

Large Scale Graph Mining

GRAPH NEURAL NETWORK

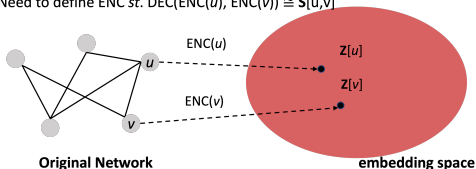
Nacéra Seghouani

Computer Science Department, CentraleSupélec
Laboratoire Interdisciplinaire des Sciences du Numérique, LISN
nacera.seghouani@centralesupelec.fr

2024-2025

Motivation & Challenges

Need to define ENC st. $\text{DEC}(\text{ENC}(u), \text{ENC}(v)) \cong \mathbf{S}[u,v]$



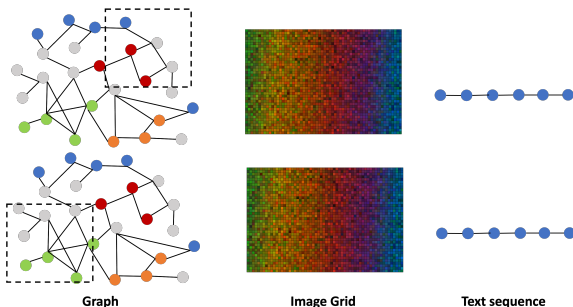
✎ The node embedding approaches discussed before use a shallow embedding approach to generate representations of nodes. Encoder is just an embedding lookup.

- ✓ Every node has its own embedding.
- ✓ Do not incorporate node features.
- ✓ No parameters shared between nodes.
- ✓ Cannot generate embeddings for nodes that are not seen during training

Motivation & Challenges

- ☞ Focus on more complex encoder models. Graph Neural Networks (GNNs)   is a general framework for defining deep neural networks on graph data.
- ☞ The key idea is to learn representations of nodes that actually depend on the structure of the graph, as well as any feature information we might have.
- ☞ Tasks:
 - ✓ Node classification
 - ✓ Link prediction
 - ✓ Community detection (Identify densely linked clusters of nodes)
 - ✓ Network similarity (How similar are two (sub)networks)
- ☞ The primary challenge is related to how to define complex encoders for graph-structured data.
- ☞ Usual deep learning such as Convolutional Neural Networks (CNNs) well-defined for grid-structured inputs (ex. images) and Recurrent Neural Networks (RNNs) well-defined for sequences (ex. text) are not relevant for graph data.

Motivation & Challenges



- 👉 Networks are far more complex
 - ✓ Arbitrary size and complex topological structure (i.e., no spatial locality like grids)
 - ✓ There is no fixed notion of window sliding or locality on the graph. No fixed node ordering or reference point
 - ✓ Often dynamic and have multimodal features
- 👉 Need to define a new deep learning architecture: $ENC(v)$ is multiple layers of non linear transformations based on graph structure

Deep Neural Network Basics

- Loss function: $\text{Min}_{\theta} \mathcal{L}(y, \sigma_{\theta}(x))$
where σ is a linear layer or no linear such as MLP (a multilayer perceptron), x a minibatch sample of input data
Pick θ that minimises the loss $\mathcal{L}$
- Forward propagation: Compute $\mathcal{L}$ given x (epoch=forward pass)
- Back-propagation: Compute/estimate gradient $\nabla_{\theta} \mathcal{L}$ using a chain rule. Update θ
- Stochastic gradient descent (SGD) to minimise $\mathcal{L}$ for weights θ over many iterations.

Permutation Invariance and Equivariance

- ☞ One naive approach to define a deep neural network over graphs would be to simply use the adjacency matrix as input to a deep neural network.
 - ✓ For example, to generate an embedding of an entire graph we could simply flatten the adjacency matrix and feed the result to a multi-layer perceptron (MLP):
 $\mathbf{z}_{\mathcal{G}} = \text{MLP}(\mathbf{A}_1 \oplus \mathbf{A}_2 \oplus \dots \oplus \mathbf{A}_{|V|})$ where $\mathbf{A}_i \in R^{|V|}$ denotes a row of the adjacency matrix. $\oplus$ is vector concatenation (don't confuse with direct sum)
- ☞ The issue is that depends on the order of nodes in $\mathbf{A}$.

Permutation Invariance and Equivariance

- ☞ In mathematical terms, any function f that takes $(\mathbf{A}, \mathbf{X})$ as input (resp. adjacency matrix and node features vector) should ideally satisfy one of the two following properties:

- ✓ Permutation Invariance: permute the input, the output stays the same

$$f(\mathbf{PAP}^T, \mathbf{PX}) = f(\mathbf{A}, \mathbf{X})$$

- ✓ Permutation Equivariance: permute the input, output also permutes accordingly

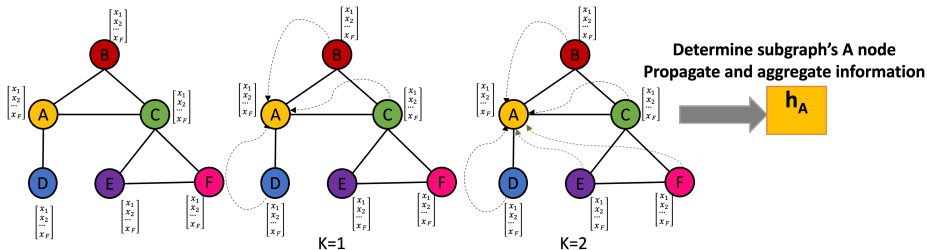
$$f(\mathbf{PAP}^T, \mathbf{PX}) = \mathbf{P}f(\mathbf{A}, \mathbf{X})$$

where $\mathbf{P}$ is a permutation matrix.

- ☞ The shallow encoders discussed are examples of permutation equivariant functions.
- ☞ Graph neural networks consist of multiple permutation equivariant / invariant functions.

Problem Definition

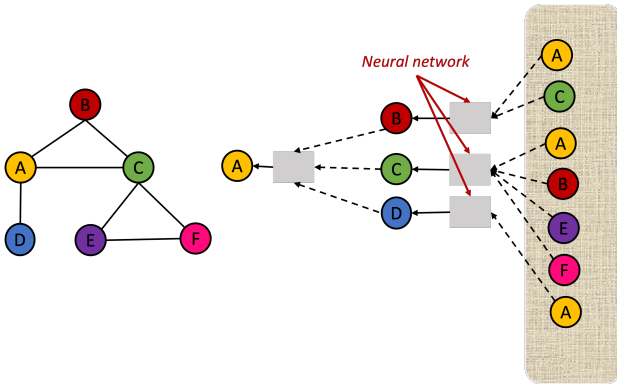
- Given $G(V, E)$, its adjacency matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ and feature matrix $\mathbf{X} \in \mathbb{R}^{n \times F}$. A graph, Node attributes/feature vectors, (part of nodes are labeled).
- Examples: Social networks (user profile), Biological networks (gene expression profiles, gene interactions), ...
- Idea: Homophily. Connected nodes are informative nodes $\Rightarrow$ Node's neighbourhood needs a subgraph computation.



$\Rightarrow$ Learn how to propagate information across the graph to compute node features

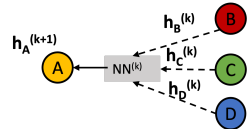
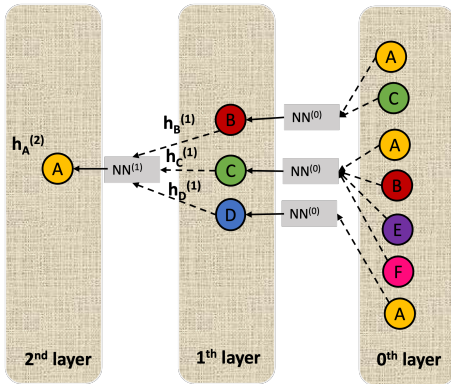
Problem Definition

- Generate node embeddings based on local neighbourhoods
- Aggregate information from neighbours using neural network



Problem Definition

- 👉 Model of arbitrary depth
- 👉 Layer- k embeddings gets information from nodes that are k hops away



Neural Message Passing

Design graph neural networks that are permutation invariant / equivariant by passing and aggregating information from neighbours

- ☞ Recursively update the node features by aggregating information from their neighbouring nodes.
 - ✓ The process performs series of message passing steps, where each node sends a message to its neighbors, which are then aggregated to produce an updated feature representation for each node.
 - ✓ At each iteration, every node aggregates information from its local neighborhood, and as these iterations progress each node embedding contains more and more information from further reaches of the graph.
 - ✓ After iteration k every node embedding contains information about its k -hop neighborhood.
- ☞ But what kind of “information” do these node embeddings actually encode? Generally, this information comes in two forms:
 - ✓ On the one hand there is structural information about the graph: after GNN (k) message passing, the embedding $\mathbf{h}_u$ of u might encode information about the degrees of all the nodes in u 's k -hop neighborhood.
 - ✓ The other key kind of information captured by GNN node embedding is feature- based. Each node also encode information about all the features in their k -hop neighborhood, and possibly edge representations.

Neural Message Passing

☞ $\mathbf{h}_u^k$ embedding of u at k -th layer

$$\mathbf{h}_u^{(k+1)} = \mathbf{g}^{(k)} \left(\mathbf{h}_u^{(k)}, \mathbf{f}^{(k)}(\{\mathbf{h}_v^{(k)}, \forall v \in \mathcal{N}(u)\}) \right) = \mathbf{g}^{(k)} \left(\mathbf{h}_u^{(k)}, m_u^{(k)} \right)$$

where $\mathbf{f}^{(k)}$ and $\mathbf{g}^{(k)}$ are resp. aggregation and transformation (update) function at k -layer, differentiable functions (i.e., neural networks) and $m_u^{(k)}$ is the message that is aggregated from u 's $\mathcal{N}(u)$.

- ✓ The initial embeddings at $k = 0$ are set to the input features for all the nodes, i.e., $\mathbf{h}_u^0$. After running K iterations of the GNN message passing, we can use the output of the final layer to define the embeddings for each node, i.e., $\mathbf{z}_u = \mathbf{h}_u^K, \forall u \in V$
- ✓ For ex., rich node features to use are gene expression features in biological networks or text features in social networks. However, where no node features there are several options using node statistics using metrics discussed before.

The basic GNN message passing

$$\mathbf{h}_u^{(k)} = \sigma \left(W_{self}^{(k)} \mathbf{h}_u^{(k-1)} + W_{neigh}^{(k)} \sum_{v \in \mathcal{N}_u} \mathbf{h}_v^{(k-1)} + \mathbf{b}^{(k)} \right)$$

where $W_{self}, W_{neigh} \in \mathbb{R}^{d \times d}$ are trainable parameter matrices and σ denotes an element wise non-linearity (e.g., a tanh or ReLU). The bias term $\mathbf{b}^{(k)} \in \mathbb{R}^d$ (omitted for notation simplicity)

Basic GNN:

$$m_u^{(k)} = \sum_{v \in \mathcal{N}_u} \mathbf{h}_v^{(k)}$$

$$g(\mathbf{h}_u, m_u) = \sigma \left(W_{self} \mathbf{h}_u + W_{neigh} m_u \right)$$

$$\mathbf{H}^{(k)} = \sigma \left(\mathbf{H}^{(k-1)} W_{self}^{(k)} + \mathbf{A} \mathbf{H}^{(k-1)} W_{neigh}^{(k)} \right) \text{ where } \mathbf{H}^{(k)} \in \mathbb{R}^{|V| \times d}$$

The basic GNN message passing

Message Passing with Self-loops: add to the neighbours the node itself for aggregation. No need to define an explicit update, as it is implicitly included in the aggregation.

$$m_u = \sum_{v \in \mathcal{N}_u} \mathbf{h}_v^{(k)}$$

Adding self-loops is equivalent to sharing parameters between the W_{self} and W_{neigh} matrices:

$$\mathbf{H}^{(k)} = \sigma \left(\mathbf{H}^{(k-1)} W^{(k)} + \mathbf{A} \mathbf{H}^{(k-1)} W^{(k)} \right) \text{ where } \mathbf{H}^{(k)} \in \mathbb{R}^{|V| \times d}$$

Simplifying the message passing alleviates overfitting, but also limits the GNN expressivity, as the information coming from the node's neighbours cannot be differentiated from the one coming from the node itself.

Simplified GCN

- Need to normalize

$$m_u^{(k)} = \frac{1}{|\mathcal{N}_u|} \sum_{v \in \mathcal{N}_u} \mathbf{h}_v^{(k)}$$

- Symmetric normalization

$$m_u^{(k)} = \sum_{v \in \mathcal{N}_u} \frac{\mathbf{h}_v^{(k)}}{\sqrt{|\mathcal{N}_u| |\mathcal{N}_v|}}$$

$$\mathbf{h}^{(k)} = \sigma \left(W^{(k)} \sum_{v \in \mathcal{N}_u} \frac{\mathbf{h}_v^{(k)}}{\sqrt{|\mathcal{N}_u| |\mathcal{N}_v|}} \right)$$

- More sophisticated aggregation: Set pooling, ...
- Neighbourhood attention strategy to improve the aggregation by assigning an attention weight or importance/influence to each neighbour.

R-GCN Multi-relational Graphs

- Commonly known approach is Relational Graph Convolutional Network (R-GCN, Schlichtkrul et al, 2017):
- augment the aggregation to accommodate multiple relation types by specifying a separate transformation matrix per relation type:

$$m_u^{(k)} = \sum_{r \in R} \sum_{v \in \mathcal{N}_u(r)} \frac{W_r \mathbf{h}_u}{f_{norm}(\mathcal{N}_u, \mathcal{N}_v)}$$

f_{norm} is normalisation function, depends on the neighbourhood of u as well as v being aggregated over. Several normalisation strategies analogous to those discussed earlier.

- Drawback: The number of parameters to learn (one matrix to learn per relation) increases
⇒ overfitting on rare relations and slow learning.
⇒ a separate aggregation parameter per relation, is applicable for discrete edge features. Otherwise we can leverage these features in attention or by concatenating this information with the neighbour embeddings