

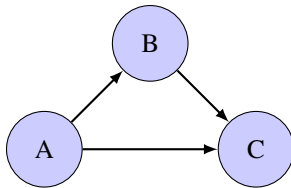
Graph Neural Network

Lecture 1



What Are Graph Neural Networks?

- GNNs are a type of deep learning model designed for graph-structured data
- Graphs consist of:
 - Nodes (Vertices): Represent entities (e.g., people in a social network).
 - Edges/Links: Represent relationships or interactions (e.g., friendships).



- GNNs leverage the structure of graphs to learn meaningful representations.

Message Passing:

Nodes aggregate information from neighbors.

Node Embeddings:

Transform features into a low-dimensional vector space.

Graph Aggregation:

Pool node embeddings to form a graph-level representation.

Key Idea:

- Nodes aggregate information from their neighbors.
- Enables nodes to learn representations based on local graph structure.

Process:

- Each node receives messages (feature vectors) from its neighbors.
- Messages are combined using a differentiable function (e.g., sum, mean, max).
- The aggregated message is used to update the node's state.

Mathematical Formulation:

$$h_v^{(l+1)} = \text{UPDATE} \left(h_v^{(l)}, \text{AGGREGATE} \left(\{h_u^{(l)} \mid u \in \mathcal{N}(v)\} \right) \right)$$

Where:

- $h_v^{(l)}$: Embedding of node v at layer
- $\mathcal{N}(v)$: Neighbors of node v .

Key Idea:

- Transform node features into a low-dimensional vector space.
- Captures structural and feature-based information.

Process:

- Each node starts with an initial feature vector.
- Through message passing, embeddings are refined over multiple layers.
- Final embeddings encode both local and global graph context.

Mathematical Formulation:

$$h_v^{(l)} = \sigma \left(W^{(l)} \cdot \text{AGGREGATE} \left(\{h_u^{(l-1)} \mid u \in \mathcal{N}(v)\} \right) \right)$$

Where:

- $W^{(l)}$: Learnable weight matrix at layer l .
- σ : Non-linear activation function (e.g., ReLU).

Key Idea:

- Pool node embeddings to form a graph-level representation.
- Enables tasks like graph classification or regression.

Process:

- Combine embeddings of all nodes in the graph.
- Common pooling methods: sum, mean, max, or attention-based.
- The resulting vector represents the entire graph.

Mathematical Formulation:

$$h_G = \text{POOL} \left(\{h_v^{(L)} \mid v \in V\} \right)$$

Where:

- h_G : Graph-level embedding.
- $h_v^{(L)}$: Final embedding of node v at layer L .
- V : Set of all nodes in the graph.

- Graph data is everywhere in real-world applications.
- Traditional neural networks struggle with non-Euclidean data.
- GNNs enable learning directly on graph structures, capturing both:
 - Node features.
 - Topological relationships (connectivity).

Examples of Graph Data

- Social networks: Users as nodes, friendships as edges.
- Molecular graphs: Atoms as nodes, chemical bonds as edges.
- Knowledge graphs: Entities as nodes, relationships as edges.
- Transportation networks: Locations as nodes, roads as edges.

- General structure:
 - **Input:** Graph data (nodes, edges, and features).
 - **Hidden layers:** Message passing and aggregation.
 - **Output:** Node embeddings, edge predictions, or graph-level classifications.
- Iterative information exchange across graph layers.
- Key insight: Combining node features with graph topology.

Graph Neural Networks (GNNs) are used in various structured data problems. These tasks are categorized into:

- Node-Level Tasks
- Edge-Level Tasks
- Graph-Level Tasks
- Dynamic and Spatio-Temporal Graph Tasks
- Other Applications

- **Node Classification:** Predicting categories of nodes (e.g., detecting fake accounts in social networks).
- **Node Clustering / Community Detection:** Identifying closely connected groups (e.g., social media groups).
- **Anomaly Detection:** Detecting unusual nodes (e.g., fraud detection in financial transactions).

- **Link Prediction:** Predicting missing or future connections (e.g., friend recommendations on Facebook).
- **Edge Classification:** Classifying relationships between nodes (e.g., type of citation between research papers).

- **Graph Classification:** Predicting the category of an entire graph (e.g., drug discovery by classifying molecular structures).
- **Graph Similarity / Matching:** Comparing graphs (e.g., plagiarism detection in research papers).
- **Graph Regression:** Predicting numerical properties of a graph (e.g., estimating molecular solubility).

Dynamic and Spatio-Temporal Graph Tasks

- **Dynamic Graph Learning:** Learning from evolving graphs (e.g., predicting social network interactions over time).
- **Spatio-Temporal Forecasting:** Using spatial and temporal dependencies (e.g., traffic prediction in road networks).

- **Recommendation Systems:** Using graph-based collaborative filtering (e.g., movie recommendations on Netflix).
- **Knowledge Graph Completion:** Predicting missing relations in knowledge graphs (e.g., completing facts in Wikidata).
- **Computer Vision with GNNs:** Using scene graphs for object detection.
- **NLP with Graphs:** Enhancing text representation (e.g., citation graphs for text classification).
- **Robotics and Control Systems:** Path planning and robot perception using scene graphs.

Advantages:

- Captures graph topology.
- Flexible and powerful.
- Handles irregular data.

Challenges:

- Computationally expensive.
- Scalability to large graphs.
- Over-smoothing in deep GNNs.

- Graph Neural Networks generalize deep learning to graph-structured data.
- Applications span diverse domains such as social networks, biology, and recommendation systems.
- Ongoing research addresses scalability and optimization challenges.