



Graph Data and Features

Graph ML 01 | Modelling data as a graph, storage formats (COO, CSR, adjacency) and their memory, graph types, homophily, node and edge features (degree, clustering, PageRank, centralities, neighbour aggregates) computed by hand, tools from NetworkX to cuGraph, and leakage-safe feature pipelines

#GraphML #FeatureEngineering #PageRank #NetworkX

From tables to a graph

A graph dataset

In one sentence
●○○○

Rows become nodes, foreign keys and interactions become edges.

Practitioner
●●○○

Decide what a node is, what an edge means, which direction and weight it carries and when it existed.

- accounts, devices, cards as nodes
- logins, payments, shares as edges
- timestamp on every edge

Engineer
●●●○

Store edges as COO for building and CSR for neighbour lookups; keep node ids contiguous 0..n - 1 and a mapping table back to business keys.

Under the hood
●●●●

Real graphs are sparse ($E \ll n^2$) with heavy-tailed degrees: a few hubs touch millions of edges and dominate cost and features unless capped.

Analogy

A spreadsheet lists the guests; the graph is the seating plan showing who talks to whom.

Graph data at four depths

Prerequisites, recalled in one line where used

Type	Example	In PyG	Watch out
Undirected	co-authorship, molecules	both directions in edge_index	count each edge once in stats
Directed	citations, payments	edge_index src → dst	in- vs out-degree differ
Weighted	transaction amount, call minutes	edge_weight or edge_attr	normalise heavy tails (log)

Bipartite	users-items, accounts-devices	HeteroData with 2 node types	no triangles: clustering is 0
Heterogeneous	user, item, shop; buys, follows	HeteroData, to_hetero()	one feature schema per type
Multigraph	repeated payments A → B	duplicate columns	aggregate or keep as events
Temporal	time-stamped interactions	TemporalData (sheet 08)	never use future edges

Graph types you will meet and how to store them

Storage formats and memory

	Dense adjacency	COO edge list	CSR
Memory	n^2 entries	2E ids	$(n + 1) + E$ ids
Neighbours of v	scan a row: $O(n)$	scan all edges	slice: $O(\deg v)$
Add an edge	$O(1)$	append: $O(1)$	rebuild
Used by	tiny graphs, maths	building, PyG edge_index	samplers, cuGraph, DGL

Three ways to hold the same graph (n nodes, E edges)

Format	Bytes	ogbn-products (n = 2.45 M, 61.9 M undirected edges, E = 123.7 M)
Dense bool	n^2	6.0×10^{12} B $\approx$ 6 TB: impossible
COO	$2 \times E \times 8$	$\approx$ 1.98 GB
CSR	$(n + 1 + E) \times 8$	$\approx$ 1.01 GB

where n = number of nodes (nodes); E = number of stored directed edges (an undirected edge counts twice) (edges)

Memory of a graph's structure with int64 ids (8 bytes each)

TIP

Node **features** usually dwarf the structure: $n \times d \times 4$ bytes for float32. ogbn-products with $d = 100$ needs $2.45 \text{ M} \times 100 \times 4 \approx 0.98 \text{ GB}$, as much as the CSR; at $d = 768$ (text embeddings) it is 7.5 GB. Store features as float16 or in a memory-mapped array when they do not fit.

Taddata | tadata.fr | contact@tadata.codes

Node features computed by hand

PROBLEM Running example used on this whole page: 4 accounts, A, B, C form a triangle and D hangs off C; A and D were flagged for fraud. Compute degree, clustering coefficient and closeness of every node, then the average clustering.

GIVEN

- Edges: A-B, A-C, B-C, C-D (undirected)
- Clustering $C_v = 2 \cdot T_v / (k_v \cdot (k_v - 1))$, T_v = edges among v 's neighbours, k_v = degree; 0 when $k_v < 2$
- Closeness $c_v = (n - 1) / \sum_u \text{dist}(v, u)$, $n = 4$ nodes, dist = hops

1 Degrees

$k_A = 2$, $k_B = 2$, $k_C = 3$, $k_D = 1$ (sum 8 = 2×4 edges)

Why: each edge adds 1 to two nodes

2 Clustering of A

neighbours B, C; B-C exists $\rightarrow T = 1$; $2 \cdot 1 / (2 \cdot 1) = 1$

Why: A sits in a closed triangle

3 Clustering of C

neighbours A, B, D; only A-B $\rightarrow T = 1$; $2 \cdot 1 / (3 \cdot 2) = 0.33$

Why: D is not linked to A or B

4 Clustering of B, D

B = 1 (same as A); D: $k = 1 \rightarrow 0$

Why: convention for degree below 2

5 Average clustering

$(1 + 1 + 0.33 + 0) / 4 = 0.58$

Why: NetworkX average_clustering does this

6 Closeness of C

dist 1, 1, 1 $\rightarrow 3 / 3 = 1.0$

Why: C reaches everyone in one hop

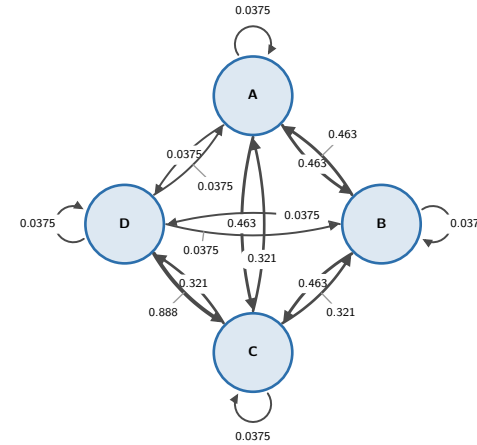
7 Closeness of D

dist to C, A, B = 1, 2, 2 $\rightarrow 3 / 5 = 0.6$

Why: peripheral node

ANSWER Degree (2, 2, 3, 1); clustering (1, 1, 0.33, 0); closeness (0.75, 0.75, 1.0, 0.6); average clustering 0.58

CHECK ☒ C has the lowest clustering but the highest closeness: it is the broker between the triangle and D, and it also has the highest betweenness (all 2 shortest paths A-D and B-D pass through it).



Distribution after n steps

n	A	B	C	D
π_0	0.250	0.250	0.250	0.250
π_1	0.215	0.215	0.463	0.108
π_2	0.260	0.260	0.312	0.169
π_3	0.236	0.236	0.402	0.126
π	0.246	0.246	0.367	0.141

π_n is within 0.001 of π after $n = 11$ steps (from any start it converges).

π : stationary distribution, the unique solution of $\pi = \pi \cdot P$.

$\pi_{n+1} = \pi_n \cdot P$, i.e. $\pi_{n+1}(j) = \sum_i \pi_n(i) \cdot P(i \rightarrow j)$
 $\pi_1(A) = 0.25 \cdot 0.0375 + 0.25 \cdot 0.463 + 0.25 \cdot 0.321 + 0.25 \cdot 0.0375 = 0.215$
 $\pi = \pi \cdot P$ and $\sum_i \pi(i) = 1 \rightarrow \pi = (A \ 0.246, B \ 0.246, C \ 0.367, D \ 0.141)$

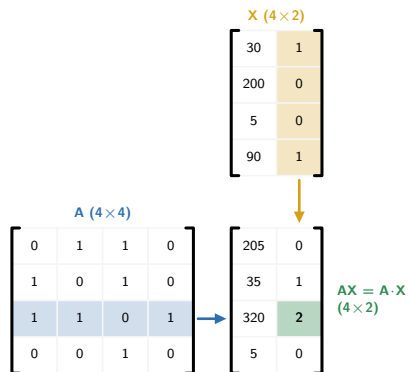
Where: $P(i \rightarrow j)$: probability of moving from state i to state j in one step (each row of P sums to 1); $\pi_n(j)$: probability of being in state j after n steps (π_0 = start); π : stationary distribution (one more step leaves it unchanged)

PageRank of the running example as a Markov chain: from node i , follow a random edge with $d = 0.85$, teleport to any of the 4 nodes with 0.15 (so each entry adds $0.15 / 4 = 0.0375$). The stationary distribution is the PageRank vector

Quantity	Formula	Reading
PageRank	$PR(v) = \frac{1-d}{n} + d \sum_{u \rightarrow v} \frac{PR(u)}{k_{out}(u)}$	importance flows along edges
Transition	$P_{uv} = \frac{1-d}{n} + d \frac{A_{uv}}{k_{out}(u)}$	row u of the chain above
Undirected, $d = 1$	$PR(v) = \frac{k_v}{2E}$	here (2, 2, 3, 1) / 8

where $PR(v)$ = PageRank of node v ; all values sum to 1 (probability); d = damping factor, probability of following an edge (NetworkX alpha = 0.85 by default) (0-1); n = number of nodes (nodes); $k_{out}(u)$ = out-degree of u (degree if undirected) (edges); $A_{uv} = 1$ if the edge $u \rightarrow v$ exists, else 0 (0/1); E = number of undirected edges (edges)

PageRank, the formula behind the chain



Shapes: A is 4×4 , X is $4 \times 2 \rightarrow AX = A \cdot X$ is 4×2 (the inner sizes 4 = 4 must match)

$AX[3,2] = \sum A[3,k] \cdot X[k,2]$ for $k = 1 \dots 4 = 1 \cdot 1 + 1 \cdot 0 + 0 \cdot 0 + 1 \cdot 1 = 2$

Cost: $m \cdot n \cdot p = 4 \cdot 4 \cdot 2 = 32$ multiplications for the whole of AX

Where: A : $m \times n$ matrix ($m = 4$ rows, $n = 4$ columns); X : $n \times p$ matrix ($p = 2$); $AX[i,j]$: entry in row i , column j of AX = dot product of row i of A with column j of X ; k : index over the n shared columns of A / rows of X ; Σ : sum

Neighbour aggregates are one sparse product: $A \cdot X$ sums each node's neighbours' features. X columns = (account age in days, flagged 0/1). Row C, column 2: C has 2 flagged neighbours (A and D)

Feature catalogue and pipeline

Feature	Captures	Cost	Good for
Degree, weighted degree	activity, popularity	$O(E)$	every task; log-transform
Neighbour aggregates ($A \cdot X$, $A^2 \cdot X$)	what the neighbourhood looks like	$O(E \cdot d)$ per hop	fraud, churn, credit
Share of flagged neighbours	guilt by association	$O(E)$	fraud, abuse; needs label-time care
Triangles, clustering	tight groups, closure	$O(E^{1.5})$ worst case	spam, social graphs
PageRank, personalised PageRank	global or seed-relative importance	$O(E)$ per iteration	ranking, retrieval, fraud seeds
Betweenness	brokers, bridges	$O(n \cdot E)$ exact; sample k sources	money mules, supply chains
k-core number	embeddedness in a dense core	$O(E)$	bots, collusion
Connected component size	ring size	$O(n + E)$ with union-find	fraud rings (sheet 06)
Community id (Louvain, Leiden)	group membership	about $O(E)$ per pass	segments, graph RAG (sheet 09)

Graph features worth trying first (n nodes, E edges); all in NetworkX, igraph, cuGraph, Neo4j GDS

Compute features with NetworkX, join them to the table (small graphs; cuGraph mirrors the API on GPU)

```
import networkx as nx, pandas as pd
G = nx.Graph([("A","B"), ("A","C"), ("B","C"), ("C","D")])
feats = pd.DataFrame({
    "degree": dict(G.degree()),
    "clustering": nx.clustering(G),
    "pagerank": nx.pagerank(G, alpha=0.85),
    "betweenness": nx.betweenness_centrality(G),
    "core": nx.core_number(G),
})
flagged = {"A": 1, "B": 0, "C": 0, "D": 1}
feats["flagged_nbrs"] = [
    sum(flagged[u] for u in G[v]) for v in feats.index]
print(feats.round(3))
```

⚠ Leakage through the graph

Compute every graph feature on the graph as of the prediction time and with training labels only. Common leaks: PageRank or degree computed on the full graph including future edges; 'share of flagged neighbours' using labels of test nodes or labels that arrived after the decision (chargebacks arrive weeks later); the target edge itself counted in common-neighbour features (sheet 04).

Leakage-safe graph feature pipeline

✓ baseline / mandatory □ recommended – verify in your context

✓ Snapshot the edge table at each training cut-off date features for day t use edges with timestamp $< t$

✓ Mask labels of validation and test nodes before label-based features

□ SHOULD Cap or log-transform hub degrees a few hubs dominate sums and sampling cost

□ SHOULD Measure edge homophily per label low homophily: prefer features that separate self from neighbours

✓ Compare against a no-graph baseline same model without graph columns

ⓘ How do you measure whether a graph helps?

Train the same gradient-boosted model with and without graph features on a time-based split. Then check edge homophily: if linked nodes share labels much more than chance, neighbour aggregates and GNNs help; if not, the graph may still help through structure (degree, cores) but averaging neighbours will dilute the signal.

🚩 Misconceptions and recap

YOU MAY HEAR

"An adjacency matrix is the natural way to store a graph"

BUSTED

✓ ACTUALLY

Only for tiny graphs

A 2.45 M-node graph would need about 6 TB dense; CSR needs about 1 GB.

YOU MAY HEAR

"High degree means important"

BUSTED

✓ ACTUALLY

Degree counts links, not their quality

PageRank and betweenness weight who you are connected to and where you sit.

YOU MAY HEAR

"Graph features are leak-free because they use no labels"

BUSTED

✓ ACTUALLY

Future edges leak too

Degree computed on next month's edges encodes next month's activity.

YOU MAY HEAR

"Neighbours always look alike"

BUSTED

✓ ACTUALLY

Fraud and marketplaces are heterophilous

Measure homophily before averaging neighbours.

What people get wrong about graph data

Tick a box only when you could explain the point to someone else



Pick nodes and edges from the prediction question

and stamp every edge with a time



COO to build, CSR to query

features usually cost more memory than structure



Degree, PageRank, aggregates, cores, components

the five families to try first



Point-in-time features only

no future edges, no test labels

Q1 Shape of PyG's edge_index for E directed edges?

A [2, E] int64: row 0 sources, row 1 destinations; an undirected edge is stored twice.

Q2 CSR memory for n nodes and E edges with int64?

A $(n + 1 + E) \times 8$ bytes: one row-pointer array plus one column array.

Q3 Clustering coefficient of a node with degree 3 and one edge among its neighbours?

A $2 \times 1 / (3 \times 2) = 0.33$.

Q4 What is PageRank in Markov-chain terms?

A The stationary distribution of a random walk that follows an edge with probability d and teleports uniformly otherwise.

Q5 PageRank of an undirected graph without teleport?

A Proportional to degree: $k_v / 2E$.

Q6 What does row v of A·X contain?

A The sum of the feature vectors of v's neighbours (A is the adjacency matrix, X the node features).

Q7 Edge homophily ratio?

A Share of edges whose two endpoints have the same label; low values mean heterophily.

Q8 Most common graph-feature leak?

A Computing features on the full graph, including edges or labels that appear after the prediction time.

Q9 Why is betweenness costly and how is it approximated?

A Exact Brandes is $O(n \cdot E)$; sample k source nodes (`betweenness_centrality(G, k=...)`) for an estimate.