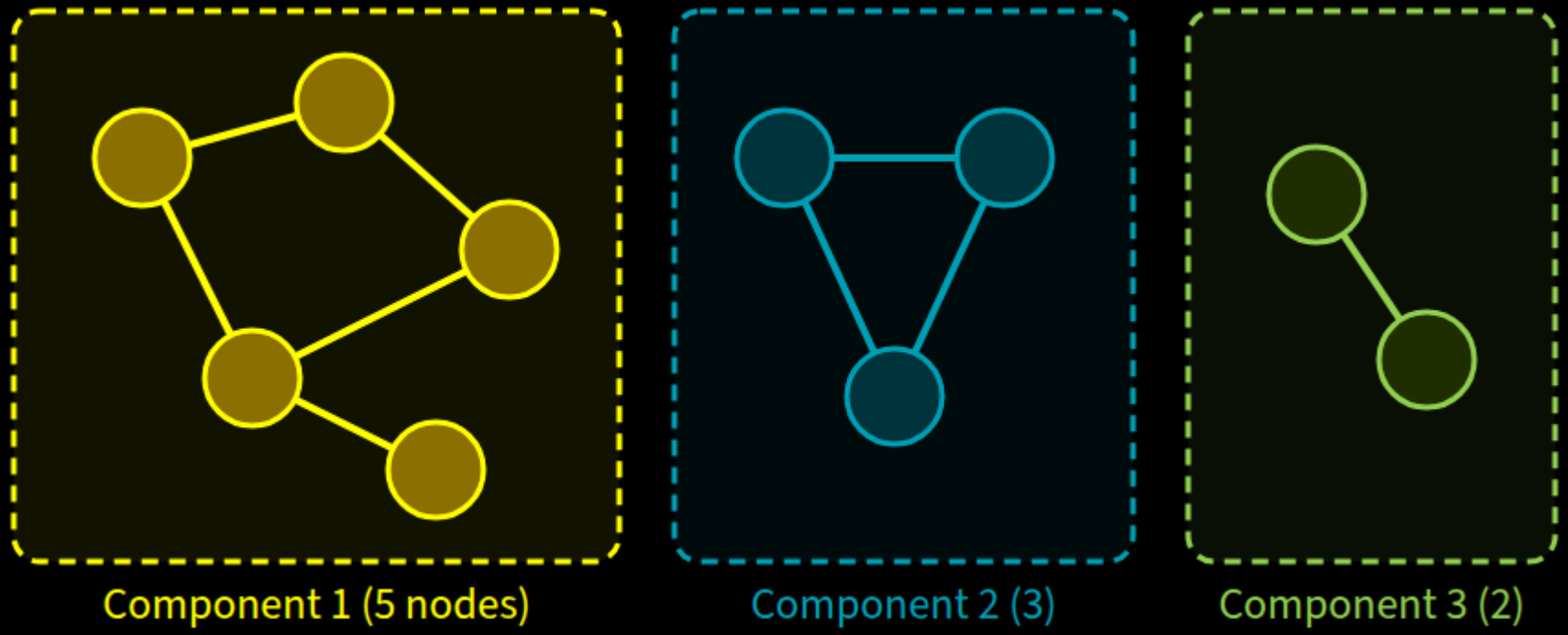
$$d(u, v)$$



- Unweighted graph: minimum number of edges
- Weighted graph: minimum total cost along a route

Connectivity

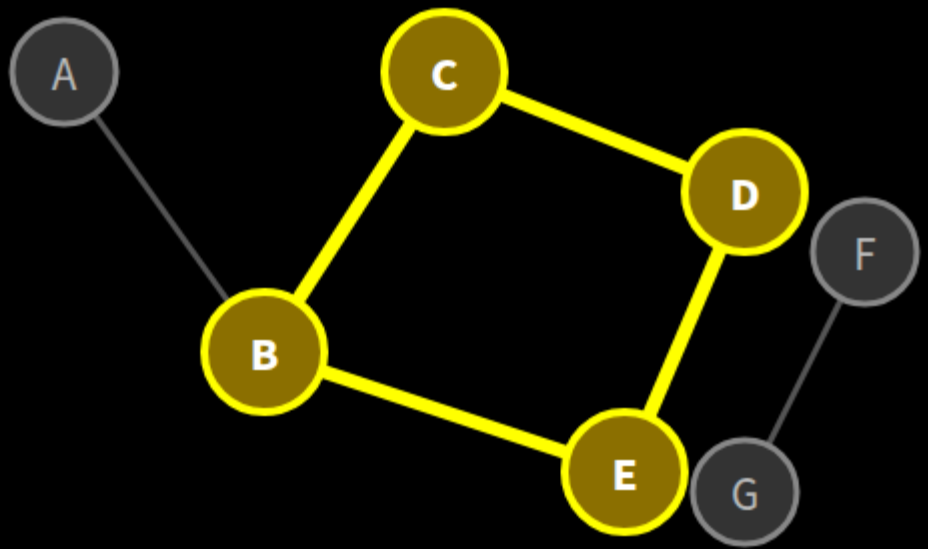
- Two nodes are connected if a path exists between them
- A connected component is a maximal group of mutually reachable nodes



No path crosses between components — they are separate worlds.

Cycles

A cycle is a path that returns to its starting node.



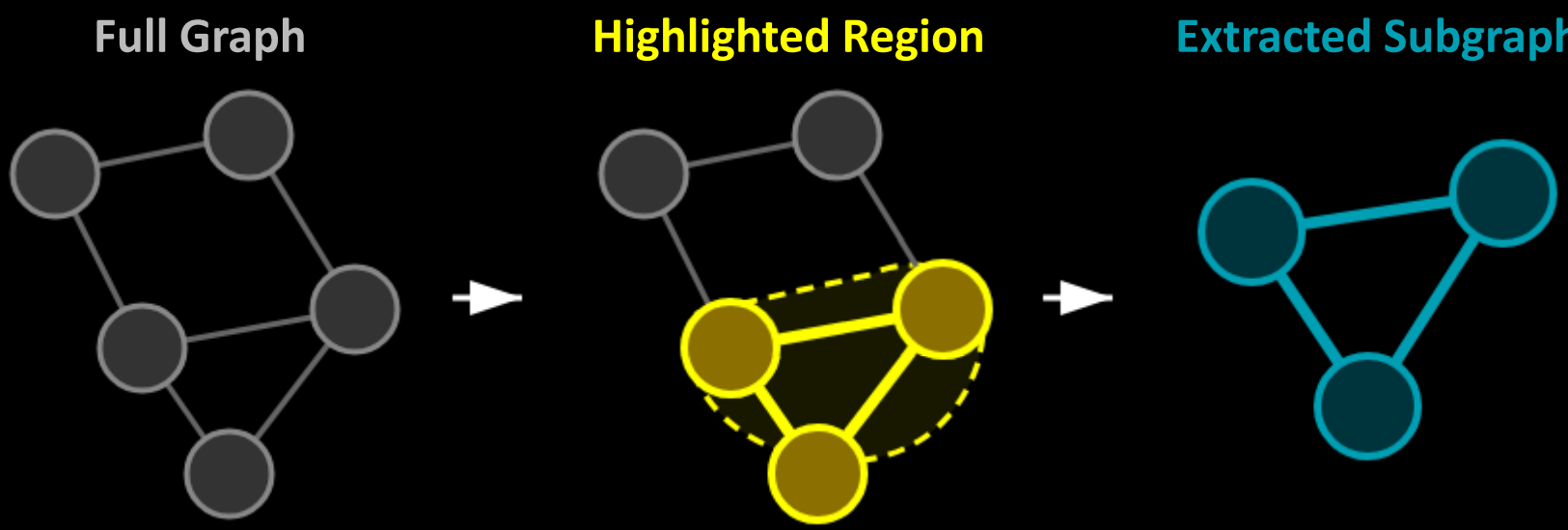
Intuitive examples

- circular dependencies
- feedback loops
- closed interaction patterns
- ring structures in molecules

Cycles are basic vocabulary for describing graph shape.

Subgraphs

A subgraph uses a subset of the nodes and edges of a larger graph.

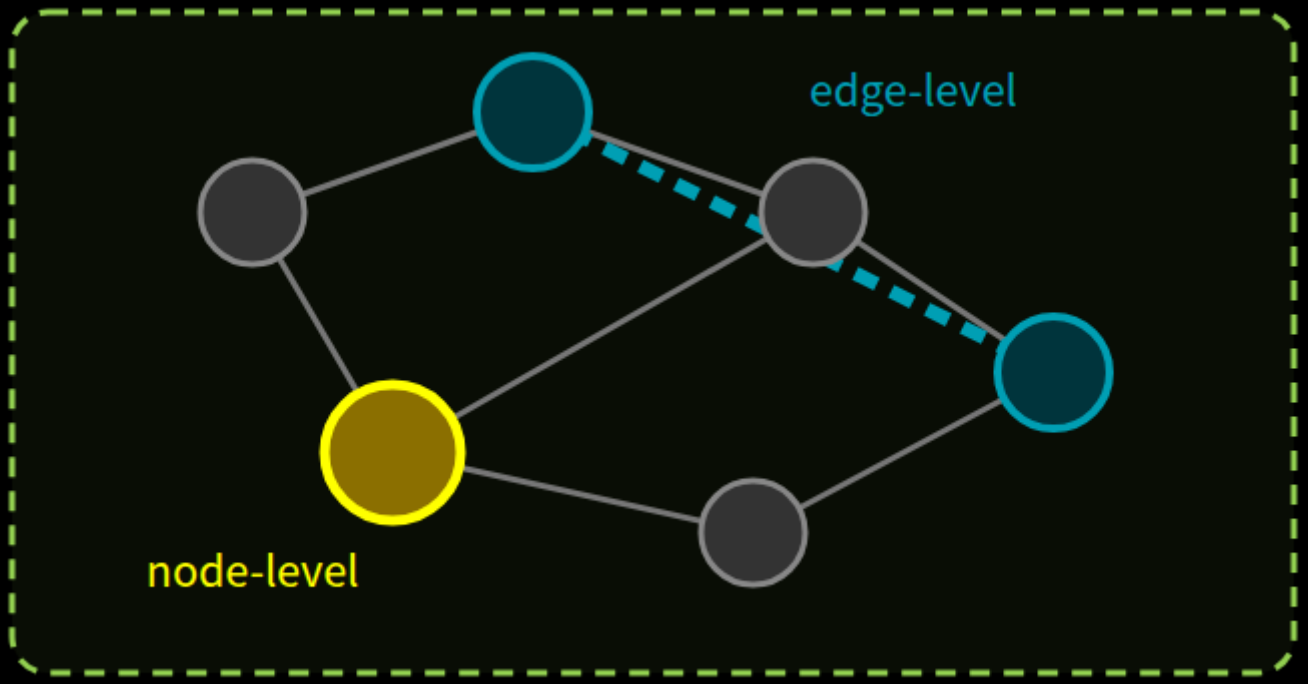


Examples

- local social community
- molecular functional group
- retrieved knowledge-graph neighborhood

Many graph tasks operate on subgraphs, not the entire network.

What Can We Predict on a Graph?



1. Node-level
a property of one node

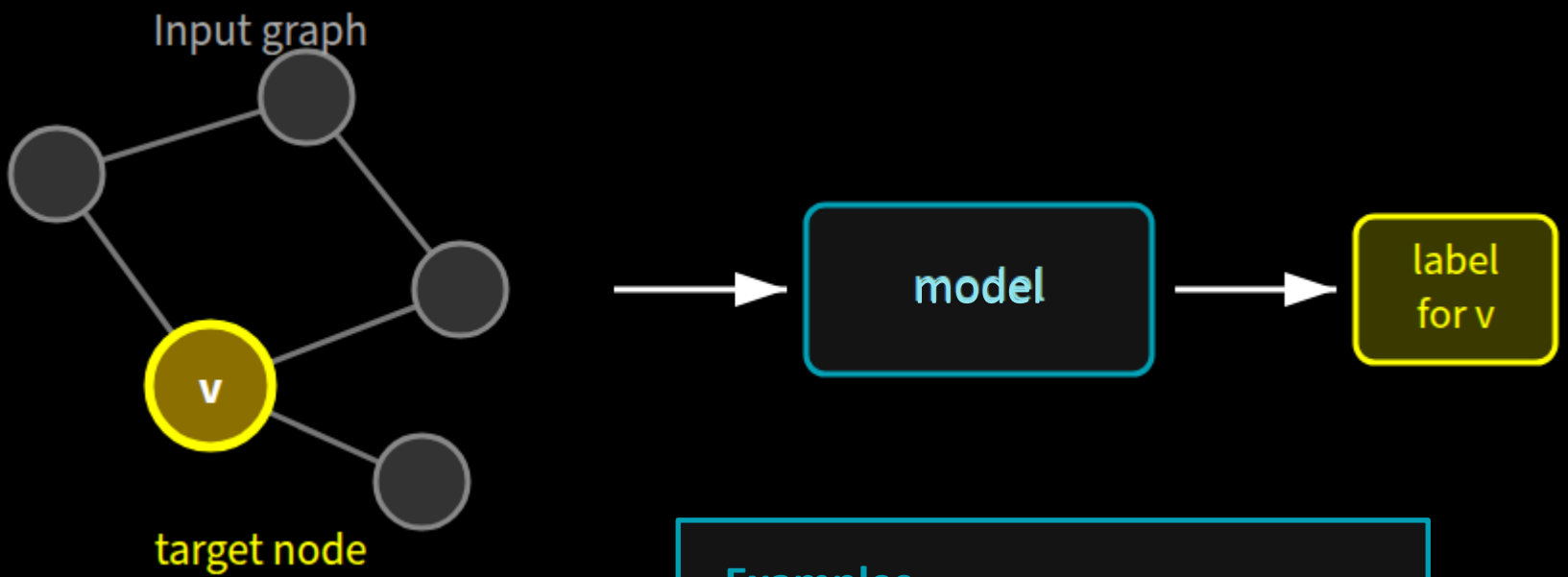
2. Edge-level
a property of a node pair

3. Graph-level
a property of the graph

The prediction level defines what the model outputs.

Node-Level Prediction

Predict a property or label for a node.

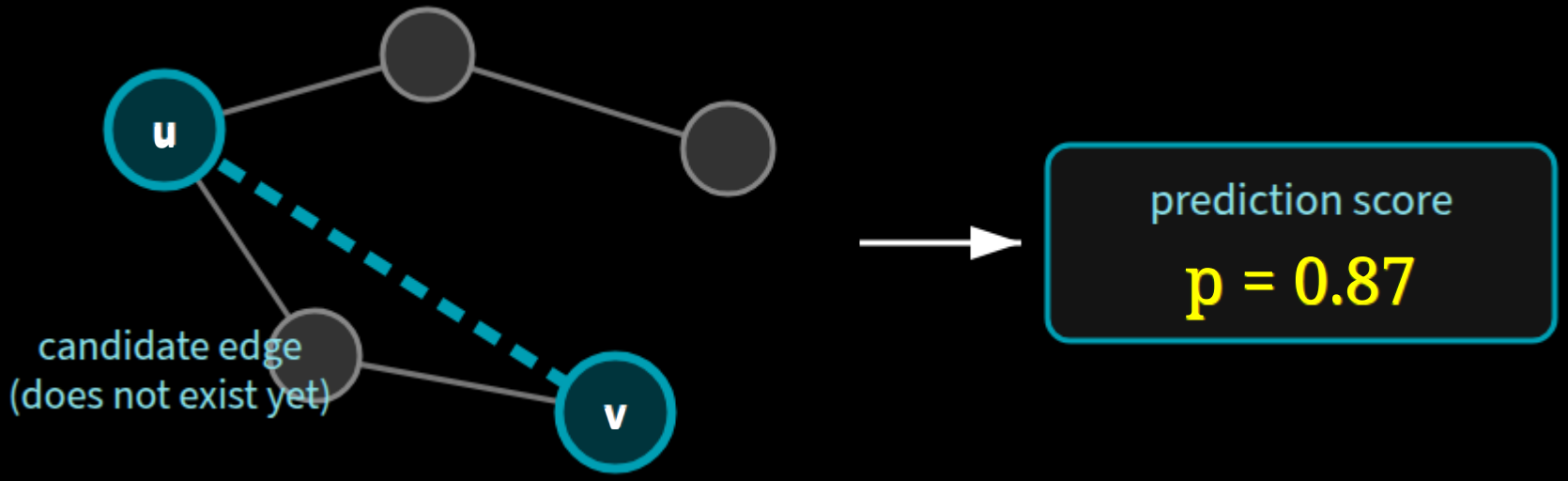


Examples

- classify the topic of a paper
- detect a fraudulent account
- predict a property of an atom

Edge-Level Prediction

Predict a property of an edge, or whether an edge should exist.



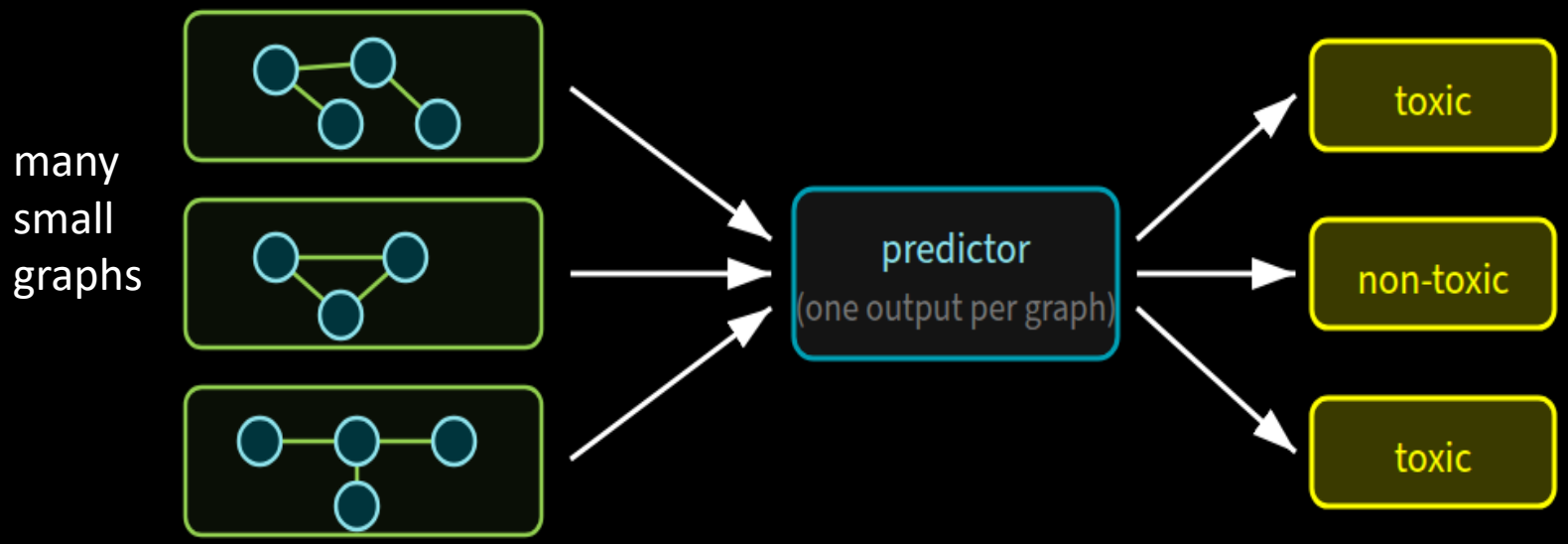
Predicting a missing or future edge is called **link prediction**.

friend recommendation · user–item interaction prediction

drug interaction prediction

Graph-Level Prediction

Predict a property of the entire graph.



Examples

- molecular toxicity or activity
- classify a program, circuit, or network as a whole

Matching Problems to Prediction Levels

Real-World Question	Prediction Level
Is this account fraudulent?	Node
Will these two users connect?	Edge / Link
Is this molecule toxic?	Graph
What topic does this paper belong to?	Node
Which item should we recommend to this user?	Edge / Link

Defining the prediction target is part of formulating a graph-learning problem.

Graph Learning = Features + Structure

Node / Edge Features

age = 27

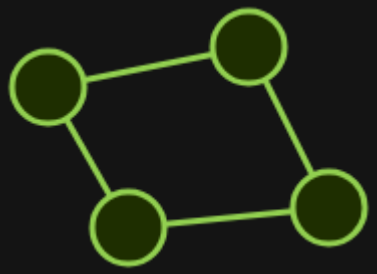
topic = GNN

amount = 120

attributes of entities and relations

+

Graph Structure



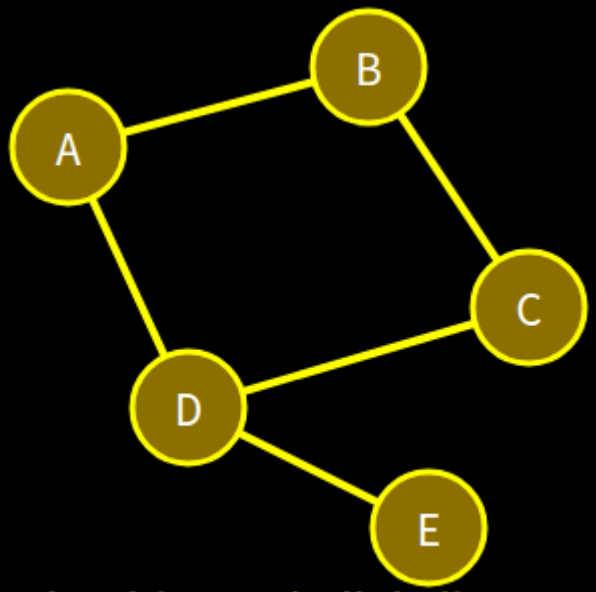
who connects to whom

Graph Learning

Graph learning differs from feature-only learning because relational structure is part of the input.

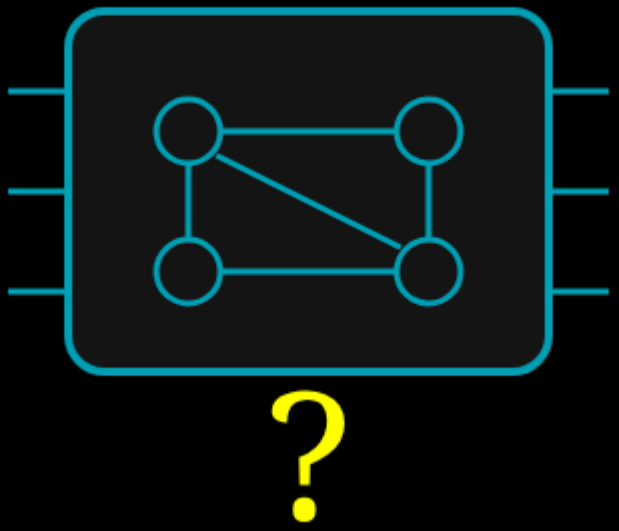
Humans See a Graph — What Does a Computer See?

Human View



an intuitive node-link diagram

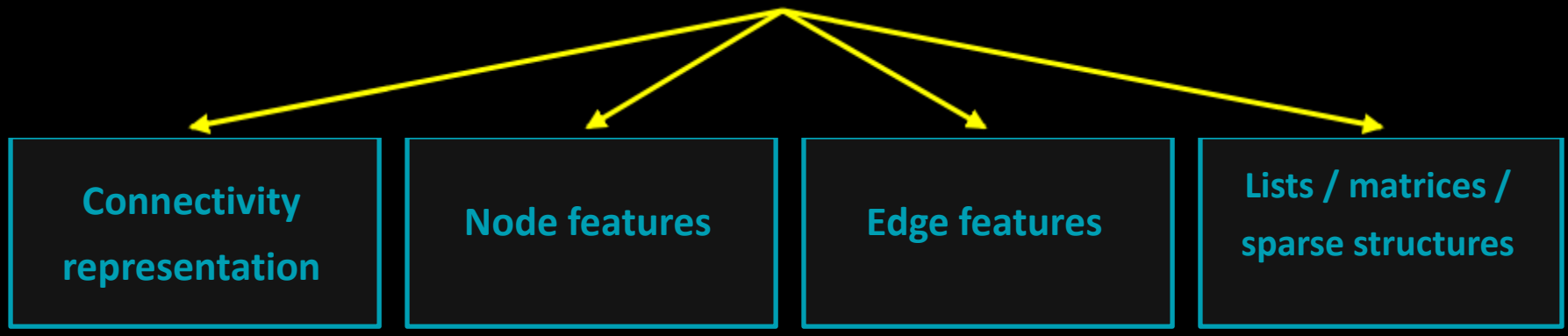
Computer View



How can a computer process nodes, edges, attributes, and structure?

How Can We Represent a Graph?

$$G = (V, E)$$



a visual hint only — details come in the next lecture

adjacency matrix

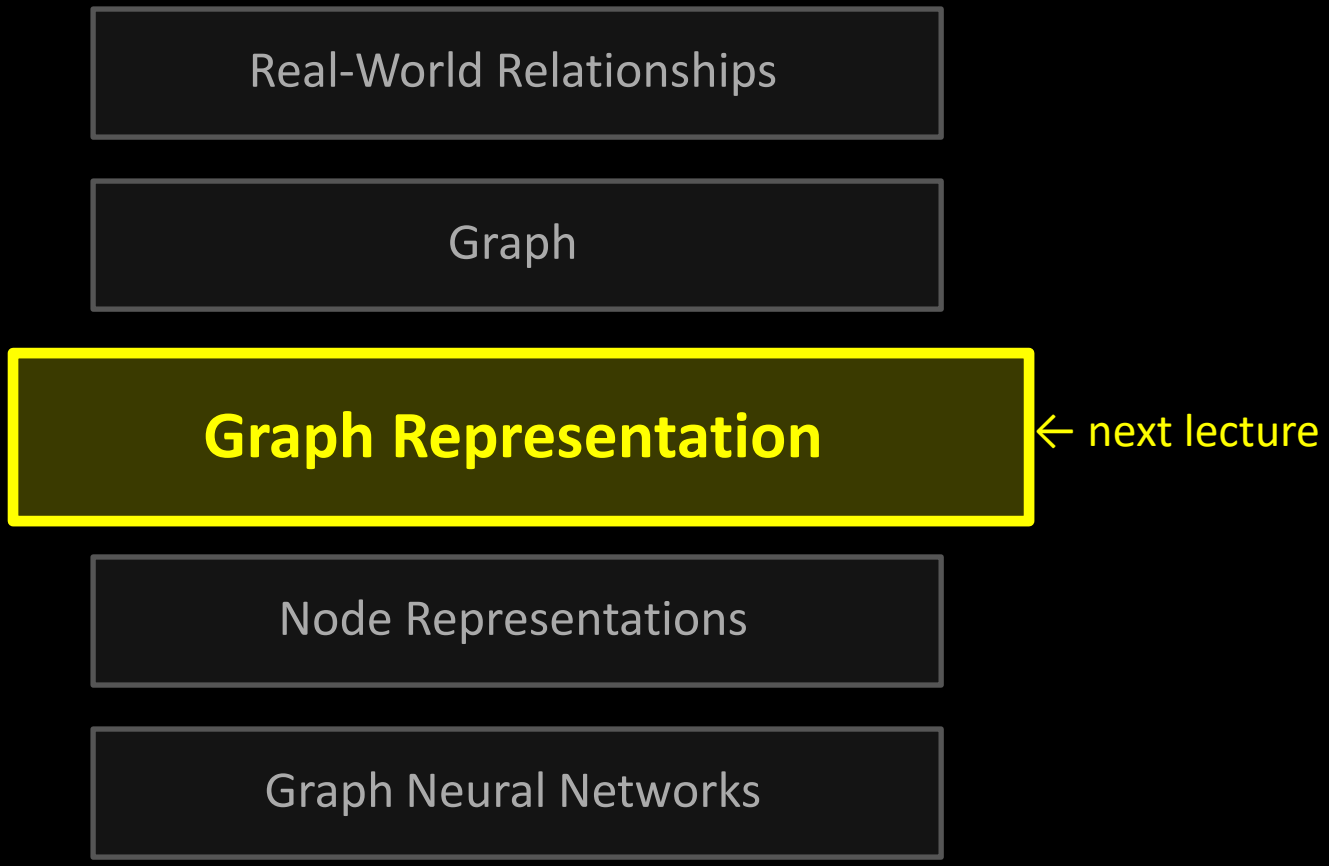
0	1	0	1
1	0	1	0
0	1	0	1
1	0	1	0

edge list

(A, B)
(B, C)
(C, D)
(A, D)

A graph must be converted into a computational representation before it can be processed by machine learning models.

Next: Graph Representation



How do we encode graph structure and attributes for computation?