Introduction to Graph Deep Learning

Guest lecture for CS 7643 Deep Learning, Fall 2023

Jiaxuan You
Incoming Assistant Professor at UIUC CS
Interconnected world

Gap

Modern ML
How to Represent Interconnected Data?

Interconnected world

Graph: The language for describing entities with relations

Graph-structured data

A node

An edge

Introduction to Graph Deep Learning, Jiaxuan You, UIUC CS
Goal of Graph Deep Learning
Enable DL research for the interconnected data
Graph: Ubiquitous across Disciplines

Molecule design

Protein interaction

Drug discovery

Social network

Recommender systems

Economic network

Policy making

- **Graphs:** *flexible* and *expressive*

- **Graphs** can *bridge interdisciplinary data*
Machine Learning with Graphs is Hard

- Arbitrary size and topological structure
- Nodes have no fixed ordering

Introduction to Graph Deep Learning, Jiaxuan You, UIUC CS
Graph Machine Learning Tasks

**Node-level prediction**

“Classify user by their type in a social network”

**Graph-level prediction**

“Predict which molecules are drug-like”

**Edge-level prediction**

“Recommend item nodes to user nodes”
Graph ML Tasks

- **Node-level prediction**
- **Graph-level prediction**
- **Edge-level prediction**

Key Idea: **Node Embeddings**

**Intuition:** Map nodes to $d$-dimensional embeddings such that similar nodes in the graph are embedded close together.
Graph ML Tasks

Key Idea: Node Embeddings

Node-level prediction

Graph-level prediction

Edge-level prediction

Graph Neural Networks (GNNs)

original network

ENC(u)

encode nodes

ENC(v)

embedding space

z_u

z_v
Deep Graph Encoders

Output: Node embeddings. Also, we can embed subgraphs, graphs.
Graph ML Setup

- Assume we have a graph $G$:
  - $V$ is the vertex set
  - $A$ is the adjacency matrix (assume binary)
  - $X \in \mathbb{R}^{m \times |V|}$ is a matrix of node features
    - Social networks – user attributes, molecule – atom types, ...
    - When there is no node feature in the graph dataset:
      - One-hot encodings – cannot generalize to new nodes
      - Vector of constant 1: [1, 1, ..., 1] – inductive, but less expressive
  - **Edge feature** can be incorporated as well
- $v$: a node in $V$; $N(v)$: the set of neighbors of $v$.
- Node features:
A Naïve Approach: MLP

- Join adjacency matrix and features
- Feed them into a deep neural net:

\[
O(|V|) \text{ parameters}
\]

- Issues with this idea:
  - Not applicable to graphs of different sizes
  - Sensitive to node ordering
Idea: Convolutional Networks

Goal is to generalize convolutions beyond simple lattices
Leverage node features/attributes (e.g., text, images)
Real-World Graphs

But our graphs look like this:

- There is no fixed notion of locality or sliding window on the graph
- Graph is permutation invariant
From Images to Graphs

Single Convolutional neural network (CNN) layer with 3x3 filter:

Idea: transform information at the neighbors and combine it:

- Transform “messages” \( h_i \) from neighbors: \( W_i h_i \)
- Add them up: \( \sum_i W_i h_i \)
Graph Convolutional Networks

- Graph Convolutional Networks: one of the first GNN models

Determine node computation graph

Propagate and transform information

[Graph Convolutional Networks: one of the first GNN models]

[Kipf and Welling, ICLR 2017]
Idea: Aggregate Neighbors

- **Key idea:** Generate node embeddings based on local network neighborhoods

![Input Graph and Target Node Diagram]
Idea: Aggregate Neighbors

- **Intuition:** Nodes aggregate information from their neighbors using neural networks
**Idea: Aggregate Neighbors**

- **Intuition:** Network neighborhood defines a computation graph

Every node defines a computation graph based on its neighborhood!
Deep Model: Many Layers

- Model can be of arbitrary depth:
  - Nodes have embeddings at each layer
  - Layer-0 embedding of node $u$ is its input feature, $x_u$
  - Layer-$k$ embedding gets information from nodes that are $K$ hops away

**Diagram**

- **Input Graph**
  - Nodes: A, B, C, D, E, F
  - Target Node: A

- **Layer-0**
  - Embeddings: $x_A$, $x_C$

- **Layer-1**
  - Embeddings: $x_A$, $x_C$, $x_B$, $x_E$, $x_F$

- **Layer-2**
  - Embeddings: $x_A$, $x_B$, $x_C$, $x_E$, $x_F$, $x_A$
The Math: GCN with Many Layers

- **Basic approach:** Average neighbor messages and apply a neural network

\[
\begin{align*}
    h_v^0 &= x_v \\
    h_v^{(l+1)} &= \sigma(W_l \sum_{u \in N(v)} \frac{h_u^{(l)}}{|N(v)|} + B_l h_v^{(l)}), \forall l \in \{0, \ldots, L - 1\}
\end{align*}
\]

- Initial 0-th layer embeddings are equal to node features
- Embedding of \( v \) at layer \( l \)
- Average of neighbor’s previous layer embeddings
- Non-linearity (e.g., ReLU)
- Total number of layers
- Embedding after \( L \) layers of neighborhood aggregation

Introduction to Graph Deep Learning, Jiaxuan You, UIUC CS
Training the GNN Model

How do we train the model to generate embeddings?

Need to define a loss function on the embeddings

Introduction to Graph Deep Learning, Jiaxuan You, UIUC CS
Model Parameters

Trainable weight matrices
(i.e., what we learn)

\[ h_v^{(0)} = x_v \]
\[ h_v^{(l+1)} = \sigma \left( \sum_{u \in N(v)} \frac{h_u^{(l)}}{|N(v)|} + B_l h_v^{(l)} \right), \forall l \in \{0, ..., L - 1\} \]
\[ z_v = h_v^{(L)} \]

Final node embedding

We can feed these embeddings into any loss function and run SGD to train the weight parameters

\( h_v^l \): the hidden representation of node \( v \) at layer \( l \)
- \( W_k \): weight matrix for neighborhood aggregation
- \( B_k \): weight matrix for transforming hidden vector of self

Introduction to Graph Deep Learning, Jiaxuan You, UIUC CS
How to train a GNN

- GNN provides us node embedding $z_v$
- **Supervised setting:**
  - we want to minimize the loss $\mathcal{L}$:
    \[
    \min_{\Theta} \mathcal{L}(y, f(z_v))
    \]
    - $y$: node/edge/graph label (from external sources)
    - $\mathcal{L}$ could be L2 if $y$ is real number, or cross entropy if $y$ is categorical
- **Unsupervised setting:**
  - Use graph structure/feature itself as supervision
    - E.g., link prediction, masked feature prediction, ...
Model Design: Overview

(1) Define a neighborhood aggregation function

(2) Define a loss function on the embeddings
Model Design: Overview

(3) Train on a set of nodes, i.e., a batch of computational graphs
Model Design: Overview

(4) Test time: Generate embeddings for nodes as needed

Even for nodes we never trained on!
GNN vs CNN & Transformer
GNN vs CNN

Convolutional neural network (CNN) layer with 3x3 filter:

- **GNN formulation:**
  \[
  h^{(l+1)}_v = \sigma \left( W_l \sum_{u \in N(v)} \frac{h^{(l)}_u}{|N(v)|} + B_l h^{(l)}_v \right), \forall l \in \{0, \ldots, L - 1\}
  \]

- **CNN formulation:**
  \[
  h^{(l+1)}_v = \sigma \left( \sum_{u \in N(v)} W^u_l h^{(l)}_u + B_l h^{(l)}_v \right), \forall l \in \{0, \ldots, L - 1\}
  \]

**Key difference:** We can learn different $W^u_l$ for different “neighbor” $u$ for pixel $v$ on the image.

---

*Image*  
*Graph*

---

*Introduction to Graph Deep Learning, Jiaxuan You, UIUC CS*
GNN vs CNN

Convolutional neural network (CNN) layer with 3x3 filter:

CNN can be seen as a special GNN with fixed neighbor size and ordering:
  - The size of the filter is pre-defined for a CNN.
  - The advantage of GNN is it processes arbitrary graphs with different degrees for each node.

CNN is not permutation invariant/equivariant.
  - Switching the order of pixels will lead to different outputs.

Key difference: We can learn different $W_t^u$ for different “neighbor” $u$ for pixel $v$ on the image.
Transformer

Transformer is one of the most popular architectures that achieves great performance in many sequence modeling tasks.

**Key component: self-attention**
- Every token/word attends to all the other tokens via matrix multiplication.
GNN vs Transformer

Transformer layer can be seen as a special GNN that runs on a fully-connected “token graph”!

Since each word attends to all the other tokens, the computation graph of a transformer layer is identical to that of a GNN on the fully-connected “token graph”.

Text

Fully-connected Graph
Applications of GNNs
Tasks on Networks

Tasks we will be able to solve:

- **Node classification**
  - Predict a type of a given node

- **Link prediction**
  - Predict whether two nodes are linked

- **Subgraph detection**
  - Identify certain subgraphs or paths within a graph

- **Graph classification**
  - Classify different graphs
Example (1): Financial Networks

- **Financial Networks**: Describe financial entities and their connections

**International banking**
- **Nodes**: Countries
- **Edges**: Capital flows

**Bitcoin transactions**
- **Nodes**: BTC wallets
- **Edges**: Transactions

Image credit: The Political Economy of Global Finance: A Network Model

Image credit: https://dailyblockchain.github.io/
ROLAND: GNN for Financial Networks

**ROLAND framework:**
- Transform financial networks as GNN computational graphs
- Learning from diverse objectives (node and edge level)

**Self-supervised** (from raw data)
- Will a user make a transaction? **Yes**
- What is the amount? **$500**
- When will it happen? **01/03**

**Supervised** (from external sources)
- Does a user involve fraud? **No**
- Does a user involve money laundering? **Yes**

Learn to predict

---

Introduction to Graph Deep Learning, Jiaxuan You, UIUC CS
Example (2): Recommender Systems

- **Users interacts with items**
  - Watch movies, buy merchandise, listen to music
  - **Nodes**: Users and items
  - **Edges**: User-item interactions

- **Goal**: Recommend items users might like

![Diagram of users and items with interactions and recommendations](image)
**PinSage: Graph-based Recommender**

**Task:** Recommend related pins to users

Task: Learn node embeddings $z_i$ such that $d(z_{cake1}, z_{cake2}) < d(z_{cake1}, z_{sweater})$

Predict whether two nodes in a graph are related
Example (3): Traffic Prediction
Road Network as a Graph

- **Nodes**: Road segments
- **Edges**: Connectivity between road segments

Image credit: DeepMind
Traffic Prediction via GNN

Predict the best route via Graph Neural Networks

- Used in Google Maps

THE MODEL ARCHITECTURE FOR DETERMINING OPTIMAL ROUTES AND THEIR TRAVEL TIME.

Introduction to Graph Deep Learning, Jiaxuan You, UIUC CS
Example (4): Drug Discovery

- **Antibiotics** are small molecular graphs
  - **Nodes:** Atoms
  - **Edges:** Chemical bonds


Image credit: CNN
Deep Learning for Antibiotic Discovery

- **A graph classification task**
- Predict promising molecules from a pool of existing candidates

Molecule Generation / Optimization

Graph generation: Generating novel molecules

Use case 1: Generate novel molecules with high drug likeness

Use case 2: Optimize existing molecules to have desirable properties

You et al., Graph Convolutional Policy Network for Goal-Directed Molecular Graph Generation, NeurIPS 2018
Frontiers of Graph ML Research
Designing more Expressive GNNs

Position-aware task

- GNNs fail at Position-aware tasks 😞
- $v_1$ and $v_2$ will always have the same computational graph, due to structure symmetry

- Q: Can we define deep learning methods that are position-aware?
Idea: P-GNN

- P-GNN proposes the first notion of **position embeddings for graphs**
  - Notably, Position embeddings are crucial for Transformers and LLMs

- P-GNN inspires many successful application of **Transformer + Graphs**
  - E.g., **GAT-POS** [Ma et al., 2021], **Graphormer** [Ying et al., 2021], ...
Graphs are Ubiquitous in ML problems

Graph is a superset for existing ML input data

Understand and inspire ML methods with graphs

Graph can represent novel ML applications
(1) Graphs in Missing Data Problems

- **Real-world data often exhibit missing values**
- **Idea:** Input data as heterogeneous graph
  - **Nodes:** Data points and features
  - **Edges:** Link data points with features
- **Graph offers unified solution for missing data problem**
  - Feature imputation – *edge-level prediction*
  - Label prediction – *node-level prediction*
- **10~20% lower MAE than SOTA baselines**
Can we translate any graph (e.g., brain network) to a neural network?

- Study the performance of NNs with network science tools
- Bridge deep learning with neuroscience
(2) New NN representation: Relational Graph

Relational Graph
- Translate any graph $\rightarrow$ NN
- Computation is defined as message passing over the graph

Neural network layer
Directed message computation
(3) **Graphs in Multi-task Learning Problems**

- **Graph representation for multi-task learning** (supervised/meta learning)
  - **Nodes**: Data points and ML tasks
  - **Edges**: A data point labeled by a task

- **Innovations**
  - Solve various multi-task settings via **graph ML**
  - Explore **new multi-task learning settings**: Leverage **auxiliary labels** during inference
  - **~13% improvement** with auxiliary task info
Summary

- **Why Graph Deep Learning?**
  - Enable DL for interconnected data

- **What is a GNN**
  - **Key**: iterative node neighborhood aggregation
  - CNN & Transformer can be considered as special GNNs

- **Applications of GNNs**
  - **Different levels**: Node, edge, subgraph, graph

- **Frontiers of Graph ML research**
  - Design more expressive GNNs
  - Empower general ML pipeline with graphs