Module 16 - Graph Representation Learning

Topics

16.1 Motivation

- Traditional graph analysis relies on **hand-crafted features** (degree, centrality, etc.)
- **Graph Representation Learning**: automatically learn **low-dimensional vector representations** (embeddings) of nodes, edges, or entire graphs
- Goal: encode graph structure and properties into dense vectors that can be used by ML models

Why Embeddings?

- Graphs are non-Euclidean — standard ML expects tabular/vector data
- Embeddings map nodes to $\mathbb{R}^d$ where:
  - **Similar nodes** in the graph are **close** in embedding space
  - **Dissimilar nodes** are **far** apart
- Enables use of standard ML classifiers, clustering, visualization

---

16.2 Node Embeddings

Encoder-Decoder Framework

- **Encoder**: maps node v to embedding $z_v \in \mathbb{R}^d$
- **Decoder**: reconstructs graph information from embeddings
- **Objective**: similar nodes should have similar embeddings

$$\text{similarity}(u, v) \approx z_u^T z_v$$

DeepWalk

- Use **random walks** on the graph (analogous to sentences in text)
- Apply **Skip-gram** (Word2Vec) to learn node embeddings
- Algorithm:
    1. Generate random walks from each node
    2. Treat walks as sentences
    3. Train Skip-gram: predict neighbors in the walk from the center node

Node2Vec

- Extension of DeepWalk with **biased random walks**
- Two parameters control walk behavior:
    - p **(return parameter)**: likelihood of revisiting the previous node
        - Low $p \to$ BFS-like, captures local structure
    - q **(in-out parameter)**: likelihood of exploring away from the previous node
        - Low $q \to$ DFS-like, captures community structure

$$P(v_i = x \| v_{i-1} = v) = \begin{cases} \frac{1}{p} & \text{if } d(t,x) = 0 \text{ (back to previous)} \\ 1 & \text{if } d(t,x) = 1 \text{ (same distance)} \\ \frac{1}{q} & \text{if } d(t,x) = 2 \text{ (farther away)} \end{cases}$$

- Can capture both **homophily** (community) and **structural equivalence** (role)

LINE (Large-scale Information Network Embedding)

- Preserves **first-order proximity** (direct connections) and **second-order proximity** (shared neighbors)
- Scalable to millions of nodes

Limitations of Shallow Embeddings

- Each node gets a unique embedding vector — no parameter sharing
- Cannot generalize to unseen nodes (transductive, not inductive)
- Don't leverage node features

16.3 Graph Neural Networks (GNNs)

- **GNNs**: neural networks that operate directly on graph structure
- Learn embeddings by **aggregating information from neighbors** (message passing)

Message Passing Framework

For each layer l:

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

- $h_v^{(l)}$: embedding of node v at layer l
- $h_v^{(0)} = x_v$ (initial node features)

Graph Convolutional Network (GCN)

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}H^{(l)}W^{(l)}\right)$$

- $\tilde{A} = A + I_n$ (adjacency with self-loops)
- $\tilde{D}$: degree matrix of $\tilde{A}$
- $W^{(l)}$: learnable weight matrix
- σ: activation function (ReLU)

GraphSAGE (SAmple and aggreGatE)

- **Inductive**: can generate embeddings for unseen nodes
- Aggregation functions: mean, LSTM, max-pooling
- **Sampling**: sample a fixed-size neighborhood for scalability

Graph Attention Network (GAT)

- Use **attention** to weight neighbor contributions differently
- Attention coefficient:

$$\alpha_{ij} = \frac{\exp(\mathrm{LeakyReLU}(a^T[Wh_i \| Wh_j]))}{\sum_{k \in N(i)} \exp(\mathrm{LeakyReLU}(a^T[Wh_i \| Wh_k]))}$$

- Multi-head attention for stability

Comparison

Method	Transductive/Inductive	Key Feature
DeepWalk	Transductive	Random walks + Skip-gram
Node2Vec	Transductive	Biased random walks (p, q)
GCN	Inductive	Spectral graph convolutions
GraphSAGE	Inductive	Sampling + aggregation
GAT	Inductive	Attention-based aggregation

16.4 Applications in Social Media

Application	Approach
Node classification	Predict user attributes (bot/human, political leaning)
Link prediction	Friend/follower recommendation
Community detection	Cluster nodes in embedding space
Influence prediction	Predict influence from node embeddings
Content recommendation	User-item graph embeddings
Anomaly detection	Detect unusual nodes/edges in embedding space
Knowledge graphs	Extract and reason over social knowledge graphs

Social Media-Specific Challenges

- **Scale**: billions of nodes and edges
- **Dynamic graphs**: network evolves over time — need temporal GNNs
- **Heterogeneous graphs**: multiple node/edge types (users, posts, hashtags)
- **Multi-relational**: different types of interactions (follow, like, retweet)

16.5 Review of Key Concepts (Sessions 9–15)

Session	Topic	Core Concepts
9	Information Diffusion	IC/LT models, influence maximization, epidemiological models
10	Influence & Homophily	Assortativity, types of influence/homophily, causal identification
11	Recommendation	CF, content-based, social regularization, evaluation
12	Behaviour Analytics	Individual profiling, collective behavior, echo chambers
13	Monitoring & Strategy	Social listening, KPIs, strategy frameworks
14	Strategies & Multi-modal	SNA for marketing, multi-modal GenAI analytics
15	Ethics	Privacy (GDPR), bias, misinformation, ethical frameworks

---

Key Takeaways

- Graph representation learning bridges graph data with standard ML via embeddings

- Shallow methods (DeepWalk, Node2Vec) use random walks; GNNs use message passing
- GNNs (GCN, GraphSAGE, GAT) are inductive and can leverage node features
- Applications span all areas of social media analytics: classification, recommendation, detection
- This field is rapidly evolving — heterogeneous graphs, temporal GNNs, and scalability are active research areas

---

References

- R1: William Hamilton, *Graph Representation Learning*, 2020
- CS224W of Stanford — Graph Neural Networks