Lecture 14: Graph Neural Networks

Yen-Yi Ho

Department of Statistics

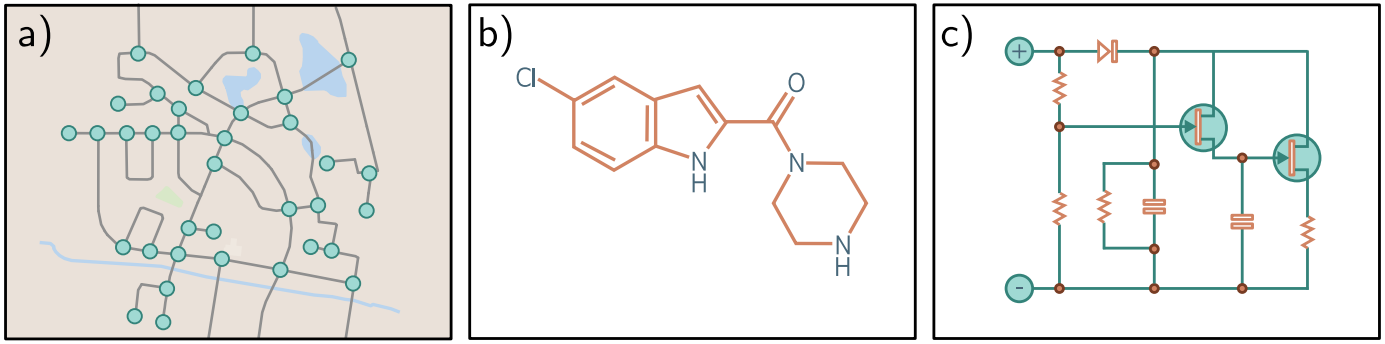
Outline

- Graph-Structured Data
- Graph Convolutional Networks
- Graph Attention Networks
- Edge Graphs

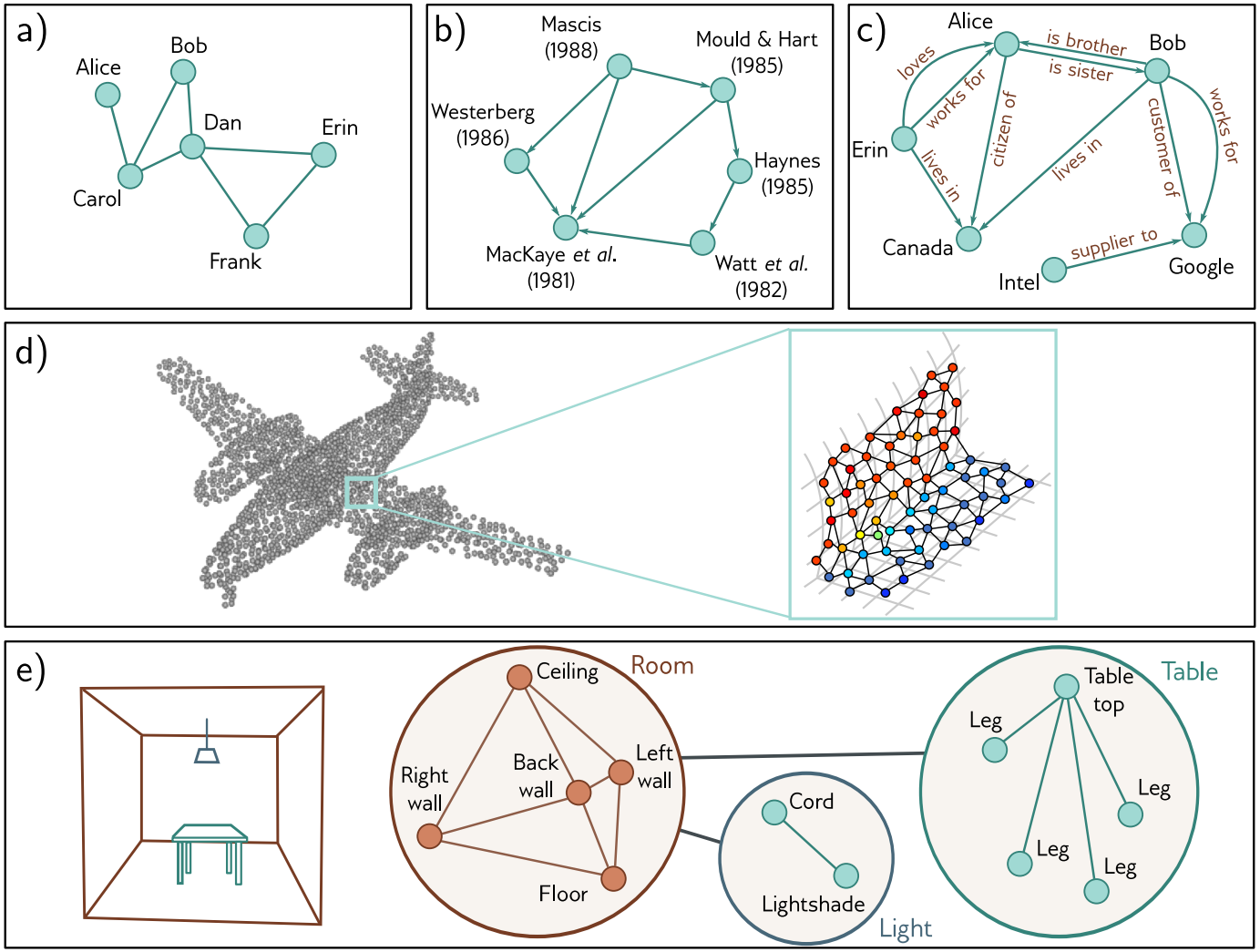
Graph-Structured Data

Graphs are a general language for describing and analyzing entities (node) with relations/interactions (edges). Examples:

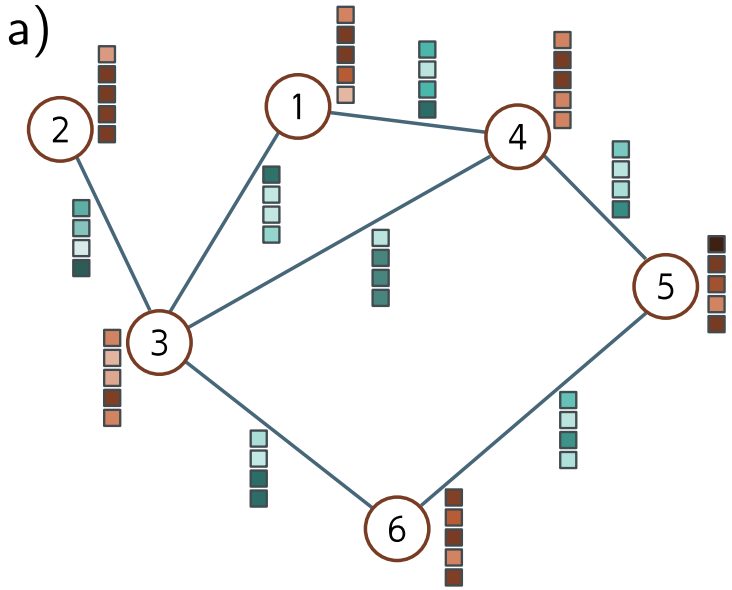
- Social networks, citation networks, multi-agent systems
- Knowledge graphs
- Electrical circuits
- Protein interaction networks
- Brain networks
- Road map



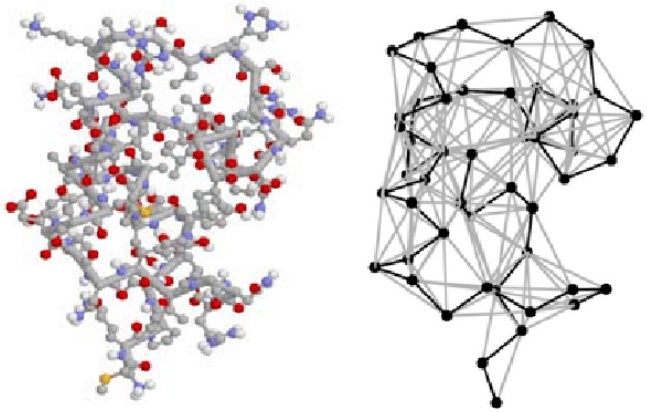
Types of Graph



Graph Representation



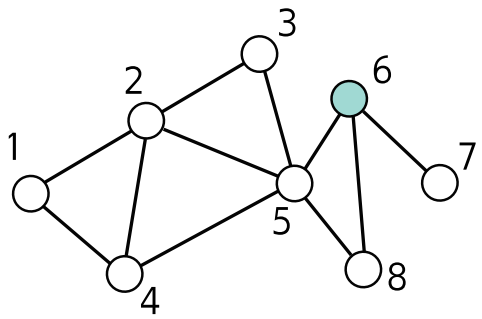
Graph Representation: Example



```
# Physicochemical property lookup (simplified, illustrative values)
# Columns: [hydrophobicity, charge, relative_size]
AA_PROPS = {
    'A': [1.8,  0.0, 0.3],   'C': [2.5,  0.0, 0.5],
    'D': [-3.5,-1.0, 0.5],   'E': [-3.5,-1.0, 0.6],
    'F': [2.8,  0.0, 0.9],   'G': [-0.4, 0.0, 0.1],
    'H': [-3.2, 0.5, 0.8],   'I': [4.5,  0.0, 0.7],
    'K': [-3.9, 1.0, 0.8],   'L': [3.8,  0.0, 0.7],
    'M': [1.9,  0.0, 0.8],   'N': [-3.5, 0.0, 0.6],
    'P': [-1.6, 0.0, 0.5],   'Q': [-3.5, 0.0, 0.7],
    'R': [-4.5, 1.0, 1.0],   'S': [-0.8, 0.0, 0.4],
    'T': [-0.7, 0.0, 0.5],   'V': [4.2,  0.0, 0.6],
    'W': [-0.9, 0.0, 1.0],   'Y': [-1.3, 0.0, 0.9],
}
```

Adjacency Matrix

a)



b)

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}$$

c)

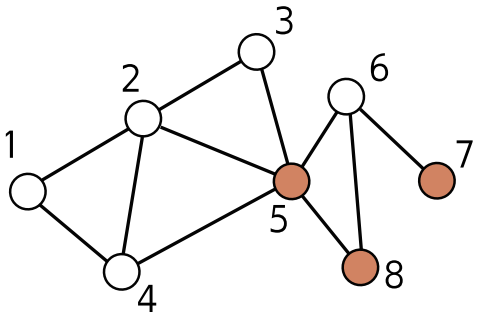
$$\mathbf{A}^2 = \begin{bmatrix} 2 & 1 & 1 & 1 & 2 & 0 & 0 & 0 \\ 1 & 4 & 1 & 2 & 2 & 1 & 0 & 1 \\ 1 & 1 & 2 & 2 & 1 & 1 & 0 & 1 \\ 1 & 2 & 2 & 3 & 1 & 1 & 0 & 1 \\ 2 & 2 & 1 & 1 & 5 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 3 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 2 \end{bmatrix}$$

d)

$$\mathbf{x} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

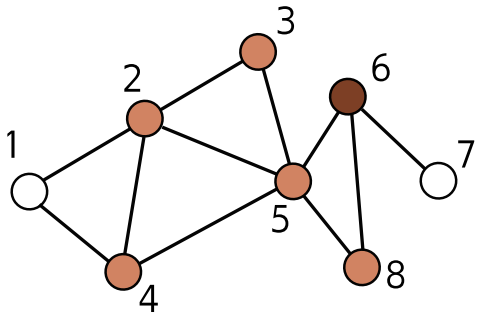
e)

$$\mathbf{Ax} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$



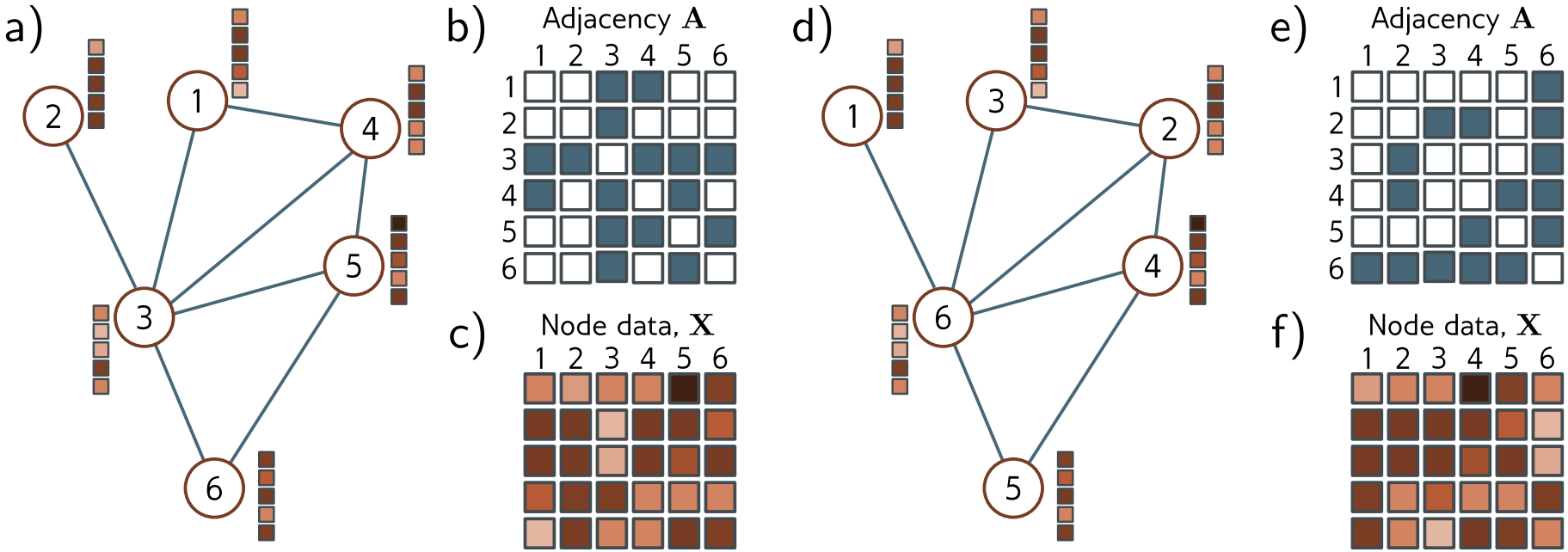
f)

$$\mathbf{A}^2\mathbf{x} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 3 \\ 0 \\ 1 \end{bmatrix}$$



A non-zero entry at position (m, n) in A^L indicates that the distance from m to n must be less than or equal to L.

Permutation of Node Indices

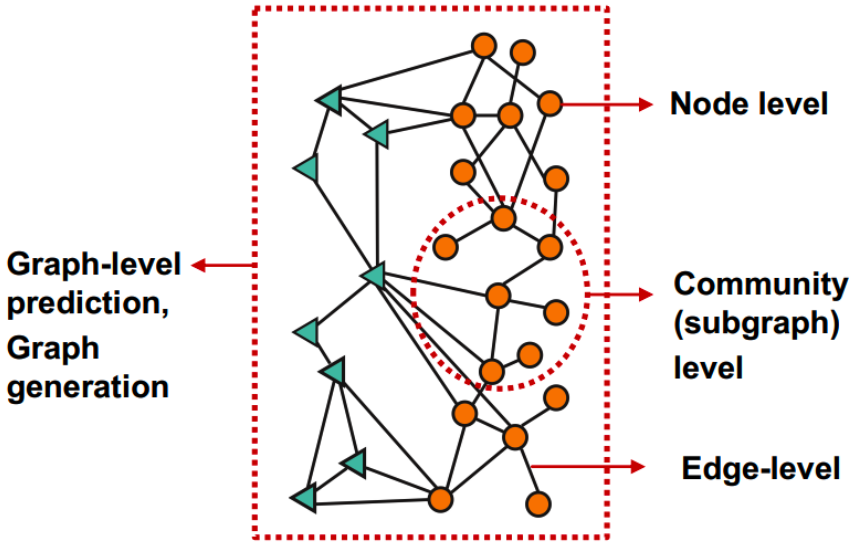


$$
\begin{aligned}
X' &= XP \\
A' &= P^T AP.
\end{aligned}
$$

Different Types of Tasks

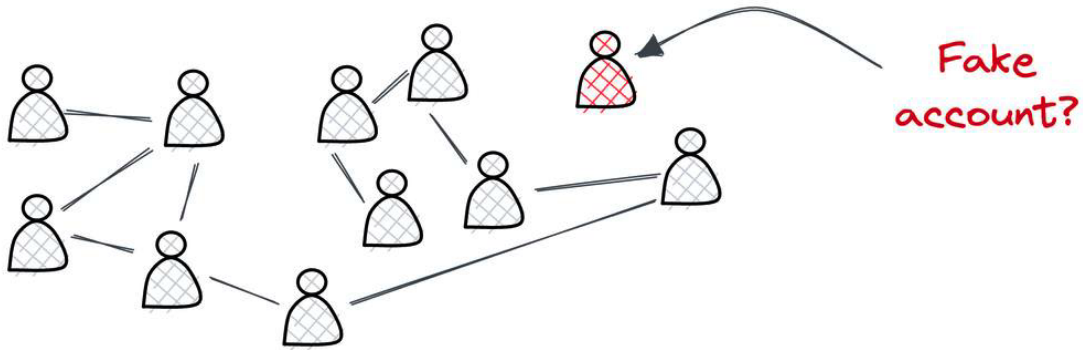
Graph-based machine learning involves multiple tasks categorized by the focus of analysis. The main categories of tasks include:

- Node-Level Tasks: Predicting properties of individual nodes
- Edge-Level Tasks: Inferring relationships between node pairs.
- Community-Level Tasks: Detecting and analyzing groups of closely connected nodes
- Graph-Level Tasks: Understanding global graph properties

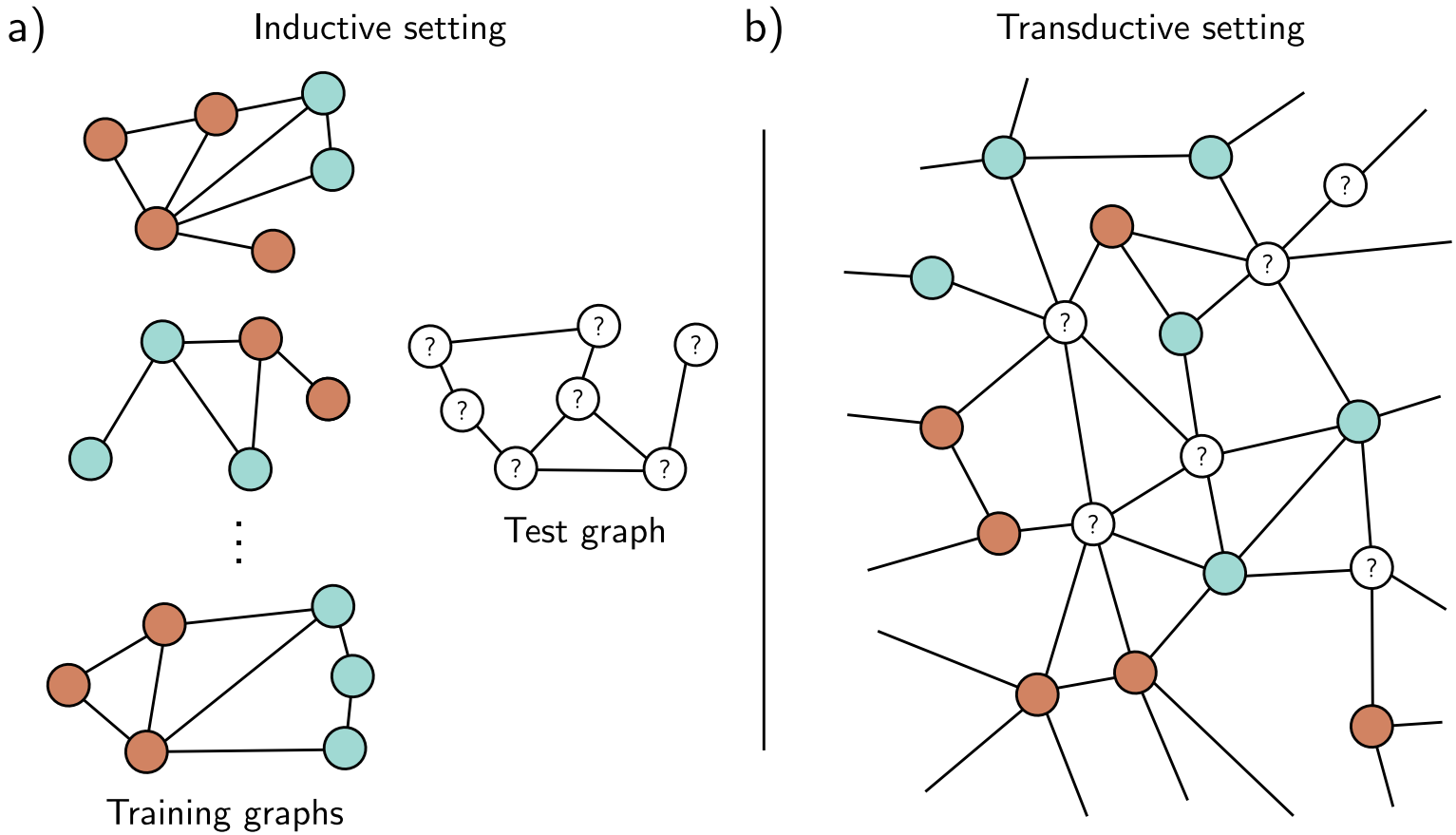


Node-Level Tasks

- Node Classification: Assign labels to nodes (e.g., fraud detection in financial networks).
- Node Regression: Predict continuous values (e.g., influence score in social networks).



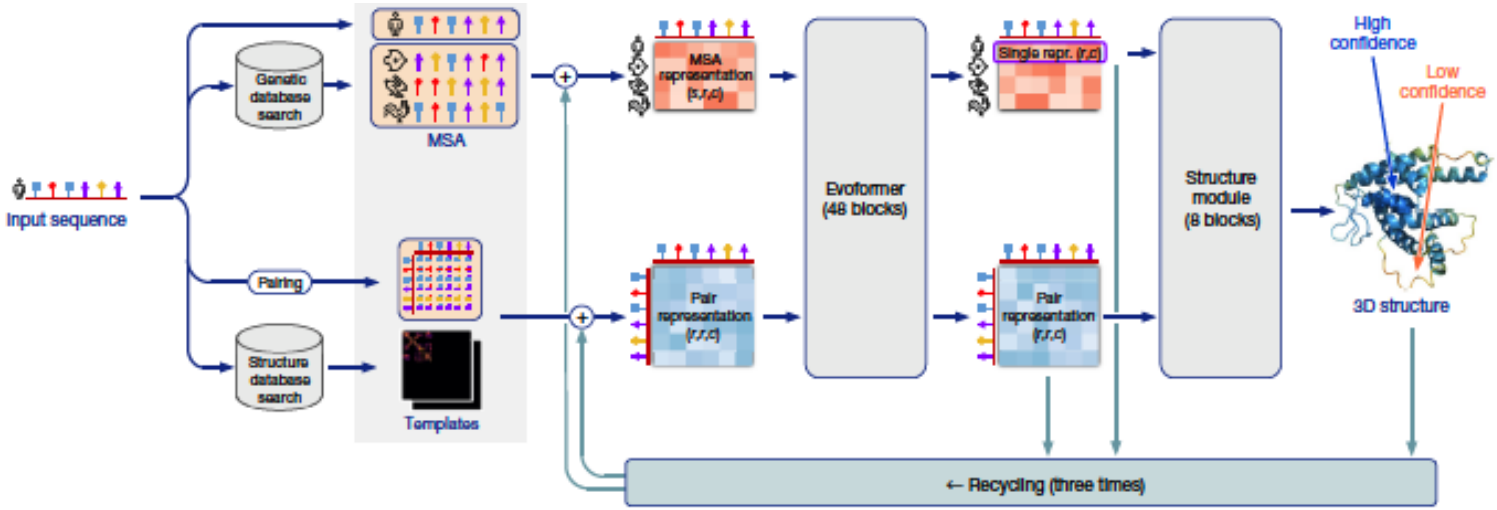
Inductive versus Transductive Models



Node-Level Task Example: AlphaFold

AlphaFold is a deep learning-based model for protein structure prediction

- It represents protein structures as spatial graphs where:
  Nodes: Amino acids (residues) in a protein sequence.
  Edges: Defined by spatial proximity between amino acids.
- These graphs capture important structural and functional relationships in proteins.

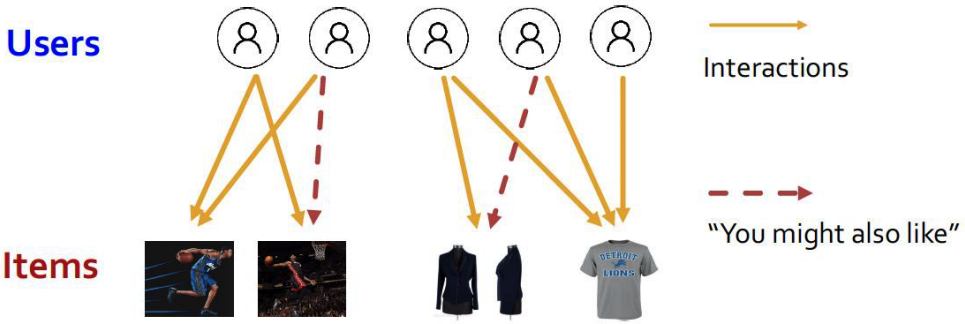


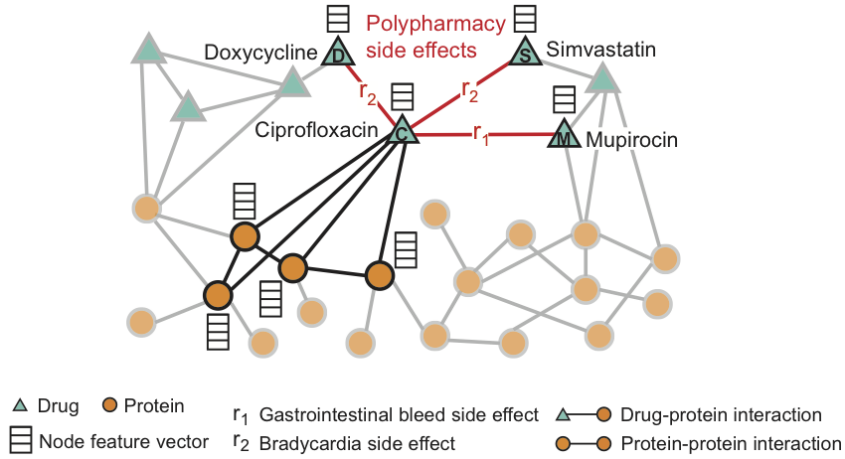
Edge-Level Task

Predict properties or existence of edges.

Common examples:

- Link prediction: Determine if an edge should exist (e.g., friend recommendations on social media).
- Edge Classification: Categorize relationships (e.g., sentiment analysis in social interactions).



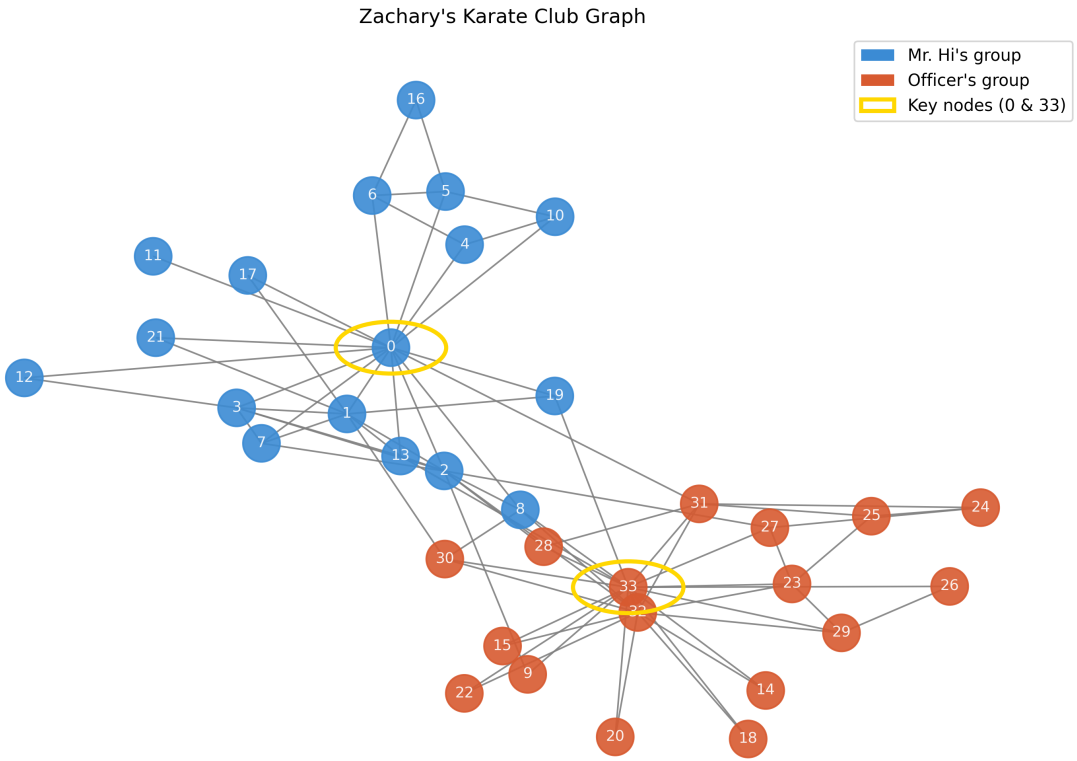


Community-Level Task

Identify groups of nodes with similar properties or high connectivity

Common examples:

- Community Detection: Identifying clusters of related nodes
- Graph Partitioning: Dividing a graph into smaller, meaningful subgraphs.

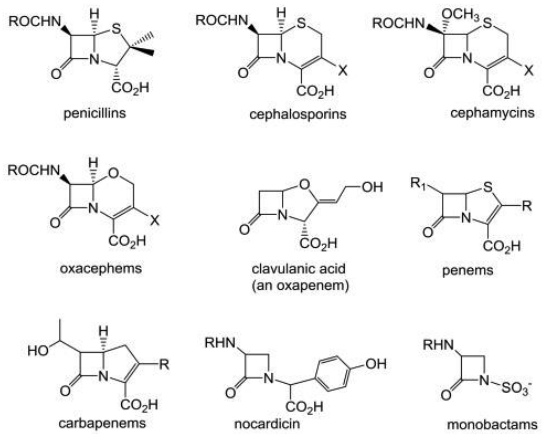


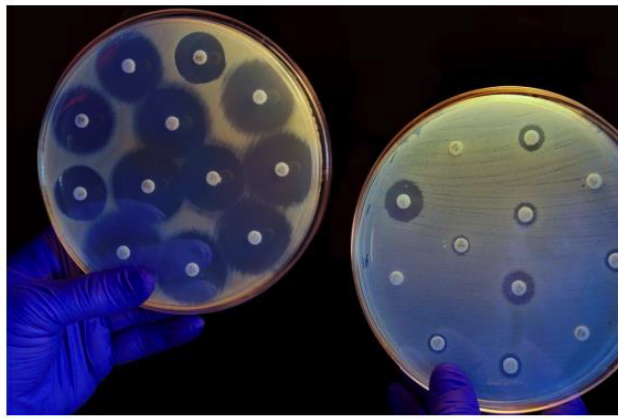
Graph-Level Task

Analyze entire graphs (or a subset) to understand global properties

Common examples:

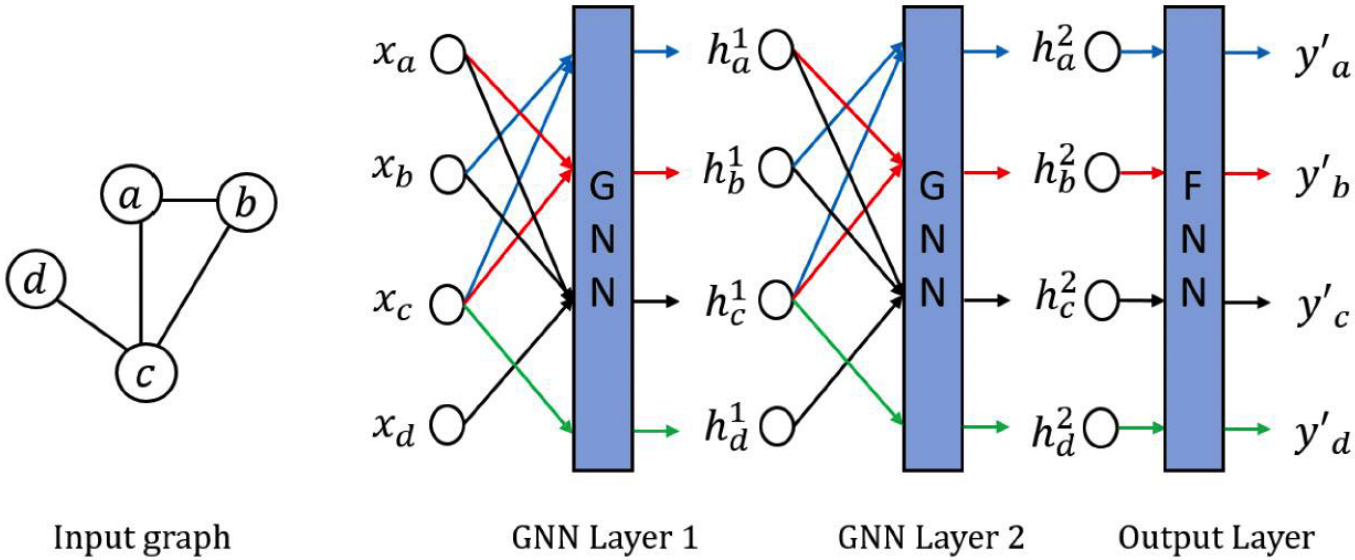
- Graph Classification: Assign labels to entire graphs (e.g., molecule toxicity prediction).
- Graph Clustering: Group similar graphs based on structure.





General Graph Neural Network (GNN) Framework

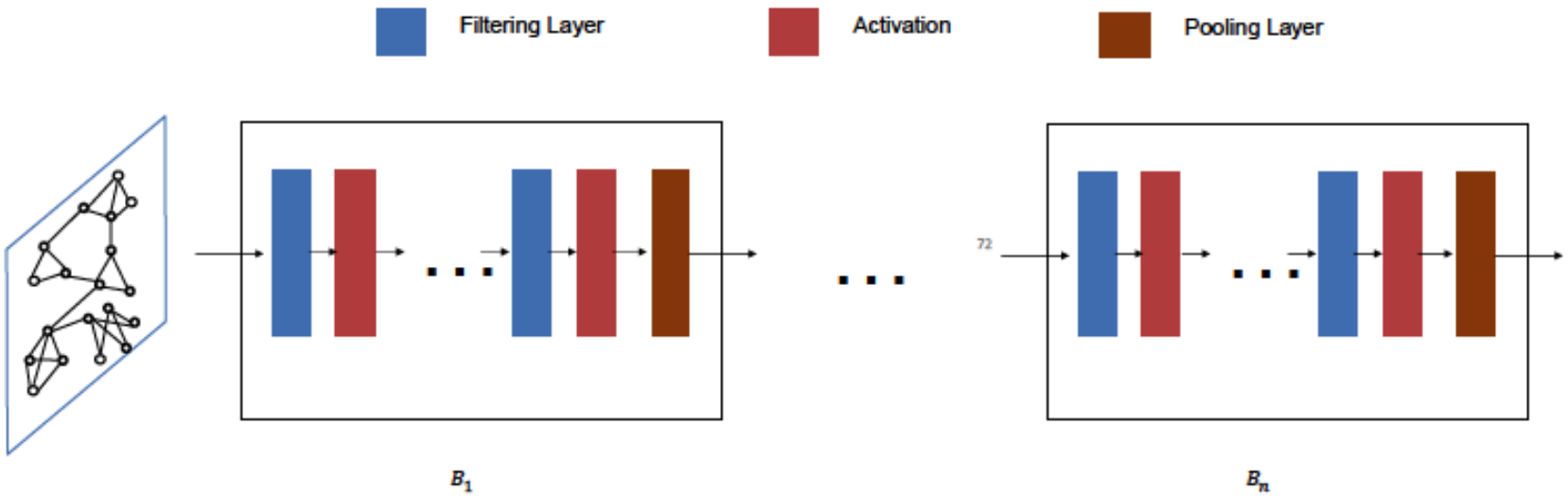
For node/edge-level task



$$Pr(y^n == 1|X, A) = sig[\beta_K + \omega_K h_K^n],$$
$$Pr(y^{mn} == 1|X, A) = sig[h_K^m h_K^n].$$

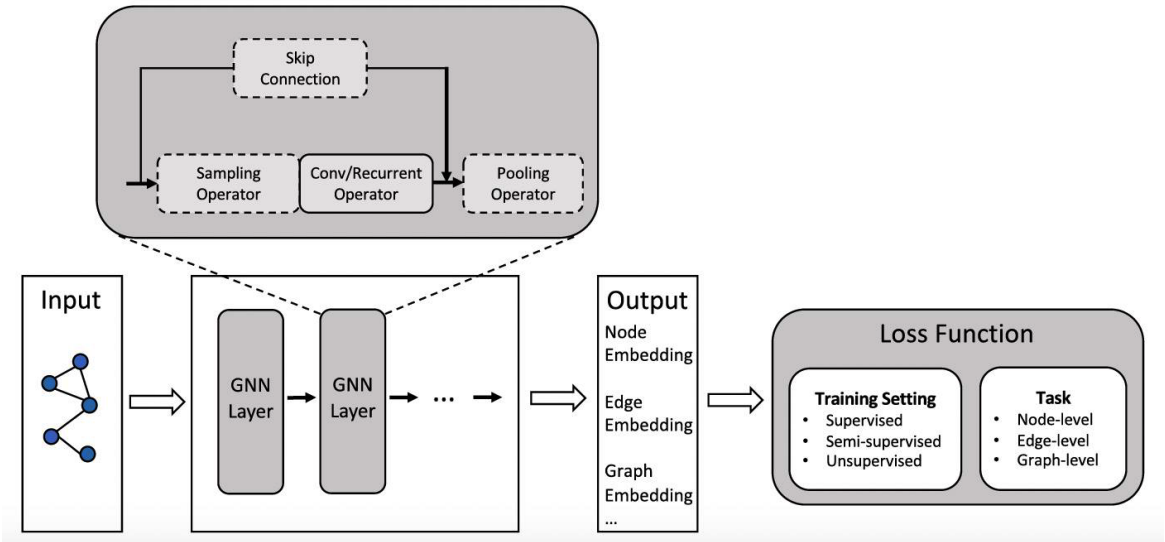
General GNN Framework

For graph-level task



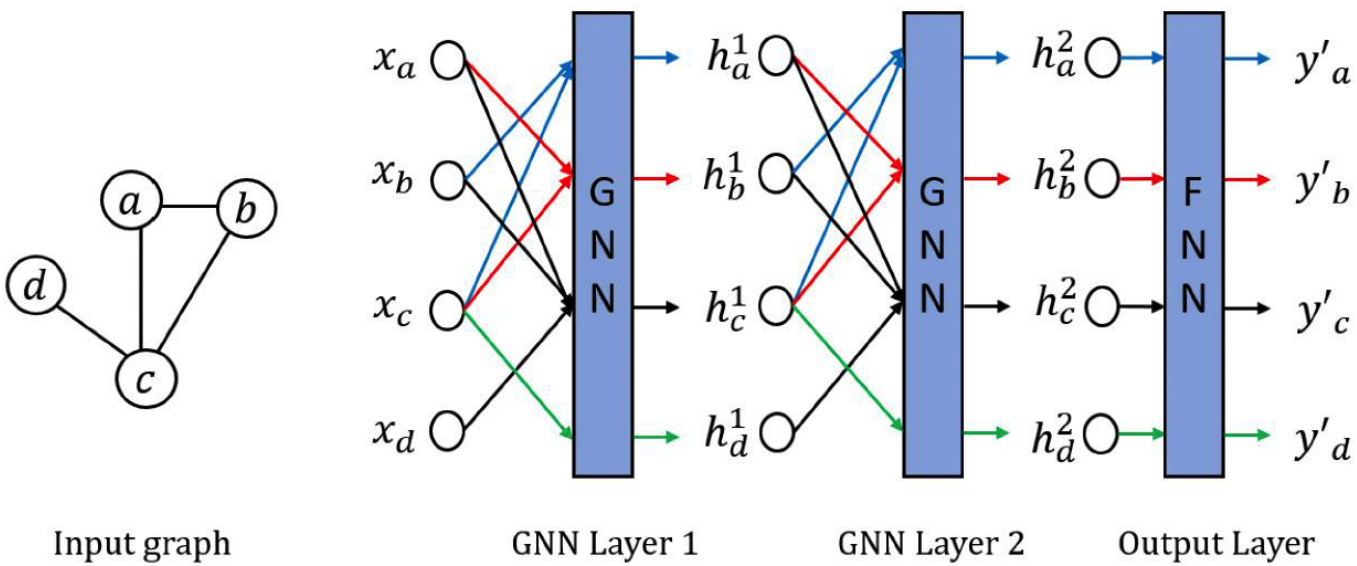
GNN Designs

- Define Graph: nodes, edges, and features
- Feature Engineering: define node and edge features
- Message Passing
- Choose Architecture: GCN, GAT, GraphSAGE, ...etc
- Loss Function: Supervised (cross-entropy), unsupervised (contrastive)
- Training
- Evaluation
- Deployment



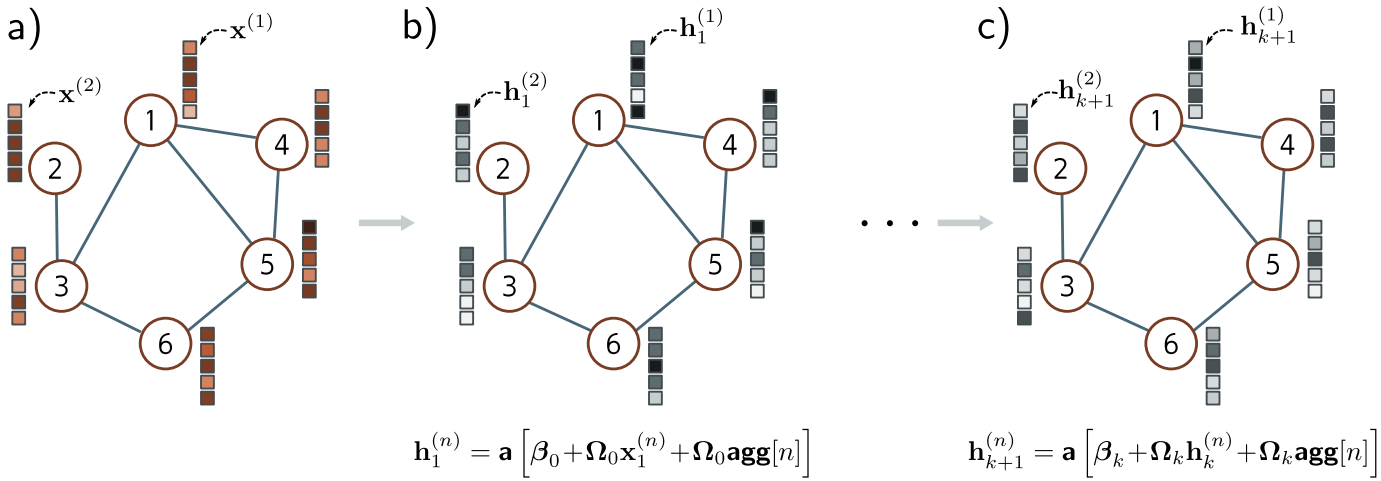
Graph Convolutional Networks

$$
\begin{aligned}
H_1 &= F[X, A, \phi_0] \\
H_2 &= F[H_1, A, \phi_1] \\
\vdots &= \vdots \\
H_K &= F[H_{K-1}, A, \phi_{K-1}]
\end{aligned}
$$



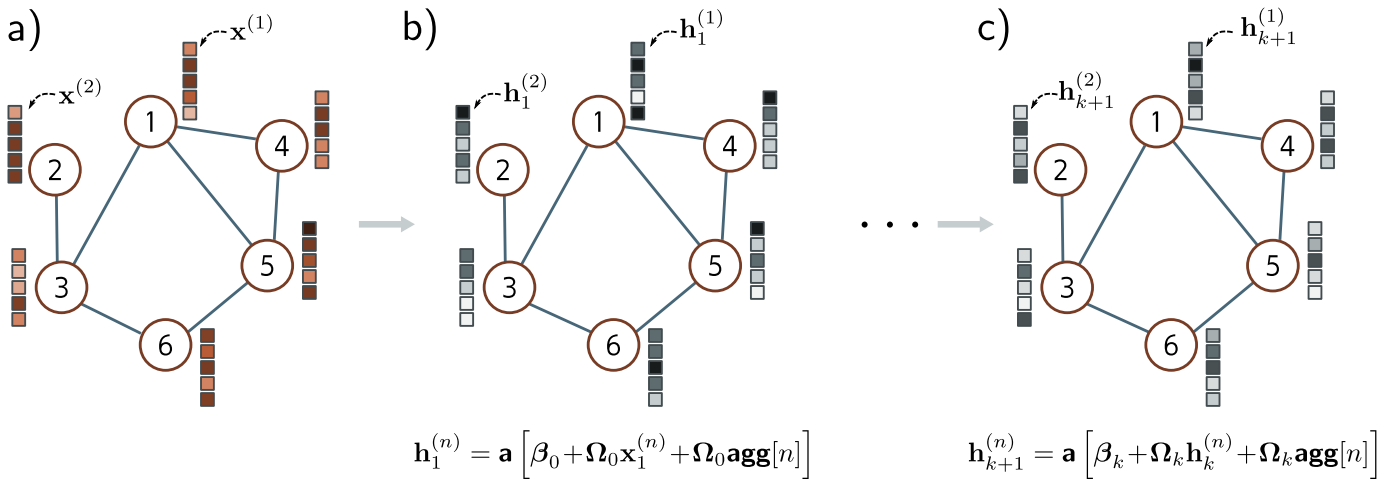
Example GCN Layer

$$
\begin{aligned}
agg[n,k] &= \sum_{m \in ne[n]} h_k^m \\
h_{k+1}^n &= a[\beta_k + \omega_k \cdot h_k + \omega_k \cdot agg[n,k]]. \\
H_{k+1} &= a[\beta_k 1^T + \omega_k H_k + \omega_k H_k A], \\
&= a[\beta_k 1^T + \omega_k H_k (A + I)].
\end{aligned}
$$



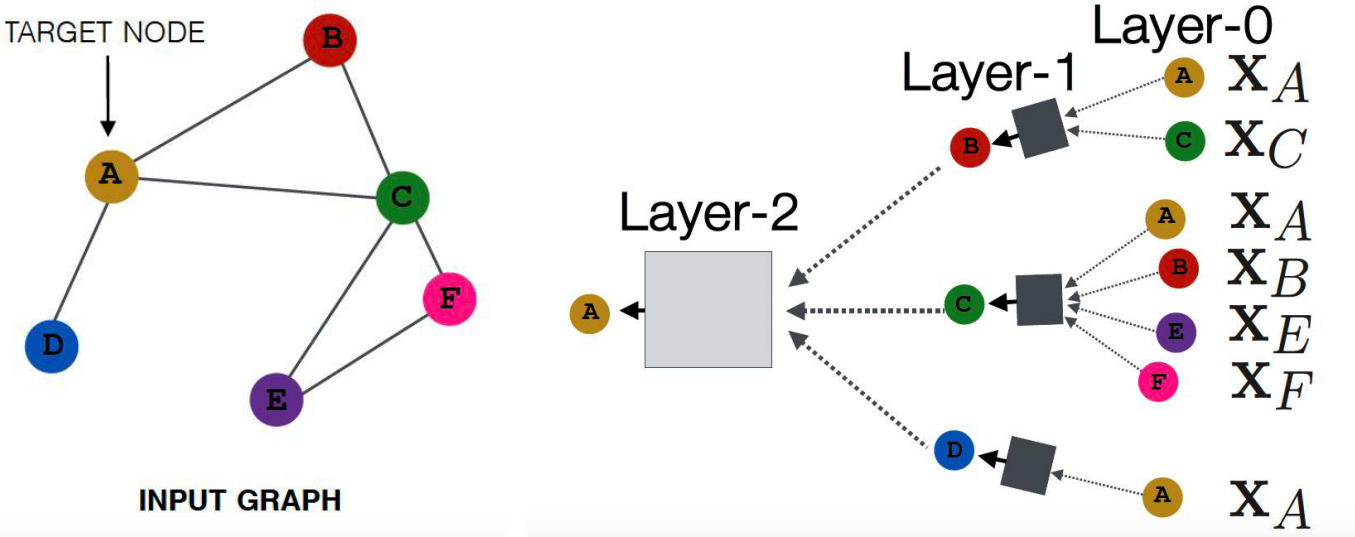
Graph Classification

$$
\begin{aligned}
H_1 &= a[\beta_0 1^T + \omega_0 X(A+I)]. \\
H_2 &= a[\beta_1 1^T + \omega_1 H_1(A+I)] \\
\vdots &= \vdots \\
H_K &= a[\beta_{K-1} 1^T + \omega_{K-1} H_{k-1}(A+I)]. \\
f[X,A,\Phi] &= sig[\beta_K + \omega_K H_K 1/N].
\end{aligned}
$$

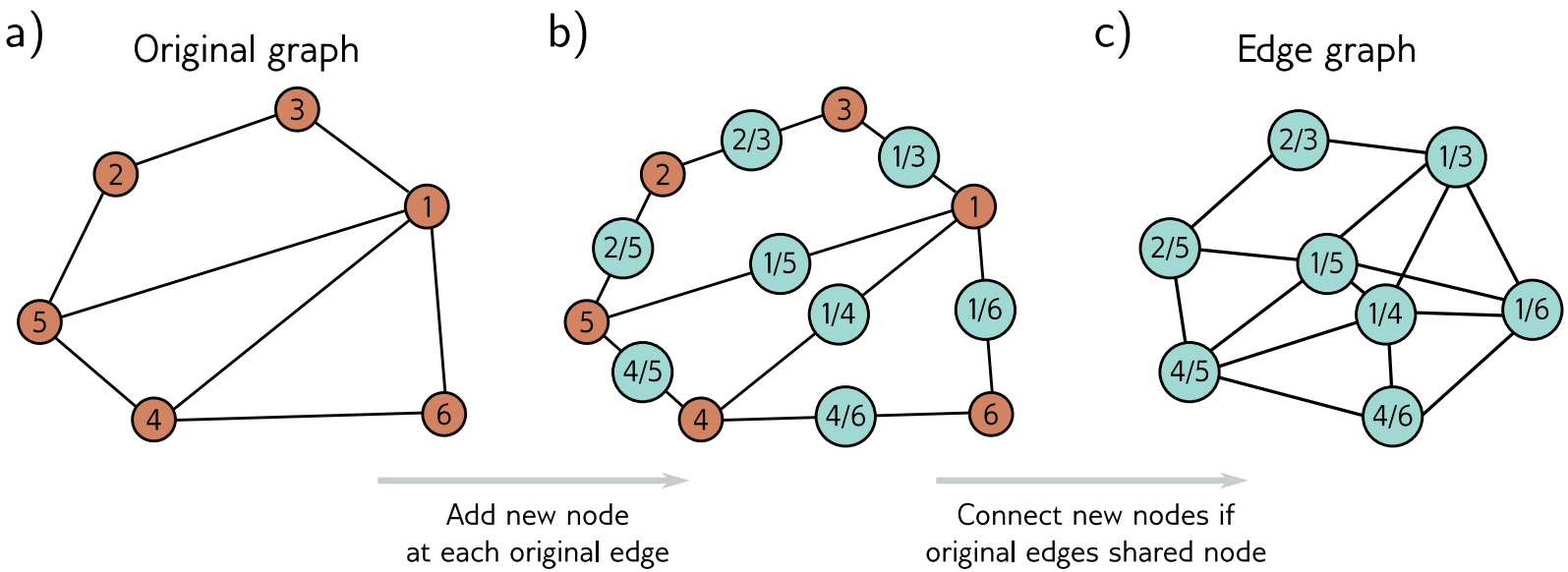


Key Modules in GNN

- Sampling Module: Aims toreduce the sizeof each node's neighborhood, especially forlarge graphs, preventing the neighbor explosion problem
- Propagation Module: Performs message passing via convolutions.
- Pooling Module: Aggregates node-level embeddings into subgraph or graph-level re-presentations, extracting higher-level features needed for tasks like graph classification.

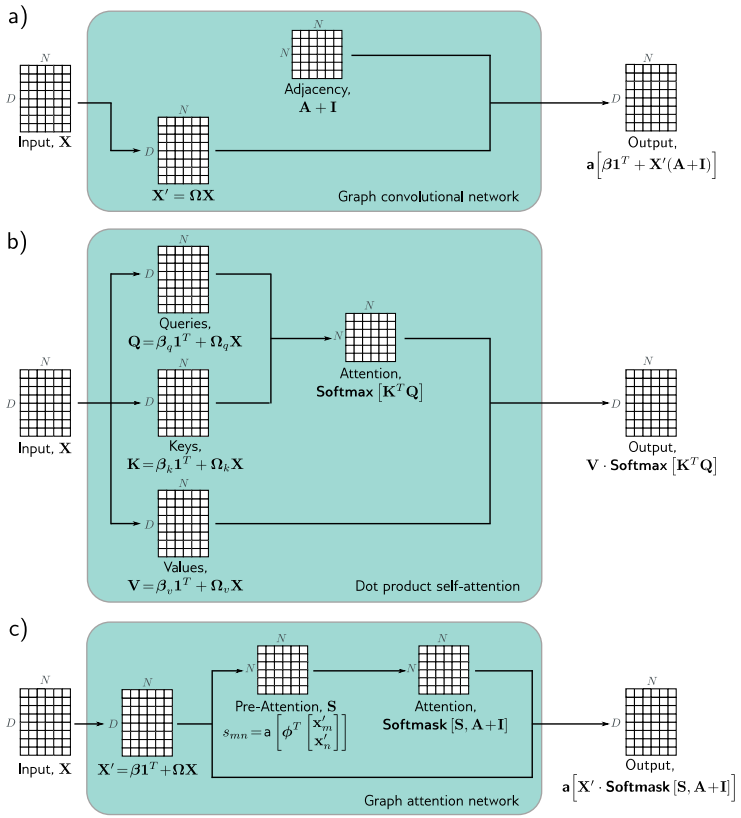


Edge Graph


a)
Original graph
3
2
1
5
4
6
b)
3
2/3
1/3
2
1
2/5
1/5
1/4
1/6
5
4/5
4
4/6
6
c)
Edge graph
2/3
1/3
2/5
1/5
1/4
1/6
4/5
4/6
Add new node
at each original edge
Connect new nodes if
original edges shared node

Graph Transformer: Aggregate by Attention

$$
\begin{aligned}
H'_k &= \beta_k 1^T + \omega_k H_k. \\
s_{mn} &= a[\phi_k^T [h'_m, h'_n]^T], \\
H_{k+1} &= a[H'_k \cdot Softmask[S, A + I]].
\end{aligned}
$$



Karate Club Graph

The Karate Club Graph is a well-known dataset in network science, commonly used in community detection, graph clustering, and GNNs research.

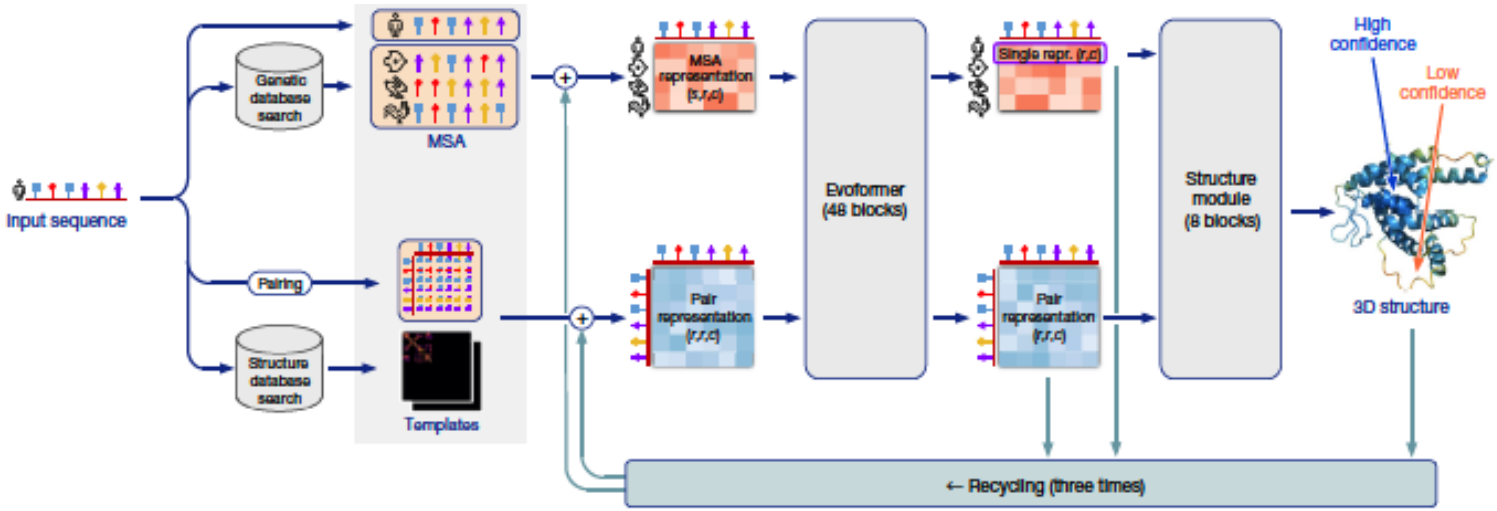
- The dataset was collected byWayne W. Zacharyin the 1970s
- It represents the social interactions of 34 members in a university karate club over a period of time.
- Due to internal conflicts, the club eventually split into two groups, forming two communities.
- Node (34): Each node represents a club member.
- Edge (78): An edge between two nodes indicates that the corresponding members interacted outside of the club
- Two Communities:
  One group followed the instructor (Leader: Node 0)
  Another group followed the club administrator (Leader: Node 33)

A Jupyter notebook example can be found on the course website.

Node-Level Task Example: AlphaFold

AlphaFold is a deep learning-based model for protein structure prediction

- It represents protein structures as spatial graphs where:
  Nodes: Amino acids (residues) in a protein sequence.
  Edges: Defined by spatial proximity between amino acids.
- These graphs capture important structural and functional relationships in proteins.



A simplified version of AlphaFold Evoformer Jupyter notebook example can be found on the course website.