Graph Neural Networks (GNNs)

Introduction, Concepts, and Applications

Ali Balaj

FZ Business Informatics

December 9, 2024

Table of Contents

What are Graph Neural Networks?

- **Definition:** GNNs are neural networks designed to operate on graph-structured data, leveraging nodes, edges, and relationships.
- **Purpose:** They encode graph structures into feature-rich representations for tasks like node classification, link prediction, and graph classification.
- **Example Applications:**
  - Social Network Analysis
  - Recommendation Systems
  - Molecular Graphs in Drug Discovery

Why Graph Neural Networks?

- Traditional neural networks are not suited for graph data.
- GNNs leverage the rich connectivity and structural information inherent in graphs.
- Key advantages:
  - Capturing complex relationships between entities.
  - Directly encoding graph structures into embeddings.

Graph Representation

- **Graph Components:**
  - **Nodes (Vertices):** Entities in the graph (e.g., users, molecules).
  - **Edges:** Relationships or connections between nodes.
  - **Node Features:** Attributes of nodes (e.g., user preferences, atom types).
  - **Edge Features:** Attributes of edges (e.g., weights, interaction types).
- **Mathematical Representation:**
  - A graph $G = (V, E)$, where V is the set of nodes and E is the set of edges.
  - Node features: $X \in \mathbb{R}^{|V| \times d}$, where d is the feature dimension.
  - Adjacency matrix: $A \in \mathbb{R}^{|V| \times |V|}$.

Key Operations in GNNs

- **Message Passing:**
  - Nodes exchange information with their neighbors.
  - Messages are aggregated to update node embeddings.
- **Aggregation Function:**
  - Combines messages from neighbors (e.g., summation, mean, max).
  - Ensures permutation invariance.
- **Update Function:**
  - Updates node embeddings based on aggregated messages.
  - Often implemented as a neural network layer.

Graph Convolutional Network (GCN)

- **GCN Layer:**

$$H^{(l+1)} = \sigma\left(\hat{A}H^{(l)}W^{(l)}\right)$$

  where:
  - $\hat{A}$: Normalized adjacency matrix.
  - $H^{(l)}$: Node embeddings at layer l.
  - $W^{(l)}$: Trainable weights.
  - σ: Activation function (e.g., ReLU).
- **Intuition:** Each node aggregates information from its neighbors and updates its embedding.

Applications of GNNs

- **Node Classification:**
  - Predict node labels based on graph structure and features.
  - Example: Predict user behavior in a social network.
- **Link Prediction:**
  - Predict missing or future connections in a graph.
  - Example: Recommending friends on a social platform.
- **Graph Classification:**
  - Classify entire graphs.
  - Example: Classify molecules based on their properties.

Challenges of GNNs

- Scalability: Training GNNs on large graphs is computationally expensive.
- Over-smoothing: Node embeddings may become indistinguishable in deep GNNs.
- Dynamic Graphs: Adapting GNNs to handle evolving graph structures.

Future Directions

- Designing scalable architectures for large-scale graphs.
- Enhancing interpretability of GNN predictions.
- Exploring hybrid models combining GNNs with other machine learning techniques.

Questions?

Thank you!
Any questions?