Graph Neural Networks: From Foundations to Applications

Xueqin Chen

5 October 2022

Universiteit Leiden
The Netherlands

What this tutorial covers today?

Aims at answering the following questions:

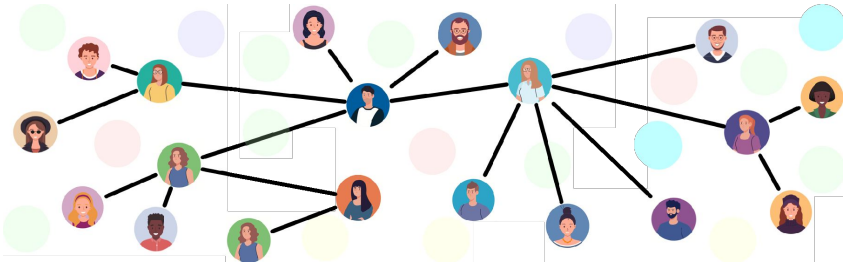
- Why GNNs?
- What GNNs?
  - -Theory
  - -Types spatial/spectral
  - -Popular architectures GCN GAT message passing
- How GNNs?
- Future directions.
- Studying roadmap.

1 Why GNNs?

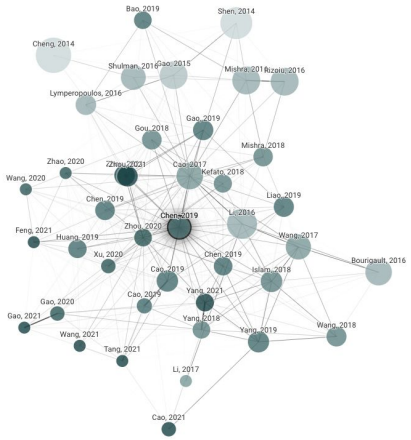
Why we learn graphs?

Graphs are a general form for describing and modeling complex systems.

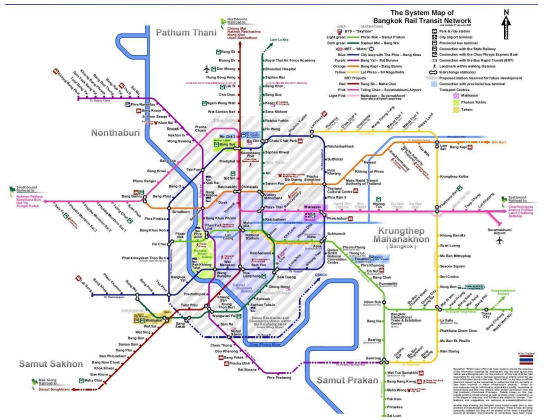
Graphs are everywhere!



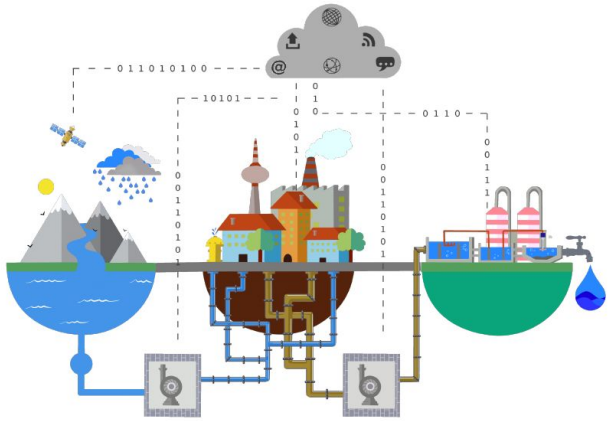
Social Graphs



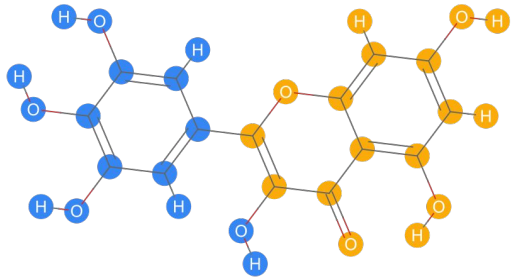
Citation Graphs



Transportation Graphs



Water Graphs



Molecule Graphs

Everything can be represented as graphs!

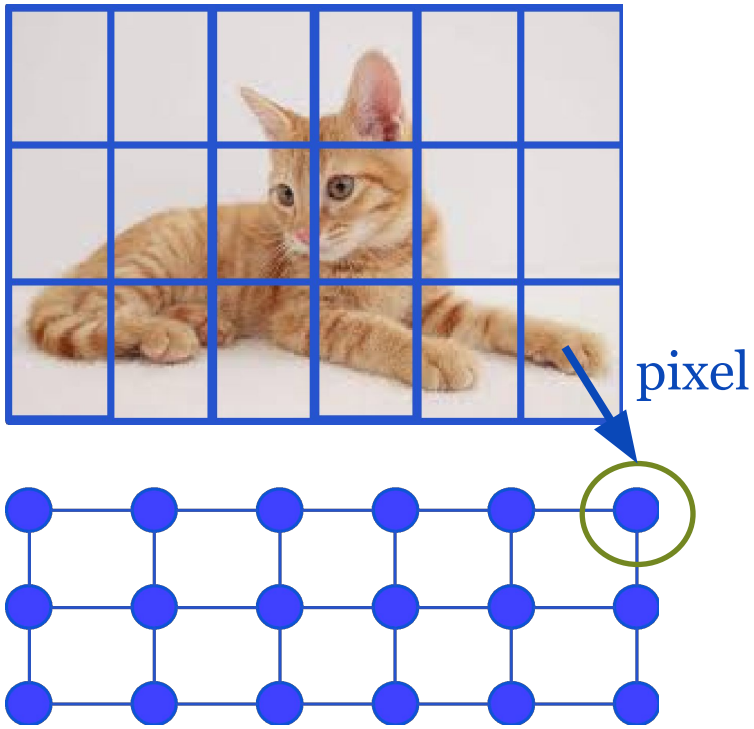
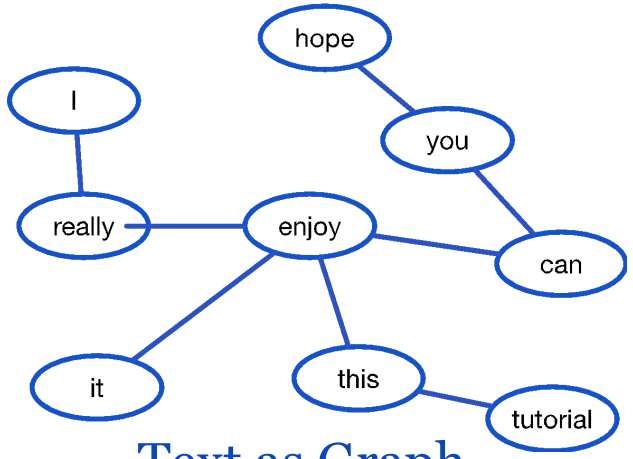


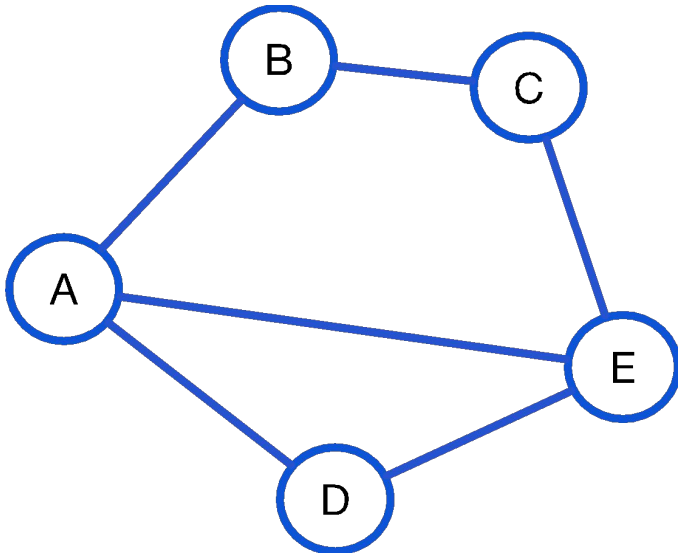
Image as Graph

A: Hope you can enjoy this tutorial!
B: I really enjoy it!



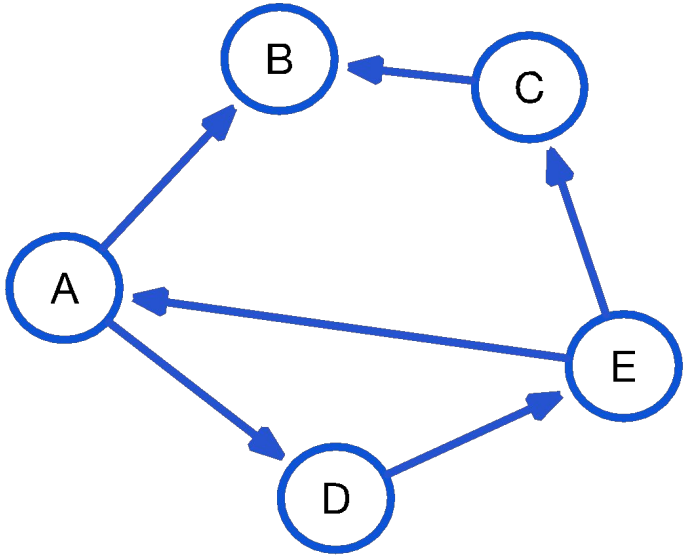
Text as Graph

Types of graphs



Undirected Graphs

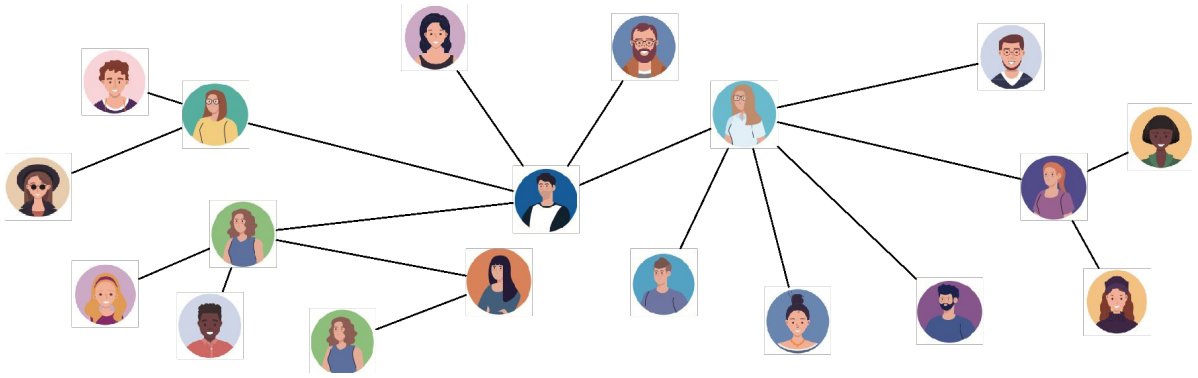
Undirected edge



Directed Graphs

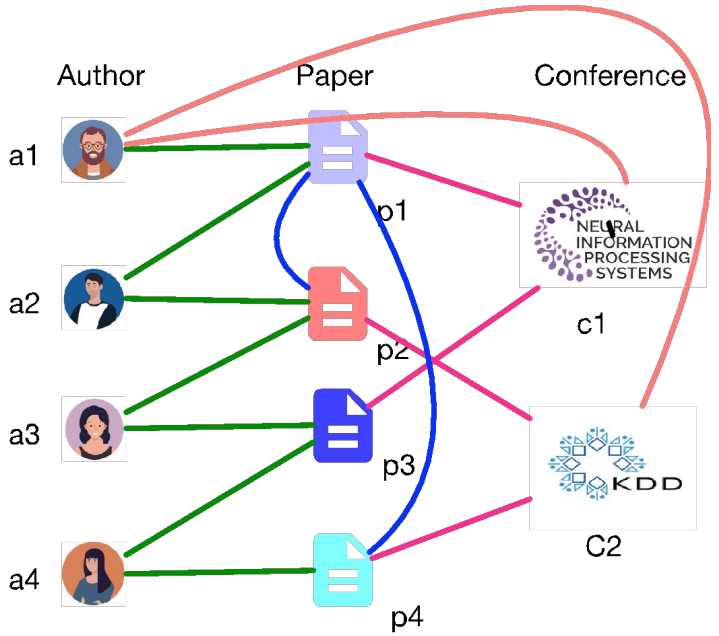
Directed edge

Types of graphs



Homogeneous Graphs

Node type:1 (User)
Edge type: 1 (following)



Heterogeneous Graphs

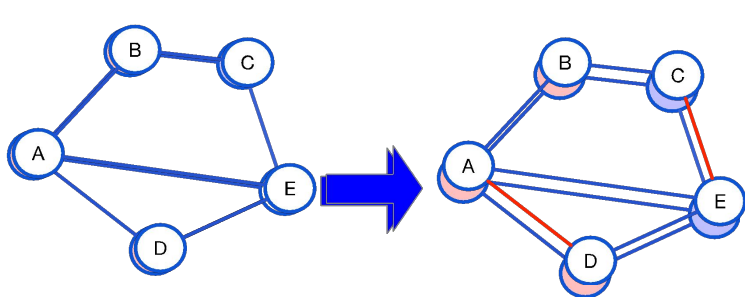
Node type: N (author, paper, conference)
Edge type: M (e.g., a->p: writing; p1->c1:publishing)

Too many!

- Dynamic graphs
  - -nodes, edges changed over time
- Knowledge graphs
  - -collections of interlinked descriptions of concepts, entities, relationships, and events
- Line graphs
  - -edges are treated as nodes
- Hypergraphs
  - -an edge can connect more than 2 nodes

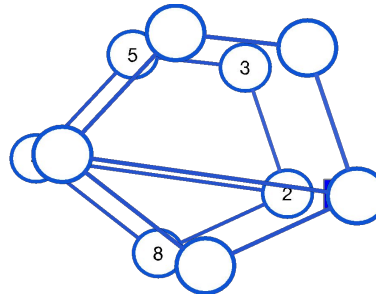
Learning graphs

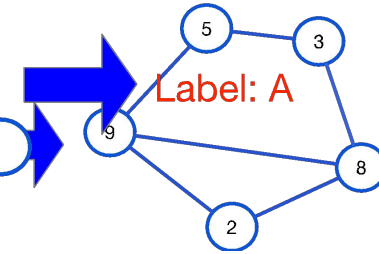
What can we do with graphs?



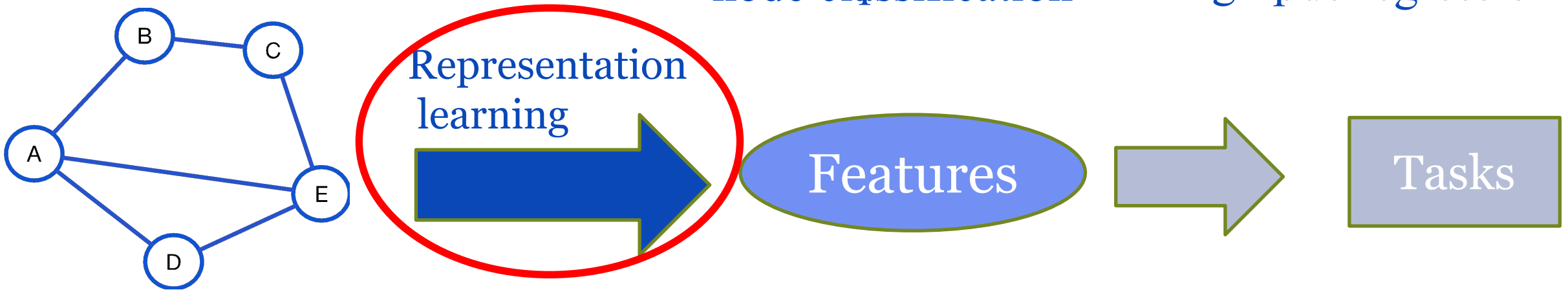
node classification

link prediction





graph classification



Aims at extracting features from the graph for graph-based tasks.

Representation learning on graphs

- Traditional
  - -Achieved via human efforts based on the prior knowledge and domain expertise on the data and tasks, which is also named as feature engineering.
  - -Predefined straightforward features in graph levels (e.g., the diameter, average path length, and clustering co-efficient), node levels (e.g., node degree and centrality), or subgraph levels (e.g., frequent subgraphs and graph motifs).
  - -These methods are time-consuming and bad generalizations.

Representation learning on graphs

- Deep learning
  - -Automatically learning from data!
  - -Traditional DL is designed for simple Euclidean
    - CNNs for fixed-size images/grids
    - RNNs for text/sequences
- Challenges (Wu et al., 2022)
  - -Graphs are non-Euclidean.
  - -Diversity of graphs.
  - -No fixed node ordering.
  - -Large scale.
  - -…

So GNNs come out!

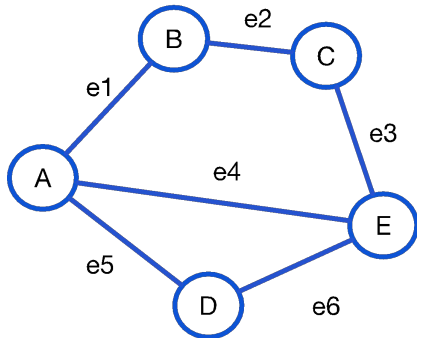
2 What GNNs?

How to represent a graph?

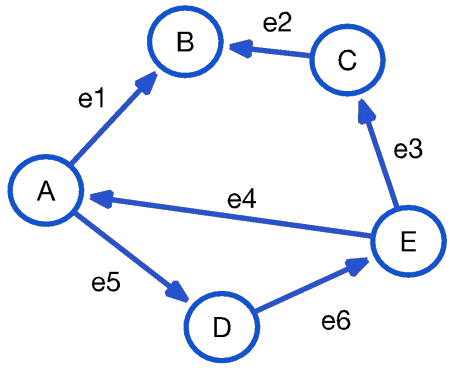
- A graph G represents the relation between a set of entities, which can be denoted as $G=\{V, E\}$, where V is node set, and $E \subseteq V \times V$ is the edge set.
- Use adjacency matrix **A** to represent graph G, where:

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1N} \\ \vdots & \ddots & \vdots \\ a_{N1} & \cdots & a_{NN} \end{bmatrix}, \quad a_{ij} = \begin{cases} 1, & if\ (i,j) \in E \\ 0, & otherwise \end{cases}$$

Note that, if there exist weight w on an edge (i, j), $a_{ij} = w$.



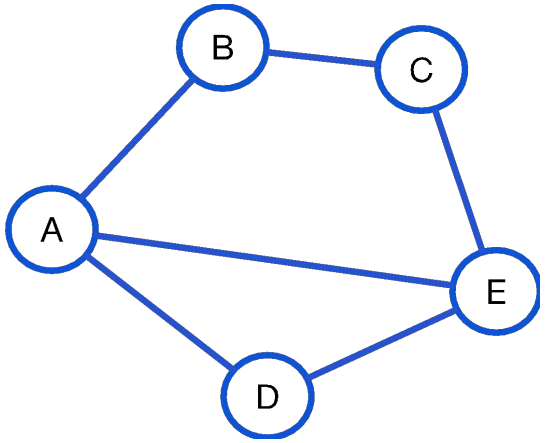
$$\begin{bmatrix} 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \end{bmatrix}$$



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

$$V = \{A, B, C, D, E\}$$
$$E = \{e1, e2, e3, e4, e5, e6\}$$

Other matrixes of a graph



Degree matrix **D**

$$d_{i,i} = \sum_{(i,j)\in E} a_{ij}$$

$$\begin{bmatrix} 3 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 3 \end{bmatrix}$$

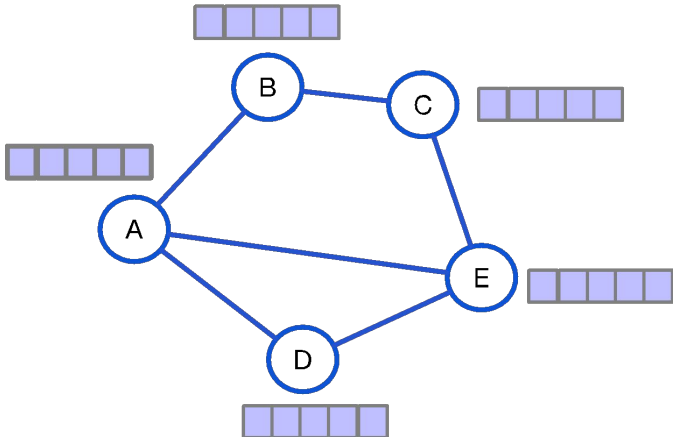
diagonal matrix

Laplacian matrix **L**

L = D - A

$$\begin{bmatrix} 3 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 3 \end{bmatrix} - \begin{bmatrix} 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 3 & -1 & 0 & -1 & -1 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & 0 & -1 \\ -1 & 0 & 0 & 2 & -1 \\ -1 & 0 & -1 & -1 & 3 \end{bmatrix}$$

Information in a graph



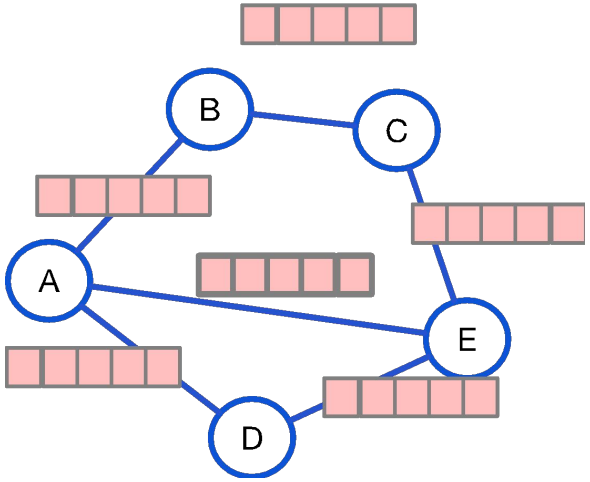
Node feature: each node in a graph can be represented as a D-dimension vector.

$$\mathbf{x}_i \in \mathbb{R}^D \text{ for node i}$$

stack them into a node feature matrix of shape N x D:

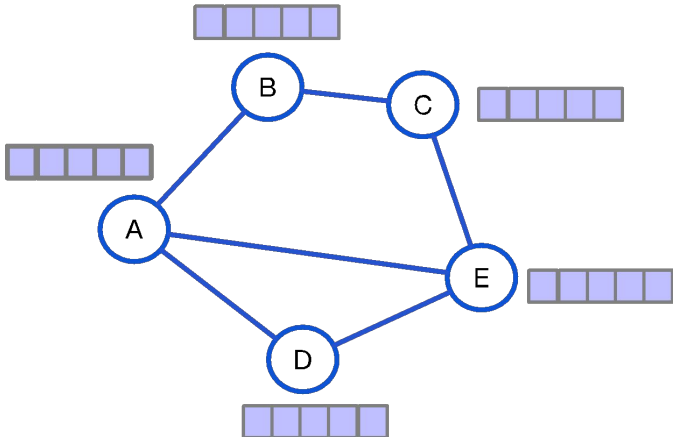
$$\mathbf{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_N\} \in \mathbb{R}^{N \times D}$$

That is, the i-th row of **X** corresponds to x_i



Edge feature: similarly, each edge in a graph can be represented as a m-dimension vector.

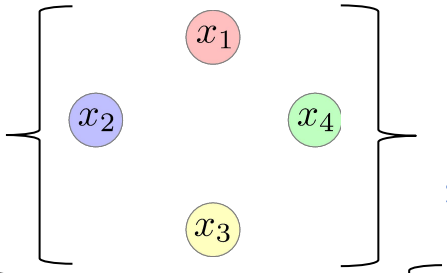
Information in a graph

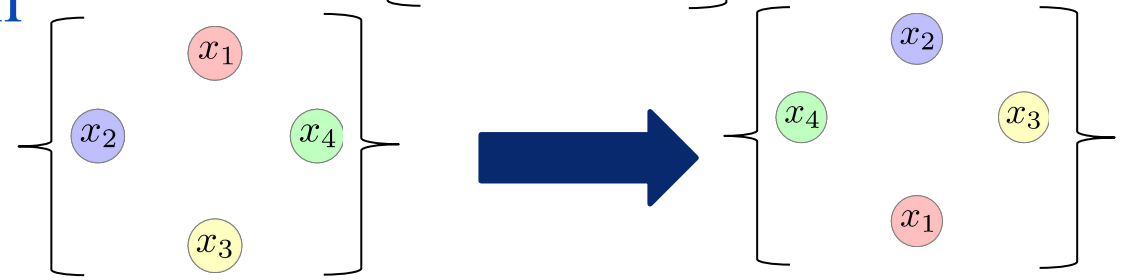


$$\mathbf{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_N\} \in \mathbb{R}^{N \times D}$$

- Note that, just by the act of stacking the nodes we have specified an order which we are visiting them!
  - We would like the result of any GNNs to not depend on this specified ordering.

Permutation on nodes

Assume we have a nodes set 

, now apply a permutation (2, 4, 1, 3) on it, then 

Permutation: change node order, each permutation defines an *n* x *n* matrix **P.**

$$\mathbf{P}_{(2,4,1,3)}\mathbf{X} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} - & x_1 & - \\ - & x_2 & - \\ - & x_3 & - \\ - & x_4 & - \end{bmatrix} = \begin{bmatrix} - & x_2 & - \\ - & x_4 & - \\ - & x_1 & - \\ - & x_3 & - \end{bmatrix}$$

This part from “Theoretical Foundations of Graph Neural Networks” by Petar Veličković, 2021

Permutation invariance

First, defined a function over nodes set as $f(\mathbf{X})$, if the output of $f(\mathbf{X})$ always hold the same whatever how we changed the order of nodes, we call this function permutation invariance, i.e., for all permutation matrices $\mathbf{P}$:

$$f(\mathbf{PX}) = f(\mathbf{X}) = \oplus_{i \in V} \mathbf{x}_i$$

The $\oplus$ is permutation invariant aggregator used to aggregate the nodes' features (such as sum, avg or max).

This part from "Theoretical Foundations of Graph Neural Networks" by Petar Veličković, 2021

Permutation equivariance

- The permutation invariance function operates on all nodes.
  - Can we still be able to identify individual nodes?

- Yes, we need a permutation equivariance.
  - A function that holds the same output whatever we do permutation before or after applying this function.

$$f(\mathbf{PX}) = \mathbf{P}f(\mathbf{X})$$

This part from “Theoretical Foundations of Graph Neural Networks” by Petar Veličković, 2021

Now adding edges!

- Permutation not only applied on nodes but also on edges.

- That is, we have both $\mathbf{PX}$ and $\mathbf{PAP}^{\mathrm{T}}$. ($\mathbf{A}$ is adjacency matrix).

- Now the invariance and equivariance change to:

$$\text{Invariance: } f\left(\mathbf{PX}, \mathbf{PAP}^{\mathrm{T}}\right) = f(\mathbf{X}, \mathbf{A})$$

$$\text{Equivariance: } f\left(\mathbf{PX}, \mathbf{PAP}^{\mathrm{T}}\right) = \mathbf{P}f(\mathbf{X}, \mathbf{A})$$

This part from "Theoretical Foundations of Graph Neural Networks" by Petar Veličković, 2021

Neighbors

- After introducing edges in the node-set, now, each node only interact with its own neighbors.
- For a node i, its direct (1-hop) neighborhood is defined as:

$$\mathcal{N}(i) = \{j | (i,j) \text{ or } (j,i) \in E\}$$

- And the stack features for i's neighbors, denoted as:

$$\mathbf{X}_{\mathcal{N}(i)} = \{\mathbf{x}_j | j \in \mathcal{N}(i)\}$$

This part from "Theoretical Foundations of Graph Neural Networks" by Petar Veličković, 2021

Construct permutation equivariant $f(\mathbf{X},\mathbf{A})$

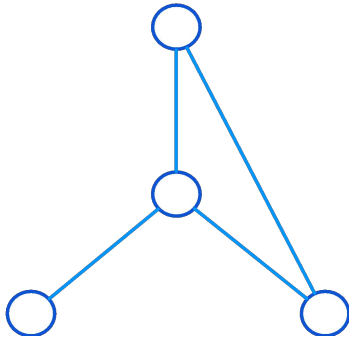
$$f(\mathbf{X},\mathbf{A}) = \begin{bmatrix} — & g(\mathbf{x}_1, \mathbf{X}_{\mathcal{N}_1}) & — \\ — & g(\mathbf{x}_2, \mathbf{X}_{\mathcal{N}_2}) & — \\ & \vdots & \\ — & g(\mathbf{x}_n, \mathbf{X}_{\mathcal{N}_n}) & — \end{bmatrix}$$

Applying permutation invariant function g on each node with its neighbors, to ensure the value of each $g(\mathbf{x}_1, \mathbf{X}_{\mathcal{N}_1})$ is not depending on order of nodes in $\mathcal{N}_i$.

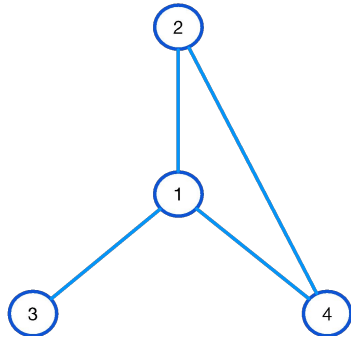
This part from "Theoretical Foundations of Graph Neural Networks" by Petar Veličković, 2021

Highlight

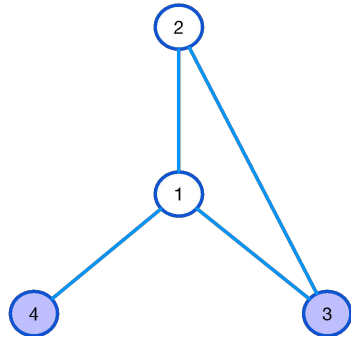
- Invariance
  - -The graph is naturally disordered;
  - -The change of node order will not affect graph features



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



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

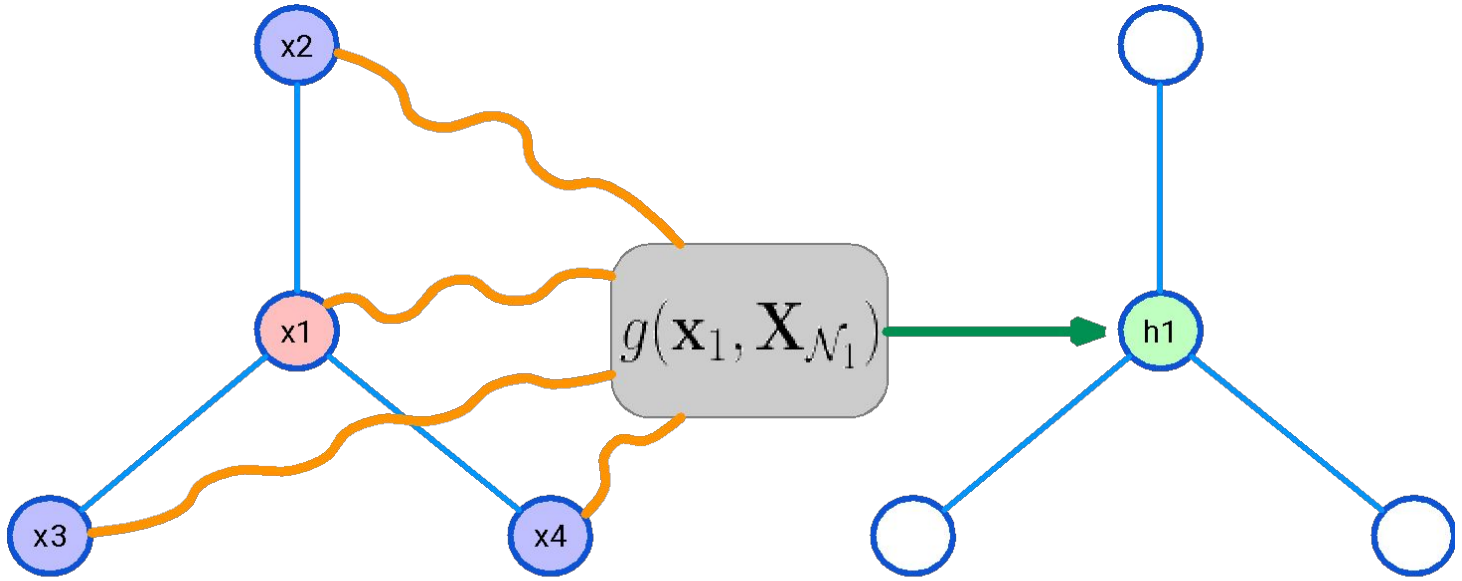


- Equivariance:
  - -If we change the node order, the node feature remains unchanged, or the order of the node feature needs to be rearranged according to the node order

Haan et al. denoted the permutation equivariance as Global Equivariance, as well as the permutation invariance as Local Invariance.(Haan et al., NeurIPS'2020)

GNNs aim at...

The goal of GNNs is to learn the node representations by aggregating the representations of node neighbors and their own representations.



$$\mathcal{N}_1 = \{\mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4\}$$

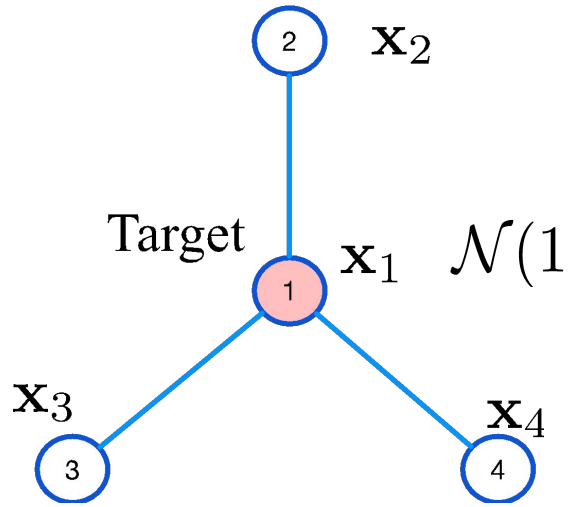
Two types of GNNs

- Spatial
  - -Using different neighbor node sampling methods and different weighted summation methods to update node features

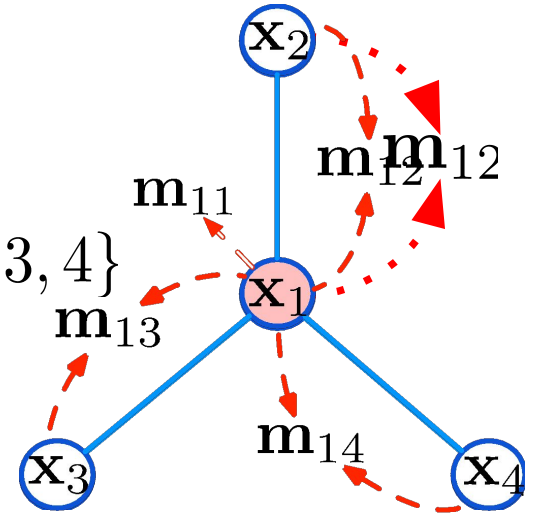
- Spectral
  - -Graph Fourier Transformation
  - -Laplacian matrix
  - -Graph spectral theory
  - -…

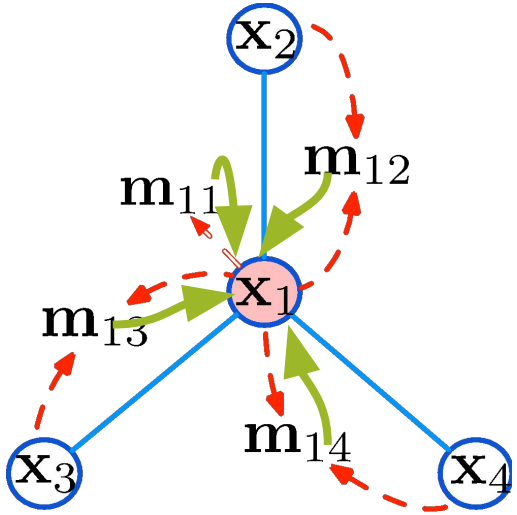
No matter in which domain, they are all based on message passing!

Message passing paradigm



$\mathcal{N}(1) = \{1, 2, 3, 4\}$





Graph

$$G = \{V, E\}$$
$$V = \{1, 2, 3, 4\}$$
$$E = \{(1, 2), (1, 3), (1, 4)\}$$
$$\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4\}$$

Message

$$\mathbf{m}_{11} = \psi(\mathbf{x}_1, \mathbf{x}_1)$$
$$\mathbf{m}_{12} = \psi(\mathbf{x}_1, \mathbf{x}_2)$$
$$\mathbf{m}_{12} = \psi(\mathbf{x}_1, \mathbf{x}_2)$$
$$\mathbf{m}_{13} = \psi(\mathbf{x}_1, \mathbf{x}_3)$$
$$\mathbf{m}_{14} = \psi(\mathbf{x}_1, \mathbf{x}_4)$$

Propagation

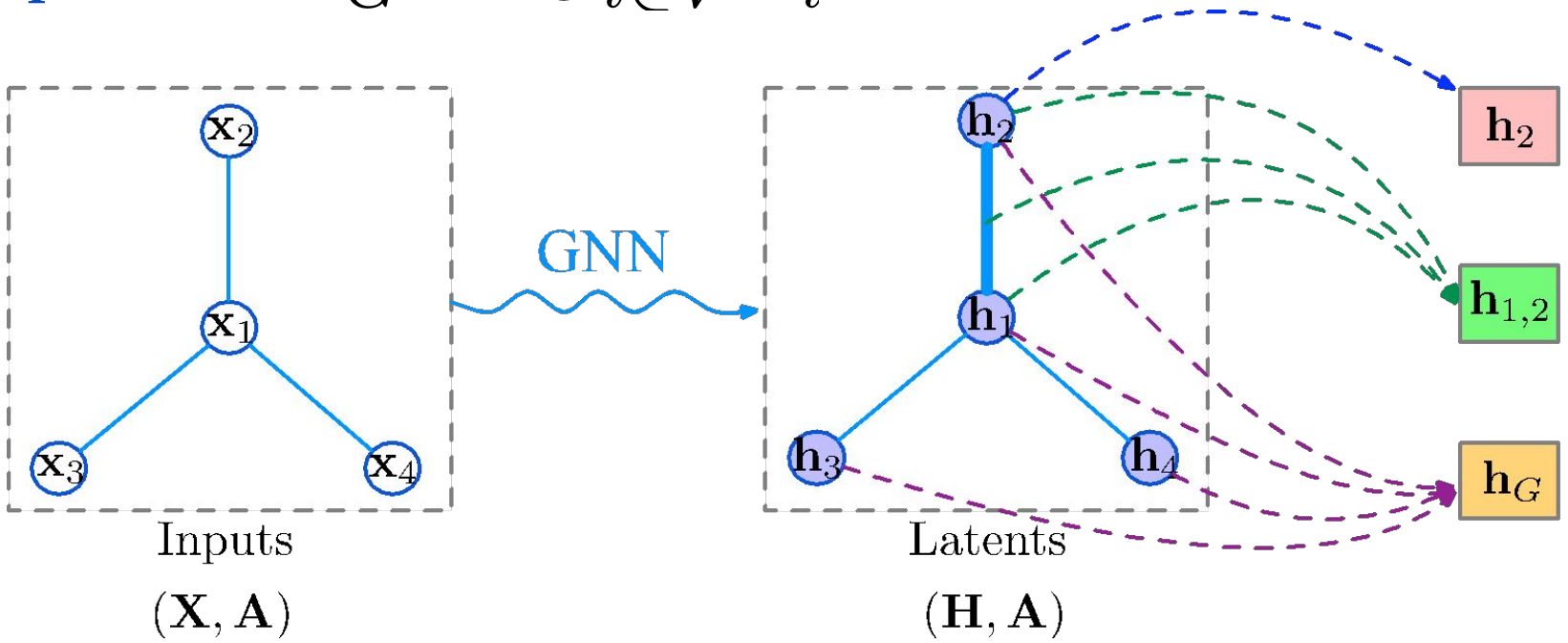
$$\mathbf{h}_1 = \phi\left(\mathbf{x}_1, \oplus_{j \in \mathcal{N}(1)} m_{1j}\right)$$

Message passing paradigm

Node-level: $\mathbf{h}_i = \phi(\mathbf{x}_i, \oplus_{j \in \mathcal{N}(i)} \psi(\mathbf{x}_i, \mathbf{x}_j))$

Edge-level: $\mathbf{h}_{i,j} = \gamma(\mathbf{h}_i, \mathbf{h}_j, e_{ij}), \ (i,j) \in E$

Graph-level: $\mathbf{h}_G = \oplus_{i \in V} \mathbf{h}_i$



Node tasks

$y_2 = f(\mathbf{h}_2)$

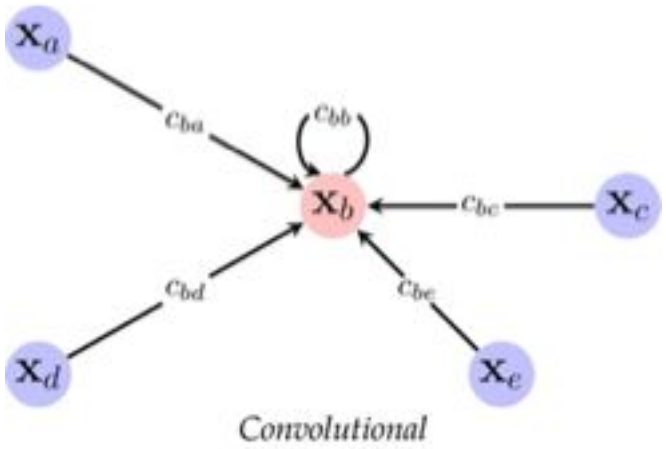
Edge tasks

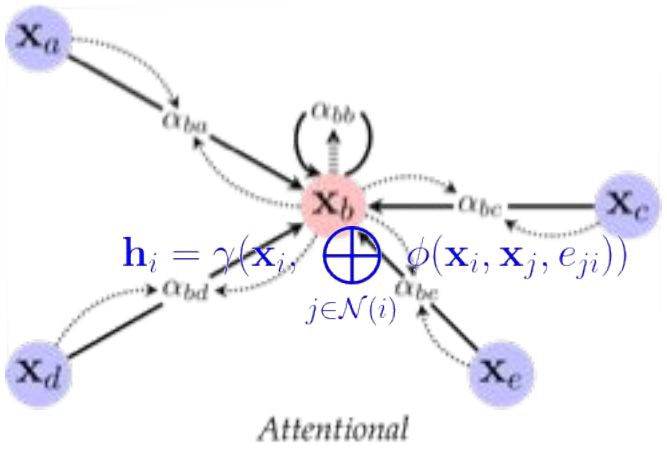
$y_{1,2} = f(\mathbf{h}_{1,2})$

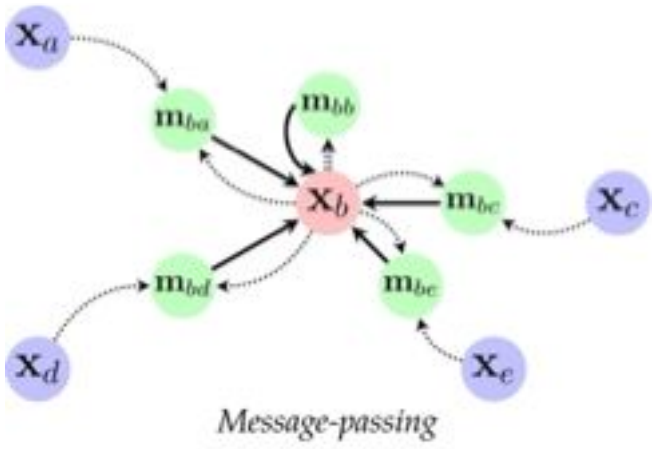
Graph tasks

$y_G = f(\mathbf{h}_G)$

Three basic flavor of Message-Passing GNNs







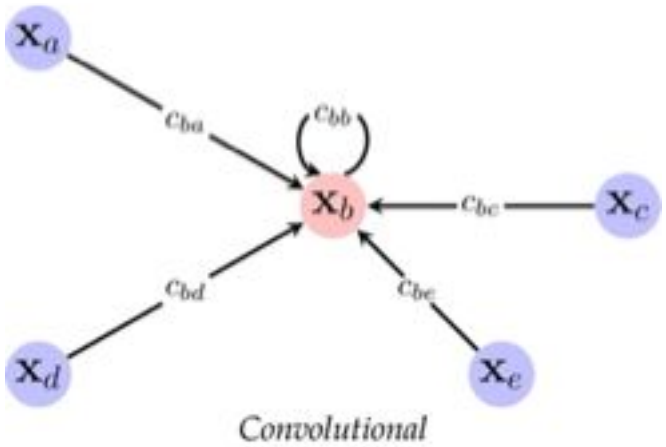
$$\mathbf{h}_i = \phi(\mathbf{x}_i, \oplus_{j \in \mathcal{N}(i)} w_{i,j} \mathbf{x}_j)$$

$$\mathbf{h}_i = \phi(\mathbf{x}_i, \oplus_{j \in \mathcal{N}(i)} a_{i,j} \mathbf{x}_j)$$

$$\mathbf{h}_i = \phi(\mathbf{x}_i, \oplus_{j \in \mathcal{N}(i)} \psi(\mathbf{x}_i, \mathbf{x}_j))$$

This part from "Theoretical Foundations of Graph Neural Networks" by Petar Veličković, 2021

Convolutional



Convolutional

$$\mathbf{h}_i = \phi\left(\mathbf{x}_i, \oplus_{j \in \mathcal{N}(i)} w_{i,j}\mathbf{x}_j\right)$$

- Features of neighbors aggregated with fixed weights, w_{ij}

$$w_{ij} = \frac{\hat{\mathbf{A}}_{ij}}{\sqrt{\hat{\mathbf{D}}_{ii}\hat{\mathbf{D}}_{jj}}}, \quad \hat{\mathbf{A}} = \mathbf{A} + \mathbf{I}$$

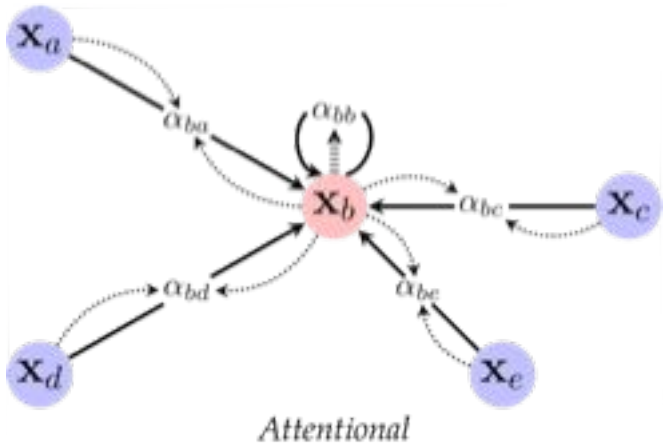
- Usually, the weights depend directly on A.

- Typical example: ChebyNet (Defferrard et al., NeurIPS'16), GCN (Kipf & Welling, ICLR'17)

- When edges encode *label similarity*
- Useful for **homophilous** graphs and **scaling up**

This part from "Theoretical Foundations of Graph Neural Networks" by Petar Veličković, 2021

Attentional



Attentional

- Features of neighbors aggregated with **implicit** weights (via *attention*)

$$e_{ij} = a(\mathbf{x}_i, \mathbf{x}_j)$$

$$a_{ij} = \mathrm{Softmax}_j(e_{ij}) = \frac{\exp(\{e_{ij}\})}{\sum_{l \in |mathcalN(i)} \exp(e_{il})}$$

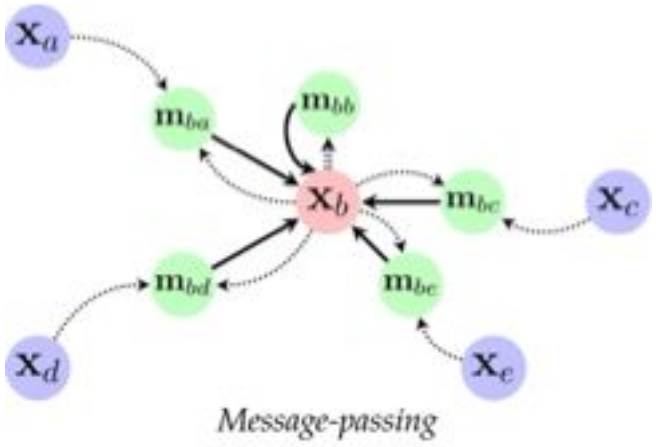
- Typical example: GAT (Veličković et al., ICLR'18)

$$\mathbf{h}_i = \phi(\mathbf{x}_i, \oplus_{j \in \mathcal{N}(i)} a_{i,j}\mathbf{x}_j)$$

- Edges need not encode homophily
- But still compute scalar value in each edge

This part from "Theoretical Foundations of Graph Neural Networks" by Petar Veličković, 2021

Pure message passing



- Compute **arbitrary vectors** (*"messages"*) to be sent across edges

$$\mathbf{h}_i = \phi\left(\mathbf{x}_i, \bigoplus_{j \in \mathcal{N}(i)} \psi\left(\mathbf{x}_i, \mathbf{x}_j\right)\right)$$

- Messages computed as $\mathbf{m}_{ij} = \psi(\mathbf{x}_i, \mathbf{x}_j)$

  MPNN (Gilmer et al., ICML'17)
  GraphSAGE (WL Hamilton et al., NeurIPS'17)

- Most **generic** GNN layer
  - May have *scalability* or *learnability* issues
  - Ideal for *computational chemistry*, *reasoning* and *simulation*

This part from "Theoretical Foundations of Graph Neural Networks" by Petar Veličković, 2021

Higher-order interactions

Stack multiple GNN layers to extract higher-order interactions for nodes.

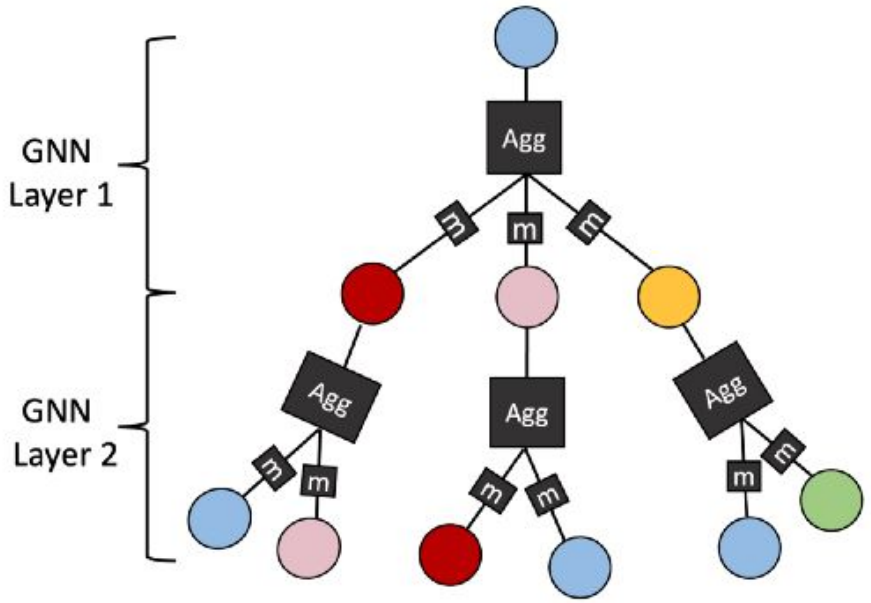


Figure from website

3 Applications

Applications

3.1 Social networks

3.2 Recommendation

3.3 Geoscience

3.4 Bioinformatics

3.5 PDEs

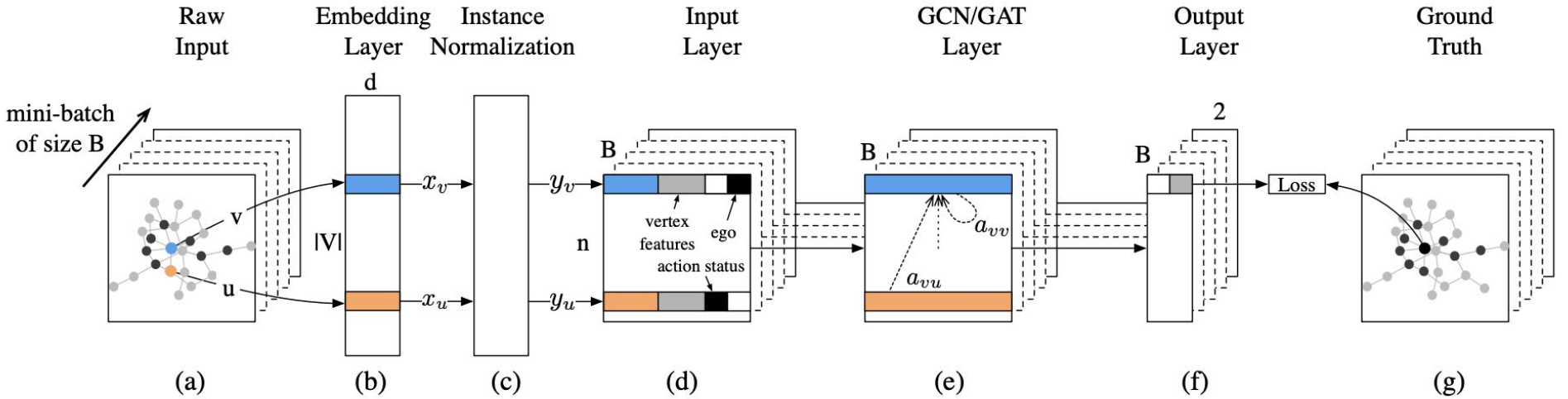
3.1 GNNs for Social Networks

GNNs for social networks

- One central problem in social network analysis is how to extract useful features from non-Euclidean structured networks, to enable the deployment of downstream machine learning prediction models for specific analysis.

- Typical studies:
  - Social influence analysis; (Qiu et al., KDD'2018)
  - Diffusion prediction; (Chen et al., ICDE'2019)
  - Fake news detection; (Bian et al., AAAI'2020)
  - Community detection; (Chen et al., ICLR'2019)
  - Viral marketing;
  - Else…

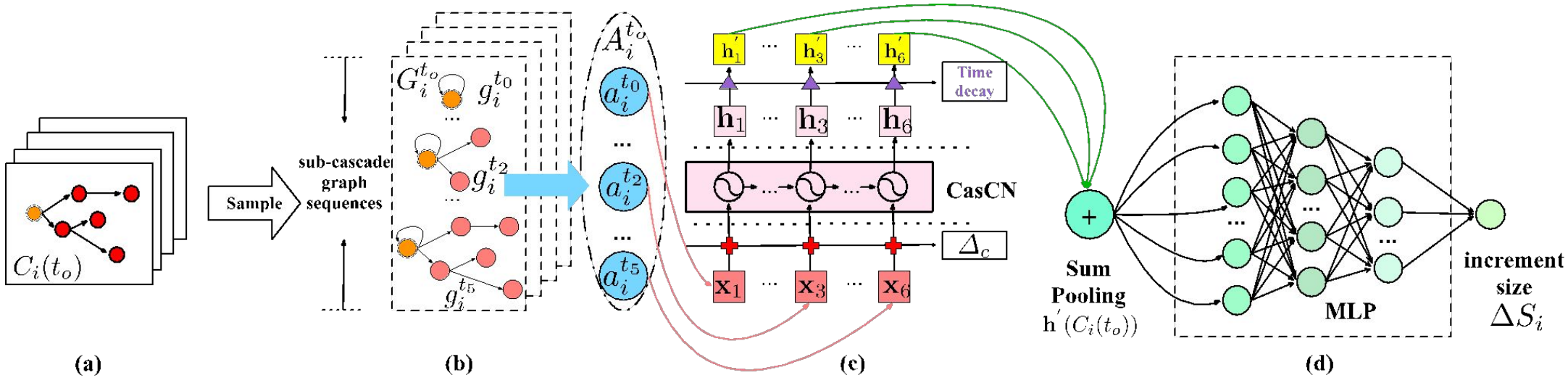
GNNs for social influence analysis



Qiu J, Tang J, Ma H, et al. Deepinf: Social influence prediction with deep learning[C]//KDD. 2018: 2110-2119.

DeepInf learns a node-level representation for users. Combined with handcrafted features, and makes predictions on social influences.

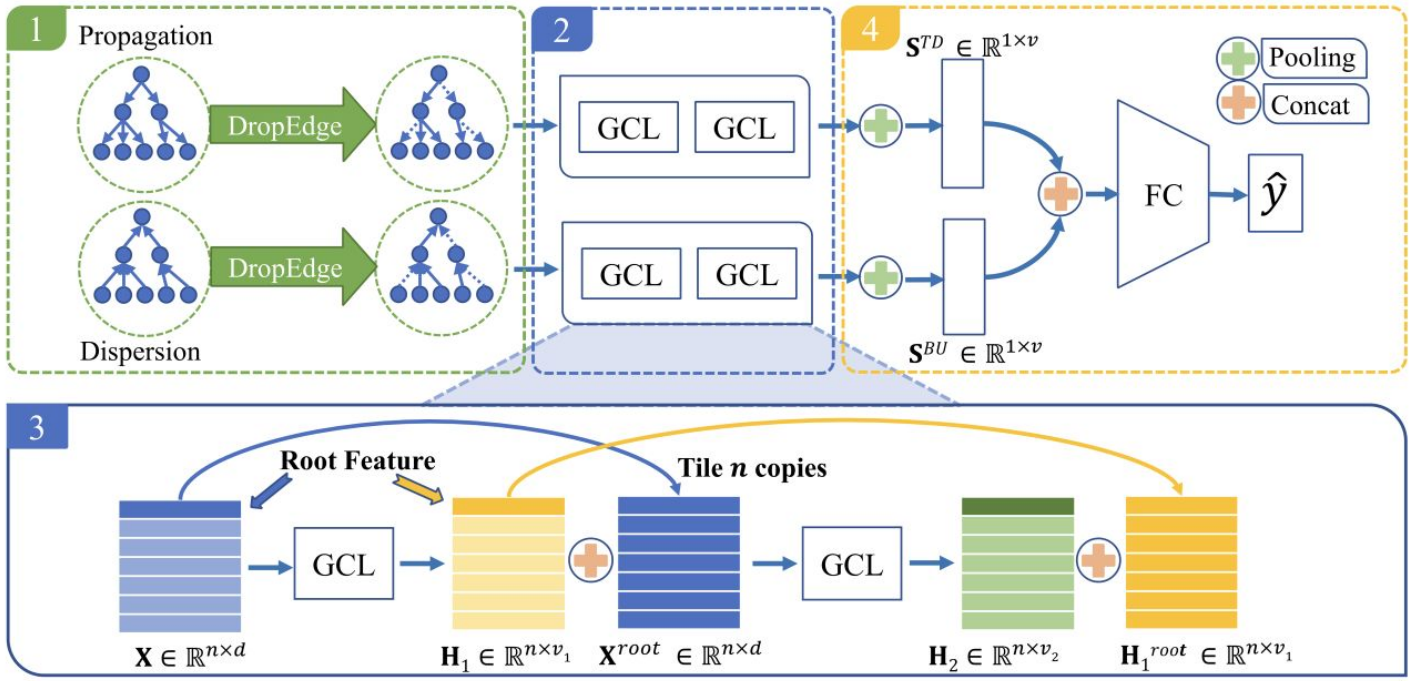
GNNs for diffusion prediction



Chen X, Zhou F, Zhang K, et al. Information diffusion prediction via recurrent cascades convolution[C]//ICDE. 2019: 770-781.

CasCN extracts spatial-temporal features from the observed information cascades, and uses the learned representation for predicting the incremental size of the cascades.

GNNs for rumor detection



Bian T, Xiao X, Xu T, et al. Rumor detection on social media with bi-directional graph convolutional networks[C]//AAAI. 2020, 34(01): 549-556.

Bi-GCN obtains the features of propagation and dispersion via the Top-Down graph convolutional Networks (TD-GCN) and Bottom-Up graph convolutional Networks (BU-GCN), respectively. And using the learning graph representation for rumor detection.

3.2 GNNs for Recommendation

Recommender systems

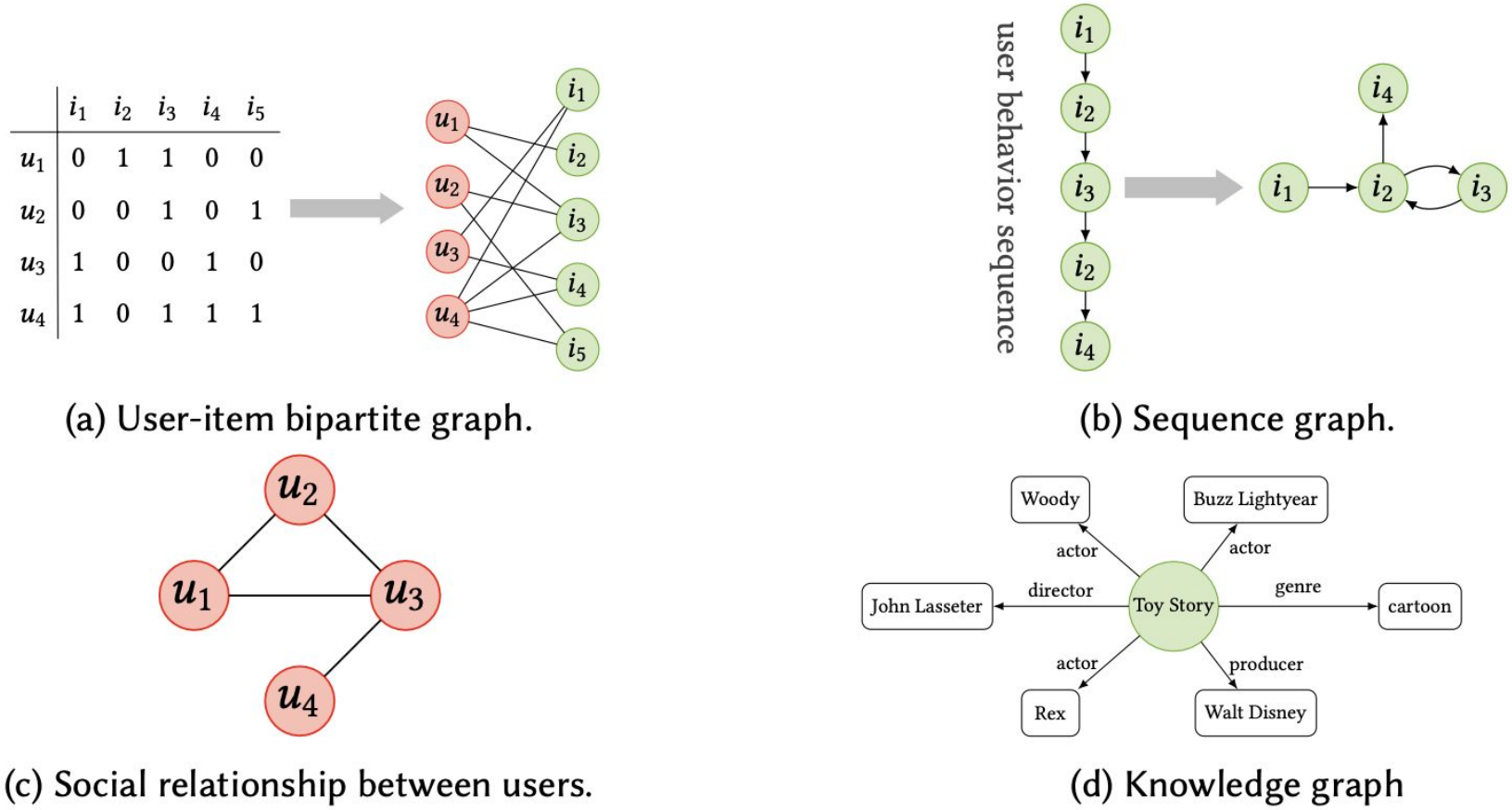
- Recommender systems infer users' preferences from user-item interactions or static features, and further recommend items that users might be interested in.
- the task is to estimate her/his preference for any item $i \in \mathrm{I}$ by the learnt user representation $\boldsymbol{h}_u^*$ and item representation $\boldsymbol{h}_i^*$, i.e.,

$$y_{u,i} = f(\mathbf{h}_u^*, \mathbf{h}_i^*)$$

- where score function $f(\cdot)$ can be dot product, cosine, multi-layer perceptions, etc., and $y_{u,i}$ denotes the preference score for user u on item i, which is usually presented in probability.

Wu S, Sun F, Zhang W, et al. Graph neural networks in recommender systems: a survey[J]. ACM Computing Surveys (CSUR), 2020.

Graphs in recommendation



(a) User-item bipartite graph.

(b) Sequence graph.

(c) Social relationship between users.

(d) Knowledge graph

Fig. 1. Representative graph structures in recommender systems.

Wu S, Sun F, Zhang W, et al. Graph neural networks in recommender systems: a survey[J]. ACM Computing Surveys (CSUR), 2020.

Recommendation tasks

- User-Item collaborative filtering
- Sequential recommendation
- Social recommendation
- Knowledge graph-based recommendation

- Other tasks:
  - POI recommendation
  - Group recommendation
  - Bundle recommendation
  - Click-through rate prediction
  - Multimedia Recommendation

Wu S, Sun F, Zhang W, et al. Graph neural networks in recommender systems: a survey[J]. ACM Computing Surveys (CSUR), 2020.

User-item collaborative filtering

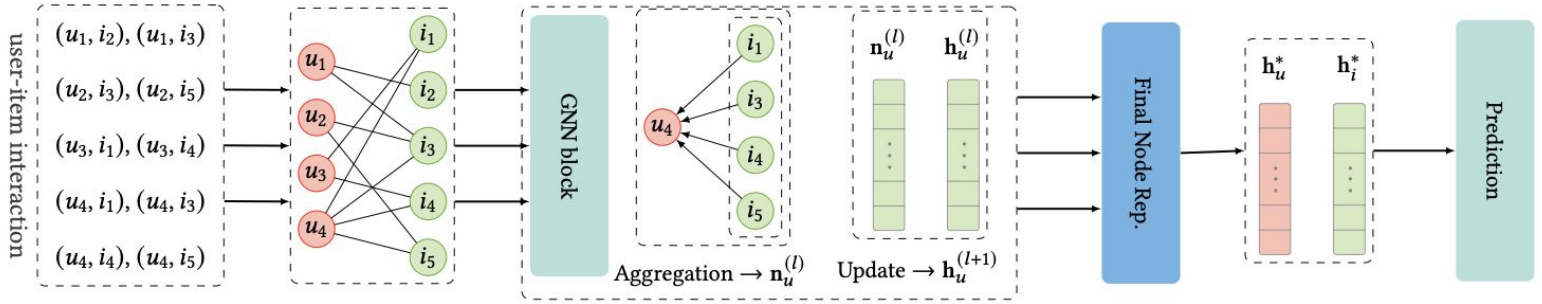


Fig. 2. The overall framework of GNN in user-item collaborative filtering.

1. Graph construction - user-item bipartite graph
2. Neighbor aggregation
3. Information update
4. Final node representation

GNNs

Using the items interacted by users to enhance user representations and using the users once interacted with items to enrich item representations

Wu S, Sun F, Zhang W, et al. Graph neural networks in recommender systems: a survey[J]. ACM Computing Surveys (CSUR), 2020.

Sequential recommendation

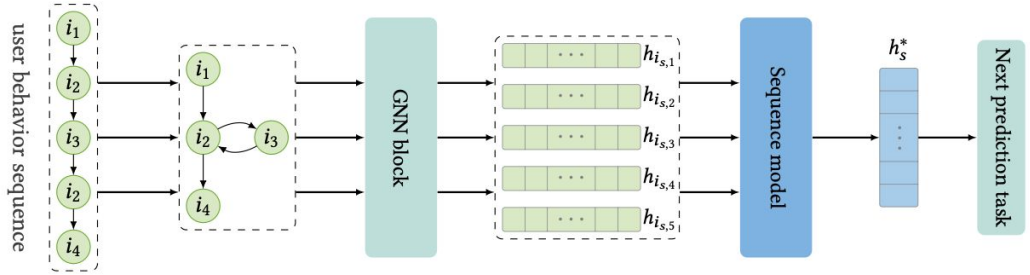


Fig. 4. The overall framework of GNN in sequential recommendation.

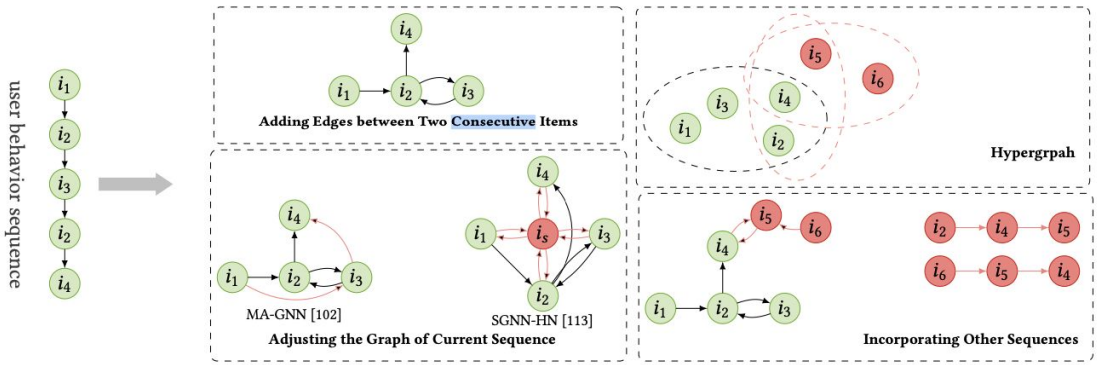
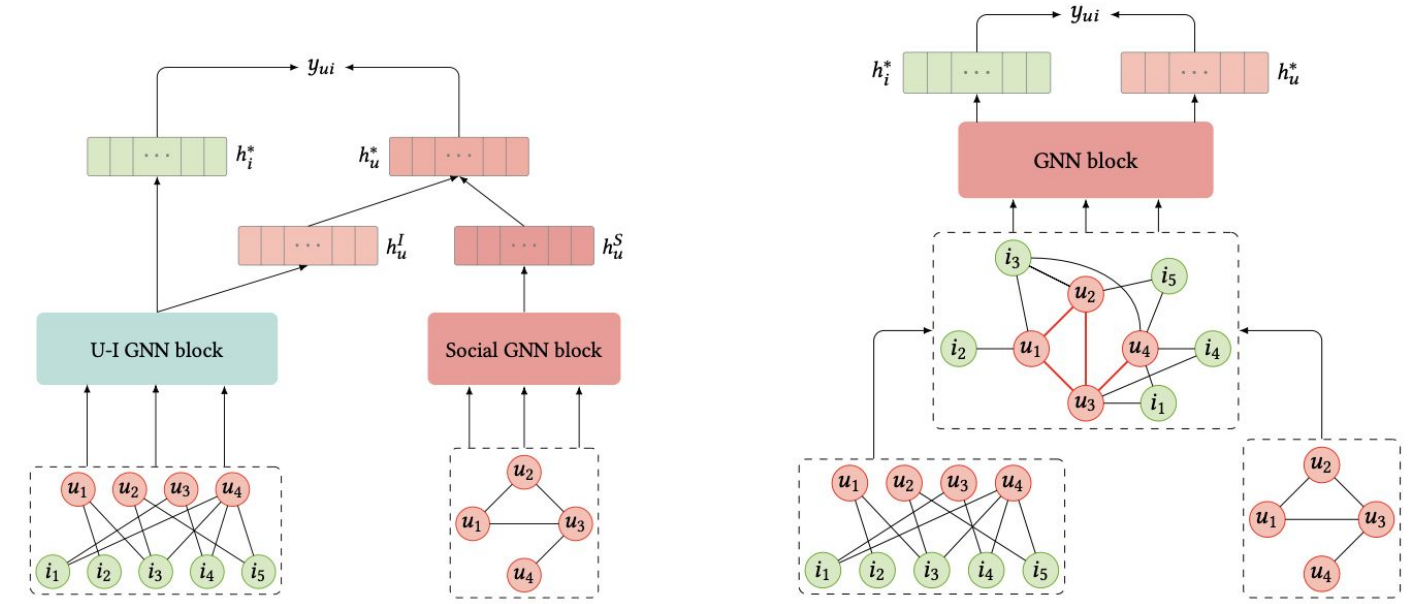


Fig. 5. Representative graph construction methods in the sequential recommendation, where the original nodes and edges are in green, and the additional ones are in red.

1. Graph construction
   - add edges between the two consecutive items
   - additional sequences enrich the graph
2. Information propagation
   - GNN
3. Sequential perference
   - Attention
   - RNN

Wu S, Sun F, Zhang W, et al. Graph neural networks in recommender systems: a survey[J]. ACM Computing Surveys (CSUR), 2020.

Social recommendation



(a) The framework of GNN on the bipartite graph and social network graph separately.

(b) The framework of GNN on the unified graph of user-item interactions and social network.

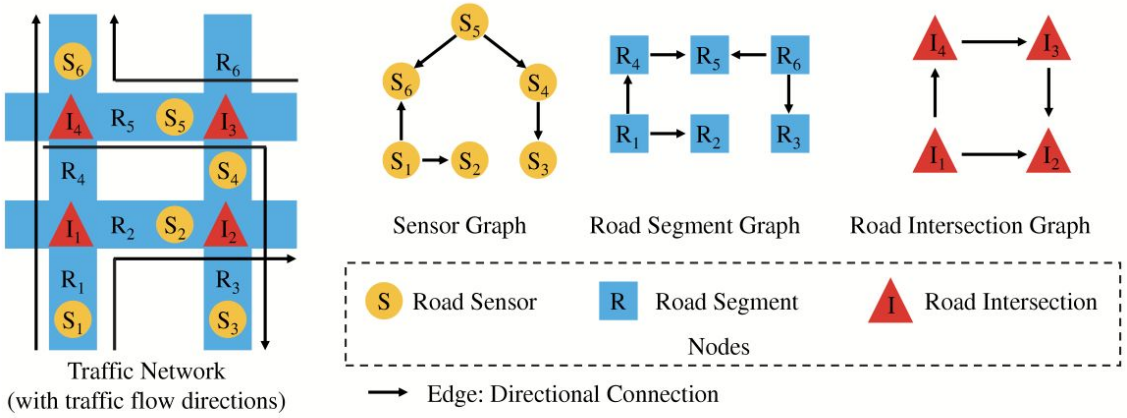
Fig. 7. Two strategies for social enhanced general recommendation.

Social recommender systems have been proposed to utilize each user's local neighbors' preferences to enhance user modeling.

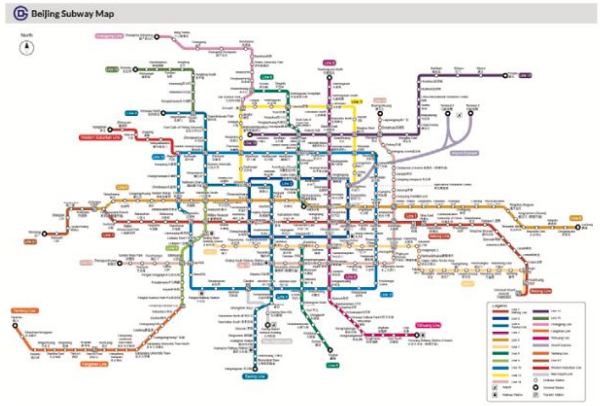
Wu S, Sun F, Zhang W, et al. Graph neural networks in recommender systems: a survey[J]. ACM Computing Surveys (CSUR), 2020.

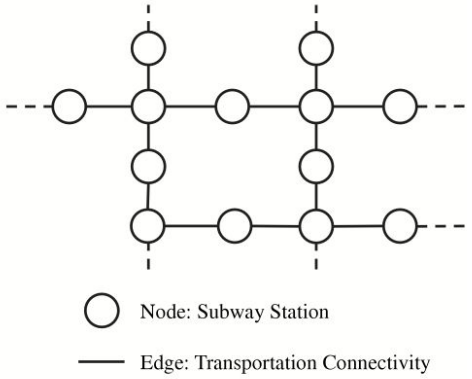
3.3 GNNs for Geoscience

Graphs in transportation field



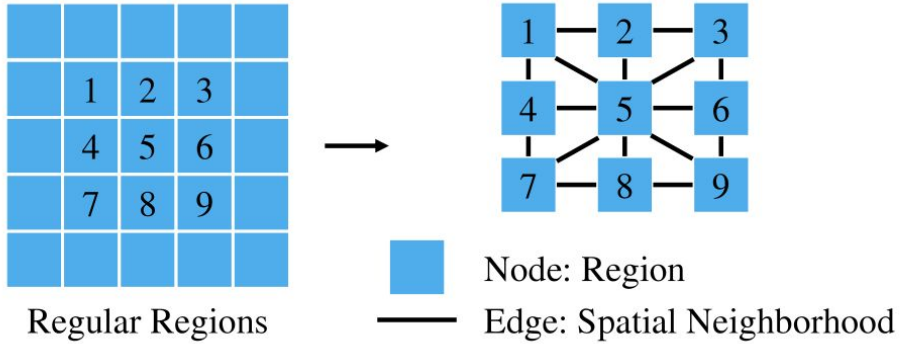
1. Road-level graph





(a) (b)

3. Station-level graph

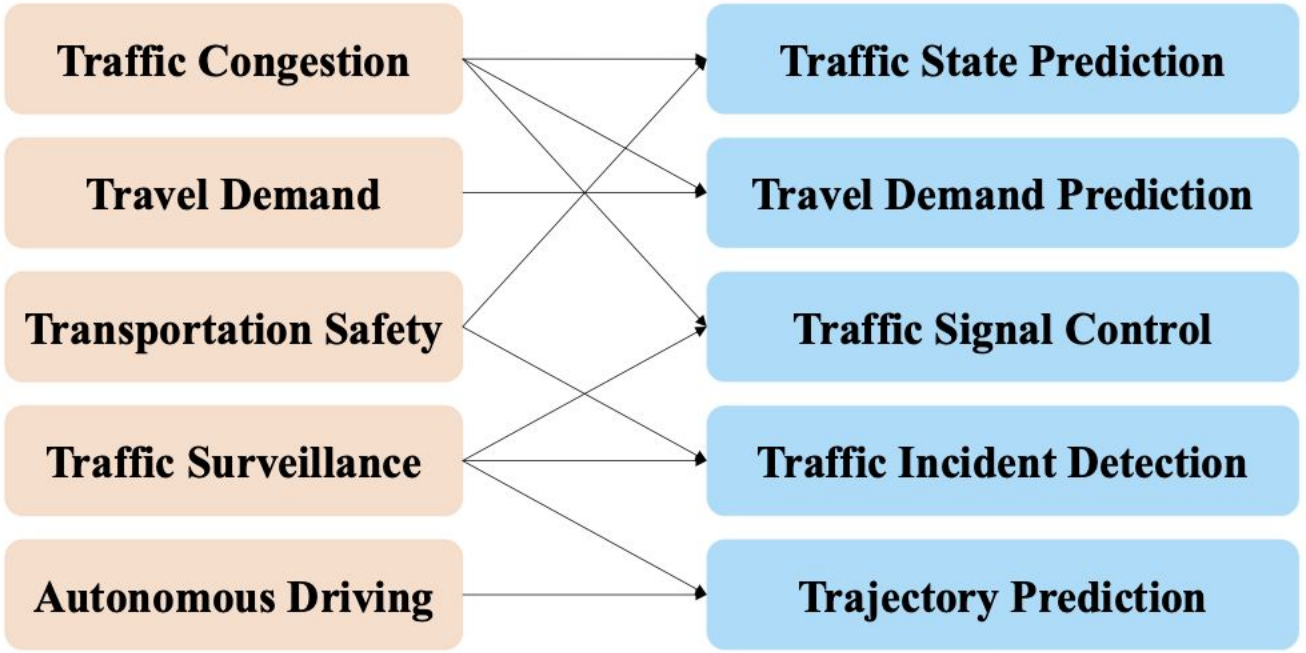


(a) (b)

2. Region-level graph

Jiang W, Luo J. Graph neural network for traffic forecasting: A survey[J]. Expert Systems with Applications, 2022: 117921

GNNs for Transportation

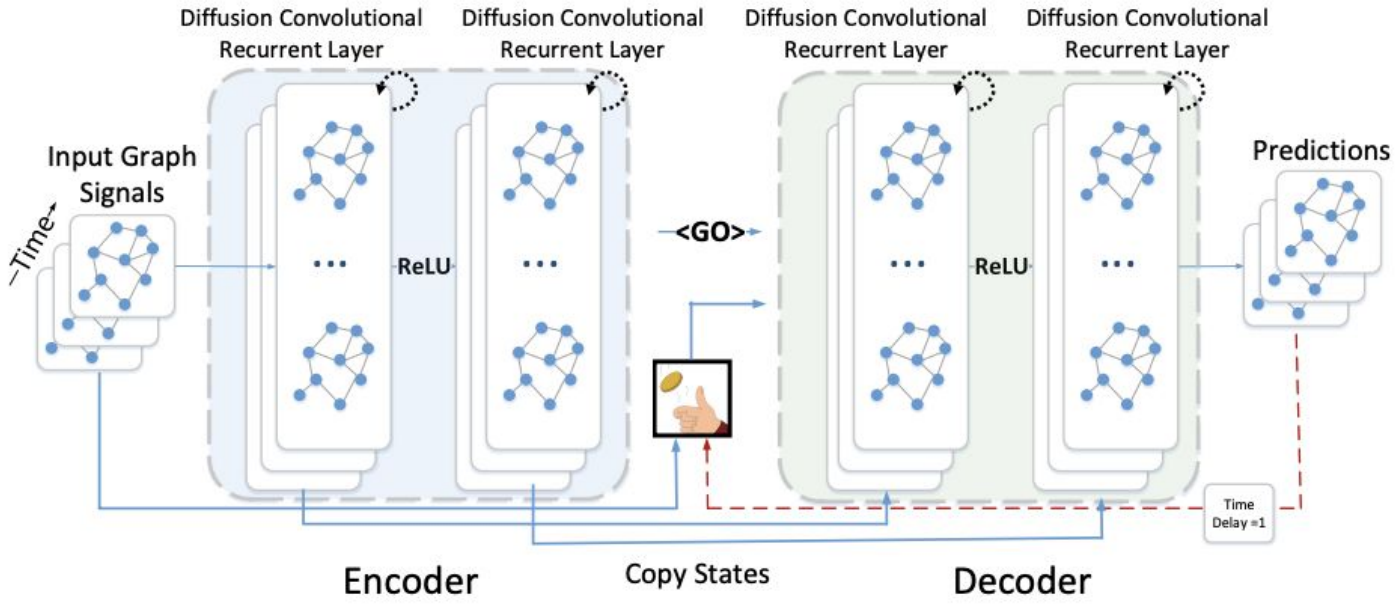


Typical traffic problems and the corresponding research directions

Ye J, Zhao J, Ye K, et al. How to build a graph-based deep learning architecture in traffic domain: A survey[J]. IEEE Transactions on Intelligent Transportation Systems, 2020.

GNNs for Transportation

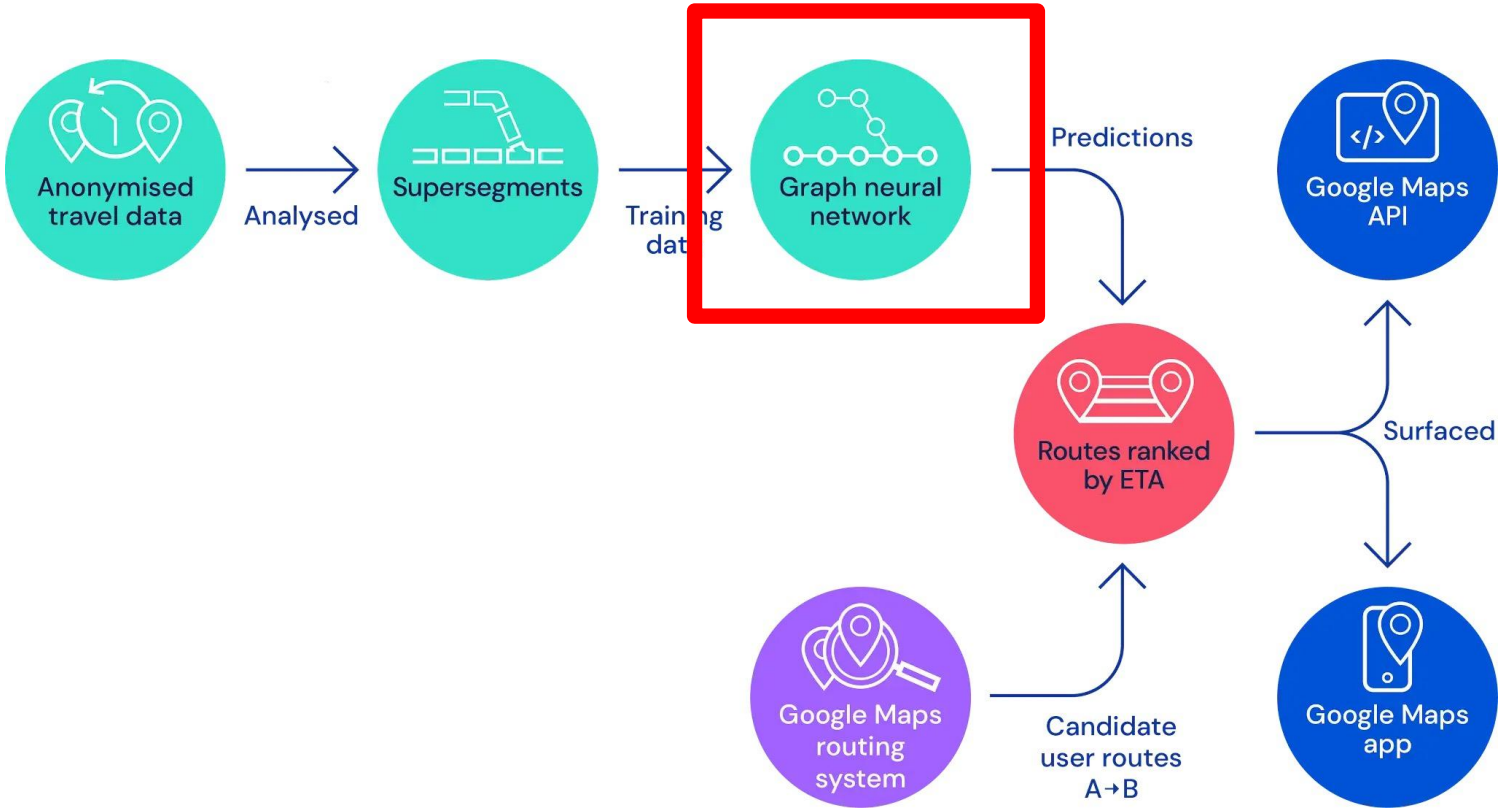
DCRNN incorporates spatial (spatial dependency on roadways) and temporal dependency (changing road conditions) in the traffic flow for traffic forecasting. Sensors in the roads are modeled as nodes in a graph.



Li Y, Yu R, Shahabi C, et al. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting[J]. arXiv preprint arXiv:1707.01926, 2017.

GNNs for Transportation

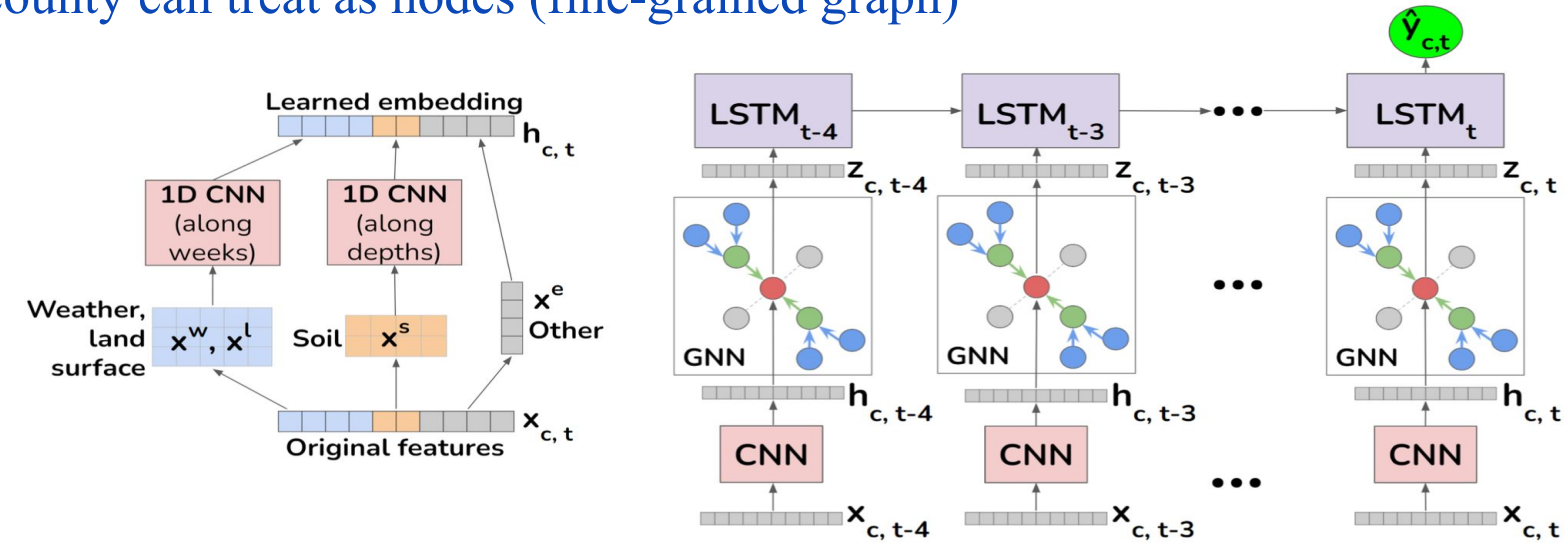
Google DeepMind uses GNN to estimate the traveling time and plan routes according.



https://www.deepmind.com/blog/traffic-prediction-with-advanced-graph-neural-networks

GNN for agriculture - A case study of crop yield prediction

- Crop yields depend on numerous factors, use these factors as input features:
  - weather $\mathbf{x}^w$, land surface $\mathbf{x}^l$, and soil quality $\mathbf{x}^s$, crop production index $\mathbf{x}^e$.
- Similar to region-level graph in transportation part, we can
  - Counties as nodes, an edge exists between two neighboring countries. (coarse-grained graph);
  - Regions in a county can treat as nodes (fine-grained graph)

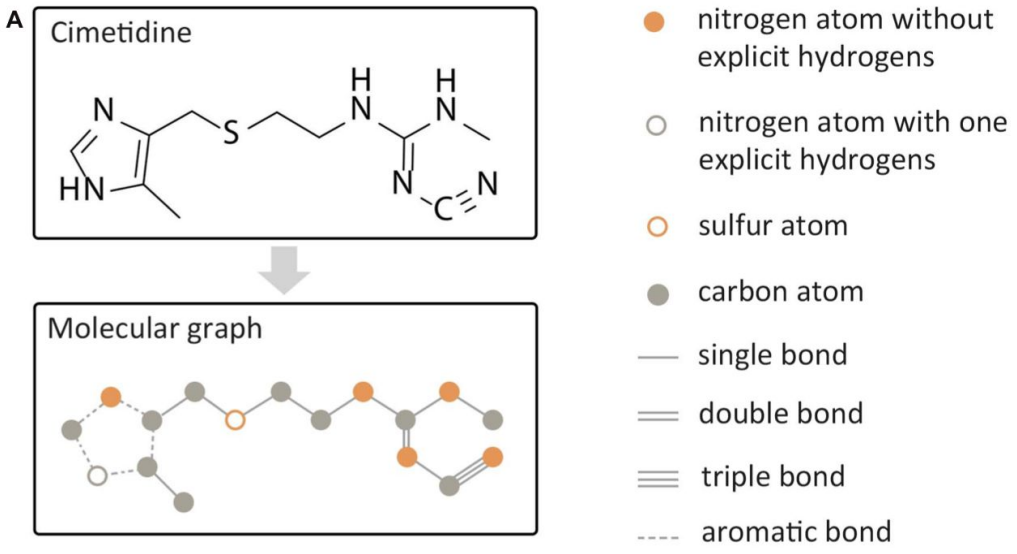


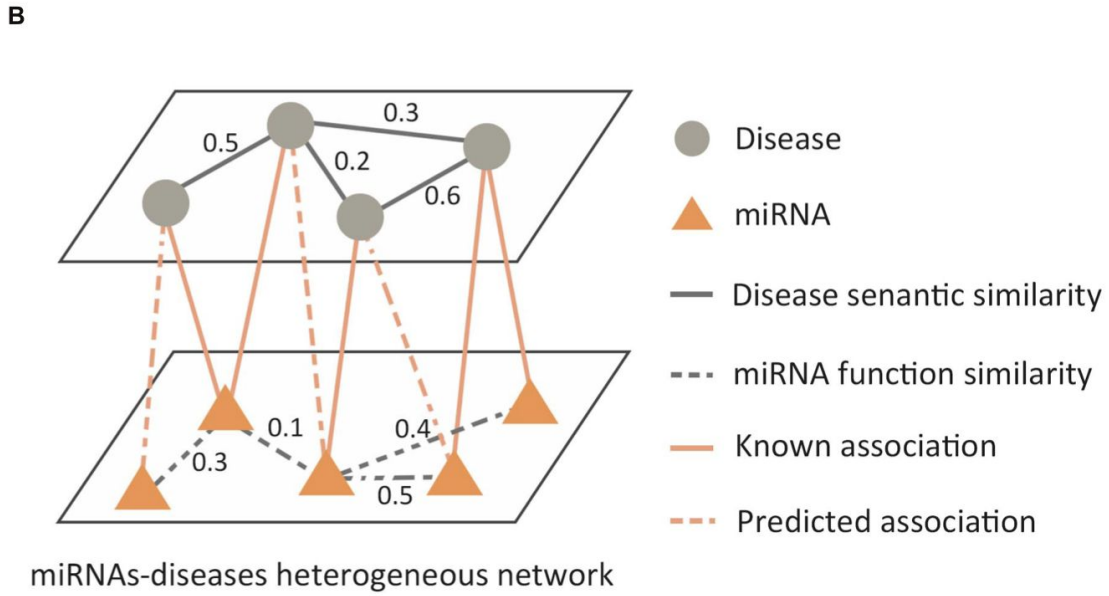
Fan J, Bai J, Li Z, et al. A GNN-RNN approach for harnessing geospatial and temporal information: application to crop yield prediction[C]// AAAI. 2022, 36(11): 11873-11881.

3.4 GNNs for Bioinformatics

Graphs in bioinformatics

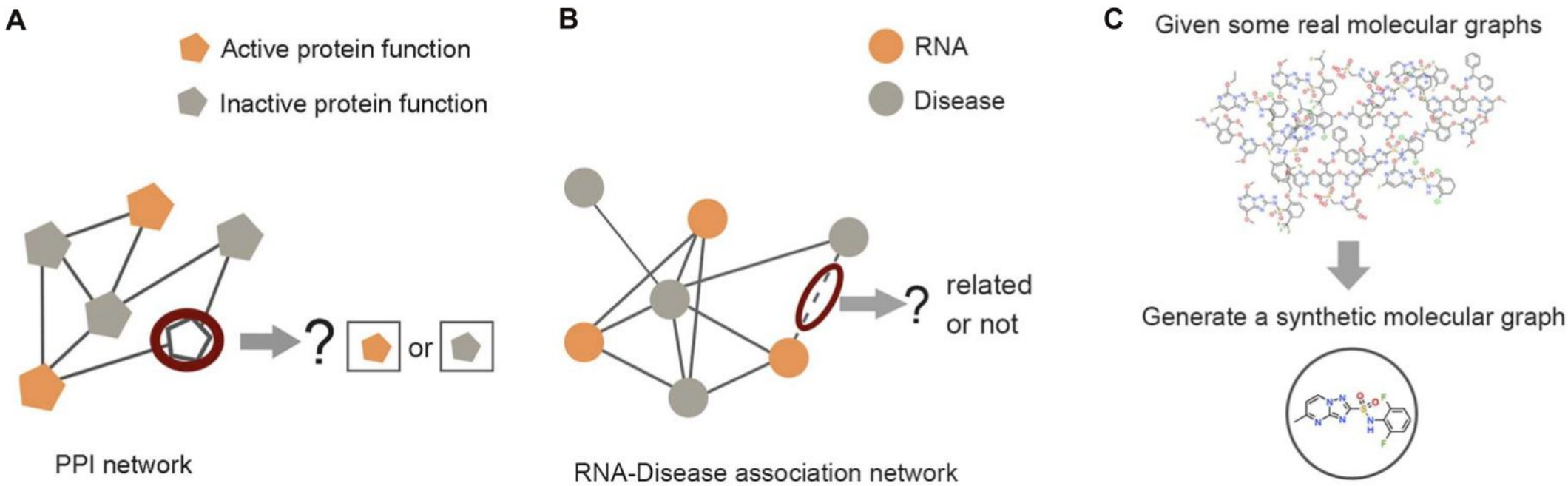
- Molecular graphs
  - -Atoms as nodes;
  - -Bonds as edges.
- Biological graphs
  - -gene, disease, RNA, etc., as nodes
  - -Known association between nodes.





Zhang X M, Liang L, Liu L, et al. Graph neural networks and their current applications in bioinformatics[J]. Frontiers in genetics, 2021, 12

Examples



Zhang X M, Liang L, Liu L, et al. Graph neural networks and their current applications in bioinformatics[J]. Frontiers in genetics, 2021, 12

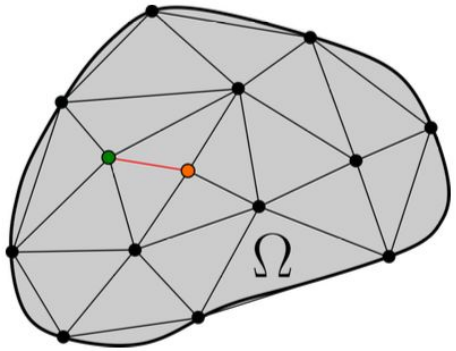
3.5 GNNs for PDEs

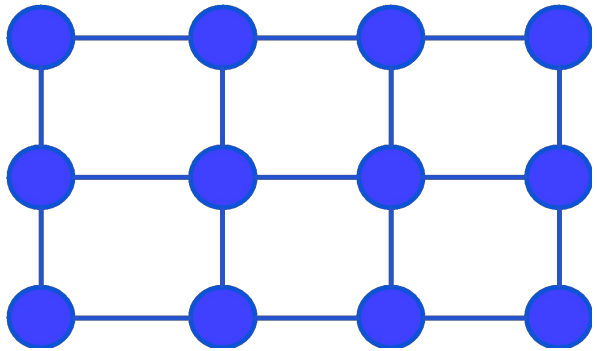
GNNs for PDEs

- A general form of the d-dimensional temporal PDEs can be expressed as follows:

$$\frac{\partial \mathbf{u}}{\partial t}(t, \mathbf{x}) = \mathcal{D}(\mathbf{u})(t, \mathbf{x})$$

- Numeric methods, such as FDM, FVM, FEM, discretize the time into time steps and space domain for each time step into grid or mesh.





General framework

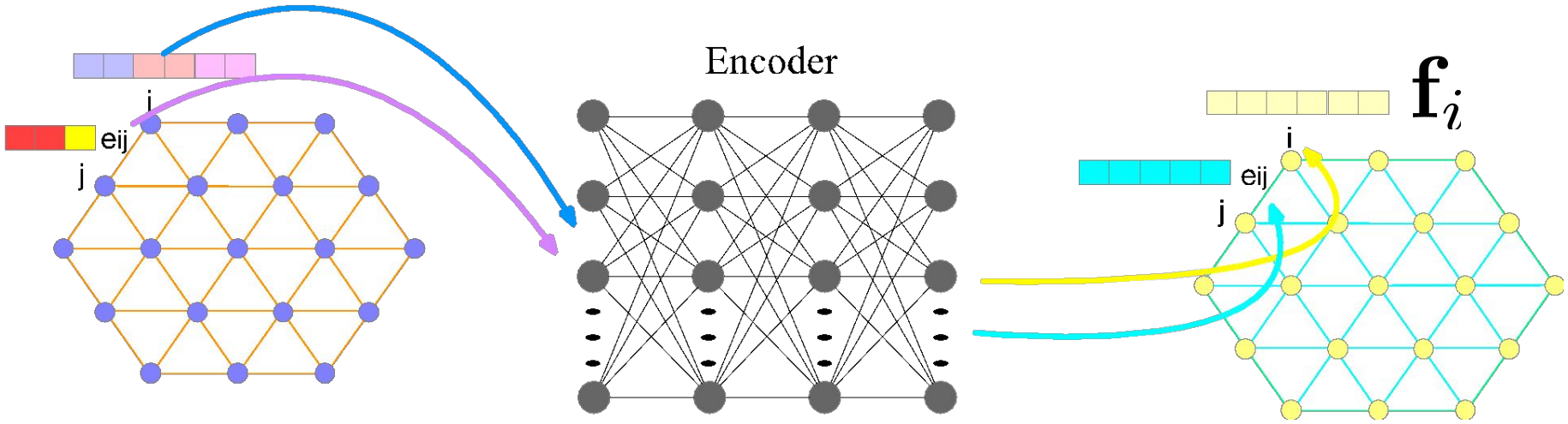
- Model the mesh as graph G=(*V, E*), nodes represent cells and the edges define local neighborhoods. Each time step t corresponds a graph Gt.

- The task is to learn a forward model of the dynamic quantities of the mesh at time t+1 given the current mesh G_t and (optionally) a history of previous meshes $\{M_{t-1},...,M_{t-h}\}$.

- Encoder-Processor-Decoder
  - "Encoding" generates node and edge embeddings from features, "processing" takes care of message passing, aggregation, and updating, and decoding is the post-processing step that gives final predictions, as described below.

GNNs for PDEs

- Encoder maps physical information into node embeddings and edge embeddings.
  - Physical information: solution values (e.g., velocity at time t_k), node type n_i, node position x_i, current time t_k, PDE coefficients and other attributes.



$$\mathbf{f}_i = \text{Encoder}(\mathbf{u}_i^{t_k}, \mathbf{x}_i, \mathbf{n}_i, t_k, \theta_{PDE}, \cdots)$$

$$\mathbf{f}_{e_{ij}} = \text{Encoder}(\mathbf{u}_i^{t_k}, \mathbf{u}_j^{t_k}, \mathbf{e}_{ij}, t_k, \cdots)$$

GNNs for PDEs

The processor computes M steps of learned message passing:

Compute message:

$$\mathbf{m}_{ij}^{m} = \phi(\mathbf{f}_{i}^{m}, \mathbf{f}_{j}^{m}, \mathbf{f}_{e_{ij}}^{m}, \cdots)$$

Update nodes:

$$\mathbf{f}_{i}^{m+1} = \psi(\mathbf{f}_{i}^{m}, \oplus_{j \in \mathcal{N}(i)} \mathbf{m}_{ij}^{m}, \cdots)$$

Update edges:

$$\mathbf{f}_{e_{ij}}^{m+1} = \gamma(\mathbf{f}_{i}^{m}, \mathbf{f}_{j}^{m}, \mathbf{f}_{e_{ij}}^{m}, \mathbf{e}_{ij}, \cdots)$$

$\emptyset, \psi$ and γ are multilayer perceptrons.

GNNs for PDEs

After message passing, use a decoder to output the node state of point x_i at next timestep t_{k+1}.

$$\mathbf{u}_i^{t_{k+1}} = \text{Decoder}(\mathbf{u}_{t_k}, \mathbf{f}_i^M, \cdots)$$

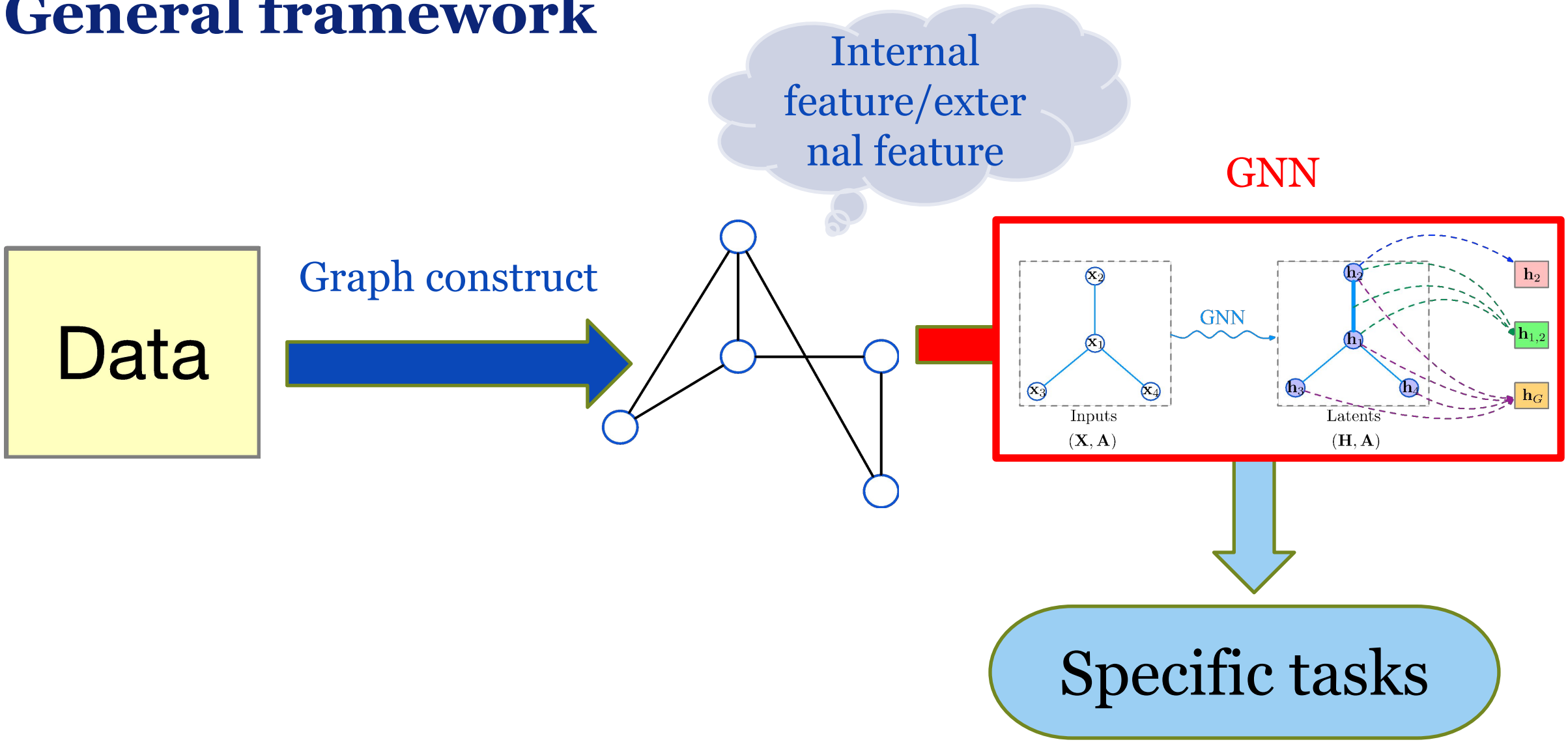
The Decoder can be MLP, CNN or other neural networks.

Example

A simulations of incompressible Navier-Stokes flow trajectories over a circular cylinder over time.

https://medium.com/stanford-cs224w/learning-mesh-based-flow-simulations-on-graph-networks-44983679cf2d

General framework



4 Future directions

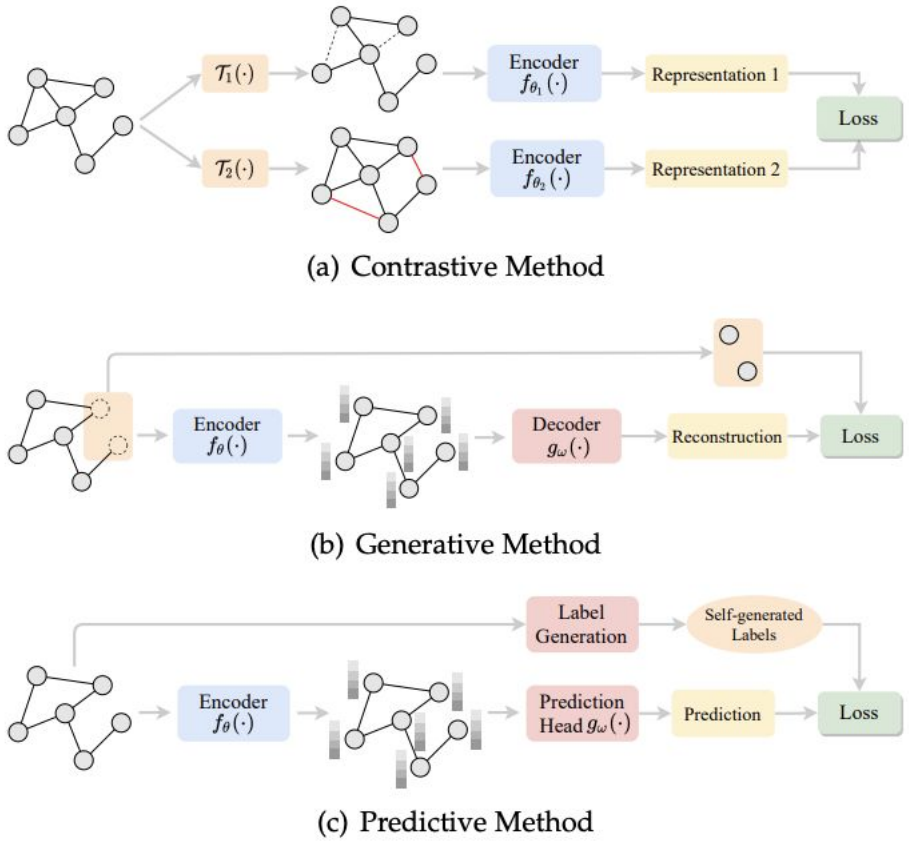
Self-supervised learning on graphs

Supervised learning: training a model that maps an input to an output based on the ground-truth input-output pairs provided by a labeled dataset.

Requires a decent amount of labeled data!
Expensive manual effort!

Self-supervised learning (SSL) is emerging as a new paradigm for extracting informative knowledge through well-designed pretext tasks unlabeled data.

Taxonomy of SSL on graphs



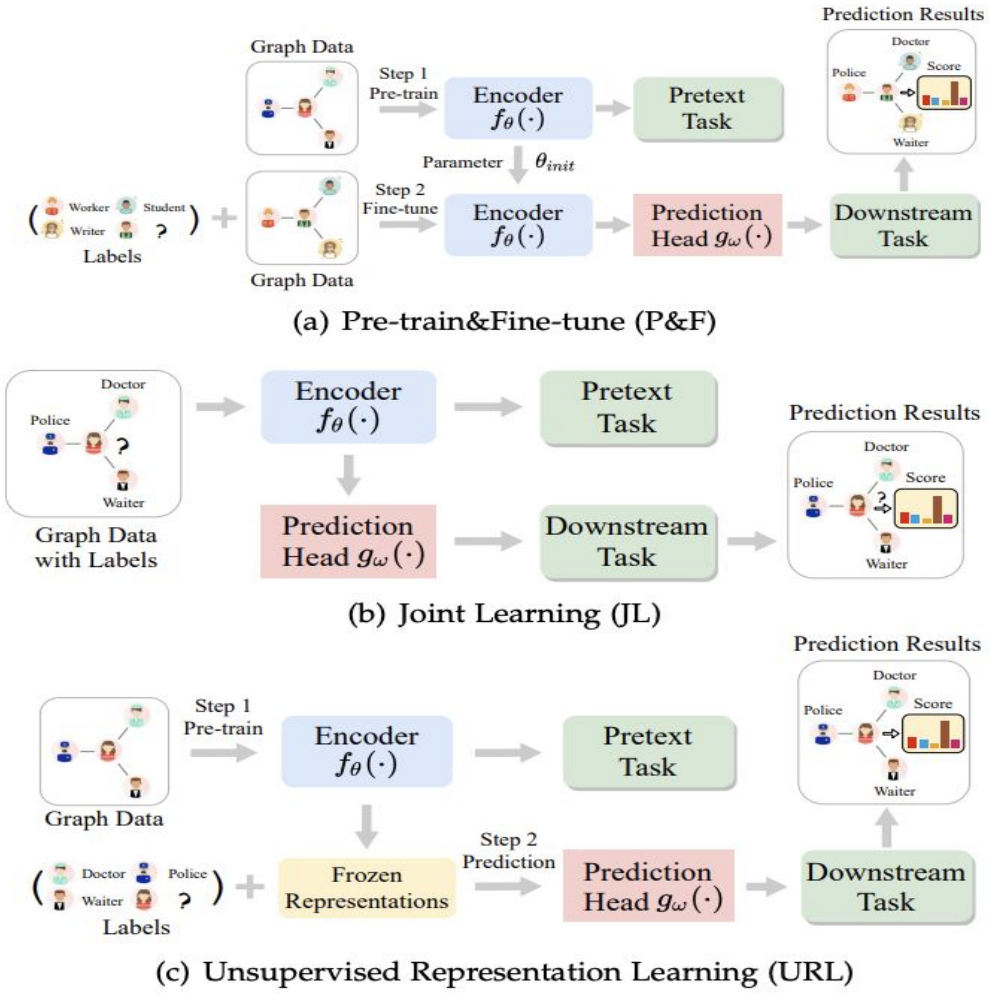
(a) Contrastive method contrasts the views generated from different augmentation T1 (·) and T2 (·). The information about the differences and sameness between (inter-data) data-data pairs are used as self-supervision signals.

(b) Generative method focuses on the (intra-data) information embedded in the graph, generally based on pretext tasks such as reconstruction, which exploit the attributes and structures of graph data as self-supervision signals.

(c) Predictive method generally self-generates labels by some simple statistical analysis or expert knowledge, and designs prediction-based pretext tasks based on self-generated labels to handle the data-label relationship.

Wu L, Lin H, Tan C, et al. Self-supervised learning on graphs: Contrastive, generative, or predictive[J].TKDE, 2021.

Taxonomy of training strategy



Wu L, Lin H, Tan C, et al. Self-supervised learning on graphs: Contrastive, generative, or predictive[J].TKDE, 2021.

(a) **Pre-train&Fine-tune** first pre-trains the encoder $f_\theta(\cdot)$ with un-labeled nodes by the self-supervised pretext tasks. The pre-trained encoder's parameters θ_{init} are then used as the initialization of the encoder for supervised fine-tuning on downstream tasks.

(b) **Joint Learning** strategy, an auxiliary pretext task is included to help learn the supervised downstream task. The encoder is trained through both the pretext task and the downstream task simultaneously.

(c) **Unsupervised Representation Learning** strategy, it first pre-trains the encoder $f_\theta(\cdot)$ with unlabeled nodes by the self-supervised pretext tasks. The pre-trained encoder's parameters θ_{init} are then frozen and used in the supervised downstream task with additional labels.

Explainability

- Deep learning methos are “black-boxes”
- So that, people cannot understand the mechanism underlying them.

- Explainability, it is the ability to explain a black- box model, i.e., to make the black-box model easier to understand or reveal the reason for its effectiveness.
- Important for trustworthy AI.

Li P, Yang Y, Pagnucco M, et al. Explainability in Graph Neural Networks: An Experimental Survey[J]. arXiv preprint arXiv:2203.09258, 2022.

Explainability

Which input edges are more important? which input nodes are more important? which node features are more important? what graph pat- terns will maximize the prediction of a certain class?

1. Instance-level methods - provide input-dependent explanations for each input graph
   - -Gradients/features-based methods (SA, Guided BP, CAM, Grad-CAM)
   - -Perturbation-based (GNNExplainer, PGExplainer, ZORRO, GraphMask)
   - -Decomposition methods (LRP, Excitation BP, GNN-LRP)
   - -Surrogate methods (GraphLime, RelEx, PGM-Explainer)
2. Model-level methods - explain graph neural networks without respect to any specific input example
   - -Graph generation (XGNN)

Yuan H, Yu H, Gui S, et al. Explainability in graph neural networks: A taxonomic survey[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2022.

Geometry

Developing equivariant GNNs to keep symmetry!

Important in the physical and bioinformatics field!

Geometric Deep Learning Blueprint instances of interest

Domain, Ω	Metric, g	Symmetry group, $\mathfrak{G}$	Architecture
Grids	L_∞	Translations	**CNNs**
Spheres	Great circle	3D rotations, SO(3)	**Spherical CNNs**
SO(3)	Geodesic		
Gauges	Geodesic	{Gauge transform.}	**Gauge Equivariant Mesh CNNs**
Graphs	Shortest path	Permutations, Σ_n	**GNNs**
Sets	$+\infty$	Permutations, Σ_n	**Deep Sets**
	0	Permutations, Σ_n	**Transformers**

GNNs for large-scale graphs

Large-scale: billions of nodes and edges…

It is challenging to directly apply the traditional GNN due to the large memory usage and long training time.

Scalability of GNN in learning large-scale graphs

-Reduce the size of the graph; (GraphSAGE , PinSage)

-Design scalable and efficient GNNs;

From applications perspective

- Robust and stable GNNs
  - it is a common phenomenon that the relationships between nodes are not always reliable;
  - the vulnerability of GNN to noisy data;

- Privacy Preserving
  - federated learning to train recommender systems without uploading users' data to the central server

On various types of graphs

Developing GNNs for specific types of graphs or Developing a uniform framework to handle various types of graphs!

- Knowledge graph
- Dynamic graph
- Heterogeneous graph
- Line graph
- …

5 Studying roadmap

Useful materials

Books:

- Graph Neural Networks: Foundations, Frontiers, and Applications
  - Lingfei Wu, Peng Cui, Jian Pei, Liang Zhao
- Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges
  - Michael M. Bronstein, Joan Bruna, Taco Cohen, Petar Veličković
- Graph Representation Learning Book
  - William Hamilton

Useful materials

Reading groups:

- LoGaG: Learning on Graphs and Geometry Reading Group, https://hannes-stark.com/logag-reading-group
- Graph Representation Learning Reading Group at Mila - Quebec AI Institute, https://grlmila.github.io/

Github repository:

- Try “awesome-gnn” in Github search, see what happens!

Courses

- Machine Learning with Graphs, http://web.stanford.edu/class/cs224w/
  - Instructor: Prof Jure Leskovec
  - Topics: representation learning and Graph Neural Networks; algorithms for the World Wide Web; reasoning over Knowledge Graphs; influence maximization; disease outbreak detection, social network analysis
- Representation Learning on Graphs and Networks, https://www.cl.cam.ac.uk/teaching/2122/L45/
  - Instructor: Prof Pietro Liò, Dr Petar Veličković
  - Topics: fundamentals, theoretical principles, design graph neural networks, geometric deep learning framework
- GDL Course, https://geometricdeeplearning.com/lectures/
  - Instructor: Prof Michael M. Bronstein et al.
  - Topics: geometric deep learning
- NYU's Deep Learning course week 13, https://atcold.github.io/pytorch-Deep-Learning/en/week13/13/
  - Instructor: Xavier Bresson and Alfredo Canziani
  - Topics: GCN and its implementation.

You can follow

- Prof Max Welling, (AMLab http://amlab.science.uva.nl/), fundamental GNN, theory of GNN, geometric deep learning
- Prof Michael Bronstein, https://www.cs.ox.ac.uk/people/michael.bronstein/, geometric deep learning
- Dr Petar Veličković, https://petar-v.com/, author of GAT, geometric deep learning
- Dr Thomas Kiof, http://tkipf.github.io/, author of GCN, GNN for physical
- Prof Jure Leskovec, SNAP (http://snap.stanford.edu/), GNN for social networks, expressive of GNN, large scale GNN, graph dataset
- Prof Jie Tang, https://keg.cs.tsinghua.edu.cn/jietang/, GNN for social networks, recommendation system, knowledge graph
- Prof Pietro Liò, https://www.cl.cam.ac.uk/~pl219/, GNN for bioinformatics
- Prof Jiliang Tang, https://www.cse.msu.edu/~tangjili/, fundamental GNN, GNN applications
- Prof Philip S. Yu, https://www.cs.uic.edu/PSYu, heterogeneous graph

Toolkits

https://pytorch-geometric.readthedocs.io/en/latest/

https://www.dgl.ai/

https://www.dgl.ai/

https://graphneural.network/

Jraph,
https://github.com/deepmind/jraph

TorchDrug
https://torchdrug.ai/

Graph Nets library
https://github.com/deepmind/graph_nets

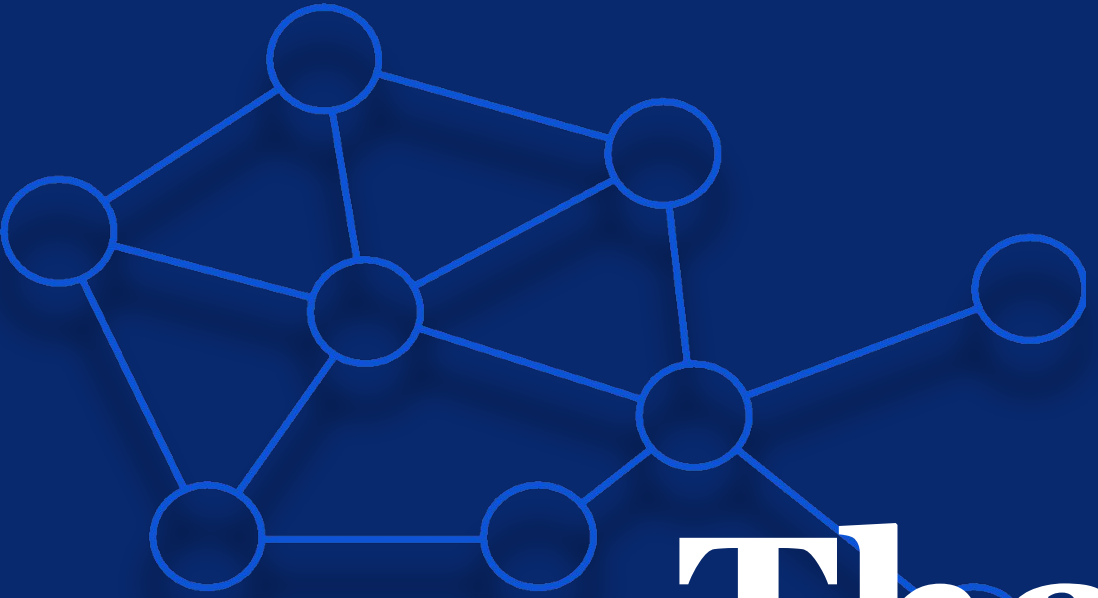
Datasets

https://ogb.stanford.edu/

OGB datasets are large-scale, encompass multiple important graph ML tasks, and cover a diverse range of domains, ranging from social and information networks to biological networks, molecular graphs, source code ASTs, and knowledge graphs. For each dataset, we provide a unified evaluation protocol using meaningful application-specific data splits and evaluation metrics.

https://chrsmrrs.github.io/datasets/

The collection consists of over 120 datasets of varying sizes from a wide range of applications. Small molecules, Bioinformatics, Computer vision, Social networks,Synthetic



Thank you!

nedchen0728@gmail.com