A gentle introduction to Graph Neural Networks

Robin Brochier, Clement Wang

2026

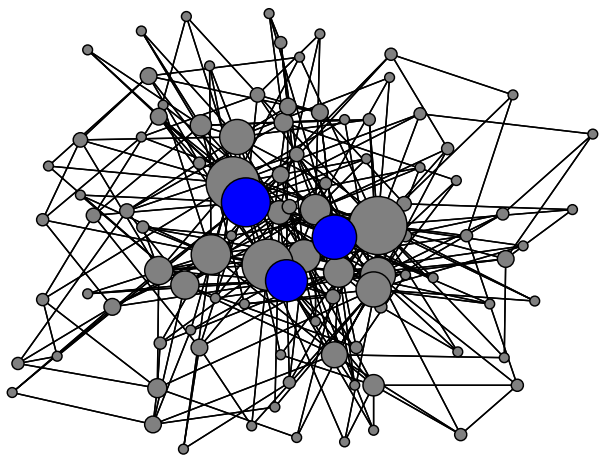
Graph data

A graph consists of **nodes** connected by **links**, with possible **attributes** on both.

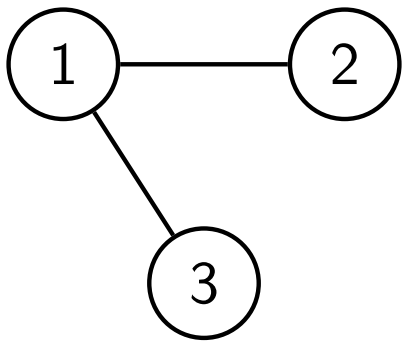
Examples include social networks, the Web, knowledge bases, documents, commerce, and biology.

Mirakl example

Products can form a graph: two products are linked if they are often purchased together.



Adjacency matrices



products bought together

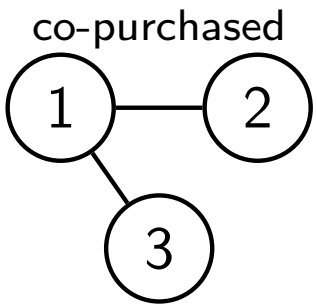
$$A = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

$A_{ij} = 1$ means co-purchased.

One row and one column per product.

Different types of graphs

Undirected



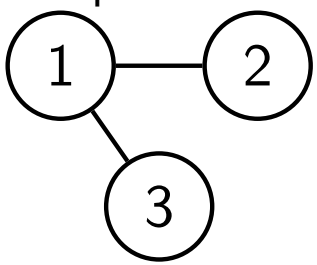
$$A = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

symmetric

Different types of graphs

Undirected

co-purchased

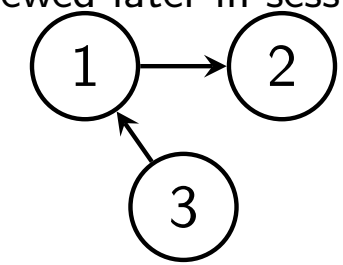


$$A = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

symmetric

Directed

viewed later in session



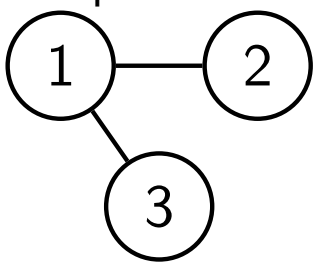
$$A = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

rows point to columns

Different types of graphs

Undirected

co-purchased

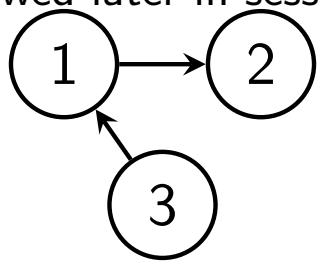


$$A = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

symmetric

Directed

viewed later in session

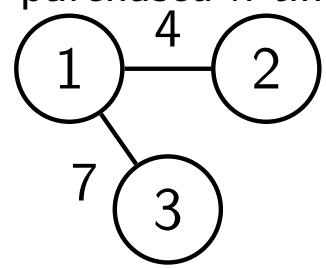


$$A = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

rows point to columns

Weighted

co-purchased k times



$$A = \begin{pmatrix} 0 & 4 & 7 \\ 4 & 0 & 0 \\ 7 & 0 & 0 \end{pmatrix}$$

counts or strength

Large graphs are sparse

Density: fraction of possible edges that are present, for undirected graphs:

$$\rho = \frac{m}{\binom{n}{2}} = \frac{2m}{n(n-1)}$$

with n nodes and m edges.

Large graphs are sparse

Density: fraction of possible edges that are present, for undirected graphs:

$$\rho = \frac{m}{\binom{n}{2}} = \frac{2m}{n(n-1)}$$

with n nodes and m edges.

Dataset	#nodes	#edges	Density
Flights	2,939	30,500	$\approx 10^{-3}$
Amazon products	335k	925k	$\approx 10^{-5}$
Actors	382k	33M	$\approx 10^{-4}$
Wikipedia	12M	378M	$\approx 10^{-6}$
Twitter	42M	1.5G	$\approx 10^{-6}$
Friendster	68M	2.5G	$\approx 10^{-7}$

First observation: large graphs tend to be very sparse.

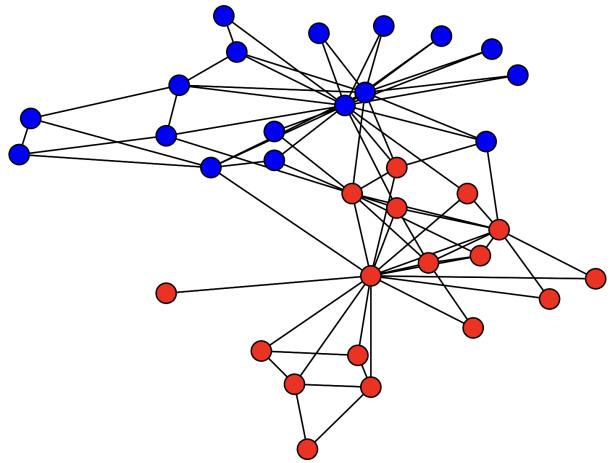
Machine learning on graphs

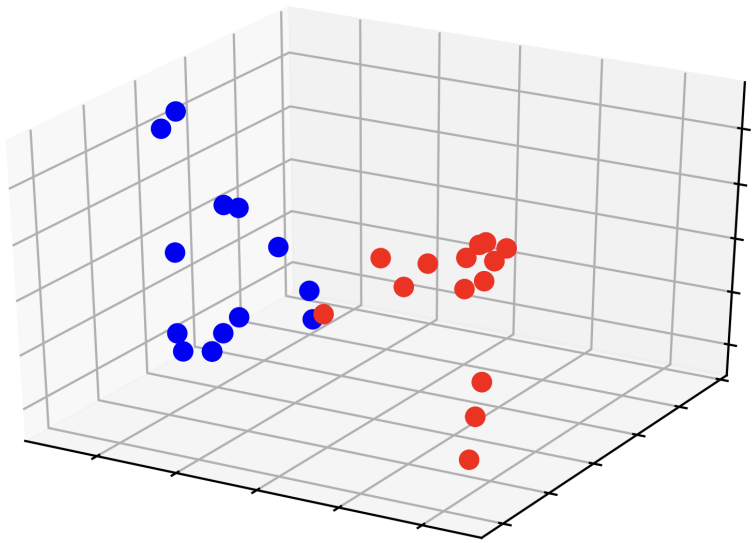
Supervised learning

- Classification
- Regression

Unsupervised learning

- Ranking
- Clustering
- Embedding
- Link prediction
- Anomaly detection





This course feels easy so far. Do not worry, it gets worse :)

Outline

Introduction and Motivations

Large Graphs and their Properties

Graph Machine Learning

Graph Neural Networks

Some Applications and Perspectives

Conclusion

Where are we?

Introduction and Motivations

Large Graphs and their Properties
- Power-law degree distributions
- Small world property

Graph Machine Learning

Graph Neural Networks

Some Applications and Perspectives

Conclusion

Large graphs are heterogeneous

Many large real-world graphs have a very unequal degree distribution: most nodes have few neighbors, while a small number of nodes are hubs.

Large graphs are heterogeneous

Many large real-world graphs have a very unequal degree distribution: most nodes have few neighbors, while a small number of nodes are hubs.

A common empirical model is a power law, often with $2 \lesssim \gamma \lesssim 3$:

$$\mathbb{P}(D = k) \propto k^{-\gamma}, \qquad k \geq k_{\min},$$

where D is the degree of a uniformly sampled node.

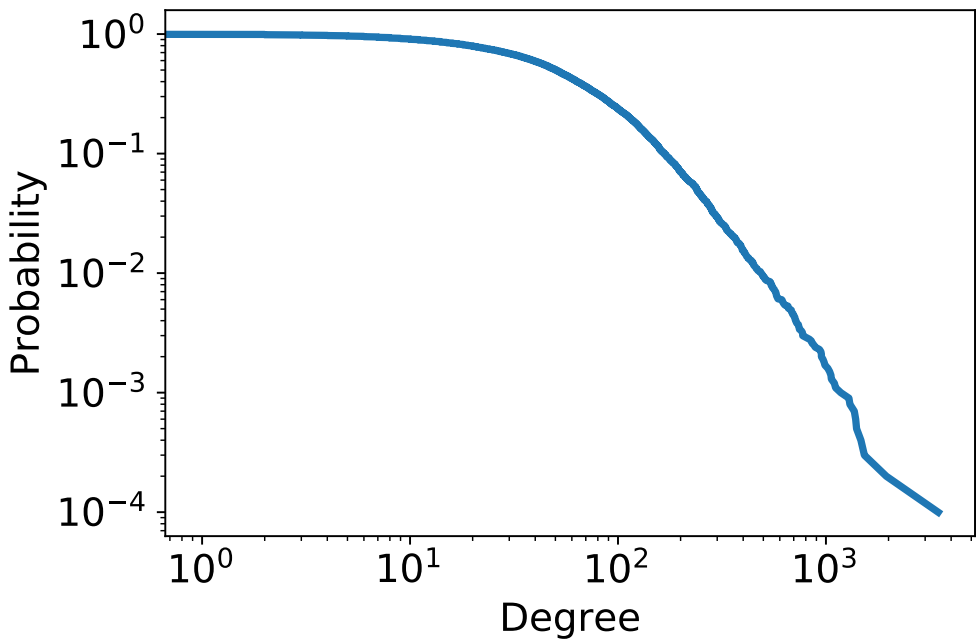
Large graphs are heterogeneous

Many large real-world graphs have a very unequal degree distribution: most nodes have few neighbors, while a small number of nodes are hubs.

A common empirical model is a power law, often with $2 \lesssim \gamma \lesssim 3$:

$$\mathbb{P}(D = k) \propto k^{-\gamma}, \qquad k \geq k_{\min},$$

where D is the degree of a uniformly sampled node.



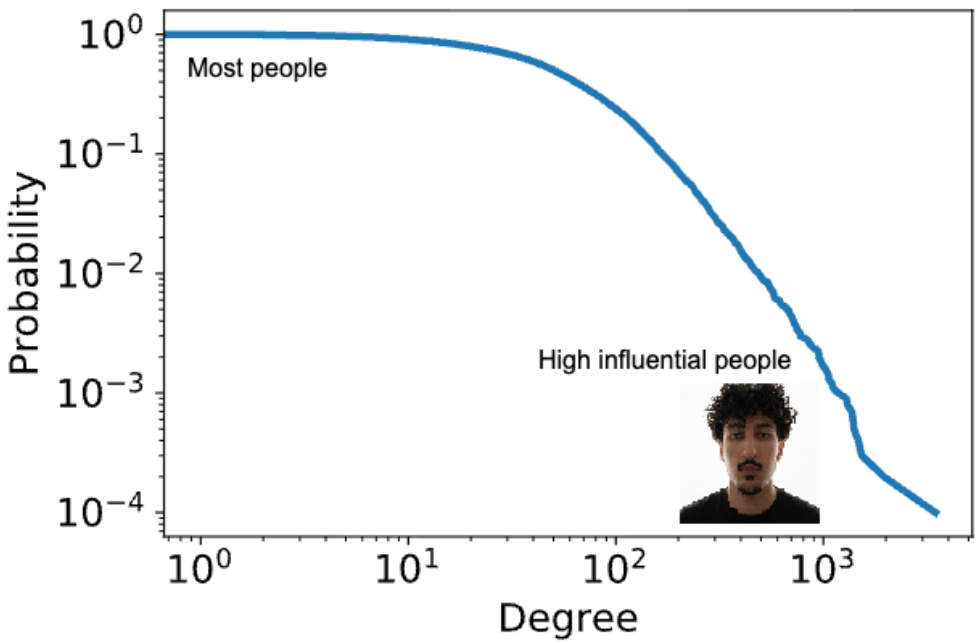
Large graphs are heterogeneous

Many large real-world graphs have a very unequal degree distribution: most nodes have few neighbors, while a small number of nodes are hubs.

A common empirical model is a power law, often with $2 \lesssim \gamma \lesssim 3$:

$$\mathbb{P}(D = k) \propto k^{-\gamma}, \qquad k \geq k_{\min},$$

where D is the degree of a uniformly sampled node.



Mirakl example: Who ate with who at lunch.

Can a simple random graph explain this?

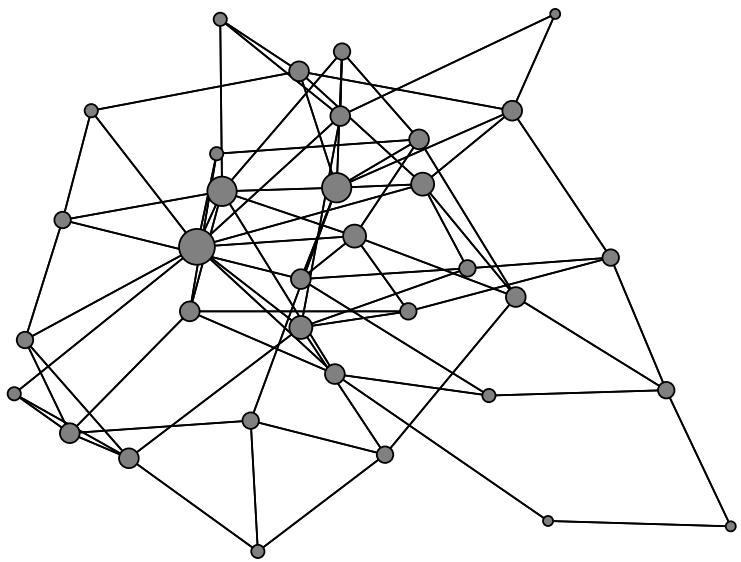
A first model is the Erdős-Rényi random graph $G(n, p)$.

We take n nodes and connect each pair independently with probability p:

$$A_{ij} \sim \text{Bernoulli}(p), \qquad i < j.$$

Equivalently:

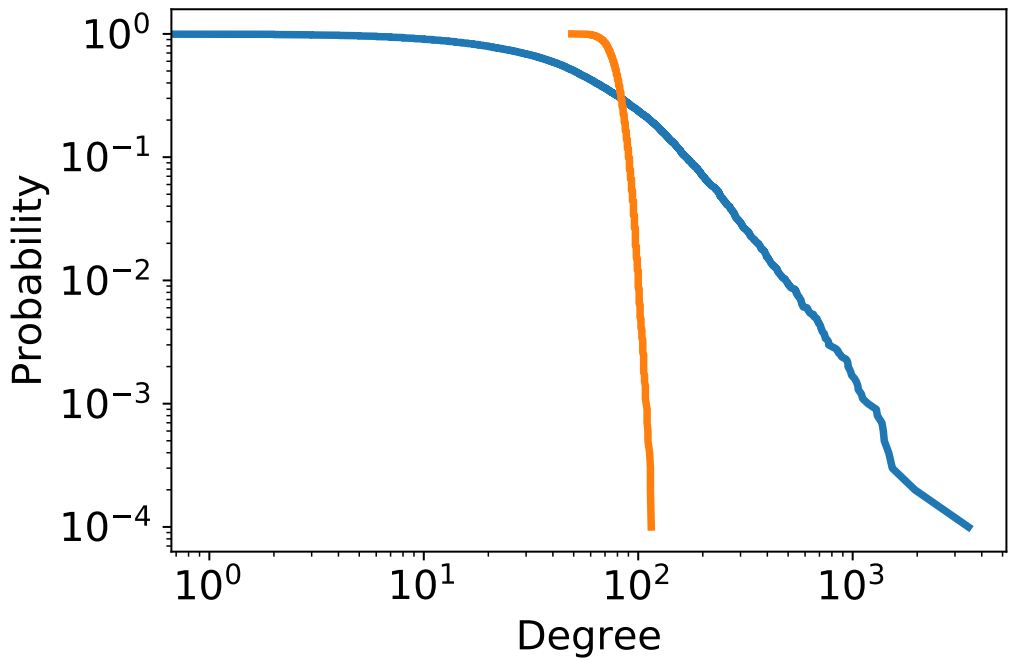
$$\mathbb{P}\big((i,j) \in E\big) = p.$$



Random graphs do not produce hubs

Wikipedia Vitals vs. Erdős-Rényi graph

same number of nodes and approximately same number of edges



Conclusion

Erdős-Rényi graphs have degrees concentrated around the mean.
Real graphs are much more heterogeneous.

Preferential attachment

To generate hubs, we need a mechanism where popular nodes become even more popular.

Barabási-Albert model:

1. Start from a clique of d nodes.
2. Add one new node at each step.
3. Connect it to d existing nodes.
4. Existing node i is chosen with probability proportional to its degree:

$$\mathbb{P}(\text{attach to } i) = \frac{\deg(i)}{\sum_j \deg(j)}.$$

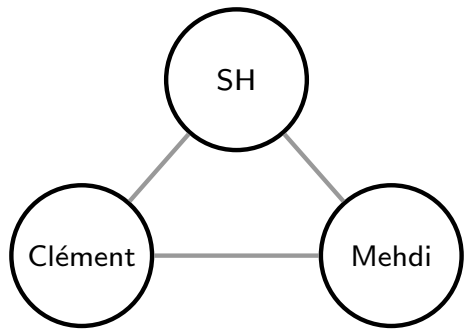
"Rich get richer"

Intuition

Nodes that already have many neighbors are more likely to attract new edges, so early lucky nodes can become hubs.

Preferential attachment

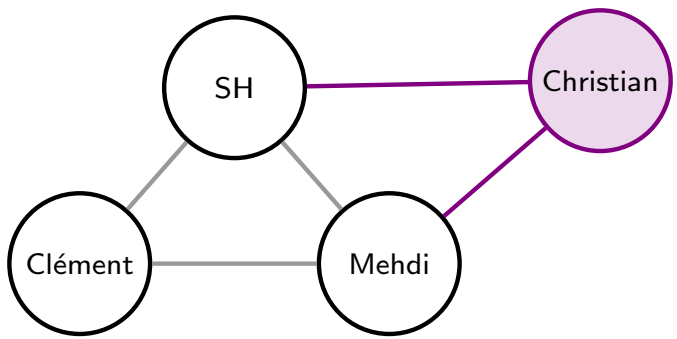
$$\mathbb{P}(\text{new node attaches to } i) = \frac{\deg(i)}{\sum_j \deg(j)}$$



Start from a small clique.

Preferential attachment

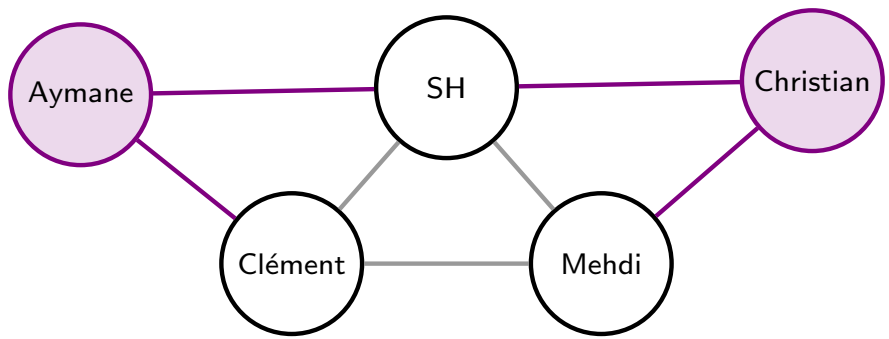
$$\mathbb{P}(\text{new node attaches to } i) = \frac{\deg(i)}{\sum_j \deg(j)}$$



A new node attaches to existing nodes.

Preferential attachment

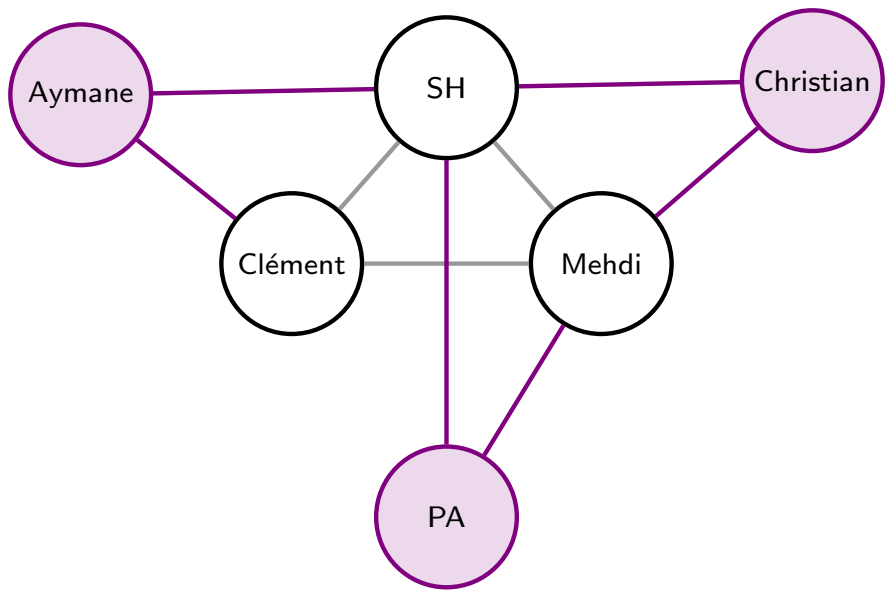
$$\mathbb{P}(\text{new node attaches to } i) = \frac{\deg(i)}{\sum_j \deg(j)}$$



High-degree nodes are more likely to receive new edges.

Preferential attachment

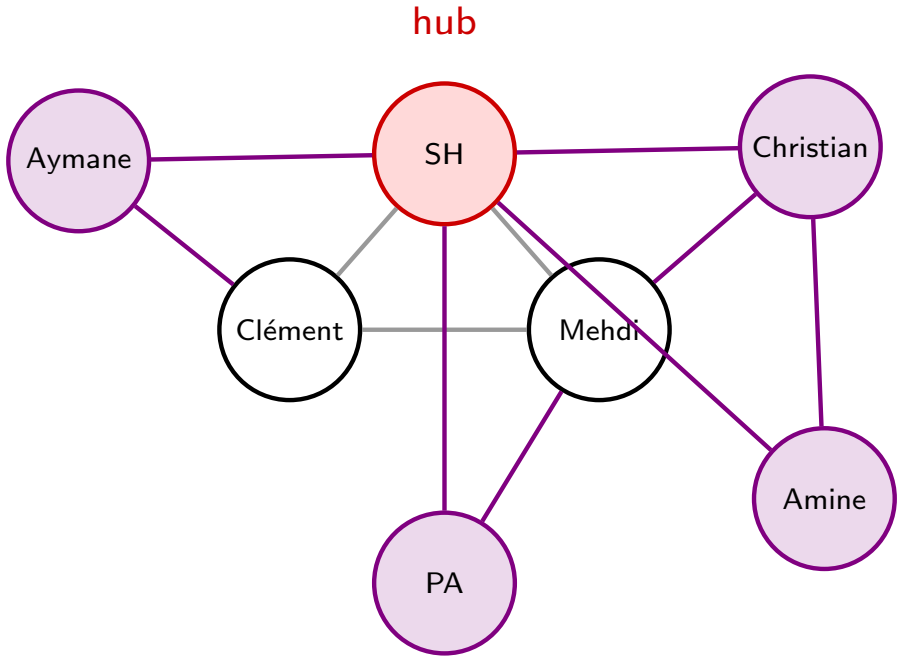
$$\mathbb{P}(\text{new node attaches to } i) = \frac{\deg(i)}{\sum_j \deg(j)}$$



The advantage compounds over time.

Preferential attachment

$$\mathbb{P}(\text{new node attaches to } i) = \frac{\deg(i)}{\sum_j \deg(j)}$$

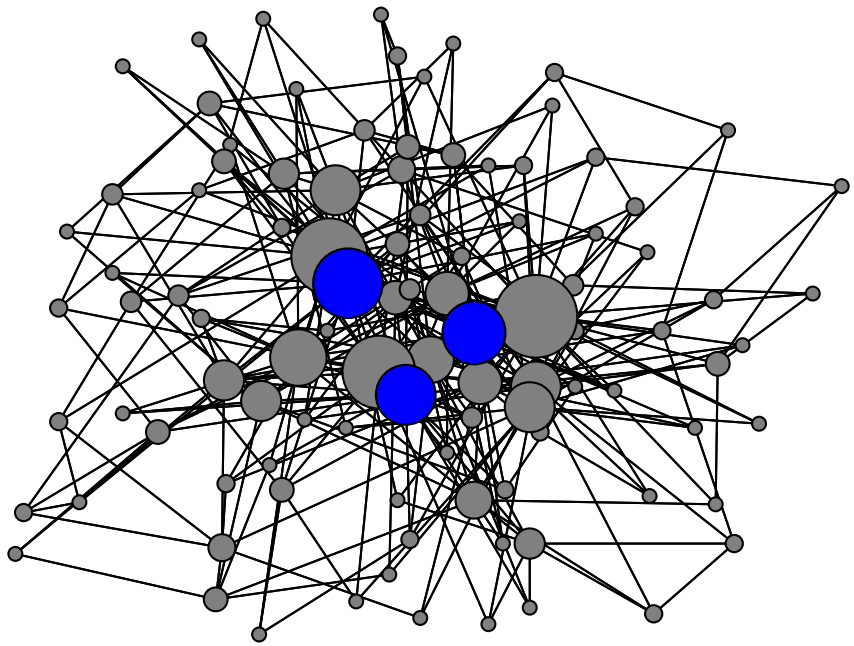


A hub emerges: **rich people get richer**

... or maybe just Sang-Hoon.

Barabási-Albert graph

Preferential attachment generates hubs.



Barabási-Albert graph with $n = 100$ and $d = 3$.

Disclaimer

- **Not all graphs follow power laws:** e.g. molecular graphs are heavily restricted by physical constraints (such as bounded atomic valency), preventing a power-law distribution.
- **Prevalence in the digital world:** particularly true for web data, including social network connections and e-commerce user-item interactions.
- **An incomplete picture:** While a power-law degree distribution is a critical structural property, it does not fully explain the underlying graph generation process. It is just one very important piece of a much larger network puzzle.

Small-world property

How many Wikipedia articles can we reach from Plato in at most k clicks?

Small-world property

How many Wikipedia articles can we reach from Plato in at most k clicks?

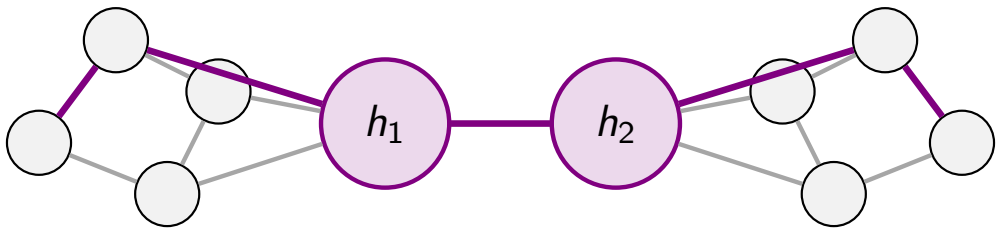
Clicks	Reachable articles	Proportion
1	392	4%
2	5,866	59%
3	9,939	99%
4	9,990	99.8%

Wikipedia Vitals: 10,011 articles, 824,999 links

In only 3 clicks, Plato reaches almost the whole graph.

Small-world property

A graph has the small-world property when typical distances are very small compared to the graph size.



Large network, short paths: hubs act as shortcuts and collapse distances.

Where are we?

Introduction and Motivations

Large Graphs and their Properties

Graph Machine Learning
- Brief History of Graph Machine Learning
- PageRank
- Graph Embedding

Graph Neural Networks

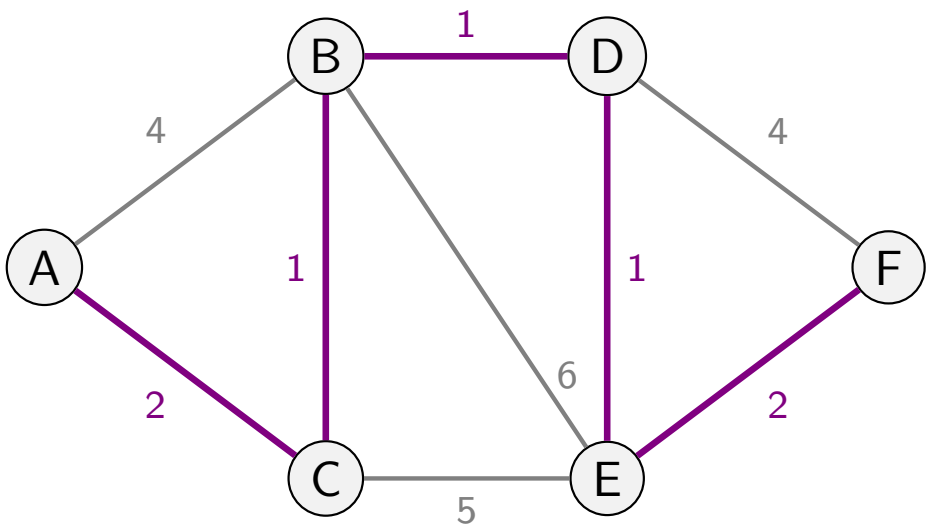
Some Applications and Perspectives

Conclusion

The Classical Era (1950s–1980s): Exact Algorithms

Early computer science focused on infrastructure, networking, and logistics (e.g., "What is the absolute shortest path for a delivery truck?").

$\Longrightarrow$ Dijkstra's shortest path (1959), Bellman-Ford, and Minimum Spanning Trees. Often have a very high time complexity.



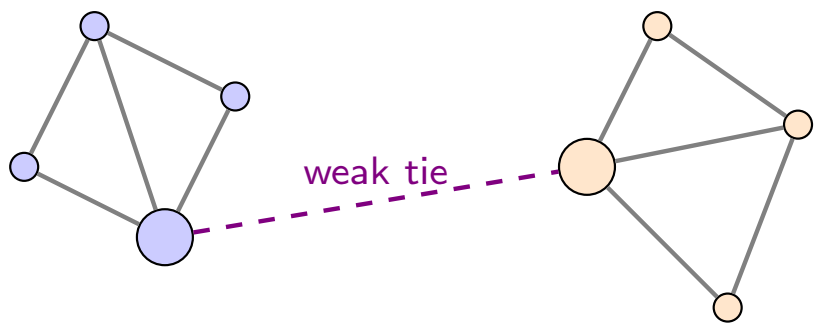
Deterministic Routing: $A \to C \to B \to D \to E \to F$ (Cost: 7)

A note on two problems in connexion with graphs (E. W. Dijkstra, Numerische Mathematik, 1959).

The Network Science Era (1990s–2000s): Macroscopic Analytics

The explosion of the Web and social networks made graphs massive, dynamic, and noisy.

$\Longrightarrow$ understanding macroscopic properties: influence, community structure, global connectivity, information flows etc. using centrality measures, Louvain (community detection) etc.



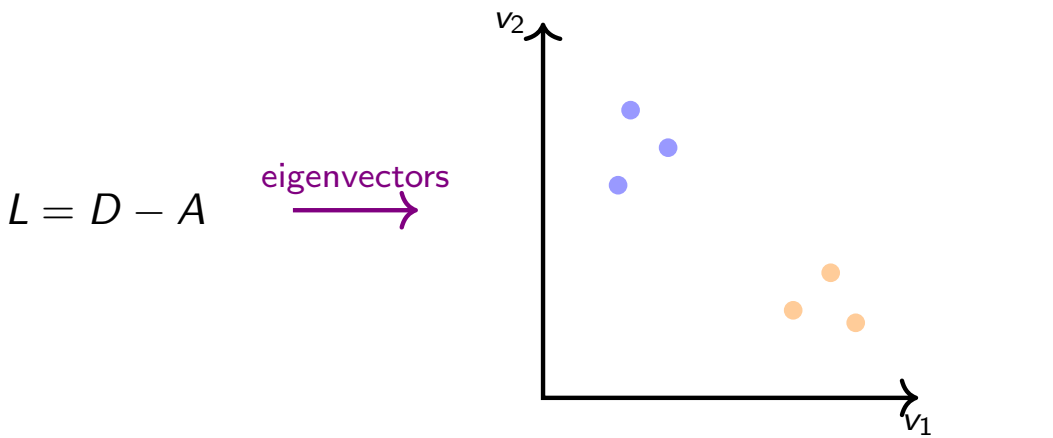
Community Detection & Centrality (Hubs)

Fast unfolding of communities in large networks (V. D. Blondel et al., J. Stat. Mech., 2008).

Early Machine Learning (2000s): Spectral Graph Theory

Needed a way to feed graph data into standard (e.g. euclidean) statistical models (like SVMs or K-Means).

$\Longrightarrow$ Laplacian Eigenmaps: using the Graph Laplacian matrix ($L = D - A$), we compute its eigenvectors capturing the graph's geometry (spectral clustering).



Nodes mapped to a 2D continuous space

Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering (M. Belkin & P. Niyogi, NeurIPS 2001).

Where are we?

Introduction and Motivations

Large Graphs and their Properties

Graph Machine Learning
- Brief History of Graph Machine Learning
- PageRank
- Graph Embedding

Graph Neural Networks

Some Applications and Perspectives

Conclusion

PageRank: motivation

In the early Web, search engines had to rank pages using both text and graph signals.

Text signal	Graph signal
Does the page match the query?	Is the page likely to be searched?

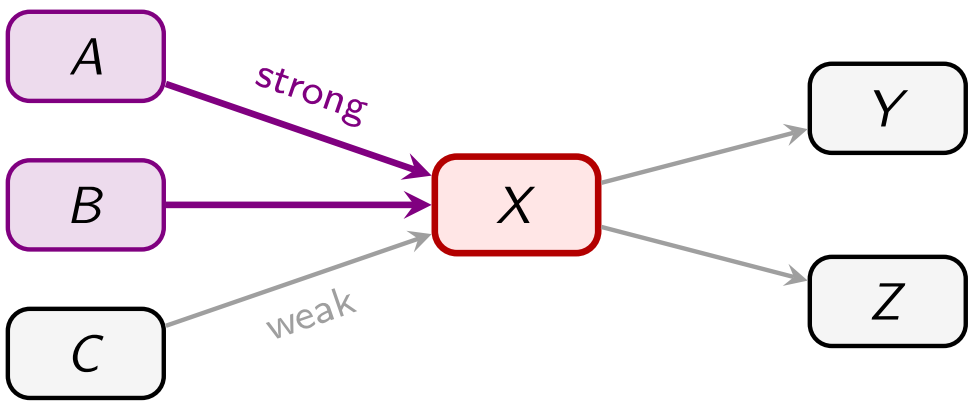
PageRank: motivation

In the early Web, search engines had to rank pages using both text and graph signals.

Text signal	Graph signal
Does the page match the query?	Is the page likely to be searched?

PageRank uses the hyperlink graph:

web pages = nodes, hyperlinks = directed edges.



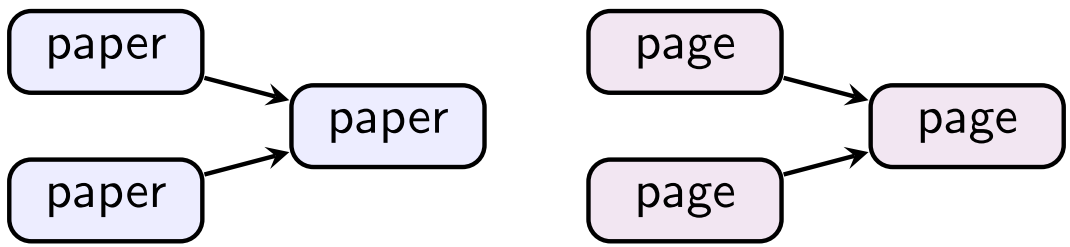
PageRank scores X using who links to it, and how important they are.

Historical context

PageRank was introduced by Larry Page and Sergey Brin at Stanford in the late 1990s.

The original idea: use the Web graph to assign a global importance score to pages, similarly to citation analysis.

link to a page $\approx$ citation of a paper.



citation graph hyperlink graph

Core idea

A page is important if important pages link to it.

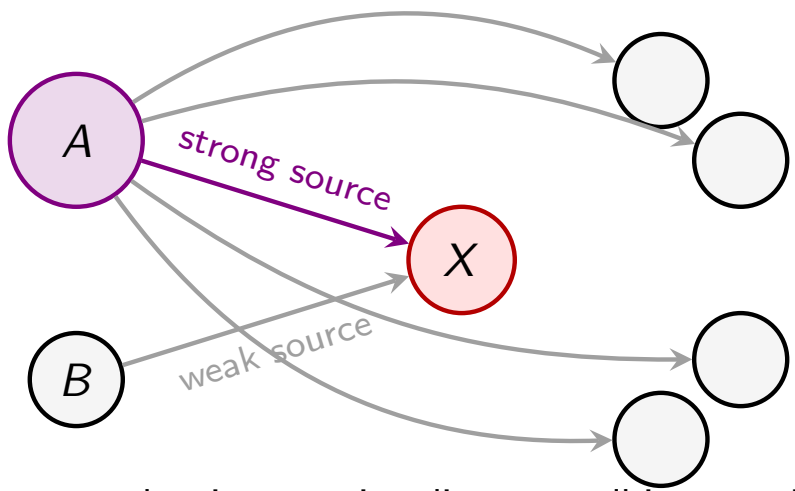
Why not just count links?

A simple baseline is in-degree:

$$\text{score}(i) = \#\{j : j \to i\}.$$

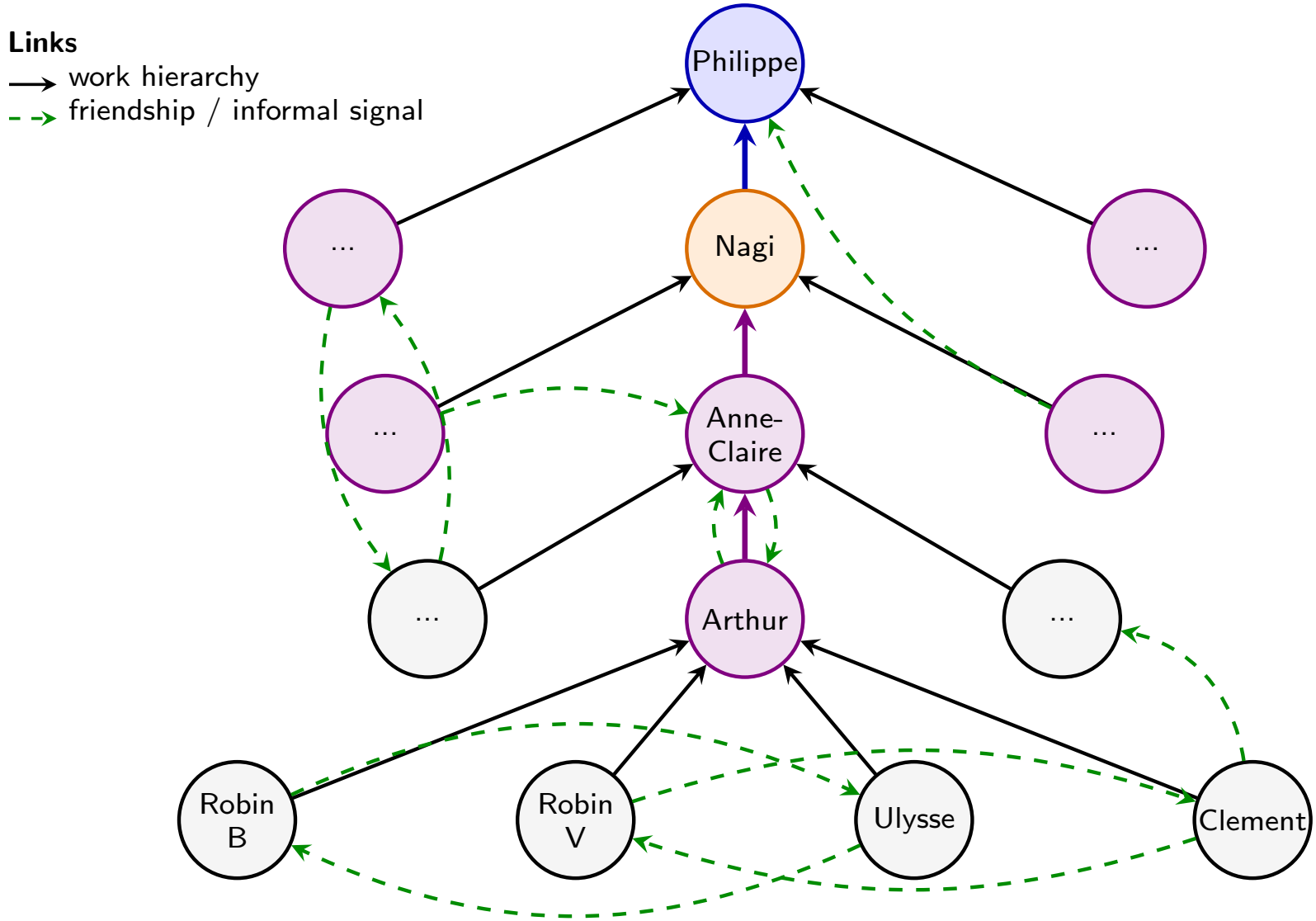
But links should not all have the same weight.

Situation	PageRank intuition
Link from an important page	stronger signal
Link from a low-score page	weaker signal
Page has many outgoing links	endorsement is diluted



A is important, but its score is split across all its outgoing links.

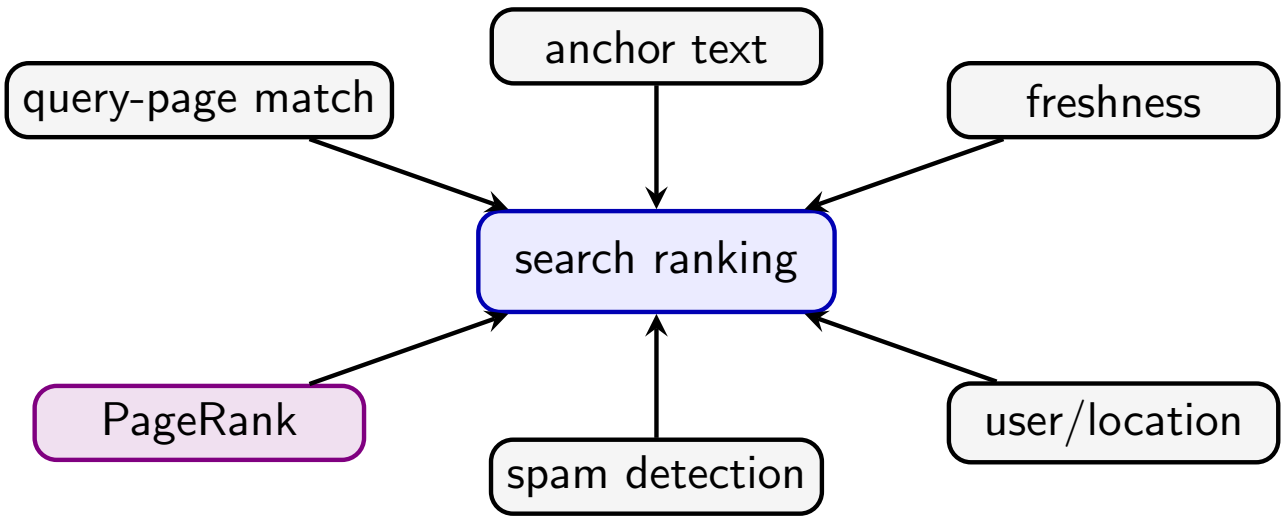
Who is the most popular at Mirakl?



What PageRank was used for

PageRank estimated page authority from the Web hyperlink graph.

In early Google, it was an important query-independent signal: the same page had the same PageRank regardless of the user query.



Public Toolbar PageRank is gone. Link-based authority remains conceptually important, but modern search ranking uses many additional signals.

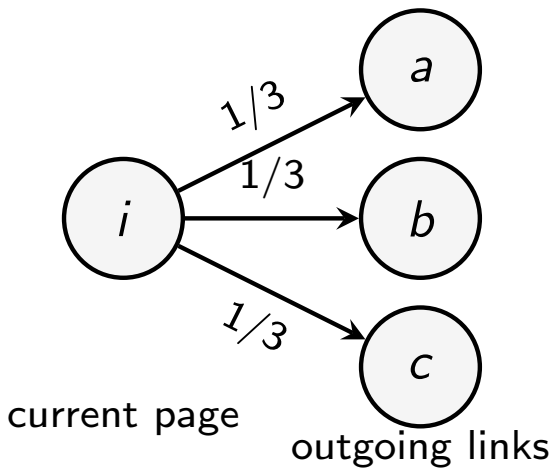
Random walk on a directed graph

Consider a directed graph with adjacency matrix A:

$$A_{ij} = 1 \quad \Longleftrightarrow \quad i \to j.$$

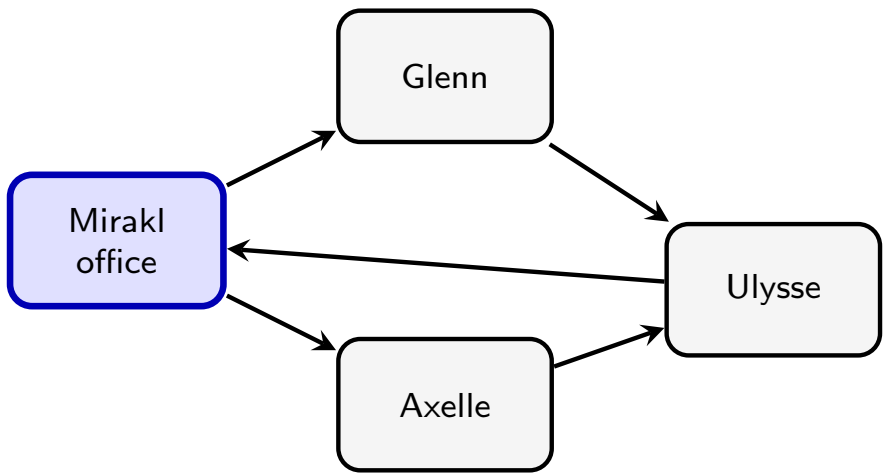
A random surfer at node i chooses uniformly among outgoing links:

$$P_{ij} = \mathbb{P}(X_{t+1} = j \mid X_t = i) = \frac{A_{ij}}{\sum_k A_{ik}}.$$



Random walk: one step at a time

A random walk is like the Perudo game at Mirakl: at each step, someone takes the game at their place.

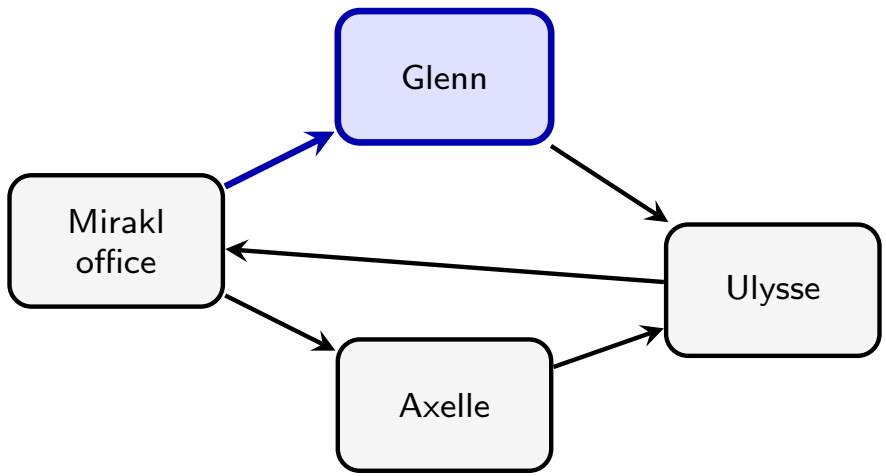


Start at the Mirakl office

$$X_0 = \text{Mirakl office}$$

Random walk: one step at a time

A random walk is like the Perudo game at Mirakl: at each step, someone takes the game at their place.

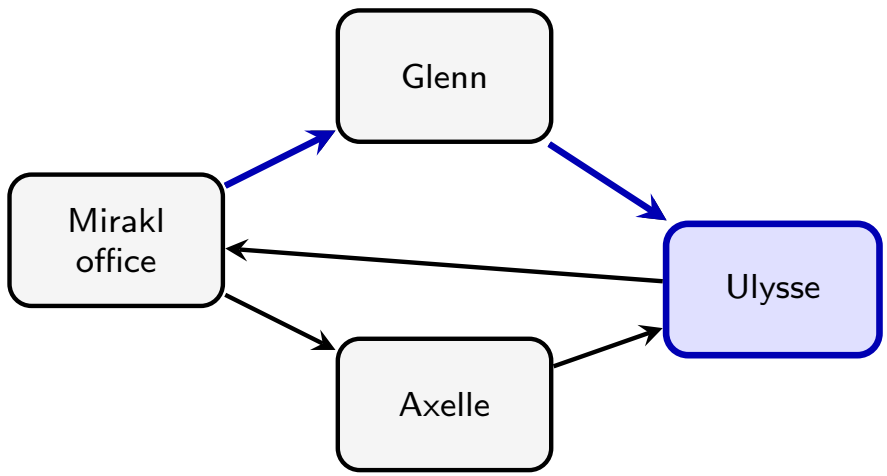


Step 1: Glenn steals the Perudo game

$$X_0 = \text{Mirakl office},\ X_1 = \text{Glenn}$$

Random walk: one step at a time

A random walk is like the Perudo game at Mirakl: at each step, someone takes the game at their place.

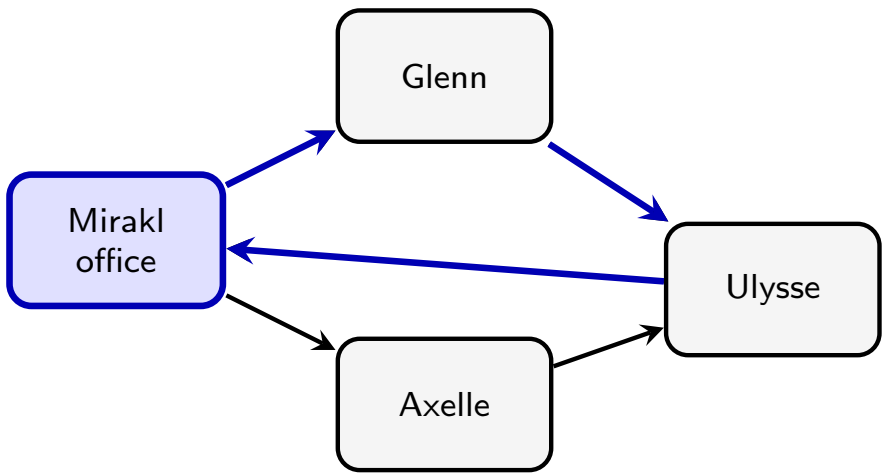


Step 2: Glenn generously gives it to Ulysse

$$X_0 = \text{Mirakl office},\ X_1 = \text{Glenn},\ X_2 = \text{Ulysse}$$

Random walk: one step at a time

A random walk is like the Perudo game at Mirakl: at each step, someone takes the game at their place.

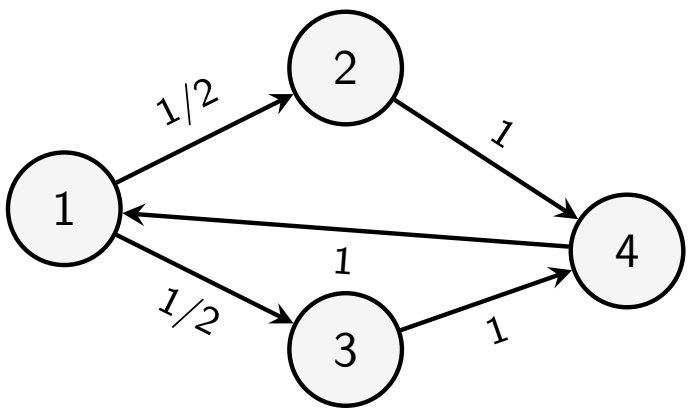


Step 3: Ulysse brings it back to the office

$$X_0 = \text{Mirakl office},\ X_1 = \text{Glenn},\ X_2 = \text{Ulysse},\ X_3 = \text{Mirakl office}, \ldots$$

Random walk as probability flow

Instead of tracking one walker, track the probability of being at each node.



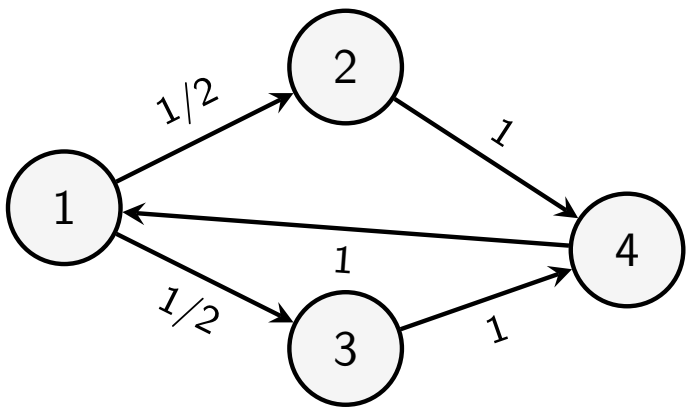
$$P = \begin{pmatrix} 0 & 1/2 & 1/2 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

$$\pi_0 = \begin{pmatrix} 1 & 0 & 0 & 0 \end{pmatrix}$$

Start with certainty at node 1.

Random walk as probability flow

Instead of tracking one walker, track the probability of being at each node.



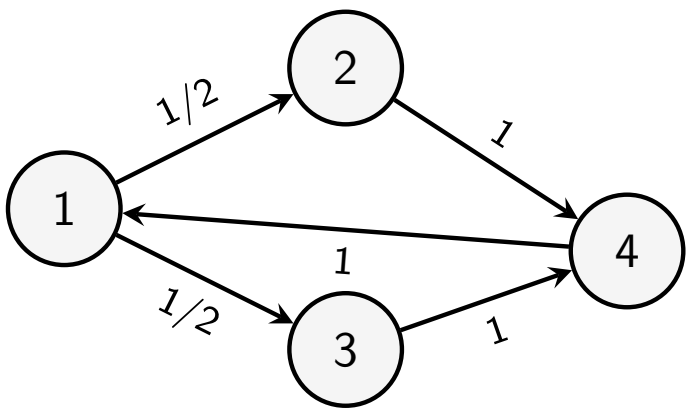
$$P = \begin{pmatrix} 0 & 1/2 & 1/2 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

$$\pi_1 = \pi_0 P = \begin{pmatrix} 0 & 1/2 & 1/2 & 0 \end{pmatrix}$$

After one step, probability splits between nodes 2 and 3.

Random walk as probability flow

Instead of tracking one walker, track the probability of being at each node.



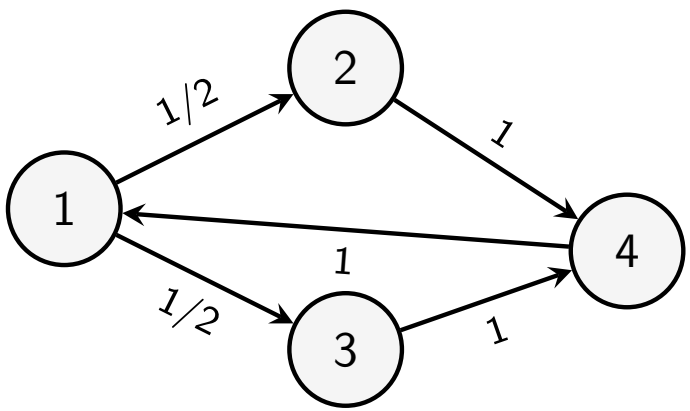
$$P = \begin{pmatrix} 0 & 1/2 & 1/2 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

$$\pi_2 = \pi_1 P = \begin{pmatrix} 0 & 0 & 0 & 1 \end{pmatrix}$$

After two steps, all probability reaches node 4.

Random walk as probability flow

Instead of tracking one walker, track the probability of being at each node.



$$P = \begin{pmatrix} 0 & 1/2 & 1/2 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

$$\pi_3 = \pi_2 P = \begin{pmatrix} 1 & 0 & 0 & 0 \end{pmatrix}$$

After three steps, probability returns to node 1.

Starting from a probability vector

We do not have to start from a single node.

A probability vector describes uncertainty about the current node:

$$\pi_0 = \begin{pmatrix} 0.4 & 0.3 & 0.2 & 0.1 \end{pmatrix}$$

Using the same transition matrix,

$$P = \begin{pmatrix} 0 & 1/2 & 1/2 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

$$\pi_1 = \pi_0 P = \begin{pmatrix} 0.1 & 0.2 & 0.2 & 0.5 \end{pmatrix}$$

Key idea

Multiplying by P moves probability mass along the edges.

Stationary distribution

Let $\pi_t(i)$ be the probability that the surfer is at page i at time t.

The distribution evolves as

$$\pi_{t+1} = \pi_t P.$$

A stationary distribution satisfies

$$\pi = \pi P, \qquad \sum_i \pi_i = 1.$$

Random-walk interpretation

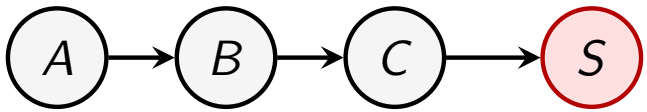
The PageRank score of a page is the long-run probability that a random surfer visits that page.

more visits $\iff$ higher rank

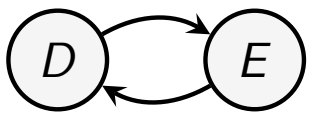
Wait, was "data scientist" supposed to include this much math?

Problem: sinks and disconnected components

A pure random walk can fail to give a useful global ranking.



dangling page



disconnected component

Two issues:

- **Dangling nodes**: pages with no outgoing links.
- **Traps**: components where the surfer can get stuck.

Teleportation

PageRank modifies the random walk with teleportation.

At each step, the surfer:

- follows an outgoing link with probability α;
- jumps to a random page with probability $1 - \alpha$.

Usually:

$$\alpha = 0.85.$$

The transition matrix becomes the Google matrix:

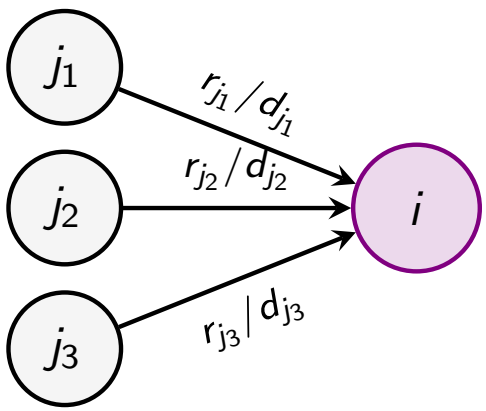
$$G = \alpha P + (1 - \alpha)\frac{1}{n}\mathbf{1}\mathbf{1}^\top.$$

link following + random restart

PageRank equation

The PageRank vector is the stationary distribution of the teleporting random walk:

$$r = rG.$$



a page receives rank from pages that link to it

Computing PageRank

PageRank is usually computed by power iteration.

Initialize

$$r^{(0)} = \frac{1}{n}\mathbf{1}.$$

Computing PageRank

PageRank is usually computed by power iteration.

Initialize

$$r^{(0)} = \frac{1}{n}\mathbf{1}.$$

Then iterate

$$\begin{aligned} r^{(t+1)} &= r^{(t)} G \\ &= \alpha\, r^{(t)} P + (1-\alpha)\frac{1}{n}\mathbf{1}. \end{aligned}$$

Computing PageRank

PageRank is usually computed by power iteration.

Initialize

$$r^{(0)} = \frac{1}{n}\mathbf{1}.$$

Then iterate

$$\begin{aligned} r^{(t+1)} &= r^{(t)} G \\ &= \alpha\, r^{(t)} P + (1-\alpha)\frac{1}{n}\mathbf{1}. \end{aligned}$$

Stop when

$$\| r^{(t+1)} - r^{(t)} \|_1 < \varepsilon.$$

Personalized PageRank

In standard PageRank, teleportation is uniform:

$$\text{teleport to page } i \quad \text{with probability} \quad \frac{1}{n}.$$

In Personalized PageRank, teleportation follows a preference vector v:

$$G_v = \alpha P + (1-\alpha)\mathbf{1}v^\top, \qquad \sum_i v_i = 1.$$

The personalized score satisfies

$$r_v = \alpha r_v P + (1-\alpha)v.$$

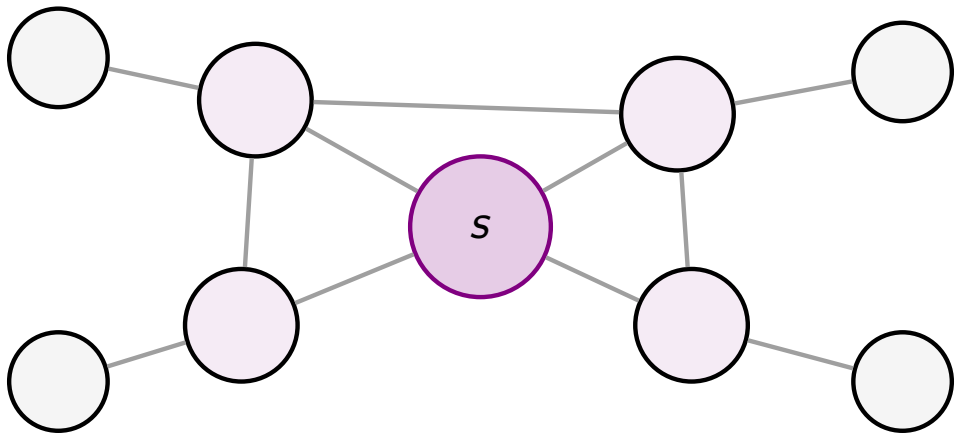
Interpretation

Rank pages by importance relative to a user, topic, seed set, or query.

Somewhere in the job description, they probably mentioned linear algebra.

Personalized PageRank: local ranking

If v is concentrated on a seed node or a seed set, the ranking becomes local.



high scores concentrate near the teleportation seeds

Used for recommendation, node proximity, graph-based retrieval, and topic-sensitive ranking.

Takeaway

PageRank turns a directed graph into node importance scores.

Input	directed graph $G = (V, E)$
Output	score r_i for each node i
Idea	stationary distribution of a random walk with teleportation

$$r = \alpha r P + (1 - \alpha) v$$

PageRank

$$v = \frac{1}{n}\mathbf{1}$$

Global importance in the graph.

Personalized PageRank

v concentrated on seeds

Importance relative to a node or group of nodes.

Graph structure $\longrightarrow$ diffusion scores

Where are we?

Introduction and Motivations

Large Graphs and their Properties

Graph Machine Learning
- Brief History of Graph Machine Learning
- PageRank
- Graph Embedding

Graph Neural Networks

Some Applications and Perspectives

Conclusion

Graph Embedding Principle

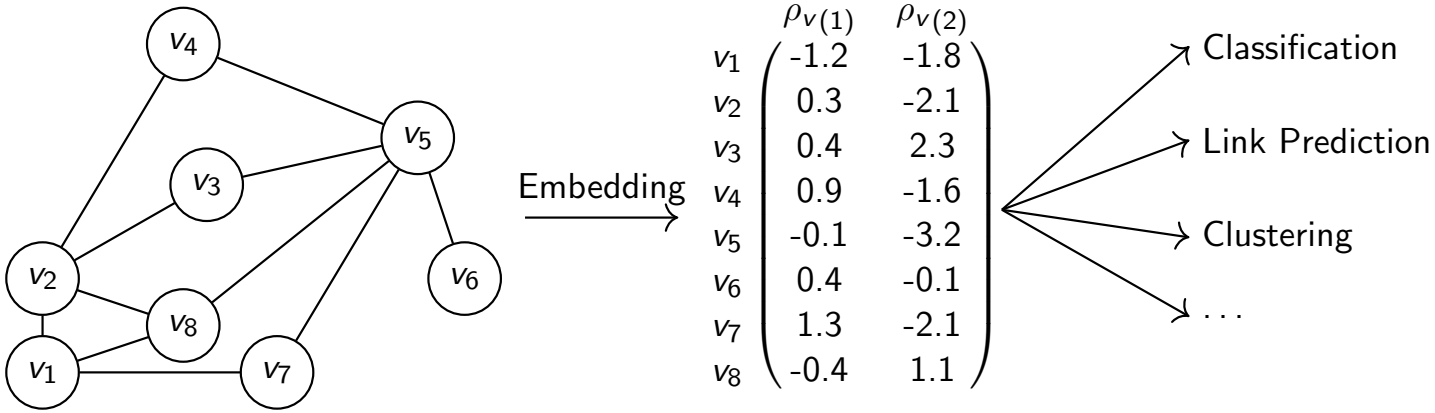


Figure: General principle of graph embedding methods. An algorithm learns a low-dimensional vector representation of dimension $\rho_v = 2$ for each node, such that the structure of the graph is preserved. These representations are then used as input features to solve downstream tasks (e.g., node classification, link prediction, graph classification) using standard machine learning algorithms.

An Example for Visualization

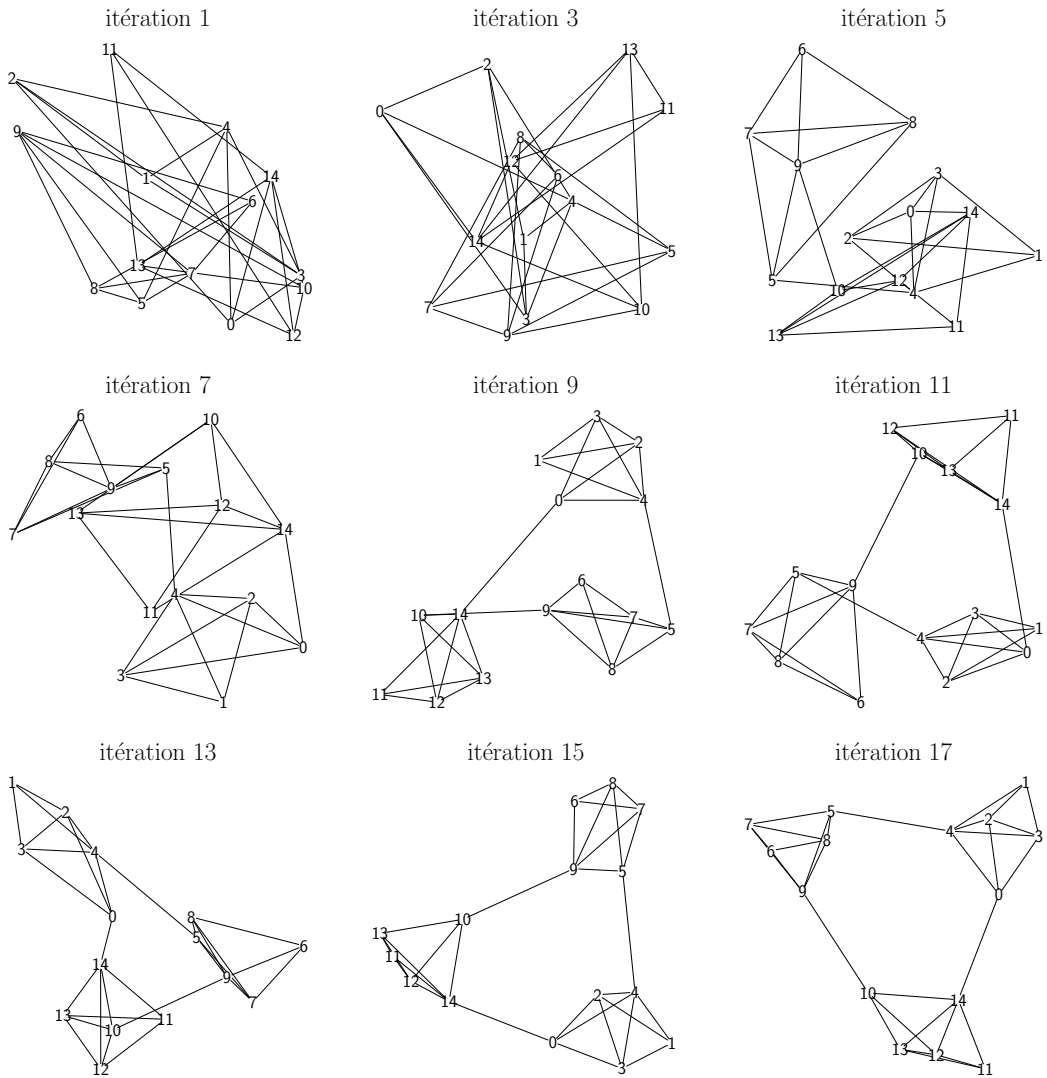


Figure: T. Fruchterman and E. Reingold, *Graph Drawing by Force-Directed Placement*, Software: Practice and Experience, 1991. https://doi.org/10.1002/spe.4380211102

The Scalability Bottleneck of Spectral Methods

- Matrix eigendecomposition has a computational complexity of $O(N^3)$, where N is the number of nodes.
- It becomes completely intractable for modern graphs with millions or billions of nodes (e.g., social networks, the Web).
- It often requires the entire graph to be loaded into memory.

Data scientists needed a method that scaled linearly with the size of the graph, and they found the answer in Natural Language Processing.

The NLP Inspiration: Word2Vec

Word2Vec with Skip-gram, 2013: inspired by he linguist J.R. Firth (1957) *"You shall know a word by the company it keeps"*. It learned dense, continuous vector representations for words by training a shallow neural network to predict a word's surrounding context in a sentence.

Translation to Graphs (DeepWalk, 2014): If we treat a graph like a document, what is a "sentence", and what is a "word"?

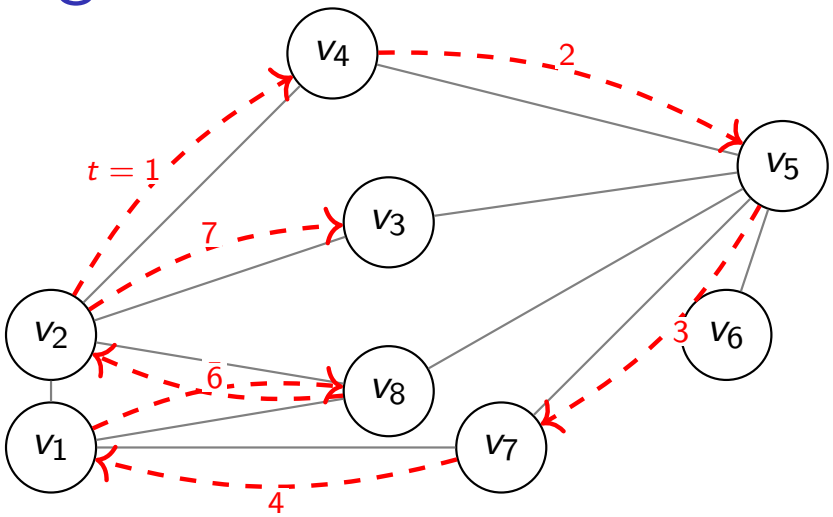
- **Word** $\Longleftrightarrow$ Node
- **Sentence** $\Longleftrightarrow$ Short Random Walk

We can use random walks on a huge graph to generate "sentences" of nodes !

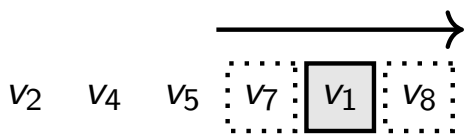
Paper: *DeepWalk: Online Learning of Social Representations* (B. Perozzi et al., KDD 2014).
`https://arxiv.org/abs/1403.6652`

DeepWalk: The Algorithm

1)



2)

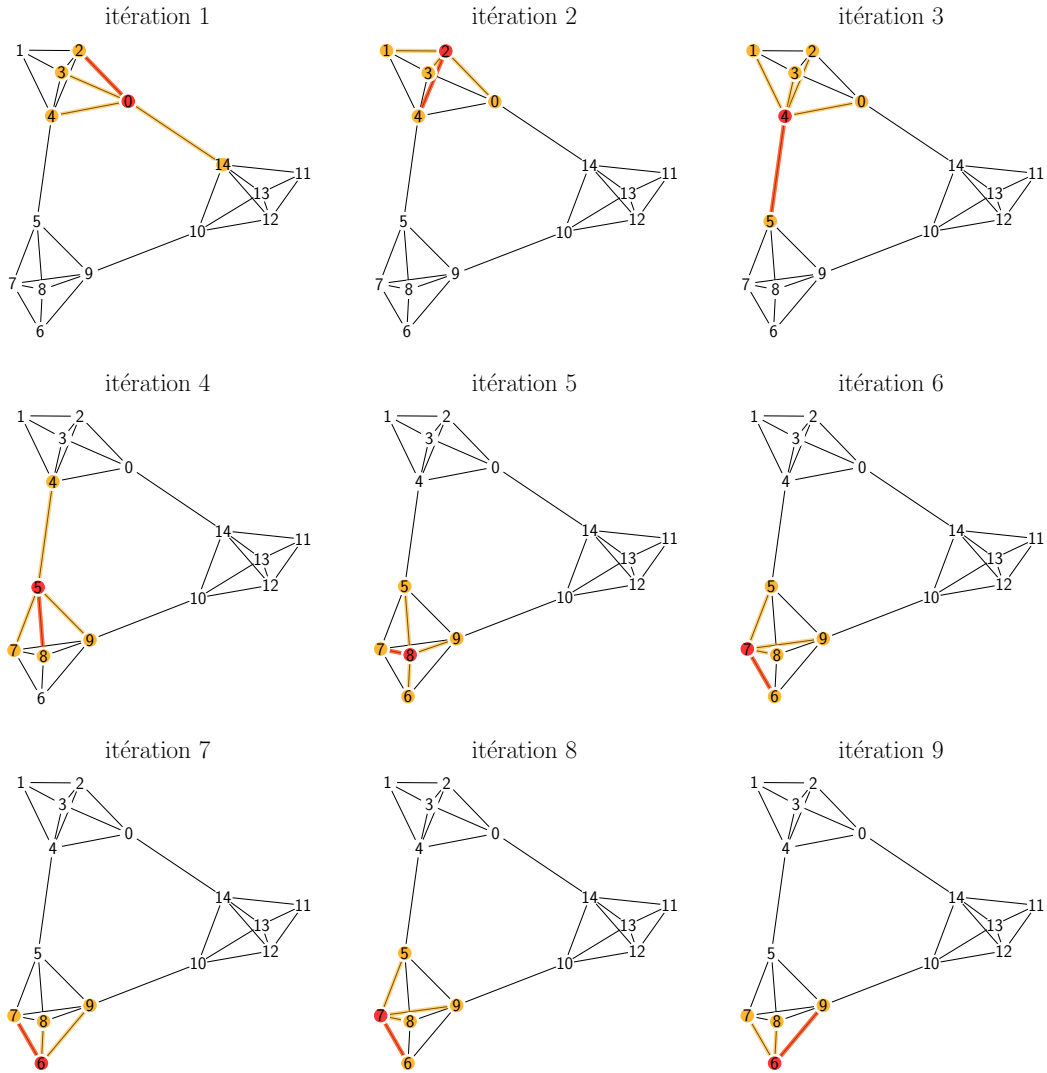


3)

target	context
v_2	v_4
v_4	v_2
...	...
$\mathbf{v_1}$	$\mathbf{v_7}$
$\mathbf{v_1}$	$\mathbf{v_8}$
...	...

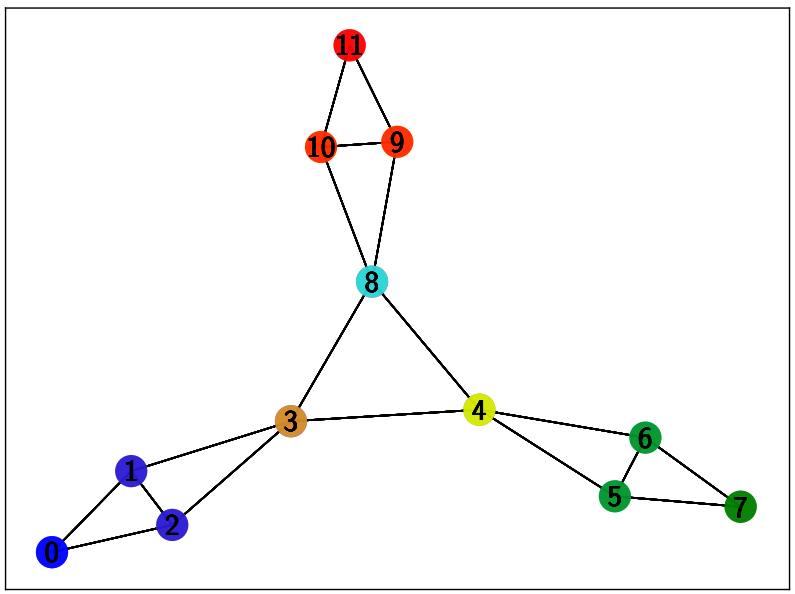
1) A random walk generates a "sentence" of nodes. **2)** A sliding window (size $\tau = 1$) captures the local context. **3)** Target-context pairs are extracted to train the Skip-gram model.

Random Walks Capture Communities and Authority

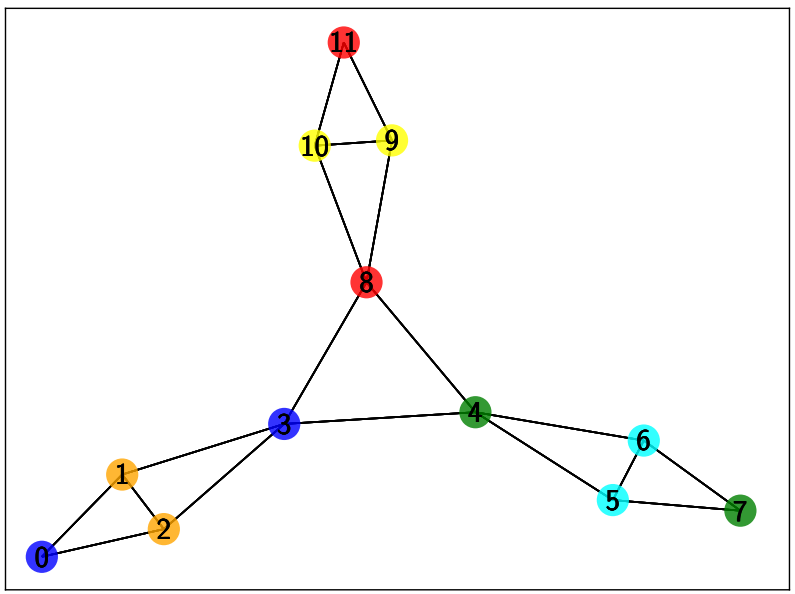


There Are Many Other Similarity Metrics

Personalized PageRank



SimRank



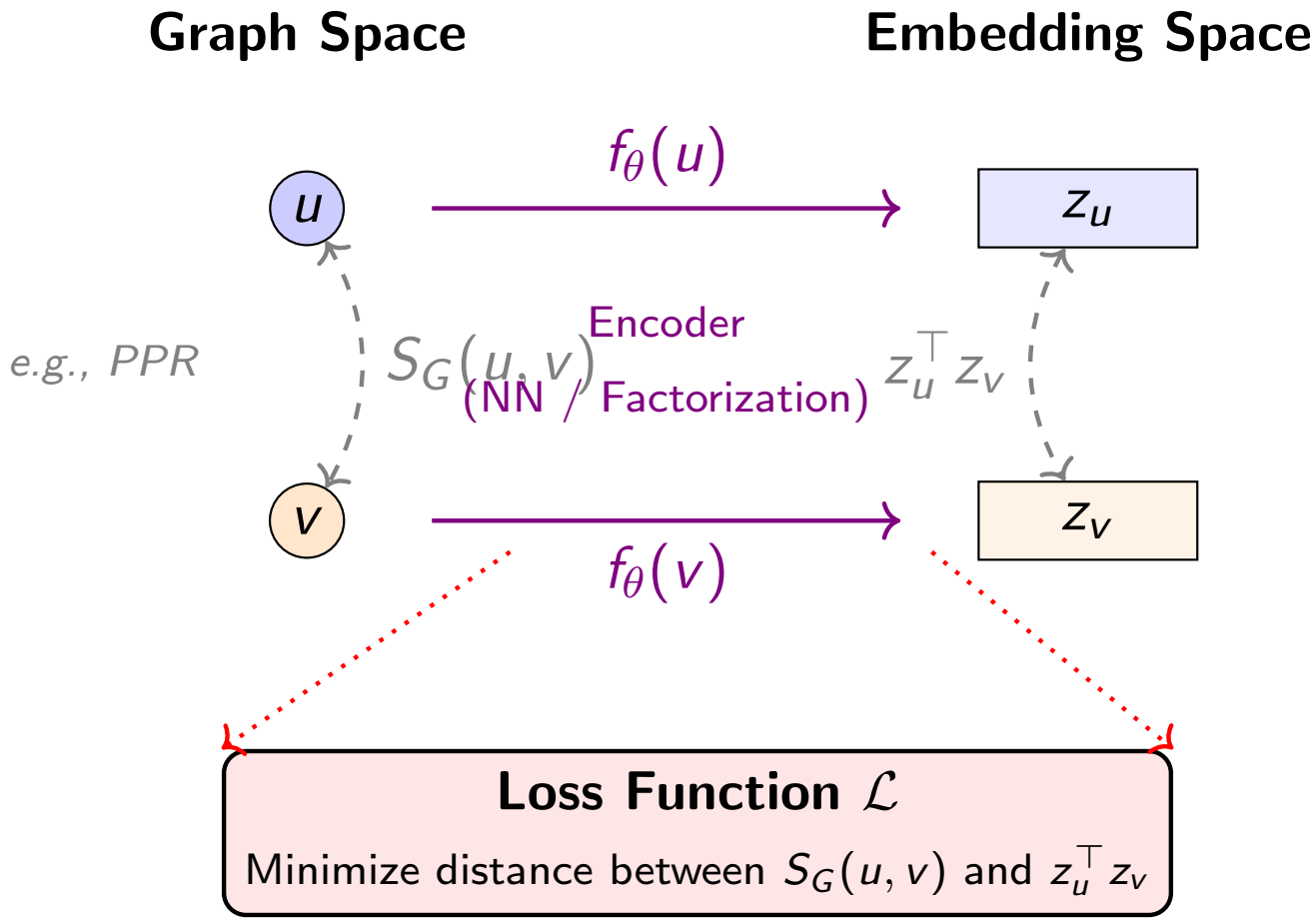
A Unified Framework for Graph Embeddings (1/2)

Learn a low-dimensional vector space designed to preserve a specific topological similarity metric.

1. **Define a Graph Metric:** Choose a pairwise similarity measure on the graph $S_G(u, v)$.
2. **Choose an Encoder:** Map nodes to dense vectors $z_u, z_v \in \mathbb{R}^d$.
3. **Optimize:** $z_u^\top z_v \approx S_G(u, v)$.

Paper: *VERSE: Versatile Graph Embeddings from Similarity Measures* (A. Tsitsulin et al., WWW 2018).
`https://dl.acm.org/doi/pdf/10.1145/3178876.3186120`

A Unified Framework for Graph Embeddings (2/2)



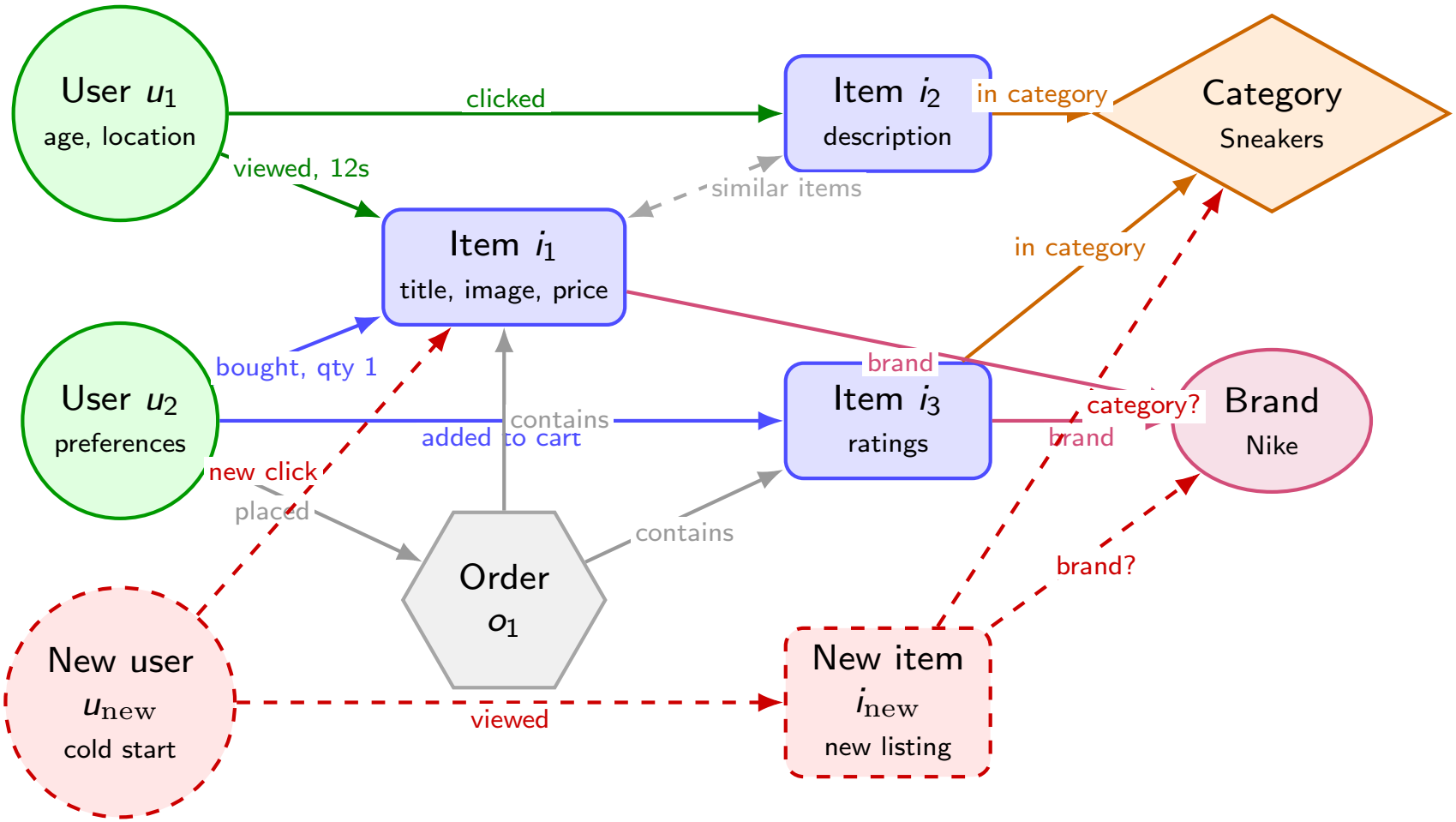
Limits of Random-Walk Embeddings

Random-walk methods (DeepWalk, Node2Vec) learn embeddings solely from graph topology. This leads to these fundamental limitations:

1. Feature-agnostic
2. Homogeneous graphs only
3. Transductive but not Inductive
4. **Still hand-crafted objective !!**

$\Rightarrow$ **Graph Neural Networks** address all these by propagating and transforming *feature vectors* along edges through learnable aggregation functions.

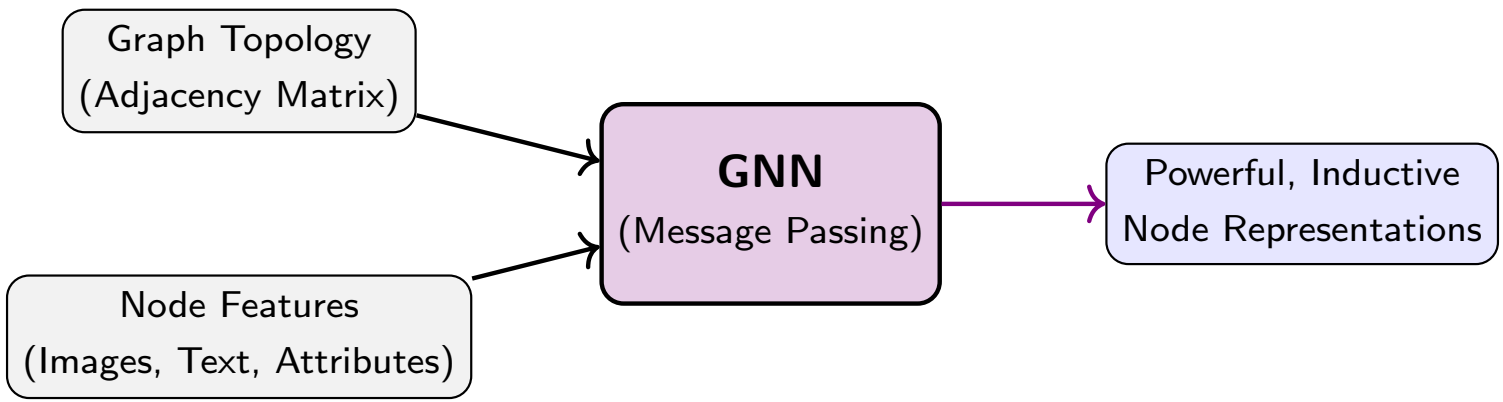
E-Commerce Graph with Features and Types



The Advent of Graph Neural Networks

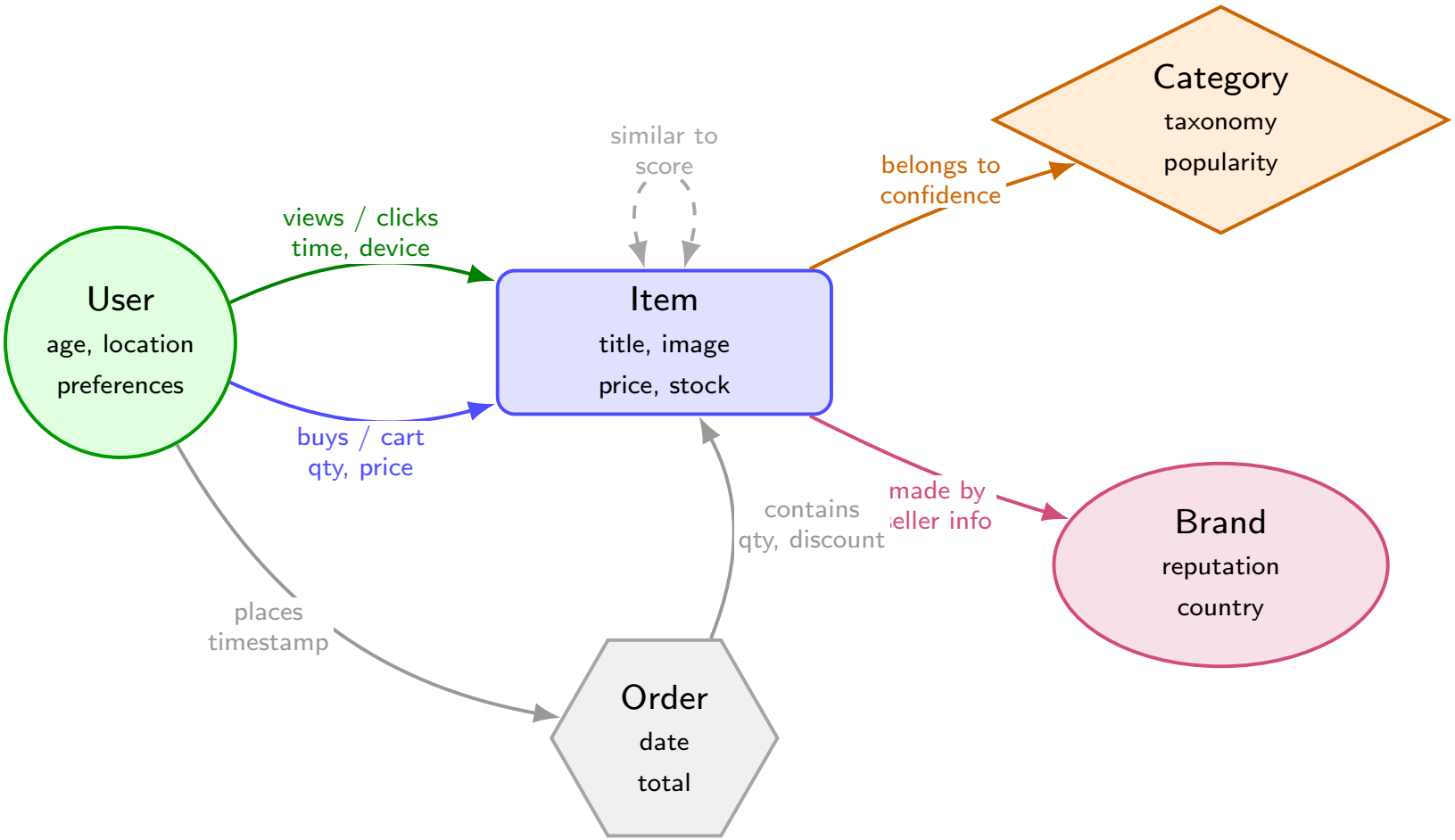
Graph Neural Networks (GNNs) emerged as the ultimate solution by unifying features and topology through **Message Passing**.

GNNs explicitly propagate the actual *node features* (text embeddings, image CNN vectors, tabular data) along the edges of the graph.



In this paradigm, the graph topology defines the computational graph of the model i.e. the data itself dictates the architecture, which can fit many scenarios.

Meta-Graph with Node and Edge Features



Where are we?

Introduction and Motivations

Large Graphs and their Properties

Graph Machine Learning

Graph Neural Networks
- Motivation
- From CNNs to message passing
- Limitations and extensions

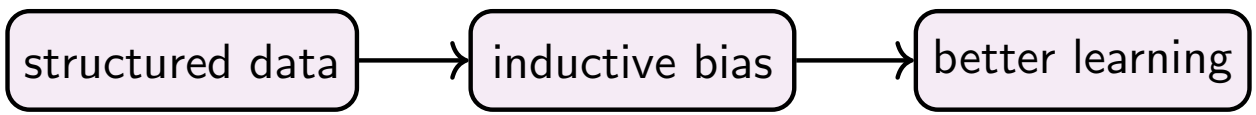
Some Applications and Perspectives

Conclusion

Why graph neural networks?

Deep learning is very effective when the architecture matches the structure of the data.

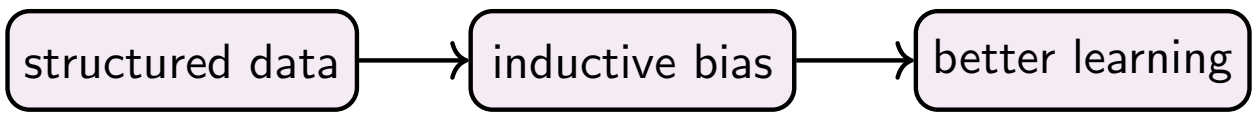
Data	Structure	Typical architecture
Images	regular grid	CNN
Text	sequence of tokens	Transformer
Graphs	nodes and edges	GNN



Why graph neural networks?

Deep learning is very effective when the architecture matches the structure of the data.

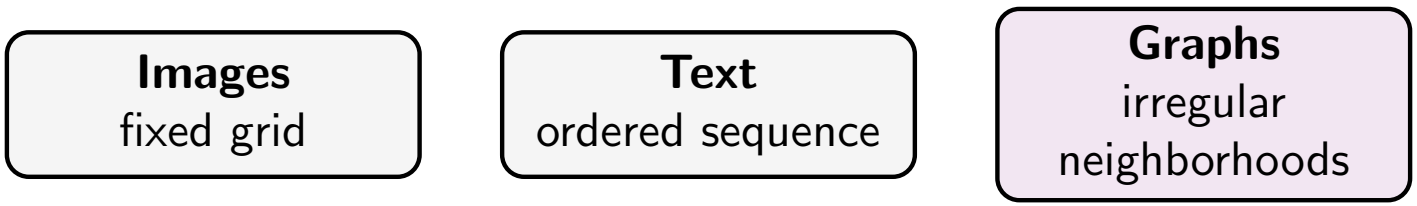
Data	Structure	Typical architecture
Images	regular grid	CNN
Text	sequence of tokens	Transformer
Graphs	nodes and edges	GNN



For graphs, the structure is the edge set.

What makes graphs different?

Graphs are not vectors, images, or sequences.



A graph model must handle:

- **arbitrary node ordering**: node IDs should not change the prediction;
- **variable degrees**: some nodes have 2 neighbors, others 10,000;
- **sparsity and scale**: real graphs can be very large;
- **local structure**: edges define which entities interact.

A GNN is a neural network that respects graph structure.

Prediction tasks on graphs

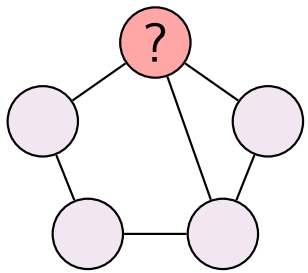
The same people graph can support different prediction tasks.

Thank you, ChatGPT, for the figures.

Prediction tasks on graphs

The same people graph can support different prediction tasks.

Node-level



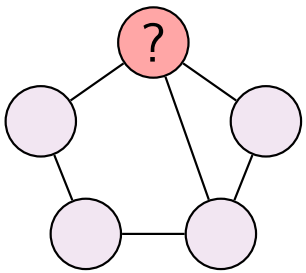
Predict the team of a person.

Thank you, ChatGPT, for the figures.

Prediction tasks on graphs

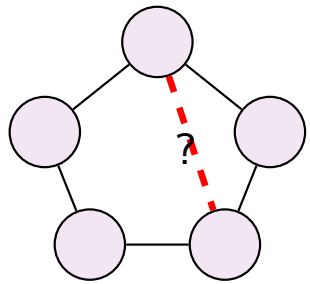
The same people graph can support different prediction tasks.

Node-level



Predict the team of a person.

Edge-level



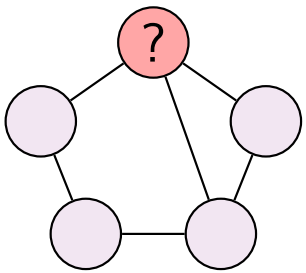
Predict if two people will work together daily.

Thank you, ChatGPT, for the figures.

Prediction tasks on graphs

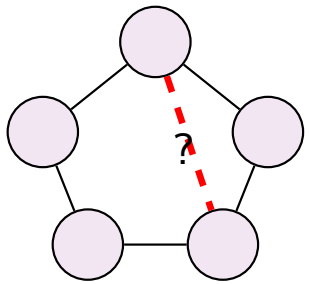
The same people graph can support different prediction tasks.

Node-level



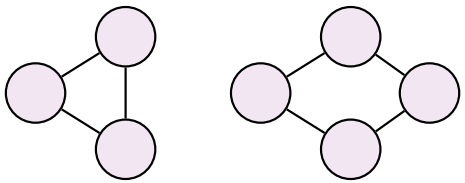
Predict the team of a person.

Edge-level



Predict if two people will work together daily.

Graph-level



label?

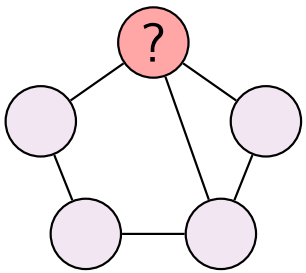
Predict how many people will be promoted in a team.

Thank you, ChatGPT, for the figures.

Prediction tasks on graphs

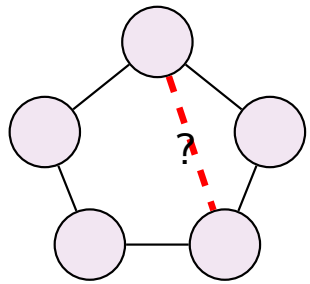
The same people graph can support different prediction tasks.

Node-level



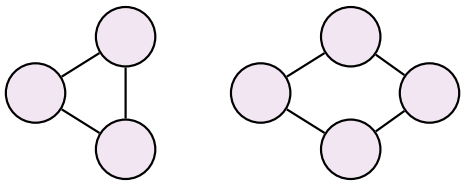
Predict the team of a person.

Edge-level



Predict if two people will work together daily.

Graph-level



label?

Predict how many people will be promoted in a team.

Examples: fraud detection, recommendation, molecules, knowledge graphs, traffic networks.

Thank you, ChatGPT, for the figures.

A supervised node classification setup

We are given one graph:

$$G = (V, E), \qquad n = |V|.$$

Each node has features:

$$x_v \in \mathbb{R}^{d_0}, \qquad X \in \mathbb{R}^{n \times d_0}.$$

A GNN outputs one prediction per node:

$$\hat{y}_v = f_\theta(G, X)_v.$$

For node classification:

$$y_v \in \{1, \ldots, C\}, \qquad \hat{y}_v \in \mathbb{R}^C.$$

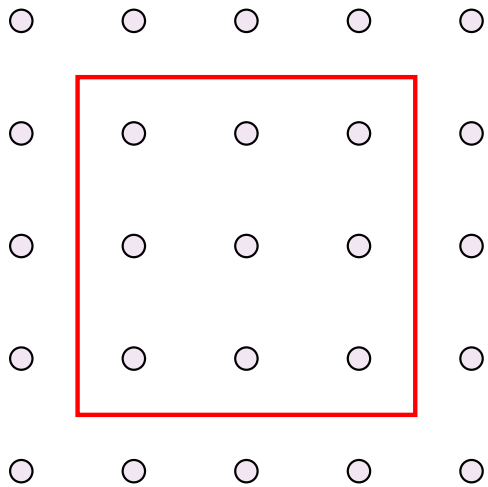
Goal: use both node features and graph structure.

From CNNs to GNNs

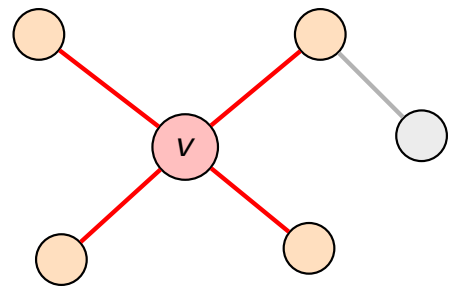
CNNs work well on images because local neighborhoods are fixed and reused everywhere.

From CNNs to GNNs

CNNs work well on images because local neighborhoods are fixed and reused everywhere.



fixed 3×3 neighborhood



graph-defined neighborhood

GNNs generalize the idea of local aggregation to irregular neighborhoods.

Message passing

A GNN computes node representations layer by layer.

Message passing

A GNN computes node representations layer by layer.

Initial representation:

$$h_v^{(0)} = x_v.$$

Message passing

A GNN computes node representations layer by layer.

Initial representation:

$$h_v^{(0)} = x_v.$$

At layer ℓ, node v aggregates information from its neighbors:

$$m_v^{(\ell)} = \mathrm{AGG}^{(\ell)}\left(\{h_u^{(\ell)} : u \in \mathcal{N}(v)\}\right).$$

Message passing

A GNN computes node representations layer by layer.

Initial representation:

$$h_v^{(0)} = x_v.$$

At layer ℓ, node v aggregates information from its neighbors:

$$m_v^{(\ell)} = \mathrm{AGG}^{(\ell)}\left(\{h_u^{(\ell)} : u \in \mathcal{N}(v)\}\right).$$

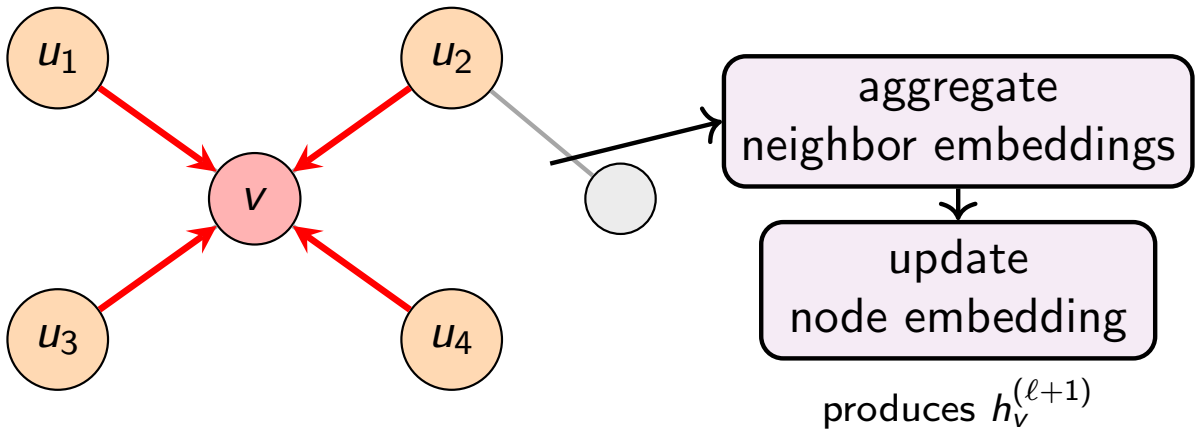
Then it updates its own representation:

$$h_v^{(\ell+1)} = \mathrm{UPDATE}^{(\ell)}\left(h_v^{(\ell)}, m_v^{(\ell)}\right).$$

neighbors send messages, nodes update states.

Message passing: visual view

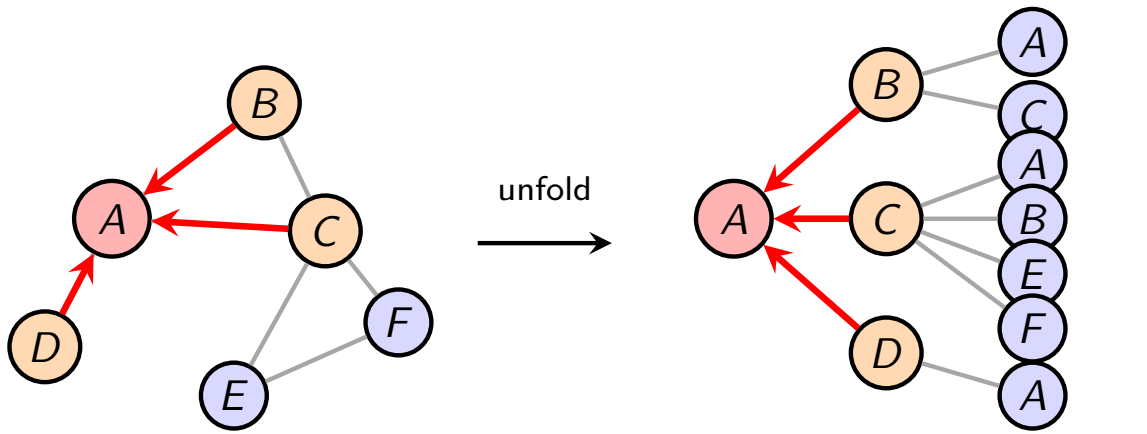
One layer updates each node using its 1-hop neighborhood.



The same aggregation and update functions are shared across all nodes.

Computation tree and receptive field

Stacking message-passing layers unfolds the graph into a computation tree.



original graph, target A two-layer computation tree for A

After k layers, node A can use information from its k-hop receptive field.

Example: GraphSAGE layer

GraphSAGE is a simple message-passing GNN.

For each node v:

$$m_v^{(\ell)} = \mathrm{AGG}\left(\{h_u^{(\ell)} : u \in \mathcal{N}(v)\}\right)$$

$$h_v^{(\ell+1)} = \sigma\left(W^{(\ell)}\mathrm{CONCAT}\left(h_v^{(\ell)}, m_v^{(\ell)}\right)\right).$$

Common choices for AGG: mean, max-pooling.

Example: GraphSAGE layer

GraphSAGE is a simple message-passing GNN.

For each node v:

$$m_v^{(\ell)} = \mathrm{AGG}\left(\{h_u^{(\ell)} : u \in \mathcal{N}(v)\}\right)$$

$$h_v^{(\ell+1)} = \sigma\left(W^{(\ell)}\mathrm{CONCAT}\left(h_v^{(\ell)}, m_v^{(\ell)}\right)\right).$$

Common choices for AGG: mean, max-pooling.

Why it is useful

The same rule is applied to every node, so it can generalize to new nodes or new graphs.

Example: GCN layer

A Graph Convolutional Network averages transformed neighbor representations.

Add self-loops and normalize:

$$\widetilde{A} = A + I, \qquad \widehat{A} = \widetilde{D}^{-1/2}\widetilde{A}\widetilde{D}^{-1/2}$$

Matrix form:

$$H^{(\ell+1)} = \sigma\left(\widehat{A}H^{(\ell)}W^{(\ell)}\right).$$

Example: GCN layer

A Graph Convolutional Network averages transformed neighbor representations.

Add self-loops and normalize:

$$\widetilde{A} = A + I, \qquad \widehat{A} = \widetilde{D}^{-1/2}\widetilde{A}\widetilde{D}^{-1/2}$$

Matrix form:

$$H^{(\ell+1)} = \sigma\left(\widehat{A}H^{(\ell)}W^{(\ell)}\right).$$

For one node, this is:

$$h_i^{(\ell+1)} = \sigma\left(\sum_{j\in\mathcal{N}(i)\cup\{i\}} \frac{1}{\sqrt{\hat{d}_j\hat{d}_i}} h_j^{(\ell)}W^{(\ell)}\right), \qquad \hat{d}_i = |\mathcal{N}(i)| + 1.$$

linear transform + normalized neighbor averaging + nonlinearity

This is the last formula. My stomach is already in the lunch break.

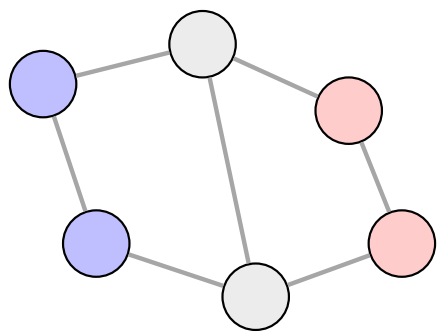
Training for node classification

A GNN predicts a class distribution for every node:

$$\hat{y}_v = \mathrm{softmax}(z_v), \qquad z_v \in \mathbb{R}^C.$$

The supervised loss is computed only on labeled training nodes:

$$\mathcal{L} = \sum_{v \in V_{\mathrm{train}}} \ell(\hat{y}_v, y_v).$$



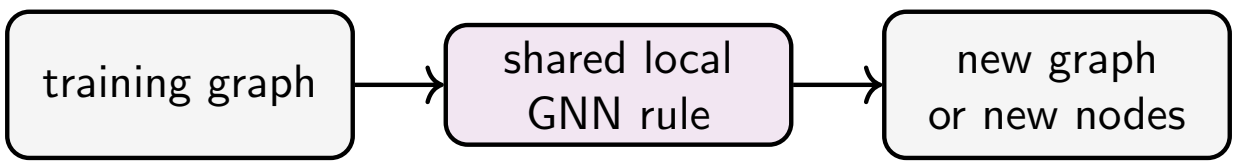
blue: labeled train
red: validation/test
gray: unlabeled

Unlabeled nodes can still contribute through message passing.

Transductive vs. inductive learning

Two common settings are useful to distinguish.

Setting	Meaning
Transductive	The graph is known during training. Some node labels are hidden and predicted later.
Inductive	The model must generalize to new nodes or new graphs not seen during training.



GraphSAGE-style message passing is naturally inductive because it learns local functions, not node-specific parameters.

Limitations of message passing

Message passing is simple and scalable, but it has limitations.

Very large graphs
hard to keep 10B nodes and features in memory

Oversmoothing
limits the expressiveness of deep GNNs

Long-range signals
distant information is hard to use

Heterophily
neighbors may have different labels

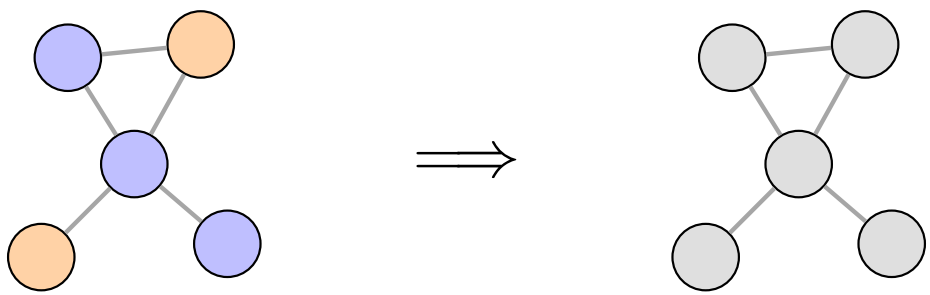
The graph is a useful inductive bias, but not always the right one.

Oversmoothing

Each layer averages information with neighbors.

With many layers, node embeddings can become too similar:

$$h_u^{(k)} \approx h_v^{(k)} \quad \text{for many nodes } u, v.$$



early layers many layers

This is one reason many practical GNNs are shallow.

Extension: Sampling for scalable GNNs

Sampling trains a GNN on a smaller part of the graph at each step.

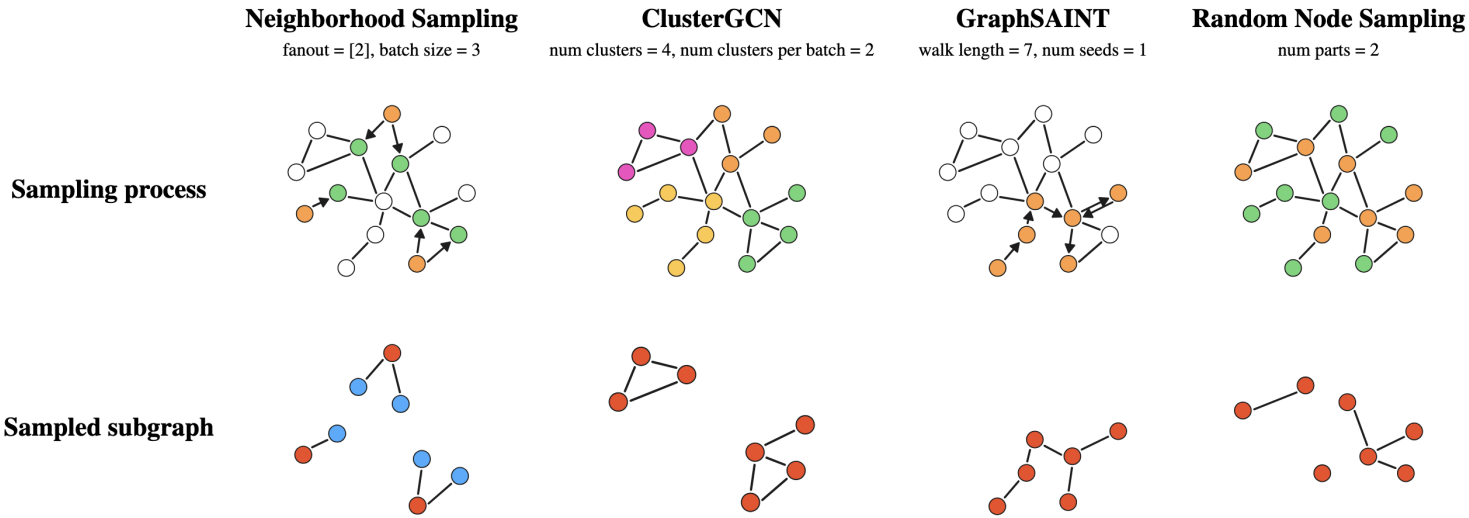


Figure 1: **Batch sampling strategies for GNN training.** Top: mini-batch construction for each method. Bottom: resulting sampled subgraph. Red nodes are targets (B^{train}), while blue nodes are auxiliary nodes used only for message passing ($B \setminus B^{\text{train}}$). See Appendix E.1.1 for details.

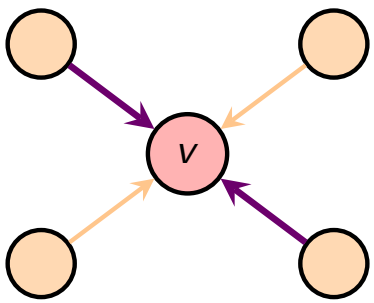
- **Neighborhood**: sample neighbors layer by layer.
- **ClusterGCN**: sample dense graph clusters.
- **GraphSAINT**: random walks sampling.
- **Random nodes**: randomly selecting nodes.

Paper under review for NeurIPS 2026, stay tuned!

Extension: Is attention all you need?

Two ways to add attention to graph learning:

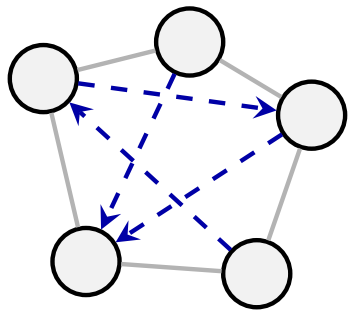
GAT: local attention



Learn weights on neighbors instead of using mean/sum.

$$h'_v = \sum_{u \in \mathcal{N}(v)} \alpha_{vu} W h_u$$

Graph Transformer: global attention



Let nodes attend beyond local neighbors.

Need tricks for graph structure and quadratic cost.

Attention is promising, but not a free lunch on large graphs.

Takeaways

1. GNNs learn representations by passing messages along graph edges.
2. A k-layer GNN uses information from the k-hop neighborhood.
3. GCN and GraphSAGE are two standard message-passing architectures.
4. GNNs are useful when edges encode predictive signal.

Where are we?

Introduction and Motivations

Large Graphs and their Properties

Graph Machine Learning

Graph Neural Networks

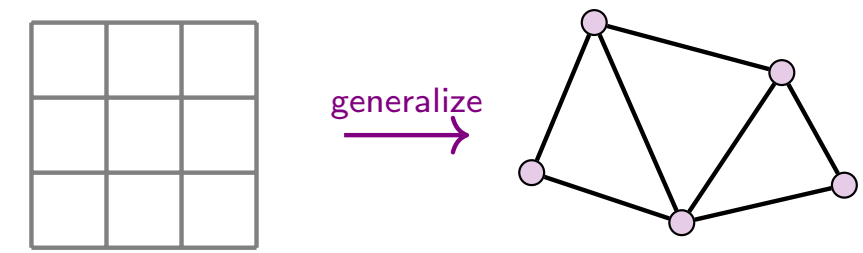
Some Applications and Perspectives

Conclusion

Board Games: AlphaZero and beyond

AlphaZero (DeepMind's chess AI) used CNNs to evaluate board states.

Replacing the CNN backbone with a GNN allows the network to become invariant to rigid grid geometry, processing arbitrary topologies and scaling to most board sizes.



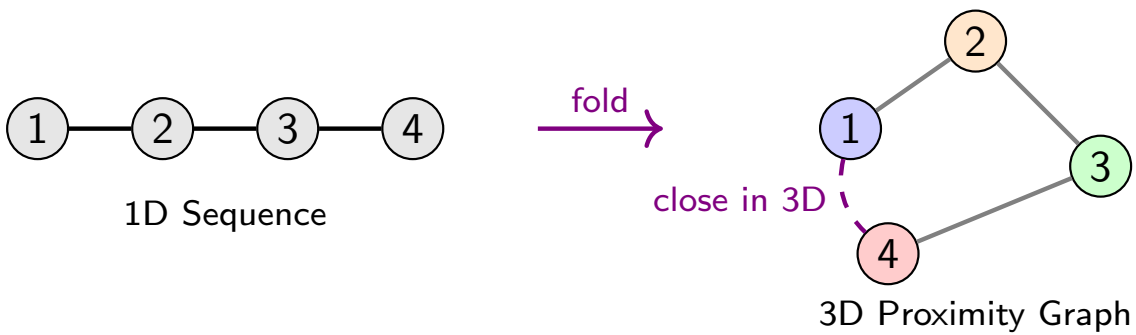
CNN (Rigid Grid) GNN (Irregular Graph)

Paper: *Scaling Up Board Games with AlphaZero and GNN* (G. de Mello et al., 2021).
`https://arxiv.org/abs/2107.08387`

Biology: AlphaFold

Predicting the 3D structure of a protein from a 1D sequence of amino acids.

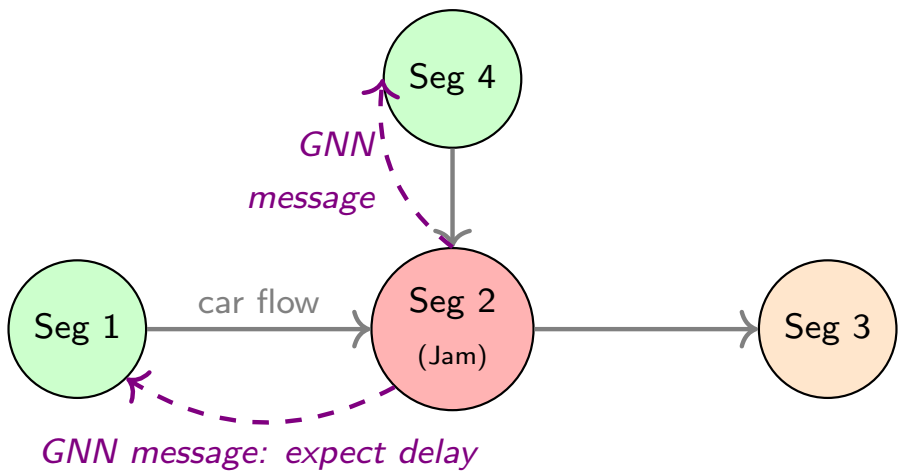
AlphaFold 2 uses graph-based attention where amino acids are nodes, and edges represent physical proximity. Message passing enforces a geometric inductive bias directly in 3D space.



Paper: *Highly accurate protein structure prediction with AlphaFold* (J. Jumper et al., Nature 2021).
`https://www.nature.com/articles/s41586-021-03819-2`

Logistics & Navigation: Google Maps ETA

Google models the road network as a graph where nodes are road segments and edges are intersections. GNN message passing allows a backed-up segment to pass a "delay message" to its upstream neighbors. This captures traffic flow dynamics and boosted ETA accuracy by up to 50%.

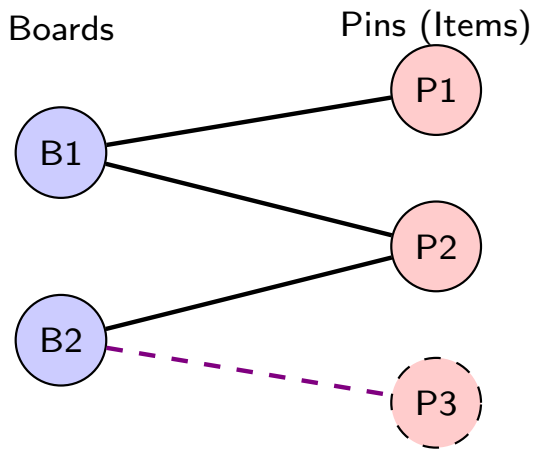


Paper: *ETA Prediction with Graph Neural Networks in Google Maps* (A. Derrow-Pinion et al., CIKM 2021). `https://arxiv.org/abs/2108.11482`

Recommender Systems: Pinterest & PinSage

Standard collaborative filtering suffers from severe cold-start problems. A newly added item with no interactions cannot be easily recommended.

PinSage creates a bipartite graph of Boards and Pins. It is inductive by aggregating features from a node's local graph neighborhood to generate high-quality embeddings even for new items with minimal interaction.



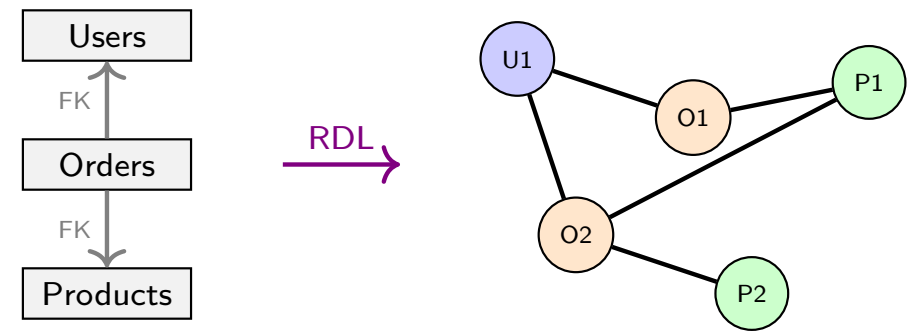
New Pin P3 aggregates data from Board B2, which shares Pin P2 with Board B1.

Paper: *Graph Convolutional Neural Networks for Web-Scale Recommender Systems* (R. Ying et al., KDD 2018).
`https://arxiv.org/abs/1806.01973`

Relational Deep Learning (RDL)

Extracting a flat tabular matrix requires complex SQL pipelines (`JOIN`, `GROUP BY`) and manual feature engineering, which destroys relational structure and risks data leakage.

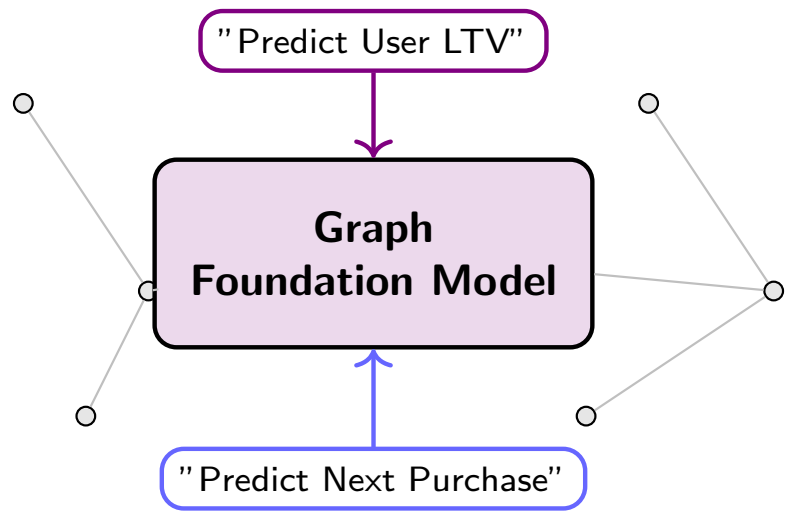
RDL treats the raw relational database directly as a heterogeneous graph. Tables are different node types, and foreign keys are the edges. The network learns aggregations automatically via message passing.



Paper: *Relational Deep Learning: Graph Representation Learning on Relational Databases* (M. Fey et al., ICML 2024). `https://arxiv.org/abs/2312.03014`

Graph Foundation Models

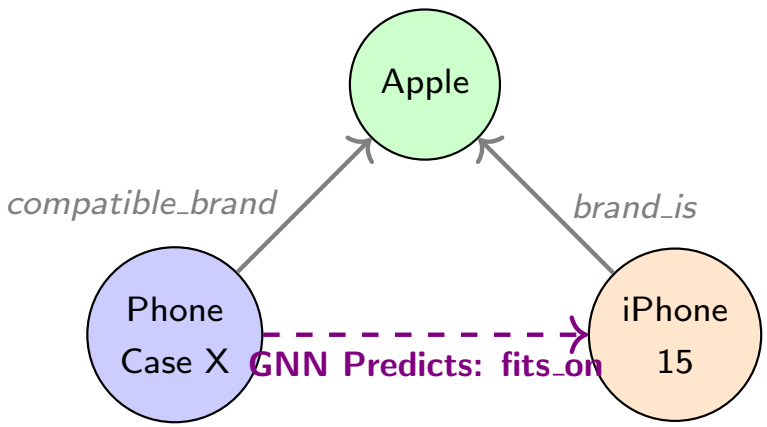
Instead of training a specific XGBoost model or custom GNN for every new business question, we pre-train a massive Graph Foundation Model over the entire multi-table schema. It the extension of foundation models from NLP to Graphs Machine Learning.



Paper: *A Foundation Model for In-Context Learning on Relational Data* (J. Leskovec et al., 2024).
`https://arxiv.org/abs/2405.04826`

Catalog & Discovery: Product Knowledge Graphs

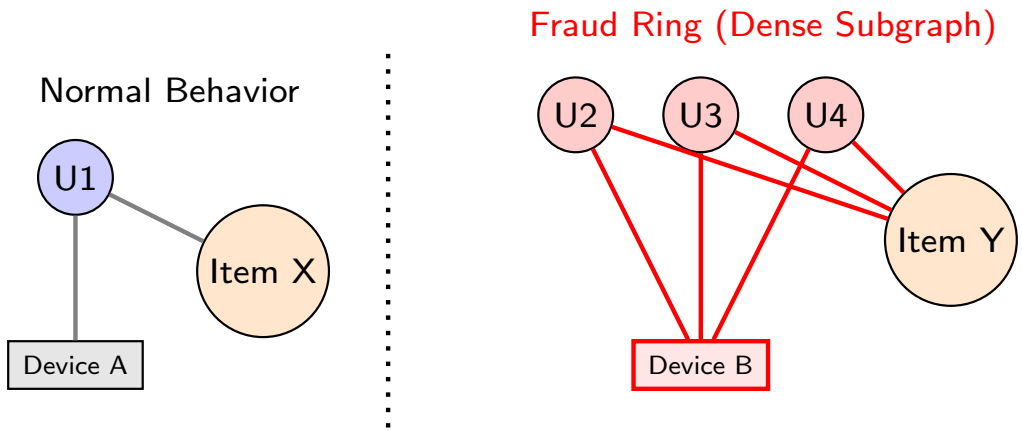
Platforms build a Product Knowledge Graph (nodes: products, brands, attributes; edges: relations like *fits_on*, *produced_by*). Relational GNNs use message passing across different edge types to perform **link prediction**, automatically inferring missing catalog connections.



Paper: *AutoKnow: Self-Driving Knowledge Collection for Products of Thousands of Types* (X. L. Dong et al., KDD 2020). `https://arxiv.org/abs/2006.13473`

Security: Fraud Detection & Fake Review Rings

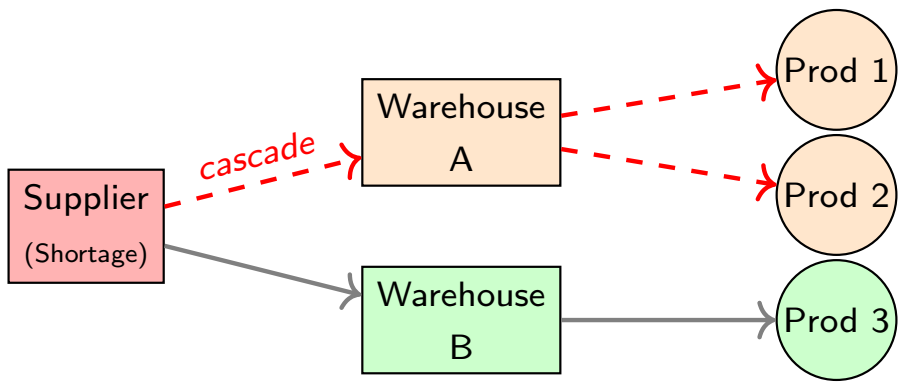
By modeling the platform as a heterogeneous graph (Nodes: Users, Devices, Items), GNNs aggregate neighborhood behavior. Even if individual users look normal, the GNN easily detects the dense, anomalous subgraphs (e.g., 50 accounts sharing 2 devices to review the same obscure item).



Paper: *FdGars: Fraudster Detection via Graph Convolutional Networks in Online App Review System* (J. Wang et al., WWW 2019). `https://dl.acm.org/doi/10.1145/3308560.3316586`

Operations: Supply Chain Vulnerability

Model the supply chain as a spatial-temporal graph (Nodes: Suppliers, Warehouses, End-Products). A GNN propagates delay or shortage features forward through the graph's edges, accurately predicting inventory cascades well in advance.



Paper: *Graph Neural Networks in Supply Chain Analytics and Optimization* (B. C. et al., 2024).
`https://arxiv.org/abs/2411.08550`

Where are we?

Introduction and Motivations

Large Graphs and their Properties

Graph Machine Learning

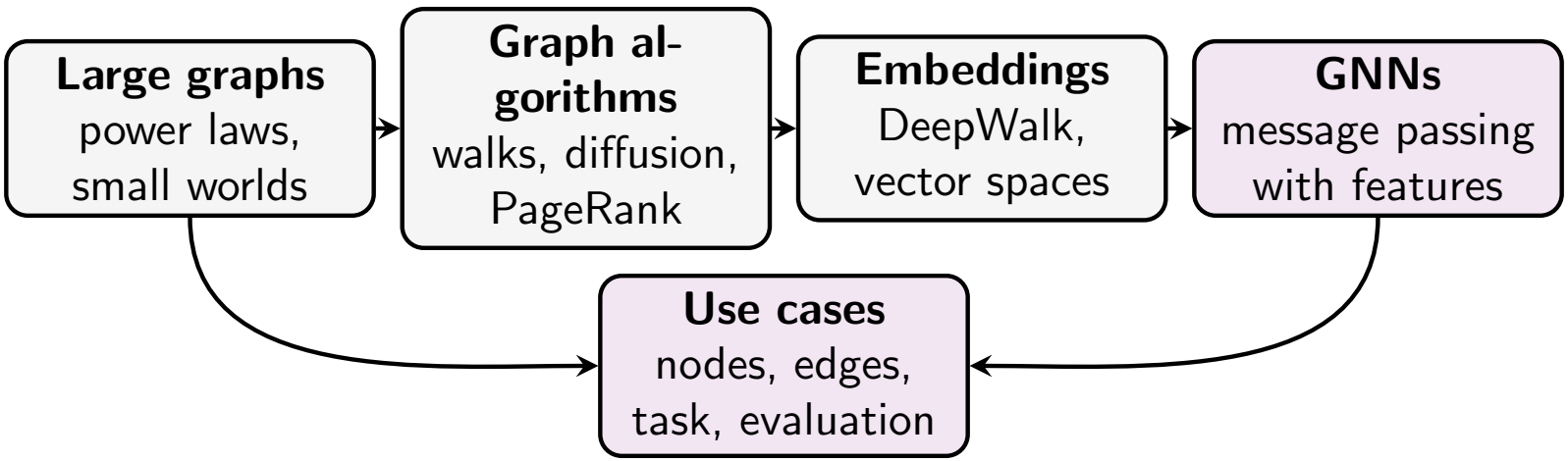
Graph Neural Networks

Some Applications and Perspectives

Conclusion

Summary

We started from graph structure and ended with prediction tasks.



Graphs give us structure; ML turns that structure into useful representations.

From graphs to GNNs

A GNN combines the graph structure with node features to make predictions.



Big picture

The graph tells the model **where information can flow**; features tell it **what information flows**.

Graphs are useful only when edges bring predictive information.

Learn more

- CS224W: Machine Learning with Graphs, *Jure Leskovec, Standford*
- Graph Representation Learning Book, *William L. Hamilton, McGill University*
- Deep Learning on Graphs, *Yao Ma and Jiliang Tang*
- Everything is Connected: Graph Neural Networks, *Petar Veličković, DeepMind/University of Cambridge*

Thank you

Questions?

